Evaluation of Cost and Effort Indicators for Combined Selection of MBT Techniques

Sandhya Kummarikunta [#1], Dr. A. Ananda Rao [*2], P. Radhika Raju [#3]

[#] *Computer Science and Engineering Department, JNTUA College of Engineering Ananthapuramu, AP- 515002, India.*

[1] sandhyachurchil@gmail.com

[3] radikaraju.p@gmail.com

[*] *Computer Science and Engineering Department*
JNTUA College of Engineering, Ananthapuramu, AP-515002, India.

[2] akepogu@gmail.com

***Abstract*—Software testing is one of the major task which plays a very important and crucial role while developing software. Selecting a single testing technique or methodology is efficient for small projects which are less in size, but in the real world, the case is different, the projects are complex and large in size which results difficulty in testing the software. Also, it is not sufficient using one testing technique to test these large software systems effectively and efficiently because, the development of the software involves on a modular basis which uses different programming languages, techniques and different processes for different modules. In such cases, to test the software it is required to use a combination of different testing techniques to test the different modules of the software system which leads to better testing results that gives the software with the quality. Since, the quality of the software mainly depends on the effectiveness and efficiency of software testing. While selecting the testing techniques, it is required to have some considerations, namely project coverage, human resource suitability, saved modeling effort, effort and cost of the testing, etc. In this proposed system a method for evaluation of cost and effort required has been proposed to test software when the test manager uses Model Based Testing (MBT) techniques to test the software where the other indicators evaluated previously. So that, it helps the test manager while selecting two or more MBT techniques where he can choose the MBT techniques based on project coverage, human resource suitability, saved modeling effort, effort and cost.**

I. INTRODUCTION

In general, after manufacturing any product, it must be tested before exposed in the market so that during the testing process, we check that whether the manufactured/developed product is manufactured/developed correctly or not, whether the product achieved the user/customer requirements or not etc.. The quality of the product can be assessed based on the testing. Therefore, in any product manufacturing/development life cycle testing plays a vital role. Similarly, in the development of the software product software testing plays a very important role. Arilo Claudio Dias-Neto et al. [1] said that software testing is a quality assurance activity in software development.

The main objectives of the software testing are systematic, focused and automated testing of the software. These three objectives of the software testing can be obtained by using Model Based Testing (MBT) technologies. MBT techniques are the techniques which are used to test software based on the models. In this model based testing, models are the specifications for the testing **[2]**. Joonas Lindholm **[3]** said that modeling is the process of developing a representation to a real system or phenomenon, simplifying and abstracting the behavior of the system. Models always define a system or any phenomena from certain point view. Even it possible to define two or models which are entirely different from one another to the same system or phenomena based on consideration of the aspects of it. Model representation can be done using different notations like using graphical charts like Unified Modelling Language (UML), using mathematical models, using specification languages like Specification and Description Language (SDL), Z, etc.

Basically, there are some specialized activities in MBT that are required to work with MBT. Those MBT activities are building a test model, generating test cases, executing test cases and comparing the obtained results with expected results **[1]**. In general, MBT incorporates various abstraction levels, behavioral or structural model of the software, model and source code relationships, technology and test case generation algorithms **[4]**. There are several kinds of MBT techniques available. They use different models which usually represents various characteristics of software. Finite State Machine (FSM) technique, Markov Chain model (MCM), Decision Tables and State charts are some categories of MBT techniques.

In general, real time projects are large in size to solve them project managers use divide and conquer method in which they divide the project into small parts technically say modules and distribute it to different persons/ teams according to their capabilities and complexity of the module etc., after developing these module again they integrate the modules and perform the testing to the project and deliver to the customers. In such scenarios, different teams/persons use different technologies to develop those modules, so that while testing the integrated project managers need to consider all the technologies. Here, it is not sufficient using the single technology to test the entire integrated project. Test managers are required to use two or more technologies to test the

project, in which using combination of two or more technologies gives effective and improved results. Arilo Claudio Dias-Neto et al. [1] discussed a process called porantim for selecting different combination of MBT techniques. Porantim consist of two parts namely a body of knowledge of MBT techniques and a process to support the selection of MBT techniques for software projects. The first part, Body of knowledge is a repository which contains MBT techniques and their characteristics which is developed based on the characterization schema. The second part consist of 5 steps which supports the selection mechanism. Steps for selection are as fallows.

