JCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol.-6, Issue-9, Sept. 2018 E-ISSN: 2347-2693

Fuzzy Expert System Based Test Cases Prioritization

Taranum Thakur[1*], Narinder Rana[2]

[1,2]Dept. Of Computer Science, Rayat Group of Institutions, Railmajra, Punjab, India

[*]Corresponding Author: thakurtaranum90@gmail.com, Tel.: +91-9805159940

Available online at: www.ijcseonline.org

Accepted: 22/Sept./2018, Published: 30/Sept./2018

Abstract- Software engineers waste a lot of time during software testing. The goal of testing is to determine error in a system. Test case generation is the procedure of developing test suites for identifying system errors. A test set is a collection of applicable test cases bunched together. It is also seen in the industry with large amount of funds being used during the software process. During software testing, we have used test case as input and has determined the final output. So, our first objective is to choose the right test case for the software testing process. In order to give correct output, it is very difficult to select test cases. So, the test case generation is an NP (non-deterministic polynomial-time hardness) problem. There are numbers of algorithms available for software testing but to choose the best algorithm as per the requirement is mostly needed. In this research work, to solve the NP hard problem of software testing, we have used Fuzzy logic classifier. Fuzzy logic is a rule based algorithm that works on if - else statement. The test input is applied as an input to the fuzzy membership function. The classifier works on the defined rules and provides us a rule based output. Fuzzy classifier helps to find error in less time on the basis of rule set. To determine the performance of the designed test case generation system the performance parameters such as accuracy, FAR (False acceptance rate) and FRR (False Rejection rate) are evaluated in MATLAB.

Keywords- Software engineering, software testing, test case prioritization, fuzzy logic, Accuracy**.**

I. INTRODUCTION

Software engineering involves building, developing, and maintaining software systems. Software engineering is a group of problem-solving techniques and technologies used in various fields to evolve and create useful systems to solve many problems such as actual problems [1]. Software engineering is divided into multiple categories and stages. It is an internal part of software development and is closely related to software quality. The main purpose of software testing is to meet the needs of users and make the system error-free. Therefore, software testing is mainly used to find the bugs to improve the quality of the system.

Testing is the procedure of observing product behaviour for detecting the difference among an existing product and the required conditions [2]. This is a procedure that identifies the integrity, correctness and validity of computer software. It validates if the expected result matches the actual result. This is the final stage of the product before it is delivered to the customer. Software testing is a significant part that discovers whether the product is effective, error-prone, works properly and as per customer's requirements [3].

II. TEST CASE PRIORITIZATION

The test case prioritization method systematize the test cases by sorting them in the most critical test case execution first, thereby improving the effectiveness of the test. Priority technology provides a way to find more errors in a resource-constrained environment, which can quickly improve the reliability of the system [4]. In addition, due to early detection of errors, software engineers have more time to fix errors and adjust project progress. Many prioritization methods have been presented to prioritize system test cases based on requirements [5]. Though, only the requirements considered cannot include key test cases. The complexity of the implementation and the complexity of the test case can also affect the priority of test cases.

Risk of some failure of test cases can be defined by two factors [6]:

Probability

It's the probability of the scenario occurred in product or how frequently or what percentage of customers may hit that scenario. The superior this likelihood is, the high the probability is.

Impact

It defined the consequence to the client/end user and/or company when the scenario is broken. The more the impact is, the high the severity is.

