Estimating the Regression Test Case Selection Probability using Fuzzy Rules

Deepak Rai, Kirti Tyagi
Department of Computer Science & Engineering
Ajay Kumar Garg Engineering College, Ghaziabad, UP, India
ramdeepakniwash@gmail.com, kirti.twins@gmail.com

***Abstract*- Software maintenance is performed regularly for enhancing and adapting the functionalities of the existing software, which modifies the software and breaks the previously verified functionalities. This sets a requirement for software regression testing, making it a necessary maintenance activity. As the evolution of software takes place the size of the test suite tends to grow, which makes it difficult to execute the entire test suite in a time constrained environment. There are many existing techniques for regression test case selection. Some are based on dataflow analysis technique, slicing-based technique, bio-inspired techniques, and genetic algorithm based techniques.**

This paper gives a regression test case selection technique based on fuzzy model, which reduces the size of the test suite by selecting test cases from existing test suite. The test cases, which are necessary for validating the recent changes in the software and have the ability to find the faults and cover maximum coding under testing in minimum time, are selected. A fuzzy model is designed which takes three parameters namely code covered, execution time and faults covered as input and produces the estimation for the test case selection probability as very low, low, medium, high and very high.

***Keywords*-** *Regression testing, Test case selection, Fuzzy logic, Selection probability.*

I. INTRODUCTION

System maintenance is the common term which is required for the proper functioning of the system. Maintenance of the software is mainly concerned with the related modifications to the system. These modifications may be due to changing user needs, error correction, improved performance, adaptation to changed environment, optimization etc. This adaptation of the software system to the changing customer needs makes a completely modified software system. Modified system breaks the previously verified functionalities of the system, which causes faults. This requires software regression testing for detecting such faults. Studies show that software maintenance activities on an average account for two third of the overall software cost. Software maintenance is frequently required to fix defects, enhance or adapt the existing functionalities of the software. One necessary maintenance activity is regression testing, which is the process of validating modified software in order to provide confidence that the software behaves correctly and the modification has not lead to degradation of software quality.

The dominant strategy for performing regression testing is to rerun the test cases that are available from the earlier version of the software. Regression testing is expensive, often accounts for almost one-half of the total cost of software maintenance [1]. Running all the test cases in a test suite requires a large amount of effort and time. A report shows that it took 1000 machine hours to execute approximately 30,000 functional test cases. Hundreds of man-hours are spent by test engineers to monitor the process of regression testing [2]. For this reason minimization of regression testing effort for reducing software maintenance costs has become an issue of considerable practical importance.

After development and release, software undergo regress maintenance phase [3].

A. Regression Testing (RT)

It is an integral part of the software development method. RT is defined as "the process of retesting the modified parts of the software and ensuring that no new regression errors have been introduced into previously unmodified part of the program". Regression test end up forming a safety net that makes refactoring easier and maintenance work less scary.

It is associated with system testing only when there is the change in the code.

There are various RT techniques shown in fig.1:

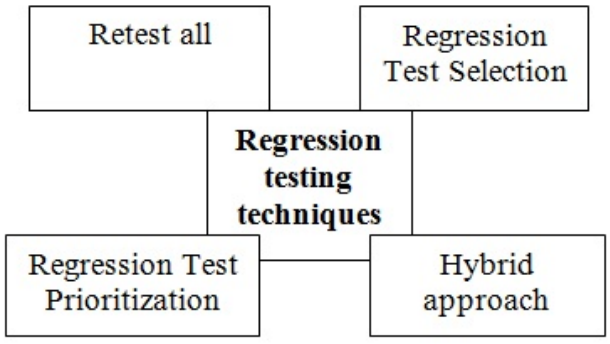


Fig.1. Techniques of regression testing.

1) *Retest all:* It is one of the conventional methods of regression techniques. This method reruns all the test cases

ISBN:978-1-4799-1024-3/13/$31.00 ©2013 IEEE

existing in the test suite. But, it consumes excessive time and resources as compared to other techniques.

2) *Regression Test Selection (RTS):*Due to expensive nature of "retest all" technique an alternative approach called Regression Test Selection or we can say selective retest is performed [4]. In this technique instead of rerunning the whole test suite, it selects a subset of valid test cases from an initial test suite that are necessary to test the modified program [5].

It attempts to reduce the time required to retest a modified program and also reduces the testing costs in environment where the program undergoes frequent modifications. Formally, RTS problem is defined as follows:

Let P be an application program and P' be a modified version of P. Let T be the initial test suite for testing P. An RTS technique aims to select a subset of test cases T' **subset of** T to be executed on P', such that every error detected when P' is executed with T is also detected when P' is executed with T' [6].

RTS consists of two major activities:

i) Identification of the affected part.

ii) Test Case Selection.

RTS divides the existing test suite into Obsolete, Retestable, and Reusable test cases [7].

- Obsolete test cases are not valid for the modified program.
- Retestable test cases execute the modified and the affected parts of the program and need to be rerun during regression testing.
- Reusable test cases execute only the unaffected parts of the program.