Step 1. Characterizing the Software Project
Step 2. Calculating the Adequacy Level
Step 3. Indicating Adequate MBT Techniques
Step 4. Analyzing Their Combination
Step 5. Selecting MBT Techniques

While analyzing the different combination of the MBT techniques it is required have some indicators so that manager can analyze and select the required combination of MBT techniques. Arilo Claudio Dias-Neto et al. [1] suggested three indicators to support the combined selection of MBT techniques which are namely project coverage, saved modeling effort and human resource suitability. To provide more support, estimation of the cost and effort has been proposed as the fourth indicator which helps in the selection of MBT technique.

The reason for considering the effort and cost estimation as a fourth indicator is, the primary aim of the software engineering is to deliver high quality software with low cost/price also cost is an important criteria in the business world. As we know that testing is the important quality assurance activity. It is necessary to estimate the development cost of the project before developing the project. Estimation is necessary for bidding the software project. Estimation is nothing but the predicting the required time, cost and effort to develop a software product. To estimate the effort of testing, it is enough to calculate the effort percentage of the testing over the development effort..

II. RELATED WORK

Various approaches have been proposed previously to select the different software technologies for the purpose of general selection of software technologies and requirements elicitation techniques. But none of these approaches gives support to the selection of different combination of two or more techniques for single project. There are different activities in the field of software engineering where applying two or more techniques gives improved results, Arilo Claudio Dias-Neto et al. [1] suggested that software testing is one of those activities. Dias-Neto et al. [1] proposed a work, which supports the selection of combination of two or more MBT techniques for a single project. As a part of his work to support the selection, he defined three indicators to provide the support for selection of MBT techniques. Those three indicators are namely project coverage, saved modeling effort and human resource suitability. To improve the effectiveness of this combined selection techniques our work proposes fourth indicator, namely effort and cost which is required to test the software system.

Boehm **[5]** proposed a model, namely COCOMO, to evaluate the effort and cost of the entire software development process including testing. Based on this model effort of testing can be evaluated by calculating the percentage of testing in development. The following works are done previously to calculate effort and cost of the software testing.

In 1998, a neural network concept was proposed by Dawson [6] to calculate the test effort of software. But its main drawback is training of data set which reduces its effectiveness. After that in 2007, Aranha and Bobra [7] proposed a model to estimate test effort based on test specification the project. In this model test cases should be predefined which is the main drawback for this model.

Srivastava et al. [8] also tried to estimate the cost of the project testing based on the fuzzy logic, which are integrated with Halstead metrics. In this fuzzy model, each and every testing metric has been specified with the help of membership values and calculates the different effort drivers for testing using fuzzy rules. To use this model, we must have the source code of the project, which is the drawback to make this model ineffective/less effective. All these methods require the source code to calculate the test effort which is the dominant drawback for all of them. To use these methods, they require completing the initial stage of the project that means they cannot be used starting point of the project.

To overcome this disadvantage in 2011, Praveen Ranjan et al. [9] proposed an approach which is namely fuzzy criteria approach. The fuzzy criteria approach made an attempt for estimating software testing effort with the help of fuzzy logic. In this approach author proposed a fuzzy model in which he combines the fuzzy logics and COCOMO model (cost estimation model for software development process) weighting techniques and test effort drivers to calculate the test effort.

The fuzzy criteria approach cannot be effective when the software is developed based on the models and the test manager want to apply the MBT to test the software because, in this approach they did not consider the criteria regarding the models. Therefore, the proposed work in this paper considers the models while estimating the test effort of the project, which gives the appropriate results test effort estimation.