Probability + Impact = Risk

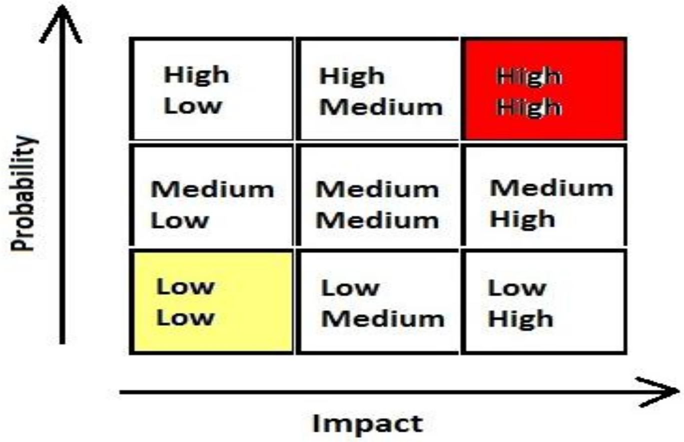


Figure 1: Test case prioritization by means of probability and impact

There are so many problems in testing software in time and cost constraint environment [7]. The most eminent challenge faced in software testing is the generation of an optimized test suite which helps in finding error in less time and path cover is more [8]. As only an appropriate test suite will results into an optimal solution. There are many algorithms and techniques available for software testing but selection of best technique according to requirement is needed [9]. To solve this problem, fuzzy logic techniques is used in this research to estimate the effectiveness of a given test case for violating an assertion based on the history of the test cases in previous testing operations. The main goal of the prioritization techniques is to increase the probability of detecting faults at an earlier stage of testing.

III. Proposed Architecture

In this research work, test case prioritization has been analyzed and accordingly, the designed principles have been formalized in algorithmic form on the basis of test case prioritization. For classification, fuzzy logic has been used to achieve improvement by means of FAR (False acceptance rate), FRR (False rejection rate) and accuracy. The methodology followed to implement the research is defined below:

Step-1: Design a model for test case prioritization using fuzzy logic to achieve high efficiency.

Step-2: Generation of Transaction number and element number has been done using fuzzy logic and later on, it is used to select software test cases based on rule set.

Step 3: Define a rule set for proposed model.

Step-4: Proposed test case prioritization technique has been implemented for being highly efficient.

Step-5: A code has been developed for the computation of performance metrics like FAR, FRR and accuracy.

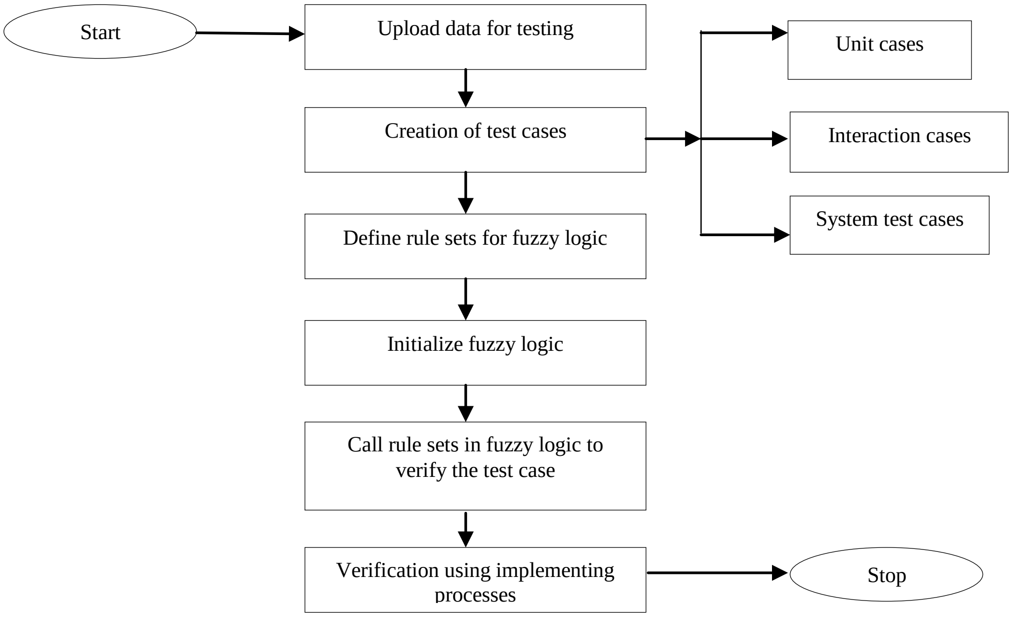
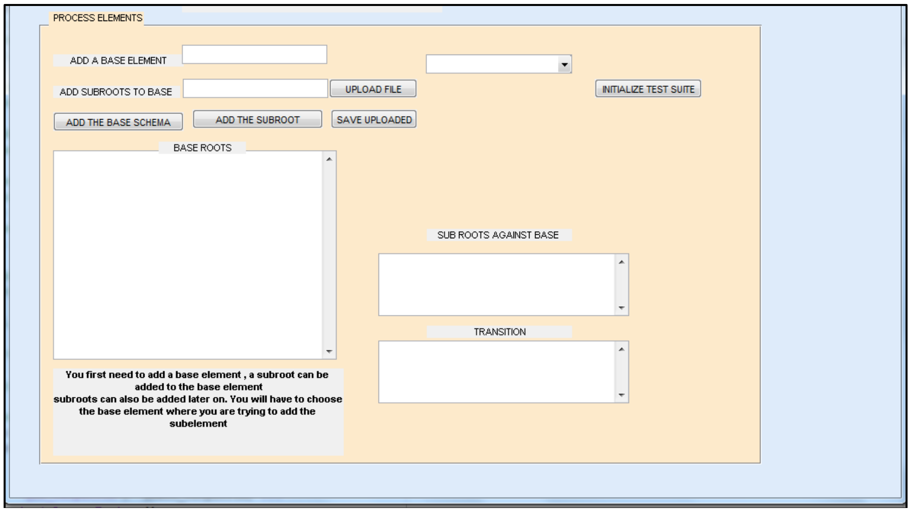