RTS techniques are broadly classified into three categories:

i. Coverage-based selection technique:

It locates program components that have been modified or affected by modifications, and select test cases that exercise those components.

ii. Minimization-based selection technique:

Similar to coverage techniques except that they select the smallest subset of test cases that can satisfy some minimum coverage criteria for the modified parts of the code [8][9][10][11].

iii. Safe Selection Technique:

Minimization techniques omit some fault-revealing test cases. To eliminate the possibility of missing faults, safe selection technique was introduced. It selects every test in T that can expose one or more faults in P'. It guarantees that the discarded test cases do not reveal faults [12] [13].

3) *Regression Test Prioritization (RTP):* It orders the test cases in such a way that the overall rate of fault detection increases. Test cases having higher fault detection capability are given higher priority and are taken up for execution earlier.

It is very much advantageous as the errors are detected and reported to the development team earlier.

4) *Hybrid Approach:* It is the combination of both RTS and RTP.

In this paper we consider only RTS.

B. Fuzzy Logic

Fuzzy logic is a convenient way to map an input space to output space through fuzzy inference process. It is basically a multivalued logic which permits intermediate values to be defined between conventional evaluations. Fuzzy logic gives the ability to quantify and reason with words having ambiguous meanings. That is why it is the best choice for managing contradicting, doubtful and ambiguous opinions.

Fuzzy logic is formed with the combinations of four concepts as shown in fig.2:

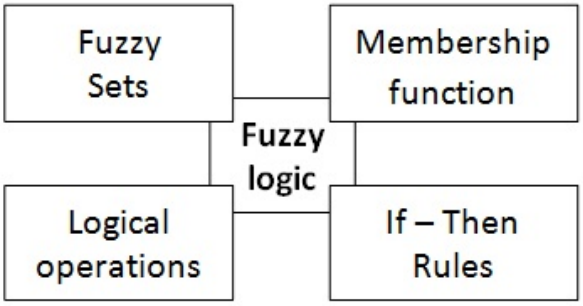


Fig.2. Fuzzy Concepts

Fuzzy sets are expressed as the set of ordered pairs [14] as shown in equation (1):

$$A = \{(x, \mu_A(x)) \mid x \varepsilon X, \mu_A(x): X \rightarrow [0, 1]\} \quad (1)$$

Where A is the fuzzy set, $(x, \mu_A(x))$is the membership function and rest is the universe of discourse. Example: Word like **Good.** There is no single value which can define the term Good, it differs from person to person. It has no clear boundary.

Simple way of forming membership functions is using straight lines. In this paper we have used triangular membership function, which is a simplest form using straight lines. It is the collection of three points forming a triangle. Logical operations are used to combine more than one inputs and conditions together for inferencing. There are three main logical operators namely AND, OR and NOT. For all possible combinations of the inputs If-Then rules are framed.

The overall fuzzy process is shown in fig.3.

This paper proceeds by describing the related work and previous works on regression testing and fuzzy logic in the next section. Section 3 discusses factors for test case selection probability estimation. Section 4 presents the proposed model.

Section 5 presents the implementation of the model, and Section 6 discusses overall conclusion and future work.

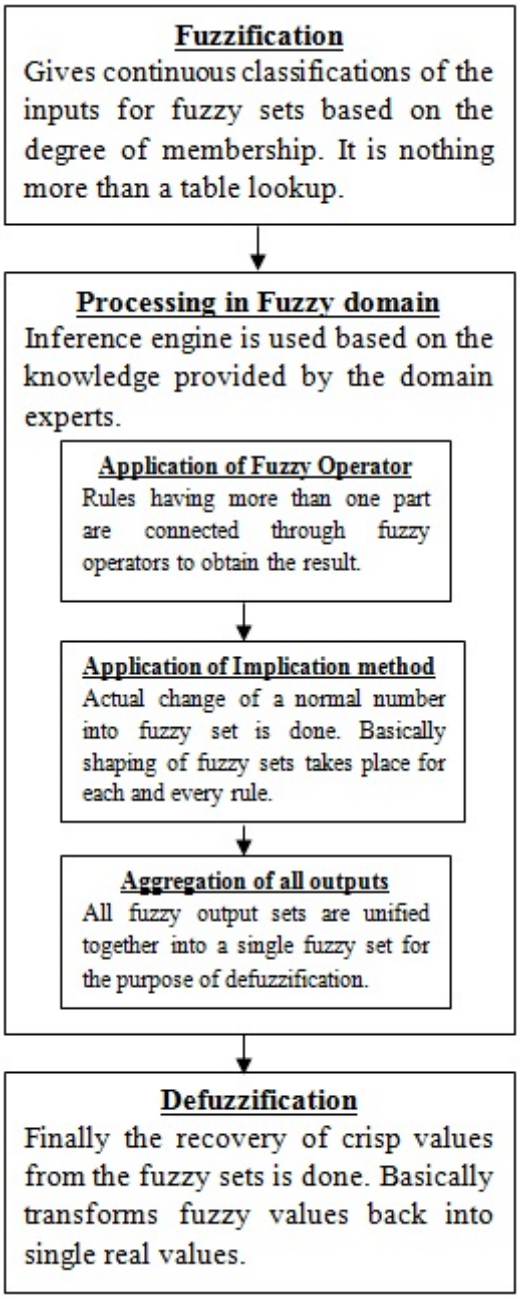


Fig.3. Stages of Fuzzy logic processing

II. RELATED WORK

Software testing tells whether a program is correct, by showing that it produces correct output over some finite subset of input. When we develop software we use development testing, when we modify software we retest it, which is called as RT. It serves many purposes with the primary one to increase confidence in the correctness and locate errors in the modified program.