III PROPOSED METHOD

To estimate the cost of the MBT, first it is required to estimate the effort of the MBT. The effort of the testing can be found by evaluating the MBT percentage in overall software development. This MBT percentage can be estimated by applying the fuzzy logics and intermediate COCOMO model to the software project. The following algorithm is used for the effort estimation.

Algorithm:

Step 1. Estimating KLOC of the project.
Step 2. Evaluating the confidence value (c).
Step 3. Evaluating newKLOC = KLOC * c.
Step 4. Evaluating Software Development Effort (SDE) using newKLOC by applying the intermediate COCOMO model.
Step 5. Identifying software test drivers.
Step 6. Evaluating the percentage of testing (P) by quantifying the test drivers using fuzzy rules.
Step 7. Evaluating Software Test Effort (STE)

STE = SDE * P.

After evaluating the effort which is calculated with the units person hours, cost can calculate based on organization scale. The cost of the software can be evaluated by multiplying the effort with organization scale.

KLOC of the software can be estimated based on the similar projects which are done similarly.

Confidence value is a real number which lies between 0 to 1. It ensures the size estimation which was estimated in the previous step. That is why in the calculation of the confidence value, project manager experience and his familiarity towards the project taken as inputs.

Steps to calculate confidence value:

Input: Number of years of experience that the project managers have and Number of similar projects he done previously.

Output: Confidence value

Step 1. Fuzzification crisp input.
Step 2. Application of fuzzy rules to the input and evaluation of the fuzzy output.
Step 3. Defuzzification of fuzzy output to get confidence value.

To fuzzify the input, membership function are to be generated which are building blocks for fuzzy set theory. Membership functions for experience and familiarity are shown in Fig 1 and Fig 2.

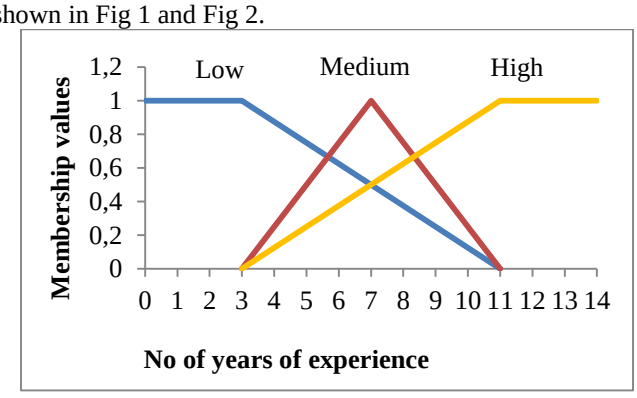

FIG 1: MEMBERSHIP FUNCTIONS FOR PROJECT MANAGER EXPERIENCE

Low, medium and high value will be obtained By applying the membership functions to the crisp value. After obtaining these values, application of fuzzy rules will be done which are shown in Table 1.

Table 1 derives the fuzzy rules as follows.

If both experience and familiarity are same i.e. both are either low, medium or high, then the value of confidence is that value, i.e. low, medium and high respectively

If it has low experience and medium familiarity or vice versa, then the resultant confidence value is low.

Similarly, low experience, high familiarity and vice versa results medium confidence.

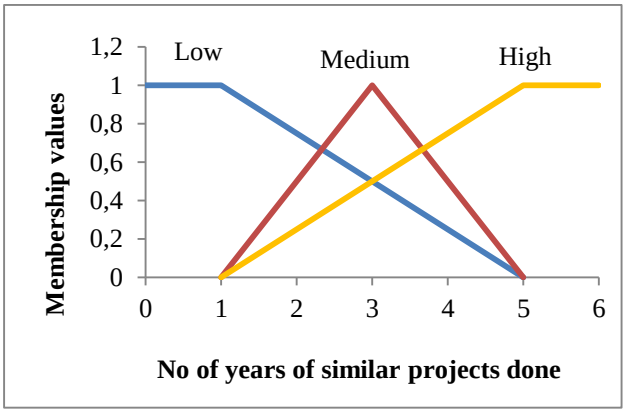

FIG 2: MEMBERSHIP FUNCTIONS FOR PROJECT MANAGER FAMILIARITY

Apart from this, medium experience and high familiarity results high confidence value, but vice versa results medium confidence value which is to be noticed.

