Asian Journal of Information Technology 15 (9): 1437-1442, 2016
ISSN: 1682-3915

Test Case Classification Using Tuned Fuzzy Logic with Test Case Reusability for Test Suite Reduction

[1] R. Kamalraj and [2] A. Rajivkannan
[1]CSE Department, SNS College of Technology, Coimbatore, Tamil Nadu, India
[2]CSE Department, KSR College of Engineering, Tiruchengode, Tamil Nadu, India

Abstract: The software testing activity needs to work perfectly in a fast manner to test the system and to find out the areas to be improved when defects present in system. Hence, the appointed testing team needs to have an effective set or optimal test cases to conduct the testing procedures in order to satisfy the customer's requirements to deliver the product before the deadline. To have optimal test cases from the test case pool the reduction technique is to be used frequently to make it perfect test cases for declaring the test suite which is to be applied. In this regard this study, is focused on test case reusability combined with risk factor of test cases using tuned fuzzy logic for generating reduced and optimized test suite. This approach can help to increase the confidence level of the testing team to perfectly manage the available resources in order to achieve the proposed target testing rates of individual as well as the whole team.

Key words: Test suite optimization, test case reusability, test case classification, fuzzy logic, India

INTRODUCTION

Software Testing is one of the most important phases in software development life cycle. It has variety of testing activities to verify whether it has reached the eligibility criteria specified by the customer. Hence, it consumes 40% of effort from software development effort for finding test suites from the test pool or design new test suites and executes test suites. In this phase, it does not mean that only checking the software boundary but also customer specification proposed when requirements elicitation was executed. That's why many of the software development companies spend much resource as well as give much preference for the testing phase before releasing the developed software system for customer experience.

In software development a single version of the software may not satisfy everything required by customer for implementing business tasks. When they found that the existing tested features not completely satisfy the required business process function and system performance then a request will be placed to the development team, for promoting the existing system to the expected level. In this state the 'Regression Testing' is conducted for verifying that the modification system function and code are satisfying the required system specification or not (Jalote, 2008). The test suite for conducting testing system attributes is to be designed or identified from the test suite pool. When new features are added or existing function procedures are modified then testing is to be done to check 'Whether there is any effect found in the modified system characteristics'. So, providing suitable and avoiding non suitable test cases is a challenging activity to testing team to effectively and efficiently use the allocated resources.

Literature review: The test suite reduction is one of the important requirements for testing team to find suitable and less size of test suite to conduct testing. The algorithm HGS is proposed for reducing the test suite size for conducting the performance evaluation. In this algorithm the occurrences of a test case in the given test set processed to give result (Harrold *et al*., 1993). The reduced test suite will have the singleton test cases for satisfying the suitable. The next algorithm is BOG and it works based on the matrix operations to reduce the taken test sets. It takes the optimal test case by applying genetic approach and that selected test case should satisfy the requirements. In this max list and min list are processed to generate the test suite with reduced size (Parsa and Khalilian, 2010). The Modified Condition/Decision Coverage (MC/DC) approach is suggesting the ideas to reduce the test suite and prioritize the test cases in the test suites (Jones and Harrold, 2003). Many simple and hybrid techniques are available to reduce the test suite to a certain size (Zhang *et al*., 2011). Among them a combined approach such as 'Multi-objective genetic and greedy approach' was

Corresponding Author: R. Kamalraj, CSE Department, SNS College of Technology, Coimbatore, Tamil Nadu, India

proposed to generate test cases and reduce the size of them (Yoo and Harman, 2012). The meta-heuristic algorithm is available to find the result of test suite minimization (Ansari *et al*., 2013; Irfan *et al*., 2013). And Redundancy based test reduction technique considers redundancy of the same test case in the same test suite. By considering and conducting the test case only once then it may reduce the testing effort (Yoo and Harman, 2010). The Orthogonal Array method also can be practiced for finding small size of test suite in order to meet the expectations of testing team (Banerji, 2012). The greedy approach with coverage based approach gives expected reduction rate but it does not follow the success rate of the test cases (Harris and Raju, 2015).

Hence, in this study the fuzzy logic with tuning parameter is suggested to enhance the performance of test suite reduction compared to the state of art algorithms.

MATERIALS AND METHODS

Proposed technique: The proposed technique is composed of 'Reusability and Risk based Test Case classification method (RRTC)' and it is a two level classification approach. The requirements coverage based test cases are identified and those will be the input to tuned fuzzy logic to reduce the test suite as perfect one to conduct required testing procedures. It is used to take decisions according to the constraints satisfaction. When fuzzy logic constraint gets satisfied then the element which satisfies it will be a part of the solution. The fuzzy logic expression is If 'X' and 'Y' then 'Z'.