Development testing and RT are different from each other in many aspects:

- Development test requires creation of test suites, whereas regression test uses existing test suites.
- Development test requires testing of all software components whereas regression test only test modified part and the part which is affected by the modification.
- Development test gets time for testing whereas regression test is performed in crisis situation, under time constraints.
- Development test is costly, performed only once whereas RT is performed many times.

Research on RT spans a wide variety of topics. The issue that has seen the greatest amount of research, however, is the selective retest problem. The process of finding minimal subset of test cases that can cover each element of the system started in the year 1977 by K. Fischer in his paper "A Test Case Selection Method for the Validation of Software maintenance modifications" [15]. Later this technique was extended in the year 1981 by Fischer, Raji and Chruscicki. They had given a methodology for retesting modified software [16]. Yau and Kishimoto had given "A method for revalidating modified programs in the maintenance phase" in the year 1987 [17]. In this a selection method was presented that depends on input partitions, and uses symbolic execution to determine tests that traverse modified blocks. Their method relies on knowledge of modifications, and is computationally expensive due to the use of symbolic execution. Ostrand and Weyuker had done analysis for regression techniques using dataflow-based regression testing methods in September 1988 [18]. Lewis, Beck and Hartmann had proposed a tool to support regression testing in September 1989 [19]. The greatest drawback of these methods is that they require prior knowledge of modifications. Leung and White provided insights into regression testing in October 1989 [7] and given a model to compare regression testing strategies in 1991. Binkley used semantic differencing to reduce the cost of regression testing in November 1992 [20]. Chen, Rothermel and Vo provided a system for selective regression testing in 1994 [21]. Rothermel and Harrold had analyzed the regression testing selection techniques in August 1996 [22] and given a safe, efficient regression test selection technique in 1997 [23]. In 2001 Rothermel, Untch, Chu and Harrold had provided a technique for prioritizing test cases for regression testing. Yoo and Harman had given method for multi-objective test case selection in2007 and in 2012 they had discussed open problems and given potential directions of future research in their survey of regression testing minimization, selection and prioritization techniques [24]. Engstrom, Runeson and Skoglund systematically reviewed the regression test selection techniques in January 2010. Amir Ngah had proposed a model for RTS using the decomposition slicing technique in his PhD thesis on RTS by exclusion in May 2012 [25]. In July, 2012 Siavash Mirarab et.al had given a multicriterion based

optimization for size-constrained RTS [26]. In the same month Jianchun Xing et.al had given a safe RTS based on program dependence graphs of a program and its modified version [27]. Previous research in the field of RTS has not focused on industrial contexts. Alex Augustsson in 2012 had introduced a framework for evaluating RTS techniques in industry [28].

Research on Fuzzy was first proposed by L. Zadeh in his paper "Fuzzy Sets" in the year 1965 [29]. More information about fuzzy logic was given by Klir and Folger in 1988. They had given uncertainty and information on fuzzy sets [30]. W. Pedycz had given fuzzy control and fuzzy systems in 1993 [31]. In the very next year Driankon, Hellendour and Reinfark had added to fuzzy control [32]. Finally in the 1999, Novak, Perilieva and Mockor had given the mathematical principles of fuzzy logic [33].

Researches using fuzzy logic for the purpose of regression test case selection and prioritization are very scant. Xu, Gao and Khoshgoftaar had firstly shown the application of fuzzy expert system in regression test selection in 2005 [34]. Later in 2011 Praveen, Sirish and Raghurama of BITS pilani had given fuzzy criteria for assessing the software testing effort [35]. In 2012 Ali M. Alakeel had proposed a fuzzy test cases prioritization technique for regression testing with assertions [36] and also fuzzy logic was used for prioritizing test cases for GUI based software [37]. Recently, in 2013 H.B. Gupta et.al used fuzzy logic for regression technique [38].

In our paper we use fuzzy rule base for the test case selection probability estimation [39].

III. FACTORS FOR TEST CASE SELECTION PROBABILITY ESTIMATION

We have taken three main factors to calculate the selection probability of a test case:

A. Code covered

B. Execution time

C. Faults covered

There may be many more factors which may be taken up for this calculation but these are the three main factors which have the most effect.

