International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
NATIONAL CONFERENCE on Developments, Advances & Trends in Engineering Sciences
(NCDATES- 09th & 10th January 2015)

RESEARCH ARTICLE OPEN ACCESS

Fuzzy Technique for Software Development Test Effort Estimation

Vishal Choudhary ,Dr.V.S.Dhaka
School of Engineering and Technology
JAIPUR NATIONAL UNIVERSITY , JAIPUR
Phones: 0141-2779016, 2754814 Fax: 2753377
e-mail : vishalhim@yahoo.com

ABSTRACT
The software are more complex and size of software increased many folds with additional functionality enhancements on the same way the complication of testing as well as the types of testing changes and new testing tools are available in the software industries . As a result, there is a major shift in the estimation of testing the effort. In the software engineering research various models are devised and still in progress to enhance the accuracy of testing tools. Here in this paper I used a fuzzy model for effort estimation for software's development. Here I taken the approximate data and devised the model for effort estimation in software development on various platform. I also used the fuzzy test case for login on different browsers.
Keywords: Fuzzy logic, test estimation, Defuzzification, fuzzy rules, effort estimation, Beta distribution

Fuzzy Logic

The theory of fuzzy logic will be used in the model and the estimation process to remove the ambiguity of the information obtained in early phases of a software development projects.[1] It will help us to manage the ambiguity about the specific meaning of linguistic values used by the "professional" when upcoming up with an approximation.

- In fuzzy precise reasoning is viewed as limiting case of approximate reasoning.
- Every statement has relative degree with which ultimate value be ascribed.
- Any system which can use logic be fuzzyfied.
- Elasticity is associated with all interpretation of knowledge.
- Every opinion how much diverge has a meaning in final interpretation.

Test Estimation and software Engineering:

Test estimation in the software engineering is a broad concept. Some time we use the test estimation in cost, other times we use for efforts and we are testing to find the errors in the final product. But actual testing is the testing which gives us an expected, best and worst output before deployment of a software . We also take in to the consideration the testing cost, testing time and testing efforts. Testing always need experts who know what is right and what is wrong. Even our tools are accurate but final human interpretation will have more importance. There are different approaches of test effort estimation some of them are Delphi technique in which we took the consideration of software size in to person per hour with a conversion factor [2].another approach is analogy based testing [3]. The procedure involves characterizing the projects for that an estimate is to be made. This classification then forms the foundation for making the decision of similar or related projects which have been completed and for them effort is known. These effort data are then used, perhaps with modification, to generate the expected value. A complexity with this method includes, finding the correlation and assesses the level of similarity.

Fuzzy approach in Testing and estimation:

New mathematical techniques, to estimate are non-algorithmic these are based on soft computing developed in mid 1990s, many researchers and scientist around the globe start working on them.. This paragraph discuss the non-algorithmic techniques for software development and effort estimation. Soft computing based on the methods centered around fuzzy logic (FL), and progression in new computations techniques (CT). These methods used in real life fuzzy applications by providing some flexibility in information processing and segregation . Soft computing logics and techniques are used by many research fraternity for software development and effort prediction to handle the fuzzy and uncertainty in data values, due to their fundamental nature. The first real use of the fuzzy logic from many application in engineering that is used in software and successfully implemented and widely used model is for constructive cost estimation model, COCOMO, was used by Fei and Liu [5]. They found in observations that an accurate estimate of knowledge delivery source instruction cannot be made before starting any software project; therefore, it is impossible to assign a predefined number for it. Jack Ryder used the application of fuzzy modeling methods for two of the most widely used applications for effort prediction;COCOMO and the Function-Points models, respectively [6]. Fuzzy Logic was applied to evaluate the cost drivers of intermediate COCOMO model because it has comparatively high estimation accuracy than the basic version

Test Effort Estimation using test case listing.

Test Effort Estimation is the *estimation* of the testing the size, testing the effort, testing the cost and testing the schedule for a particular software project in a specific environment using distinct methods, tools and techniques**.** One way is to compute software size to Test Effort. Let us say one FP needs 2 hours of testing, If the project size is 100 FP then Testing Effort is 200 Person Hours.
Let us say one UCP needs 4 hours of testing If the project size is 100 UCP then Testing Effort is 400 Person Hours. There are no benchmarked data available therefore, organizational data needs to be developed and maintained norms to be developed and maintained.

Fuzzy rules:

In general fuzzy rules are the group of statements that explain how the fuzzy inference system make the decision based on complex input dataset to output the exact and acceptable results.in fuzzy rules we generally use AND, OR, NOT operators to link the complex dataset, and these group of statements or dataset is used to identify the desirable output. Fuzzy rules are written in the form like:
if (input1 dataset)AND/OR/NOT(input 2 dataset)AND /OR/NOT(input3 data set) THEN(output$_n$ is output of desirable values).e.g
if (the application is standalone) AND(we are testing the functional point)AND (Use case point) then total effort is 0.75.