As per the above expression the conditions are organized to derive the solution 'Z' when condition gets satisfied. In this the main condition parameter and tuning parameter is 'Test case Reusability', because it reduces the steps by identifying 'Less Reused' Test cases not to be processed. So, processing time and dataset size will be reduced for further step in the proposed technique to derive the test suite with optimized test cases.

Test suite reduction using fuzzy logic: The RRTC (Reusability and Risk based Test Case Classification) model helps to find or take suitable test cases for performing testing activities to measure the quality attributes of the system for delivery. It can give a desired result by grouping the identified test cases in two different classes. In this, the defect of the test cases can be obtained when it deviates from the resource allocated for testing the system functionality. If resource exceeds from the given resource, then that test case set has to be noted for using in future. To optimize the test suite with

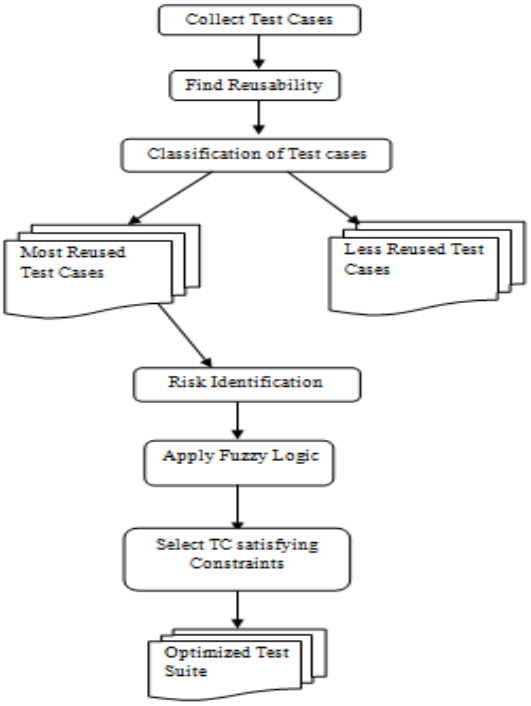

Fig.1: Fuzzy logic model with reusability and risk

appropriate test case, we recommend choosing Boolean values used with Boolean operations to represent the category of test cases based on their resource consumption rate and reusability factor. In the proposed approach, the 'tuning parameter' is the reusability of test cases. The flow of the approach is described in the below Fig. 1.

Steps of RRTC

Collecting test case: The test cases are collected as per the functionality to be tested in Regression Testing.

Find reuse frequency: In this the reuse frequency of the test case should be identified by collecting its total usage in the test procedures conducted on various similar applications. To compute the value of Reusability factor of a particular test case is:

$$\mathrm{Re}(t) = \frac{\mathrm{NTU}}{\mathrm{TNA}} \times 100\%$$

Where:
Re(t) = Reusability of Test case
NTU = No. of Times used
TNA = Total No. of times Applications tested

To select test case in final test suite this reusability value of test cases will be compared with a standard

threshold value. Hence, here 50% is considered as 'threshold' value for classifying the test cases as 'Most reused' and 'Less Reused' classes.

The below representation is used to state how a test case can be classified. The above formula can help to find the very less reused test cases occupied in the test case pool. It may waste the space required for maintaining them in the pool repository. So, by identifying the very less reused test cases and move them to separate space may give room for new recent and enhanced test cases for testing the system.

Algorithm A;test case:

If $Re(t_{id})$ >= 'Average' then
  Classified 't_{id}' as 'Most Reused' and '1' will be assigned
Else
  Classified 't_{id}' as 'Less Reused' and '0' will be assigned

Classification: This step will take the history of test cases associated with their defects or performance to group them as 'Mostly-Reused' and 'Less-Reused' test cases according to their usage in the total testing done. Then risk of the individual test cases to be found for further classification like 'Low Risks' and 'High Risks' used cases. It is a next level classification to find suitable test case which one can satisfy the available resources for conducting testing procedures.