A. Code covered (CC)

This indicates the portion of the code covered by a particular test case. This may be number of lines covered by the test case, number of statements covered by the test case, number of functions covered and number of program branches covered. Test cases with highest level of code coverage are run first. We selected code covered as a factor for estimating the test case selection probability because it is believed that the test cases which cover more code have higher rate of fault detection [40]. Hence,

CC $\propto$ test case selection probability

B. Execution time (ET)

This indicates the time required for a particular test case to complete its execution [41]. It may or may not include the loading time. Test cases having minimum execution time are given weightage and are executed first. We selected execution time as a factor for estimating the test case selection probability because it helps in selecting and reordering the execution of test cases ensuring that defects are revealed earlier in the test execution phase. Hence,

ET $\propto$ 1/ test case selection probability

C. Faults covered (FC)

Similar to code covered it indicates the number of faults covered by a particular test case. Basically, it tells about the number of uncovered faults. Test cases which are capable of detecting or we may say covering more number of faults are taken up for execution first. We selected faults covered as a factor for estimating the test case selection probability because it is the most important parameter to be considered during testing [42]. More the number of faults is detected by a particular test case more effective will be that test case. Hence,

FC $\propto$ test case selection probability

IV. PROPOSED MODEL

This paper gives a Regression Test Case Selection Technique based on fuzzy model, which uses the factors listed under section 3 for estimating the test case selection probability. Individual value of any factor maynot provide the appropriate value for selection probability. So we use fuzzy rule based approach which considers all the factors and their relative values simultaneously for estimating the selection probability.The basic fuzzy inferencing is expressed as shown in fig.4.

The model used in this paper contains three inputs namely code_covered, execution_time, faults_covered and one output i.e. selection_probability. For all the inputs and output, membership functions are chosen and the values for each membership functions corresponding to each inputs and output is defined [43].

A. Membership functions and values for Inputs

Here, Code_covered = CC ; Execution_time = ET ;

Faults_covered = FC

$$\mu_A(CC)=\begin{cases} \text{LOW} ; & 0 \le CC \le 0.38 \\ \text{MEDIUM} ; & 0.32 \le CC \le 0.65 \\ \text{HIGH} ; & 0.62 \le CC \le 1 \end{cases}$$

$$\mu_A(ET)=\begin{cases} \text{LOW} ; & 0 \le ET \le 0.28 \\ \text{MEDIUM} ; & 0.24 \le ET \le 0.59 \\ \text{HIGH} ; & 0.52 \le ET \le 1 \end{cases}$$

$$\begin{cases} \text{LOW} ; & 0 \le FC \le 0.34 \end{cases}$$

μ_A(FC)= MEDIUM ; $0.31 \le FC \le 0.59$

HIGH ; $0.50 \le FC \le 1$

B. Membership functions and values for output:

Here,Selection_probability = SP

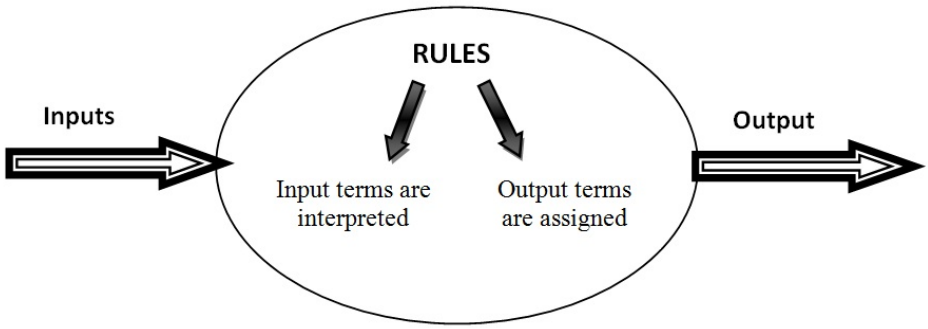


Fig.4. Fuzzy Inferencing process.

μ_A(SP)=
- VERY LOW ; $0 \le SP \le 0.14$
- LOW ; $0.12 \le SP \le 0.33$
- MEDIUM ; $0.29 \le SP \le 0.56$
- HIGH ; $0.51 \le SP \le 0.71$
- VERY HIGH ; $0.65 \le SP \le 1$

C. Rule Base

It is basically a storage space associated with the model, which stores the knowledge related to the subject in the form of "If-Then" rules. Rules are formed with the composition of inputs and output, and each rule individually represents a condition-action statement in human understandable format. In this paper we consider all possible combinations of inputs getting a total of $3^3 = 27$ sets. Based on these 27 sets of combinations a total of 27 rules are formed to constitute a complete rule base for the model.Some of the rules are as shown in fig.5.

The membership values are defined based on the data collected from the classroom projects. The rules are formed on the basis of the collected data and expert advice.

In this model we use 'Mamdani' style for inferencing, one of the two available fuzzy inferencing systems. For combining together all the obtained results we use MAX method.

Why mamdani?

Mamdani style of inferencing is chosen because it has some positive interesting characteristics:

- Works comfortably with the real values of inputs and output,
- Gives framework which allows the inclusion of expert knowledge in the form of rules, and also permits to combine rules at later stage in a very simple way,
- Provides a greater degree of freedom for the choice of inferencing system and the interface components for Fuzzification and defuzzification.

D. Working of the model

This model works as:

Step 1) Inputs corresponding to each factor is taken in crisp format, and are converted into fuzzy form.

Step2) Based on the membership functions value corresponding to each input factors, appropriate rule is fired.

Step 3) All inputs are taken together simultaneously, for this we use AND operator in order to combine the inputs together.