Start
Upload data for testing
Creation of test cases
Unit cases
Interaction cases
System test cases
Define rule sets for fuzzy logic
Initialize fuzzy logic
Call rule sets in fuzzy logic to verify the test case
Verification using implementing processes
Stop

Figure 2: Proposed Workflow

IV. Result and Analysis

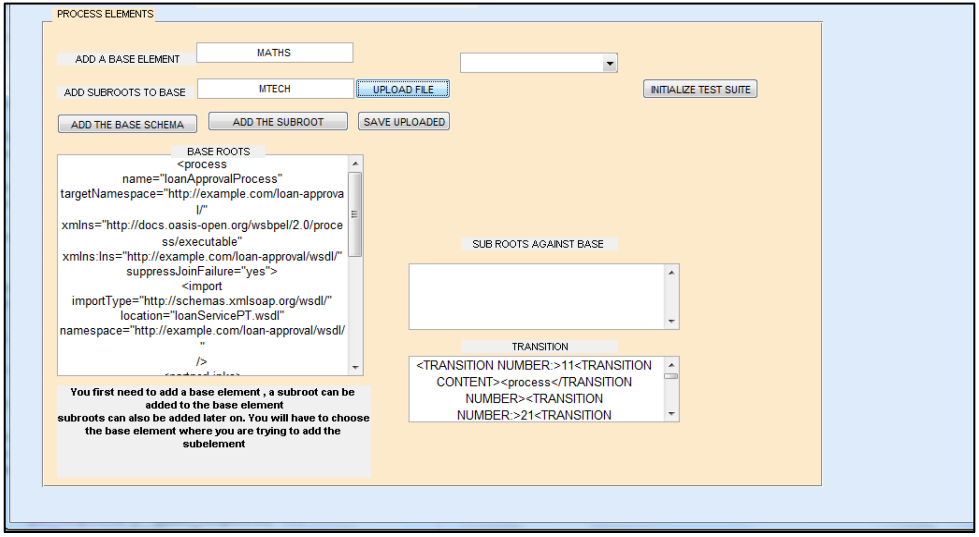
A test suite contains base and chid tag element. The designed work place is capable of adding both base and chid tags manually and it can even create base and chid tags from an uploaded file. The proposed architecture understands that both type of architecture can be utilized and hence, both the options are provided.



You first need to add a base element , a subroot can be added to the base element
subroots can also be added later on. You will have to choose the base element where you are trying to add the subelement

Figure 3: Main Working GUI

Figure 3 shows various process elements for processing of the loan like add a base element, add sub roots to base, add base schema, and add the sub root and then testing suite. These elements are added to get the sub elements. As we know that using combinations to generate test cases creates enormous amount of test cases. It can be argued that pre-requisite for GUI based user interface or windows form should have all object before further testing can be proceeded. Therefore, testing team writes test case which includes entire objects in the form but they fail to write test case which explores the association between controls used to design the form.