Applying fuzzy logic with reusability of test cases: The fuzzy logic can help to take decision as per the conditions get true or the elements satisfying those conditions. Here, conditions for picking the suitable test cases whether it can satisfy the given requirement. Here, we proposed the selection fuzzy logic condition is:

$$F(t) = \{R(t) < \text{AvgPerf AND} Re(t) = \text{true}, 1 \text{Else}_0$$

Where:

T(t) = Temporary test set
R(t) = Test case risk
Re(t) = Reusability level of test case
AvgPerf = The average performance of test cases

The constraints for test case selection are given in the above formula for fuzzy logic approach. In the proposed method first the rest cases having good reusability tuning factor will be considered to classify them as perfect test cases for executing testing procedures. It may reduce the time by eliminating the test cases to be processed in the group of 'Less Reused' test cases. By apply the Eq. 1 the most used reused groups can be identified for testing the required system functionality. The test cases having more than 50% of reusability those will be sent to further classification for final test suite.

Compute risk factor of each test case: For the identified 'mostly reused test cases', the risk factor will be computed for once again classify them for picking perfect test cases. To find the Risk level the sum of the performance attributes such as 'Execution Time' and 'System Memory' will be done. Then each test case risks will be compared with one another when they selected for put it into test suite. The Risk Matrix can be obtained by following the script

Algorithm B; risk matrix:

If $R(t_{id})$ >= 'Average' then
  Classified the test case 't_{id}' as 'High Risk' and '0'
Else
  Classified the test case 't_{id}' as 'Low Risk' and '1'

C. Test cases Algorithm for Reducing Test suite using Fuzzy Logic with Reusability Tuning Factor
Input: Output of the Success Rate approach
Output: Optimized Test suite

RM – Risk Based Test case Matrix
TRS – Temporary Resultant Matrix
OTS – Optimized Test suite set

Algorithm Begin {

Step 1 : // Coverage Matrix will be the input
  //Apply formula 1 to compute 'Reusability' of each test case
  ReM = Boolean Matrix

Step 2:
//apply formula 3 to generate Risk Matrix of each test case
  RM = Boolean Matrix

Step 3:
// Apply fuzzy logic, the formula 2 and take those values to the Boolean intersection operation to find the result
  OTM = RS n RM
Where OTM – optimized test suite matrix

Step 4: filter the test cases having '1' in the resultant matrix from OTM and store it optimized test suite set (OTS)
  OTS = { t1,t2,t3…}
} End

RESULTS AND DISCUSSION

Concept illustration: To prove the proposed concept the empirical study was done using the Login and Registration Form. These may be in traditional style as well as advanced style. So customer may prefer the traditional form in initial level. But when they want to change or update these styles by adding few more parameters then regression testing is recommended. As per this idea the following test cases are taken to demonstrate the idea by using the empirical study.

Table 1: Test cases for login and registration in a web based application

Requirements (TS)	Test cases
textfield should not be empty	t1
Email format should followed	t2
Same username should not be used	t3
Perfect zip code should be used	t4
Mandatory fields should be empty	t5
Server should give response	t6
Proper Error Msgs should be used	t7

Here, the given list of test cases is analyzed to conduct empirical study for the proposed ideas. The program units are given in the table with appropriate details. The test cases are created for validating the requirements in proposed system functionality. The test cases from 'NIST (National Institute of Standards And Technology) test dataset are also used to conduct experiments in the platform of java. But in that open source test case the datasets are not organized as per their usage level (Table 1).

The above sample test cases are applied on various web applications having same kind of functions to check the results for defining the resource consumption on testing the system (Sprenkle *et al*., 2005). Here, the Test cases 'T1' are focusing on whether the valid input domain is processed for Unit Testing. Through this the given input, if it is valid the tester may believe that functional unit should give expected perfect result as specified in the testing plan. The 'JUnit' tool was used to test the programs units and the time taken for executing the test cases with test scripts also noted for investigation purpose.

Test suite reduction based on reusability: Input Matrix 1, Coverage matrix for above test case as per their requirements coverage is TS = $\{t_1, t_2, t_3, t_4\}$; Requirements R = $\{r_1, r_2, r_3\}$:

$$C = \begin{array}{c} \\ r1 \\ r2 \\ r3 \end{array} \begin{pmatrix} t1 & t2 & t3 & t4 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

From their previous past history the usage ratio is computed for preparing the Reusability matrix (ReM). This input matrix combined with coverage matrix to find final test cases for optimization:

$$C = \begin{array}{c} \\ r1 \\ r2 \\ r3 \end{array} \begin{pmatrix} t1 & t2 & t3 & t4 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

$$TRS = C \bigcap ReM$$

$$TRS = \begin{array}{c} \\ r1 \\ r2 \\ r3 \end{array} \begin{pmatrix} t1 & t2 & t3 & t4 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