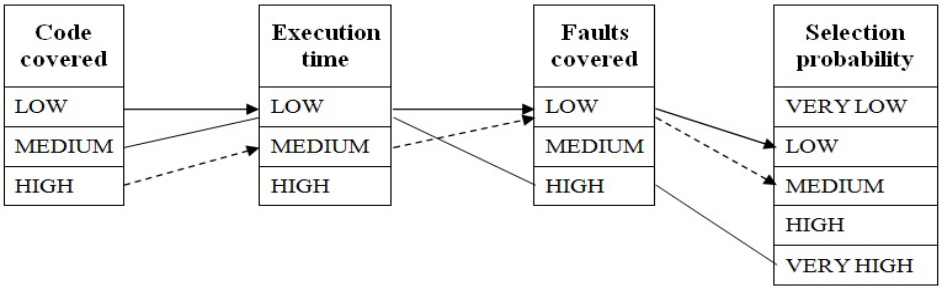
Step 4) MIN method is used for evaluating AND operator.

Step 5) All the results obtained is aggregated using MAX method.

Step 6) Finally, the aggregated result is defuzzified using centroid method.

Step 7) Step 1 to step 6 is repeated for different inputs.

Fig.5. Rule formation



model one by one. The appropriate rule is fired based on the input values and the output for the selection probability is produced for each pair of input values. Considering an example, let the inputs be:

Code_covered = 0.63 ; Execution_time = 0.25 ; Faults_covered = 0.54

These are the crisp inputs. So, firstly we convert these crisp values into fuzzy values.

A. Fuzzification

1) Code Covered:

It is seen from fig.6 that the value 0.63 belongs to both MEDIUM and HIGH set. So we need to get the fuzzy values corresponding to both MEDIUM and HIGH set.

Let 'x' represents the crisp values and 'y' represents the fuzzy values. Here, x = 0.63 and y = ???

For MEDIUM, end-points of the corresponding line is: [(0.45,1) and (0.65,0)]

So, equation (2) gives the equation of the line as:

$$(y - y_1) = \{(y_2 - y_1) / (x_2 - x_1)\} * (x - x_1) \quad (2)$$

Here, (x_1, x_2) and (y_1, y_2) are the end-points of the line.

so, y = [{(0-1) / (0.65-0.45)} * (0.63-0.45)] + 1

or, y = [(-5) * (0.18)] + 1

or, **y = 0.1**

For HIGH, end-points of the corresponding line is: [(0.62,0) and (0.75,1)]

So, equation (3) gives the equation of the line as:

$$(y - y_1) = \{(y_2 - y_1) / (x_2 - x_1)\} * (x - x_1) \quad (3)$$

Here, (x_1, x_2) and (y_1, y_2) are the end-points of the line.

so, y = [{(1-0) / (0.75 – 0.62)} * (0.63-0.62)] + 0

or, y = [(7.6923) * (0.01)] + 0

or, **y = 0.0769**

Therefore, the fuzzy value corresponding to 0.63 is 0.1 in MEDIUM set and 0.0769 in HIGH set.

Similarly, the fuzzy value for the other two inputs is found.

*2) Execution Time:*It is 0.0588 in MEDIUM set and 0.3333 in LOW set.

*3) Faults Covered:*It is 0.4545 in MEDIUM set and 0.1995 in HIGH set.

B. Rule Selection

Based on these values the rules fired are:

Here, VL = Very Low, L = Low, M = Medium, H = High and VH = Very High.

code_covered == M) & (execution_time == L) & (faults_covered == M) => (selection_probability = H).

code_covered == M) & (execution_time == L) & (faults_covered == H) => (selection_probability = VH).

code_covered == M) & (execution_time == M) & (faults_covered == M) => (selection_probability = M).

code_covered == M) & (execution_time == M) & (faults_covered == H) => (selection_probability = H).

code_covered == H) & (execution_time == L) & (faults_covered == M) => (selection_probability = H).

code_covered == H) & (execution_time == L) & (faults_covered == H) => (selection_probability = VH).

code_covered == H) & (execution_time == M) & (faults_covered == M) => (selection_probability = H).

code_covered == H) & (execution_time == M) & (faults_covered == H) => (selection_probability = VH).

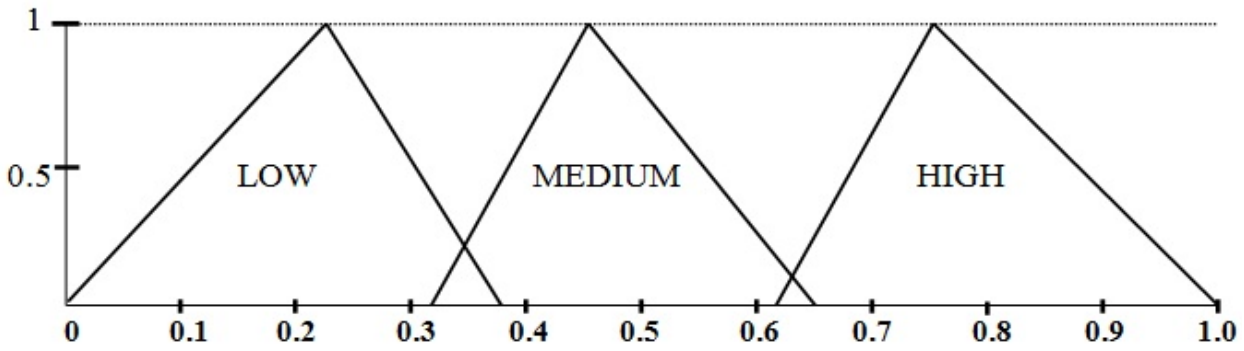


Fig.6. Membership functions for code covered

C. Rule Evaluation

As shown in fig.7, the selection probability medium, high and very high category. Therefore,