TABLE 1: FUZZY RULES

	Experience			
Familiarity		Low	Medium	High
	Low	Low Value	Low Value	Medium Value
	Medium	Low Value	Low Value	Medium Value
	High	Medium Value	High Value	High Value

After applying these fuzzy rules, Max-Min composition is to be applied to low, medium and high values. Max-min composition can find by Max (Min (Low Value), (Medium Value), (High Value)). After applying both fuzzy rules and Max-Min composition, an intermediate value is obtained usig which low, medium and high confidence value can be obtained by applying the membership function of confidence value that is shown in Fig 3.

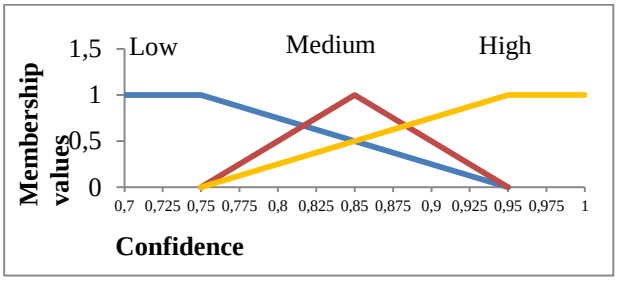

Fig 3: Membership functions for confidence.

By applying the defuzzyfication method namely Center Of Gravity (COG) method to this fuzzy value, a crisp value i.e., confidence value can be obtained.

After evaluating the confidence value, by multiplying the KLOC with this confidence value newKLOC of the project can be obtained.

i.e. newKLOC = KLOC * Confidence

Using this newKLOC, development effort of the software is calculated using the intermediate COCOMO model for which it is required to know the project type whether the project is organic, semi-detached or embedded project. Project categorization can be done based on Table 2. The fallowing formula can be applied to the project where the values of 'a' and 'b' are shown in Table 3.

$E = a * (KLOC)^b * EAF$

Where, By performing the multiplication of the rating values (shown in Table 4) of all the fifteen cost drivers (which are used in intermediate COCOMO model) where each cost driver can be rated by using six rating points namely very low, low, nominal, high, very high and extra high EAF value can be evaluated. EAF is 1, if all the cost drivers are rated as nominal. The value of EAF is in between 0.9 to 1.4.

TABLE 2: PROJECT DEVELOPMENT MODES

Development Mode	**Project Characteristics**			
	Size	**Innovation**	**Deadline/constraints**	**Development Environment**
Organic	Small	Little	Not tight	Stable
Semi-detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex hardware/customer interfaces

TABLE 3: INTERMEDIATE COCOMO MODEL PARAMETERS

Intermediate COCOMO	**a**	**B**
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

By evaluating the percentage of MBT effort for this calculated software development effort, MBT effort can be evaluated. This percentage value depends on test effort drivers which namely software complexity, software quality, schedule pressure, work force drivers and model suitability. Except the schedule pressures remaining all test drivers (which are mentioned in this paper) are directly proportional to the test effort, schedule pressure is inversely proportional to the test effort.

All these test drivers are to be quantified using the fuzzy model which is mentioned previously in the calculation of the confidence value. If the test effort drivers contains sub factors then weighted average mean method is to be used to have a single crisp value. After obtaining the unique crisp values, fuzzy problem solving methodology (discussed in step 2) is to be started to evaluate the effort percentage.