ADD A BASE ELEMENT: MATHS

ADD SUBROOTS TO BASE: MTECH

BASE ROOTS:

```
<process
name="loanApprovalProcess"
targetNamespace="http://example.com/loan-approva
l/"
xmlns="http://docs.oasis-open.org/wsbpel/2.0/proce
ss/executable"
xmlns:lns="http://example.com/loan-approval/wsdl/"
suppressJoinFailure="yes">
<import
importType="http://schemas.xmlsoap.org/wsdl/"
location="loanServicePT.wsdl"
namespace="http://example.com/loan-approval/wsdl/
"
/>
```

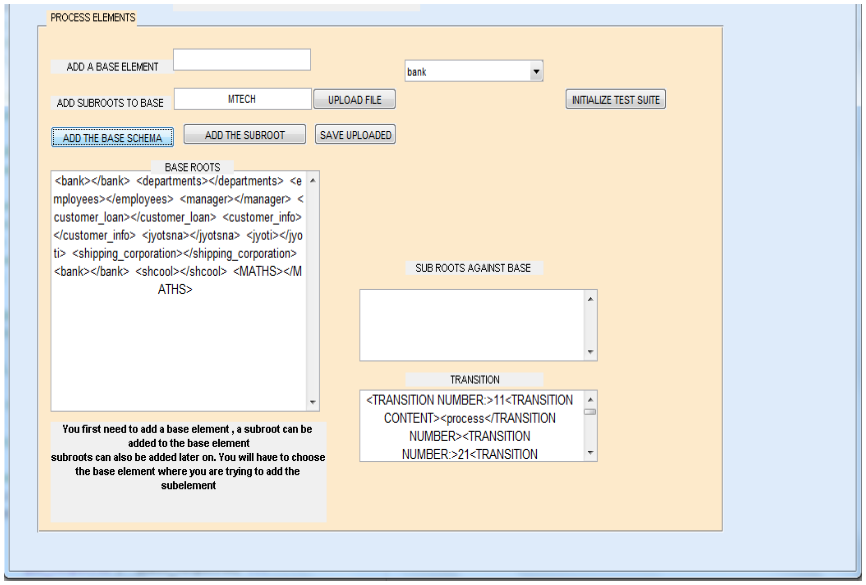
TRANSITION:

```
<TRANSITION NUMBER:>11<TRANSITION
CONTENT><process</TRANSITION
NUMBER><TRANSITION
NUMBER:>21<TRANSITION
```

You first need to add a base element , a subroot can be added to the base element
subroots can also be added later on. You will have to choose the base element where you are trying to add the subelement

Figure 4: Various Base Elements

Figure 4 shows various base roots of the banking management system like loan approval process, loan service etc. After having the base roots, different transition states have been provided for each service. Transition state indicates the execution from 1 state to another state.



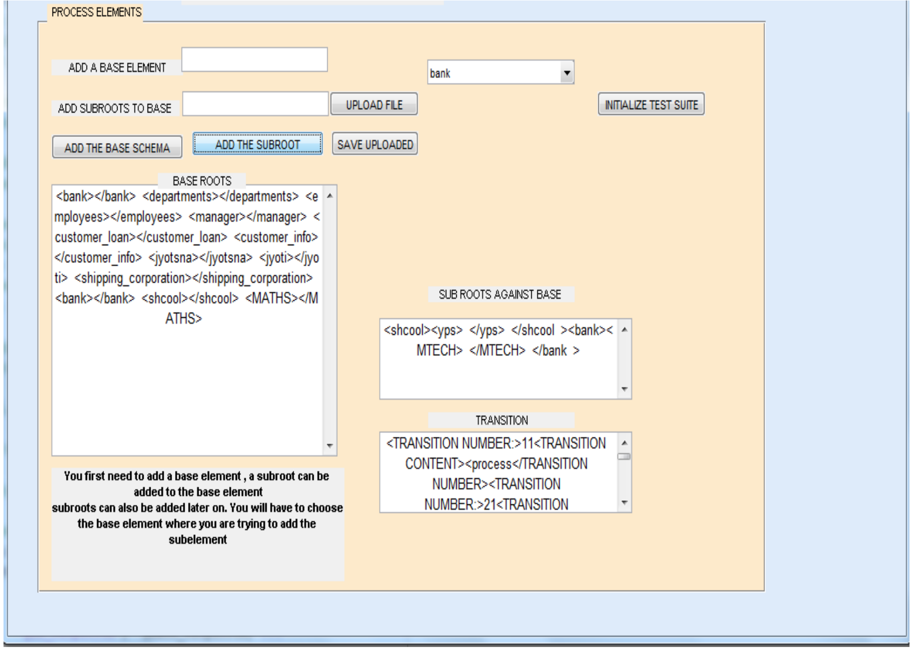
<bank></bank> <departments></departments> <employees></employees> <manager></manager> <customer_loan></customer_loan> <customer_info></customer_info> <jyotsna></jyotsna> <jyoti></jyoti> <shipping_corporation></shipping_corporation> <bank></bank> <shcool></shcool> <MATHS></MATHS>