1) $\mu_{selection\_probability=M}$= max [min { $\mu_{code\_covered=M}(0.63)$, $\mu_{execution\_time=M}(0.25)$, $\mu_{faults\_covered=M}(0.54)$ }]

= max[min{0.1, 0.0588, 0.4545}]

= **0.0588**

2) $\mu_{selection\_probability=H}$=

max[min{$\mu_{code\_covered=M}(0.63)$,$\mu_{execution\_time=L}(0.25)$, $\mu_{faults\_covered=M}(0.54)$},

min{$\mu_{code\_covered=M}(0.63)$,$\mu_{execution\_time=M}(0.25)$, $\mu_{faults\_covered=H}(0.54)$},

min{$\mu_{code\_covered=H}(0.63)$,$\mu_{execution\_time=L}(0.25)$, $\mu_{faults\_covered=M}(0.54)$},

min{$\mu_{code\_covered=H}(0.63)$,$\mu_{execution\_time=M}(0.25)$, $\mu_{faults\_covered=M}(0.54)$}]

= max[min{0.1, 0.3333, 0.4545},
min{0.1, 0.0588, 0.1905},
min{0.0769, 0.3333, 0.4545},
min{0.0769, 0.0588, 0.4545}]

= max[0.1, 0.0588, 0.0769, 0.0588]

= **0.1**

3) $\mu_{selection\_probability=VH}$=

max[min{$\mu_{code\_covered=M}(0.63)$,$\mu_{execution\_time=L}(0.25)$, $\mu_{faults\_covered=H}(0.54)$},
min{$\mu_{code\_covered=H}(0.63)$,$\mu_{execution\_time=L}(0.25)$,$\mu_{faults\_covered=H}(0.54)$},
min{$\mu_{code\_covered=H}(0.63)$,$\mu_{execution\_time=M}(0.25)$, $\mu_{faults\_covered=H}(0.54)$}]

= max[min{0.1, 0.3333, 0.1905},
min{0.0769, 0.3333, 0.1905},
min{0.0769, 0.0588, 0.1905}]

= max[0.1, 0.0769, 0.0588] = **0.1**

D. Defuzzification

The above obtained fuzzy output is finally put to defuzzification in order to get the crisp value against the output variable selection_probability. Out of several methods available for defuzzification we chose the centroid method [44]. In this the Centre of Gravity (COG) is calculated for the area under the curve using equation (4).

$$\mathbf{COG} = \frac{\sum_{x=a}^{b} \mu_A(x) * x}{\sum_{x=a}^{b} \mu_A(x)} \mathbf{(4)}$$

Here,

$$\mathbf{COG} = \frac{\{(0.0588)*(0.29+0.4+0.5)\} + \{(0.1)*(0.51+0.6+0.7)\} + \{(0.1)*(0.71+0.8+0.9+1)\}}{\{(0.0588*3) + (0.1*3) + (0.1*4)\}}$$

$$= \frac{(0.069972) + (0.181) + (0.341)}{(0.1764) + (0.3) + (0.4)}$$

$$= \frac{0.591972}{0.8764}$$

$$= \mathbf{0.6755}$$

Similarly, the selection probability is obtained against different combinations of input. Some are shown in table I.

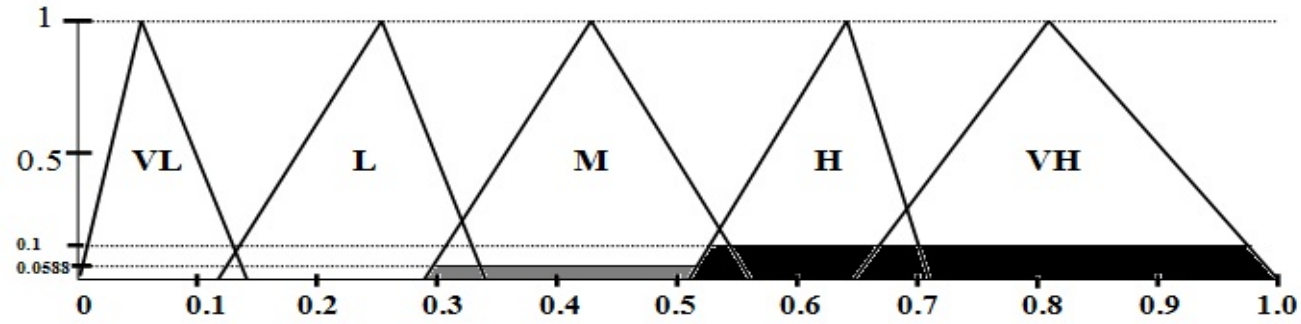


Fig.7. Aggregation of outputs.