TABLE 1 COST DRIVERS OF INTERMEDIATE COCOMO MODEL

Cost Drivers	**Ratings**					
	Very low	**Low**	**Nominal**	**High**	**Very high**	**Extra high**
Product attributes						
Required software reliability	0.75	0.88	1.0	1.15	1.40	
Size of spplication database		0.94	1.0	1.08	1.16	
Complexity of the product	0.70	0.85	1.0	1.15	1.30	1.65
Hardware attributes						
Runtime performance constraint			1.0	1.11	1.30	1.66
Memory constrints			1.0	1.06	1.21	1.56
Voltality of the virtual machine environments	0.87		1.0	1.15	1.30	
Required turnabout time	0.87		1.0	1.07	1.15	
Personal attributes						
Analysis capability	1.46	1.19	1.0	0.86	0.71	
Applications experience	1.29	1.13	1.0	0.91	0.82	
Software engineering capability	1.42	1.17	1.0	0.86	0.70	
Virtual machine experience	1.21	1.10	1.0	0.90		
Programming language experience	1.14	1.07	1.0	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.0	0.91	0.82	
Use of software tools	1.24	1.10	1.0	0.91	0.83	
Required development schedule	1.23	1.08	1.0	1.04	1.10	

Membership values for software complexity, software quality, workforce drivers and model suitability are shown in Fig 4 and Fig 5. In Fig 6 schedule pressure membership functions are shown.

Schedule pressure (SP) = $(SP_t - SP_m) / SP_t$

Where, SP_t = Estimated Schedule time by development team

SP_m = Estimated Schedule time by management

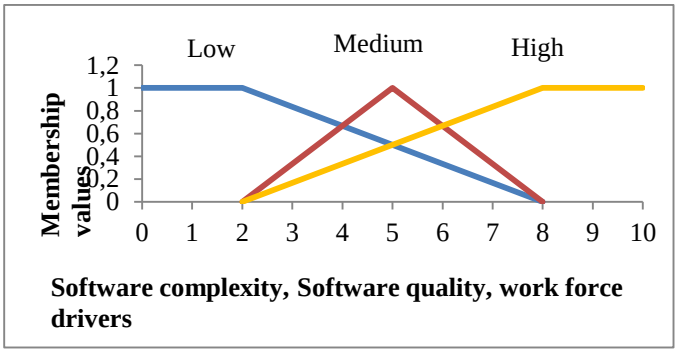

FIG 4: MEMBERSHIP FUNCTIONS OF SOFTWARE COMPLEXITY, SOFTWARE QUALITY AND WORKFORCE DRIVERS.

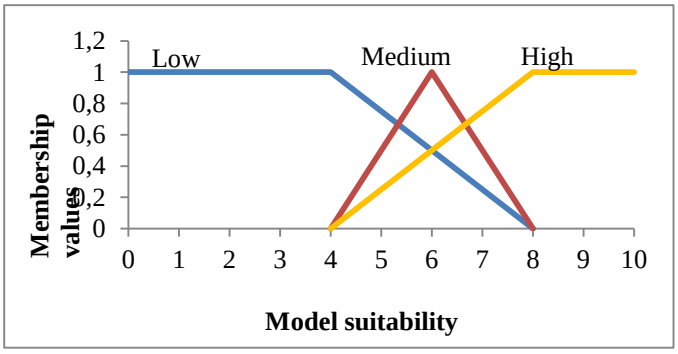

FIG 5: MEMBERSHIP FUNCTIONS FOR MODEL SUITABILITY.