<TRANSITION NUMBER:>11<TRANSITION CONTENT><process</TRANSITION NUMBER><TRANSITION NUMBER:>21<TRANSITION

You first need to add a base element , a subroot can be added to the base element
subroots can also be added later on. You will have to choose the base element where you are trying to add the subelement

Figure 5: Add Base Schema

Figure 5 shows various base roots of the banking management system like bank, departments, employees, customer _loan , customer_ info etc .It helps testers to know the actual stock and states of test data – without understanding the complex data base schemas or write SQL queries.



<bank></bank> <departments></departments> <employees></employees> <manager></manager> <customer_loan></customer_loan> <customer_info></customer_info> <jyotsna></jyotsna> <jyoti></jyoti> <shipping_corporation></shipping_corporation> <bank></bank> <shcool></shcool> <MATHS></MATHS>

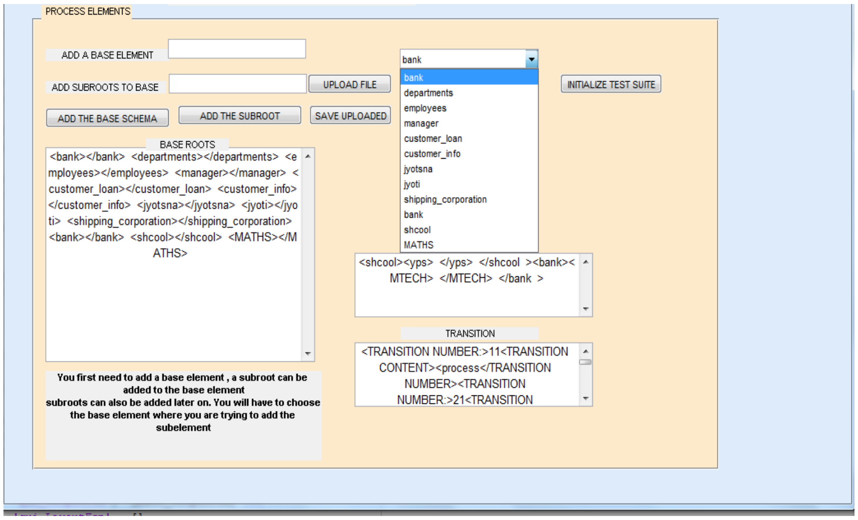
<shcool><yps> </yps> </shcool ><bank>< MTECH> </MTECH> </bank >

<TRANSITION NUMBER:>11<TRANSITION CONTENT><process</TRANSITION NUMBER><TRANSITION NUMBER:>21<TRANSITION

You first need to add a base element , a subroot can be added to the base element
subroots can also be added later on. You will have to choose the base element where you are trying to add the subelement

Figure 6: Addition of sub roots against base

Figure 6 shows the addition of sub roots against the base roots like school, bank, branch etc. In this work we have chosen the parent root whose main function is to find the sub root.



<bank></bank> <departments></departments> <employees></employees> <manager></manager> <customer_loan></customer_loan> <customer_info></customer_info> <jyotsna></jyotsna> <jyoti></jyoti> <shipping_corporation></shipping_corporation> <bank></bank> <shcool></shcool> <MATHS></MATHS>