TABLE I: Implementation Result

Input			Output	
Code_covered	**Execution_time**	**Faults_covered**	**Obtained Selection_probability**	**Modeled Selection_probability**
0.63	**0.25**	**0.54**	**0.6755**	**0.6787**
0.4	**0.15**	**0.4**	**0.63**	**0.6146**
0.56	**0.38**	**0.18**	**0.2375**	**0.2281**
0.23	**0.56**	**0.51**	**0.3583**	**0.354**
0.34	**0.72**	**0.62**	**0.41**	**0.4244**
0.78	**0.92**	**0.8**	**0.63**	**0.6126**

V. EVALUATION

The proposed rules were also processed with the designed fuzzy model in MATLAB; the selection probability against each input values were found (shown as modeled selection_probability in table 1). For the purpose of evaluation we used the Root Mean Square Error (RMSE) method, which is used to measure the difference between the actual obtained value from the calculation that is being modeled and the value predicted by the model. RMSE is defined as the square root of the mean squared error as shown in equation (5).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obtained,i} - X_{model,i})^2}{n}} \quad (5)$$

Where,

$X_{obtained}$ is the obtained value from calculation and X_{model} is the modeled value for different inputs i = 1 to n.

Here,

$$\mathbf{RMSE} = \sqrt{\frac{(.6755-.6787)^2 + (.63-.6146)^2 + (.2375-.2281)^2 + (.3583-.3540)^2 + (.41-.4244)^2 + (.63-.6126)^2}{6}}$$

$$= \sqrt{\frac{0.00086437}{6}}$$

$$= \mathbf{0.012003}$$

Hence, the percentage error is **1.2%.**

VI. CONCLUSION

This paper proposed a fuzzy model for estimation of the selection probability for regression test case. The model estimates the probability based on three important factors namely code covered, execution time and faults covered. The fuzzy approach is used to combine these inputs and reach at the estimation of the probability for selecting a test case. Fuzzy logic is a powerful tool which gives the ability to quantify with the contradicting, doubtful and ambiguous opinions. The selection of factors has been made based on some expert advice. However the results obtained are very close to the actual results. The only limitation is that, in this paper we have considered only three important test case selection factors. However there may be few more factors, which may be added. There may be the number of extensions of the model by using techniques like artificial neural network and neuro fuzzy approach. This is left as future work.

REFERENCES

[1] G. Kapfhammer. *The Computer Science Handbook*, chapter on Software testing. CRC Press, Boca Raton, FL, 2nd edition, 2004.

[2] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel. The effects of time constraints on test case prioritization: A series of controlled experiments. *IEEE Transactions on Software Engineering*, 36(5):593–617, September 2010.

[3] Yogesh Singh, Pradeep Kumar Bhatia, Omprakash Sangwan,"Predicting software maintenance using fuzzy model", published in SIGSOFT Software Engineering Notes, vol 34, July 2009.

[4] T. Graves, M. J. Harrold, J. M. Kim, A. Porter, G. Rothermel, An empirical study of regression test selection techniques, in: Proceedings of the 20th International Conference on Software Engineering, IEEE Computer Society Press, Kyoto, Japan, 1998, pp.188-197.

[5] G. Rothermel, M. J. Harrold, Empirical studies of a safe regression test selection technique, IEEE Transactions on Software Engineering 24(6) (1998) 401-419.

[6] G. Rothermel and M. Harrold. Selecting tests and identifying test coverage requirements for modified software. In *Proceedings of the InternationalSymposium on Software Testing and Analysis*, pages 169–184, August 1994.

[7] H. Leung and L. White. Insights into regression testing. In *Proceedings of the Conference on Software Maintenance*, pages 60–69, 1989.

[8] K. Fischer, "A Test Case Selection Method for the Validation of Software Maintenance Modifications," Proc. Int'l Computer Software and Applications Conf., pp. 421-426, 1977.

[9] K. Fischer, F. Raji, and A. Chruscicki, "A Methodology for Retesting Modified Software," Proc. Nat'l Telecomm. Conf., pp. B6.3.1-B6.3.5, 1981.

[10] J. Hartmann and D.J. Robson, "Techniques for Selective Revalidation," IEEE Software, vol. 7, no. 1, pp. 31-36, Jan. 1990.

[11] J. Black, E. Melachrinoudis, and D. Kaeli, "Bi-Criteria Models for All-Uses Test Suite Reduction," Proc. ACM Int'l Conf. Software Eng., pp. 106-115, 2004.

[12] G. Rothermel and M.J. Harrold, "A Safe, Efficient Regression Test Selection Technique," ACM Trans. Software Eng. and Methodology, vol. 6, no. 2, pp. 173-210, 1997.

[15] Klir, G., and T. Folger, *Fuzzy Sets, Uncertainty and Information*, Prentice Hall, 1988.

[16] Pedrycz, W., *Fuzzy Control and Fuzzy Systems*, John Wiley & Sons, 1993.

[17] Driankov, D., H. Hellendoor, and M. Reinfark, *An Introduction to Fuzzy Control*, Springer, New York, 1994.

[18] Novák, V., I. Perfilieva, and J. Močkoř, *Mathematical Principles of Fuzzy Logic*, Kluwer Academic, Dodrecht, 1999.

[19] Ganesh, M., *Introduction to Fuzzy Sets and Fuzzy Logic*, Prentice Hall, 2006.