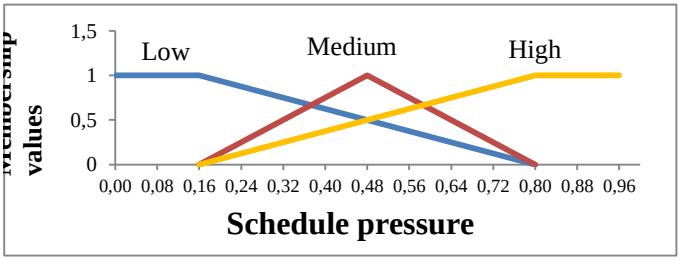

FIG6: MEMBERSHIP VALUES FOR SCHEDULE PRESSURE

Here, the preocedure is same as that is done for calculation the confidence value in step 2. After applying these membership functions, fuzzy rule (same are in step 2) application will be done which is fallowed by application of membership function of percentage as showin Fig 7.

By applying the membership function of percentage we get a fuzzy output which is a fuzzy value for test effort. By applying the defuzzyfication method namely Center Of Gravity (COG) method to this fuzzy value, a crisp value i.e., test percentage can be obtained.

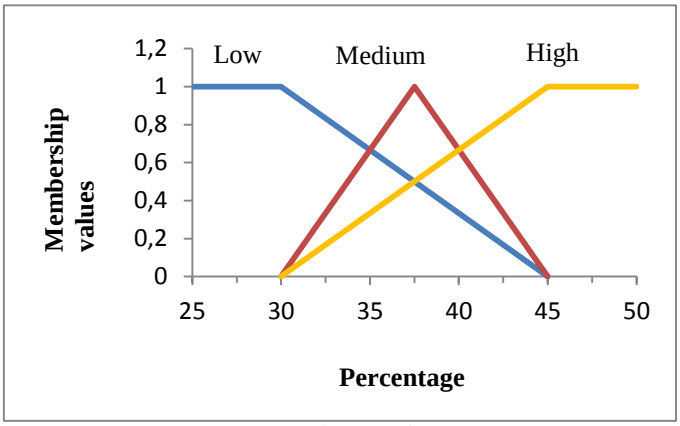

Fig 7: Membership functions for percentage

Finally, test effort can be evaluated by applying the following formula.

$$STE = P * SDE.$$

Where, STE = Software Test Effort

P = Test percentage and

SDE = Software Development Effort.

IV EXPERIMENTAL RESULTS

Table 5 shows the experimental results. For experimental purposes parking system software project has been considered which consists of 2.5 KLOC size and to test this project, three MBT techniques [10, 11, and 12] has been considered. Test drivers namely software quality, software complexity and work force drivers are rated as 7 and for calculating schedule pressure, management mandated time as 150 days and test team mandates time as 180 days. Model suitability is calculated from the saved modeling effort, where it is converted into base 10.

TABLE 5: EXPERIMENTAL RESULTS

Combination of MBT Techniques	Software Project Coverage	Saved Modeling Effort	Human Resources	Effort
MBT Tech [10]	68.23%	50%	70%	2.93
MBT Tech [11]	50.35%	0%	80%	2.97
MBT Tech [12]	63.18%	25%	50%	2.97
MBT Techs [10] + [11]	79.17%	25%	82%	2.97
MBT Techs [10] + [12]	82.44%	35.71%	80%	2.97
MBT Techs [11] + [12]	65.96%	10%	67%	2.97
MBT Techs [10] + [11] + [12]	82.44%	20%	71%	2.97

In the view of effort and cost there will be no difference in the effort when the model suitability is less than 4 because if

the model suitability is 4 or less than 4 then it is low. Similarly, the effort shows no difference when it is 8 or greater than 8 until remaining parameters are unchanged, but the difference can be found when the model suitability is between 4 and 8.

Coming to the experimental result analysis, compare to others MBT tech [10] requires less effort and have high model suitability and better human resource suitability but the project coverage is less. By observing above results, one can say that when the size of the project is constant and model suitability varies then the MBT effort varies inversely propotional to the model suitability. Effort is directly propotional to the size of the project, if the size of the project increases then the effort required to test is also increases. This effort can be reduced by using the increasing the model suitability.