From the above matrix TRS, the values in column 1 in all the rows are containing value 1. It represents that the test case t1 can test all the three requirements. And t2 is able to test 'r2' and t3 is able to test r3 requirements.

$$ts\_reduced = \{t1, t2, t3\}$$

Step 2: Test suite Optimization using Fuzzy logic with Risk Factor

Resultant Matrix from RS = {t1, t2, t3}

Risk Based Matrix of these test cases is:

$$RM = \begin{array}{c} \\ r1 \\ r2 \\ r3 \end{array} \begin{pmatrix} t1 & t2 & t3 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Apply the Fuzzy logic with Intersection operation(AND) to derive the desired results. OTM = RM n TRS, i.e.:

$$OTM = \begin{array}{c} \\ r1 \\ r2 \\ r3 \end{array} \begin{pmatrix} T1 & T2 & T3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

The intersection operation is a strict operation because it gives result when both input conditions are satisfied. By filtering the Boolean value '1' in the above OTM matrix we can obtain the final test suite which could be applied for testing the requirements r1, r2 and r3 with saved resource energy.

Therefore the final optimized test suite containing the test cases are given below:

$$OTS = \{t1, t2\}$$

This final test suite is having only two test cases such as t1 and t2 for testing those 3 different requirements because they consume only less amount of cost for testing those requirements:

$$TSRFL = [1-[(ITC-OTC)/ITC]]\times 100$$
$$=[1-[(4-2)/4]]\times 100= [1\text{-}1/2\ ]\times 100 = 50\%$$

Where:
TSR = Performance of this combined approach
ITC = Input number of Test Cases
OTC = Output number of Test Cases
TSR_{RB} = Test Suite Reduction by using fuzzy logic

The above result 67% shows that the identified test suites Fuzzy logic based on the reusability and risk of selected test suites, improving their quality in testing domain. It may lead the testing team and resource management team to act effectively. From the data analysis, for measuring the risk factor depends on the following attributes of system:

- Size of Module to be tested
- Input Boundary
- Memory Size
- CPU Speed
- Size of Test Scripts applied

The result of the empirical study is plotted in the Fig 2-4 and from that the size of the test suite is minimized to approximately 78%, in order to use it in the 'Regression Testing'. Here the HGS, BOG and proposed technique FRR (Fuzzy logic with Reusability and 'Risk Factor') Output is taken for comparing the performance. The proposed technique may be suitable when sufficient past history of the test cases is available. Else the traditional algorithms can satisfy the result with perfect test cases. By having reduced, the suitable test suite can save certain amount of energy and efforts of 'Software Testing' activity.

Merits and demerits of proposed technique: This proposed combined approach takes the test case attributes such as 'Reusability' and 'Risk Factor' with Fuzzy Logic to generate the final optimal test suite. The reduction of test suite and optimizing test suites are improved by adding procedures while identifying the test cases as per their requirements coverage. From the above empirical study the following merits and demerits are observed and listed in Table 2.

The proposed technique is designed using 'Reusability' as tuning factor for reducing the test suite and find optimized test suite. It takes more attributes of test cases for processing and finding less size and generating optimal test suite automatically.

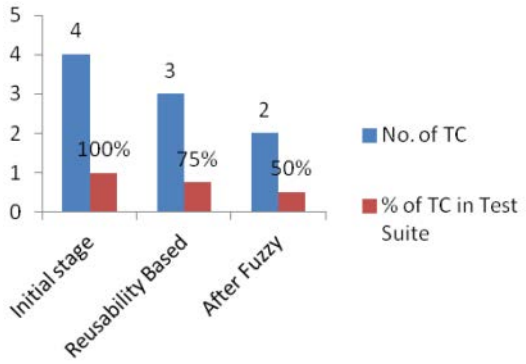

Fig. 2: Test suite reduction in proposed technique (%)

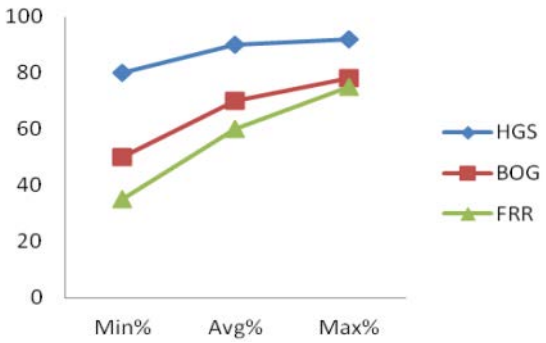

Fig. 3: Test suite requirements coverage (%)