[20] H. Agrawal, J. Horgan, E. Krauser, and S. London. Incremental regression testing. In *IEEE International Conference on Software Maintenance*, pages 348–357, 1993.

[21] A. Aho, R. Sethi, and J. Ullman. *Compilers: Principles, Techniques and Tools*. Dorling Kindersley (India) Pvt Ltd, 2nd edition, 2008.

[22] S.S Yau and Z. Kishimoto, "A method for revalidating modified programs in the maintenance phase," COMPSAC'87: the Eleventh Annual International Computer Software and Applications Conference, pp.272-7, October, 1987.

[23] T.J. Ostrand and E.J. Weyuker, "Using dataflow analysis for regression testing," Sixth Annual Pacific Northwest Software Quality Conference, pp. 233-47, September, 1988.

[24] Yoo, Shin, and Mark Harman. "Regression testing minimization, selection and prioritization: a survey." *Software Testing, Verification and Reliability* 22.2 (2012): 67-120.

[25] Ngah, Amir. "Regression test selection by exclusion." PhD diss., Durham University, 2012.

[26] Mirarab, Siavash, Soroush Akhlaghi, and Ladan Tahvildari. "Size-Constrained Regression Test Case Selection Using Multicriteria Optimization." *Software Engineering, IEEE Transactions on* 38.4 (2012): 936-956.

[27] Xing, Jianchun, et al. "Safe Regression Test Selection Based on Program Dependence Graphs." *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*. IEEE, 2012.

[28] Augustsson, Alex. "A Framework for Evaluating Regression Test Selection Techniques in Industry." (2012).

[29] R. Lewis, D.W. Beck, and J. Hartmann. Assay - a tool to support regression testing. In ESEC'89. 2nd European Software Engineering Conference Proceedings, pages 487-496, September 1989.

[30] D. Binkley. Using semantic differencing to reduce the cost of regression testing. In Proceedings of the Conference on Software Maintenance – 1992, pages 41-50, November 1992.

[31] Y.-F. Chen, D.S. Rosenblum, and K.-P. Vo, "Testtube: A System for Selective Regression Testing," Proc. ACM Int'l Conf. Software Eng., pp. 211-220, 1994.

[32] G. Rothermel and M. Harrold. Analyzing regression test selection techniques. *IEEE Transactions on Software Engineering*, 22(8):529–551, August 1996.

[33] G. Baradhi and N. Mansour. A comparative study of five regression testing algorithms. In *Proceedings of Australian Software Engineering Conference, Sydney*, pages 174–182, 1997.

[34] Z. Xu, K. Gao, Taghi M. Khoshgoftaar, "Application of Fuzzy Expert System In Test Case Selection For System Regression Test", Information Reuse and Integration, Conf, 2005. IRI -2005 IEEE International Conference on, Pages: 120-125.

[35] Srivastava, Praveen Ranjan, Sirish Kumar, A. P. Singh, and G. Raghurama. "Software testing effort: an assessment through fuzzy criteria approach."*Journal of Uncertain Systems* 5, no. 3 (2011): 183-201.

[36] Alakeel, Ali. "A Fuzzy Test Cases Prioritization Technique for Regression Testing Programs with Assertions." In *ADVCOMP 2012, The Sixth International Conference on Advanced Engineering Computing and Applications in Sciences*, pp. 78-82. 2012.

[13] G. Rothermel, Efficient effective regression testing using safe test selection techniques, PhD thesis, Clemson University, 1996.

[14] Zadeh, L.A., Fuzzy sets, *Information and Control*, vol.8, pp.338–353, 1965.

[37] Chaudhary, Neha, Om Prakash Sangwan, and Yogesh Singh. "Test Case PrioritizationUsing Fuzzy Logic for GUI based Software." *International Journal* 3 (2012).

[38] Bhasin.H., Gupta. S., Kathuria. M., "Regression Testing Using Fuzzy Logic", International Journal of Computer Science and Information Technologies (IJCSIT), vol. 4 (2), 2013, pp. 378-380.

[39] Seth, Kirti, Arun Sharma, and Ashish Seth. "Component Selection Efforts Estimation–a Fuzzy Logic Based Approach." *International Journal of Computer Science and Security,(IJCSS)* 3, no. 3 (2009): 210-215.

[40] K. K. Aggrawal, Yogesh Singh, A. Kaur, " Code coverage based technique for prioritizing test cases for regression testing," ACM SIGSOFT Software Engineering Notes, vol 29 Issue 5 September 2004.

[41] K.R.Walcott,M.L.Soa,G.M.Kapfhammer,and R.S.Roos","Time aware test suite prioritization",In Proceedings of ISSTA,pages 1-11,2006.

[42] G. Rothermel, M.J. Harrold, J. Ostrin, and C.Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In Proceedings of the international conference on Software Maintenance, 34-43, Nov.1998.

[43] Tyagi, Kirti, and Arun Sharma. "A rule-based approach for estimating the reliability of component-based systems." *Advances in Engineering Software* 54 (2012): 24-29.

[44] Hellendorn, H., and C. Thomas, Defuzzification in fuzzy controllers, *Intelligent and Fuzzy Systems*, vol.1, pp.109–123, 1993.