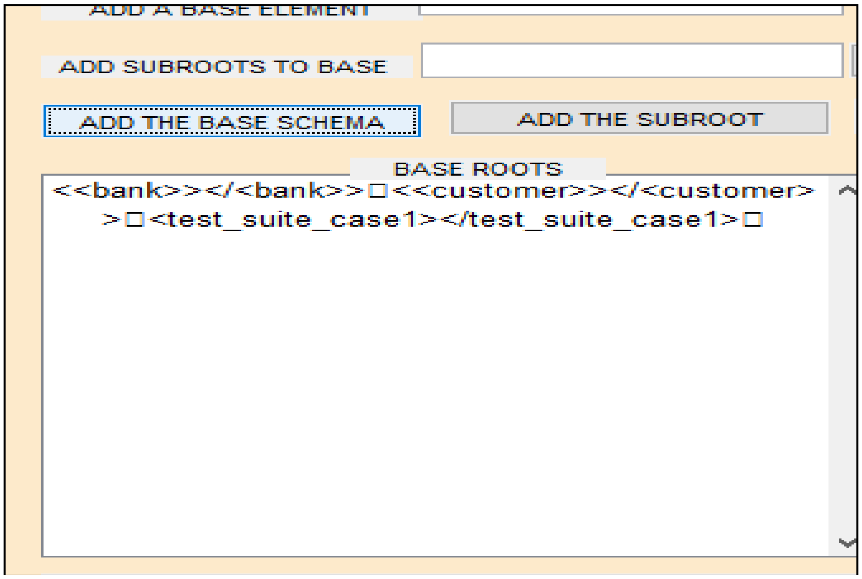
<shcool><yps> </yps> </shcool ><bank>< MTECH> </MTECH> </bank >

<TRANSITION NUMBER:>11<TRANSITION CONTENT><process</TRANSITION NUMBER><TRANSITION NUMBER:>21<TRANSITION

You first need to add a base element , a subroot can be added to the base element
subroots can also be added later on. You will have to choose the base element where you are trying to add the subelement

Figure 7: Different Functions of Banks

Figure 7 shows the various functions of the loan management process in bank like departments, employees, manager, customer_ban, Customer_info, Jyotsna, bank, shipping_corporation, bank, school etc .



<<bank>></<bank>>□<<customer>></<customer>>□<test_suite_case1></test_suite_case1>□

Figure 8: Transaction diagram

In the similar fashion, if there is already a transaction diagram available then the proposed Architecture can generate test suite from its own.

Table 1: Performance table

Parameter Name	Using Fuzzy logic
Accuracy	96.16
FAR	0.145
FRR	0.041

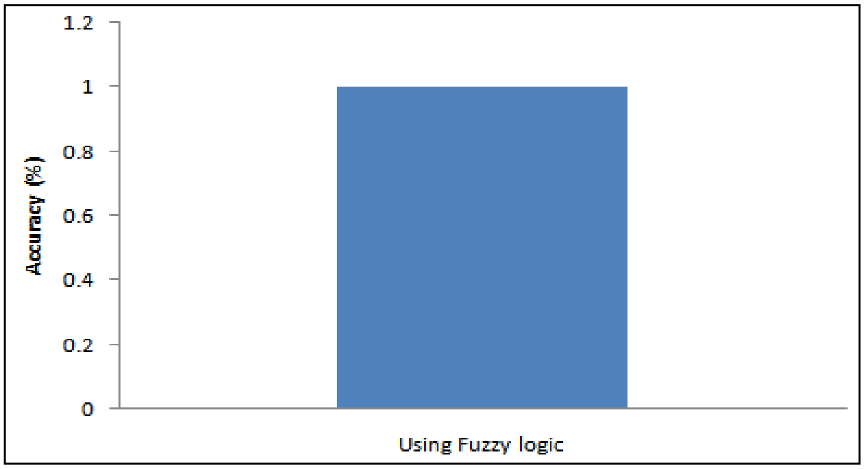


Figure 9: Accuracy of proposed work

The above figure displays the value of accuracy obtained for the proposed work. X-axis defines the Algorithm used in the proposed work whereas Y-axis defines the accuracy value of the proposed work. The accuracy obtained for the proposed work is 96.16%.