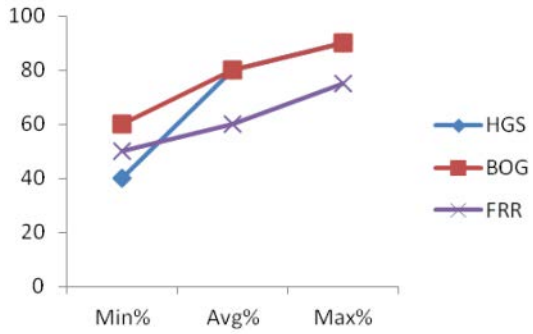

Fig. 4: Test suite reduction comparison (%)

Table 2: Merits and demerits table

Merits	Demerits
Reduced test suite	Time consuming activity
Optimal test cases	History of test cases is needed
Reducing the cost of test	
Case execution	
Handling the resources effectively	Strict approach
Effective risk management on testing	

CONCLUSION

The Software test case reduction technique is preferable not only in finding the test cases but also for

producing optimal test cases to conduct effective testing approaches (Alsmadi and Alda, 2012). In order to achieve the reduced test suite and improved optimality in generating test cases, the 'reusability' of test case taken as 'Tuning Parameter' to improve the performance of test suite reduction and optimization by adding 'Risk Factor' also in fuzzy logic. From the conducted empirical study with different program units and web applications, the proposed fuzzy logic with tuning parameter technique can be able to select and optimize the test suite with less size which covers the variety of requirements proposed to test the system functionality. This reduces the testing effort and also helps SQA team to improve the quality of testing procedures.

REFERENCES

Alsmadi, I. and S. Alda, 2012. Test cases reduction and selection optimization in testing web services. Intl. J. Inf. Eng. Electron. Bus., 5: 1-8.

Ansari, A.S., K.K. Devadkar and P. Gharpure, 2013. Optimization of test suite-test case in regression test. Proceeding of the International IEEE. Conference on Computational Intelligence and Computing Research, December 26-28, 2013, IEEE, Enathi, India, ISBN: 978-1-4799-1594-1, pp: 1-4.

Banerji, S., 2012. Orthogonal array approach for test case optimization. Orthogonal array approach for test case optimization. Intl. J. Adv. Res. Comput. Commun. Eng.,: 1-613.

Harris, P. and N. Raju, 2015. A greedy approach for coverage-based test suite reduction. Int. Arab J. Inf. Technol., 12: 17-23.

Harrold, M.J., R. Gupta and M.L. Soffa, 1993. A methodology for controlling the size of a test suite. ACM Trans. Software Eng. Methodol., 2: 270-285.

Irfan, S., V. Kulkarni and K.H. Rao, 2013. Test suite optimization for regression testing by using new meta-heuristics algorithm. IJSETR., Vol. 2,

Jalote, P., 2008. A Concise Introduction to Software Engineering. Springer Science & Business Media, Berlin, Germany, ISBN:978-84800-301-9, Pages: 265.

Jones, J.A. and M.J. Harrold, 2003. Test-suite reduction and prioritization for modified condition/decision coverage. IEEE Trans. Software Eng., 29: 195-209.

Parsa, S. and A. Khalilian, 2010. On the optimization approach towards test suite minimization. Intl. J. Software Eng. Appl., 4: 15-28.

Sprenkle, S., S. Sampath, E. Gibson, L. Pollock and A. Souter, 2005. An empirical comparison of test suite reduction techniques for user-session-based testing of web applications. Proceeding of the 21st International IEEE. Conference on Software Maintenance, September 26-29, 2005, IEEE, New York, USA., pp: 587-596.

Yoo, S. and M. Harman, 2010. Using hybrid algorithm for pareto efficient multi-objective test suite minimisation. J. Syst. Software, 83: 689-701.

Yoo, S. and M. Harman, 2012. Regression testing minimization, selection and prioritization: A survey. Software Testing Verification Reliabil., 22: 67-120.

Zhang, L., D. Marinov, L. Zhang and S. Khurshid, 2011. An empirical study of junit test-suite reduction. Proceeding of the 22nd International IEEE. Symposium on Software Reliability Engineering, November 29-December 2, 2011, IEEE, Hiroshima, Japan, ISBN: 978-1-4577-2060-4, pp: 170-179.