Cost is directly propotional to the effort. Whenever the effort is increased then the cost also increased. Whenever selecting the different combinations of MBT techniques it should be noted about the saturation of the project which means at a certain point selecting multiple MBT techniques leads to no more effectiveness of testing.

IV CONCLUSION AND FUTURE WORK

The estimation of effort and cost of model based testing for a given software project using fuzzy logics has been described in the previous sections. Using fuzzy logics the approximate estimation can be established. Here, the estimation of the effort of model based testing can be evaluated by calculating of the percentage of effort required test the software over the total development effort which can be calculated by using intermediateCOCOMO model. Cost of the project can be evaluated by using the effort where multiplication of the effot and cost units gives the cost. By estimating MBT effort and cost of the software project, test manager can have the benefit in selecting of two or more MBT techniques.

As the part of this proposed work, evaluation of the model based testing effort and cost was done based on consideration and analysis of five test drivers which are namely software quality, software complexity, schedule pressure, work force drivers and model similarity. By considering the more test drivers like type of MBT etc., the effectiveness of the proposed system can be enhanced.

The present work helps the test manager in select the combination two or more of MBT techniques for testing a single project by analyzing the results but still analyzing of the results are difficult which takes time and human effort unnecessarily. By developing a tool which analyses these results and gives appropriate MBT techniques combination as output automatic will be more helpful to the test manager which can be done in future.

REFERENCES

[1] Arilo Claudio Dias-Neto, Guilherme Horta Travassos, "*Supporting Combined Selection of Model based Testing Techniques*", IEEE transactions on software engineering.

[2] Stephan WeiBleder, "*Test Models and Coverage Criteria for Automatic Model-Based Test Generation with UML State Machines*".Ph.D thesis Humboldt-University Berlin, Germany (2009).

[3] Lindhloms, Joonas, "*Model-based testing*", university of Helsinki, Department of Computer Science, 2006.

[4] Pretschner, A. (2005), "*Model-based testing*", Proc. ICSE'05, pp. 722-723.

[5] Nancy Merlo – Schett, "COCOMO *(Constructive Cost Model)*", Seminar on Software Cost estimation WS 2002 / 2003, Requirements Engineering Research Group, Department of Computer Science, University of Zurich, Switzerland.

[6] Dawson, C.W., "*An artificial neural network approach to software testing effort estimation*", Transaction of the Wessex Institute, DOI: 10.2495/AI980361, 1998.

[7] Aranha, E., and P. Borba, "*An estimation model for test execution effort*", Proceeding of Empirical Software Engineering and Measuremen, pp.107–116, 2007.

[8] Srivastava, P., S. Saggar, A.P. Singh, and G. Raghurama, "*Optimization of software testing effort using fuzzy logic*", International Journal of Computer Sciences and Engineering Systems, vol.3, no.3, pp.179–184, 2009.

[9] Praveen Ranjan Srivastava**,** Sirish Kumar**,** A.P. Singh**,** G. Raghurama (2011), "*Software Testing Effort: An Assessment Through Fuzzy Criteria Approach*", Journal of Uncertain Systems Vol.5, No.3, pp.183-201, 2011.

[10] Nebut, C.; Fleurey, F.; Traon, Y.; Jezequel, J. (2006), "*Automatic Test Generation: A Use Case Driven Approach*". IEEE TSE, 32, 3 (March 2006), pp. 140-155. DOI=10.1109/TSE.2006.22.

[11] Peters, D.K; Parnas, D.L. (2002), "*Requirements-Based Monitors for Real-Time Systems*", IEEE TSE 28, 2 (February 2002), 146-158. DOI=10.1109/32.988496.

[12] Chen, T. Y., Poon, P. L.; Tse, T. H. (2003), "*A Choice Relation Framework for Supporting Category-Partition Test Case Generation"* IEEE TSE, 29, 7 (July 2003), pp. 577-593. DOI=10.1109/TSE.2003.1214323.

Selected Paper from International Conference on Computing (NECICC-2k15)

.