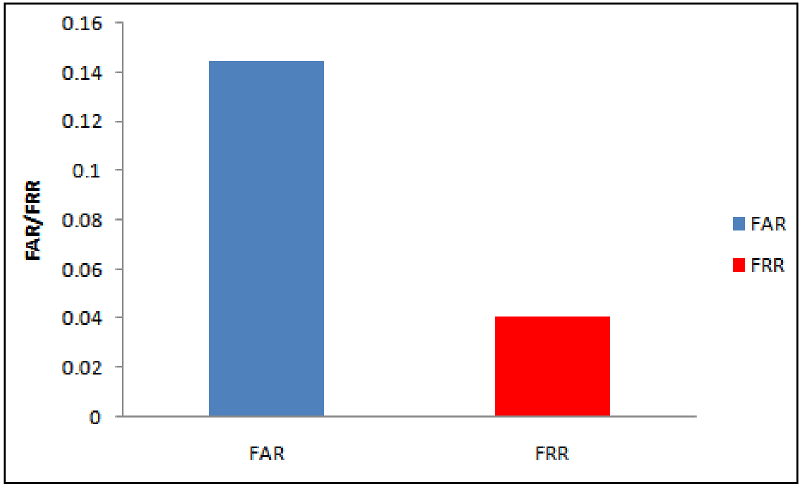


Figure 10: FAR/FRR of proposed work

The above figure defines the value of FAR/FRR of the proposed work when fuzzy logic algorithm is used as a classifier. The value of FAR and FRR of the simulated test generation system is 0.145 and 0.041 respectively.

Conclusion

In this research work, we have defined a function for each of the branch predicate. The program is written to record the functional evaluation for the corresponding inputs. A large record of such mappings from external inputs to the evaluation of branch functions can then be modelled using a fuzzy logic classifier.

The results of experiments have shown that the proposed technique works well in terms of computation parameters such as Accuracy, FAR and FRR. The values observed for accuracy, FAR and FRR are 96.16%, 0.145 and 0.041 respectively.

In the future, different meta-heuristic algorithmic rules like GA (Genetic algorithm), ACO (Ant colony optimization), and Search based optimization, search based optimization and PSO (Particle swarm optimization) with the NN (Neural network) can be used.

References

[1] Ghezzi, C., Jazayeri, M., & Mandrioli, D. (2002). Fundamentals of software engineering. Prentice Hall PTR.

[2] Conte, S. D., Dunsmore, H. E., & Shen, Y. E. (1986). Software engineering metrics and models. Benjamin-Cummings Publishing Co., Inc...

[3] Pfleeger, S. L., & Atlee, J. M. (1998). Software engineering: theory and practice. Pearson Education India.

[4] Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing. John Wiley & Sons.

[5] Briand, L., & Labiche, Y. (2004). Empirical studies of software testing techniques: Challenges, practical strategies, and future research. ACM SIGSOFT Software Engineering Notes, 29(5), 1-3.

[6] Young, M. (2008). Software testing and analysis: process, principles, and techniques. John Wiley & Sons.

[7] Srivastava, P. R. (2008). TEST CASE PRIORITIZATION. Journal of Theoretical & Applied Information Technology, 4(3).

[8] Elbaum, S., Rothermel, G., Kanduri, S., & Malishevsky, A. G. (2004). Selecting a cost-effective test case prioritization technique. Software Quality Journal, 12(3), 185-210.

[9] Chen, J., Zhu, L., Chen, T. Y., Towey, D., Kuo, F. C., Huang, R., & Guo, Y. (2018). Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering. Journal of Systems and Software, 135, 107-125.

[10] De S Campos Junior, H., Araújo, M. A. P., David, J. M. N., Braga, R., Campos, F., & Ströele, V. (2017, September). Test case prioritization: a systematic review and mapping of the literature. In Proceedings of the 31st Brazilian Symposium on Software Engineering (pp. 34-43). ACM.

[11] Dalal, S., & Hooda, S. (2017, September). A Novel Technique for Testing an Aspect Oriented Software System using Genetic and Fuzzy Clustering Algorithm. In Computer and Applications (ICCA), 2017 International Conference on (pp. 90-96). IEEE.

[12] Rhmann, W., & Saxena, V. (2017). Fuzzy Expert System Based Test Cases Prioritization from UML State Machine Diagram using Risk Information. IJ Mathematical Sciences and Computing, 2017(1), 17-27.

[13] Joseph, A. K., & Radhamani, G. (2017). Hybrid Test Case Optimization Approach Using Genetic Algorithm with Adaptive Neuro Fuzzy Inference System for Regression Testing. Journal of Testing and Evaluation, 45(6), 2283-2293.

Mrs. Taranum Thakur

Prof. Narinder Rana