One rule itself can't do much fine. What we needed are two or more rules that can be used in combination with each other. The output of each rule is a fuzzy set. The output fuzzy sets for each rule is *aggregated* into a single output fuzzy set. An at last the resulting values is *defuzzified* in to a single number. We can shows that how the whole process working in a special type of fuzzy inference system known as Mamdani type inference.

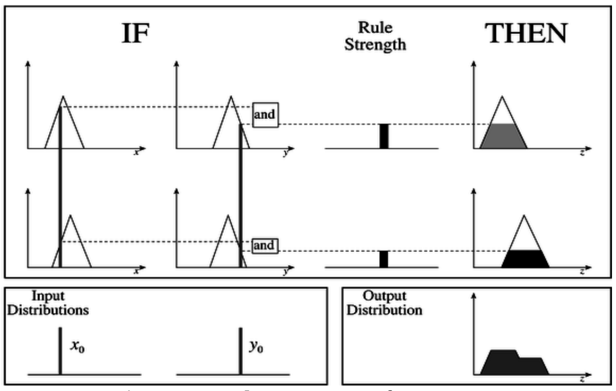


Fig 1: Mamdani Fuzzy inference

Here used the linguistic variables to check the test effort in each type of application based on its size either it will works on standalone system or it may works on client server or may be on 3-tier or 4-tier network system. The data used for these inferences is shown in table 1.

	←Person Hours per unit →					
Type of application	**Function Point**	**Use Case Point**	**Object points**	**Feature Points**	**Software size unit(SSU)**	**Average Efforts**
Stand –alone	0.25	0.5	0.25	0.25	0.5	0.35
Client-Server	0.3	0.65	0.3	0.3	0.65	0.44
3-tier Application	0.5	0.9	0.5	0.5	0.9	0.66
4-tier application	0.75	1.1	0.75	0.75	1.1	0.89

Table1: effort test case in person hours per unit.

Effort in Test case based on Size:

Table below gives approximate values of effort estimation in person per hours based on the effort on testing unit of a software project. And effort on testing unit varies with the complexity of the application. These are characterized as.
a. Function-Point: the function point is a unit to estimate the software size and complexity .once we identify the different function of software they are checked for their complexity and assigned a number known as function point.
b. **Use case point:** it is a technique for forecasting the software size and mainly used when the unified modeling language are used for design and development.
c. Object Point: this is an estimation technique based on count of raw objects, complexity of each object and weighted points

d. Feature Point: this is a technique not used today for estimation of software size based on feature a software will provides.
e. Software Size unit: this is another software estimation technique based on the component description.
Here I will apply the fuzzy rule on the raw data available in above table for test case estimation

1. If (function_point is for stand-alone system) and (use_case_point for stand-alone system) and (object_point is for standalone) and (feature_Points is for stanalone) and (SSU is for stand_alone system) then (EFFORT is low) (0.35) .
2. If (function_point is for client-server system application) and (use_case_point is for client_server application) and (object_point is for standalone application) and (feature_Points is for client_server application) and (SSU is for client_server) then (EFFORT is low) (0.44)
3. If (function_point is for 3-tier) and (use_case_point is for 3-tier) and (object_point is for 3-tier) and (feature_Points is for 3-tier) and (SSU is for 3-tier) then (EFFORT is Medium) (0.66)
4. If (function_point is 4-tier) and (use_case_point is 4-tier) and (object_point is 4-tier) and (feature_Points is 4-tier) and (SSU is 4_tier) then (EFFORT is High) (0.89)
5. If (function_point is stand-alone) and (use_case_point is client_server) and (object_point is 3-tier) and (feature_Points is 4-tier) and (SSU is 4_tier) then (EFFORT is Medium) (0.65)
6. If (function_point is 4-tier) and (use_case_point is 3-tier) and (object_point is 4-tier) and (feature_Points is 4-tier) and (SSU is 3-tier) then (EFFORT is High) (0.81)
7. If (function_point is client-server) and (use_case_point is client_server) and (object_point is 3-tier) and (feature_Points is 4-tier) and (SSU is 4_tier) then (EFFORT is Medium) (0.66)
8. If (function_point is client-server) and (use_case_point is 4-tier) and (object_point is 4-tier) and (feature_Points is client_server) and (SSU is 4_tier) then (EFFORT is High) (0.71)
   If (function_point is 3-tier) and (use_case_point is client_server) and (object_point is 3-tier) and (feature_Points is client_server) and (SSU is 3-tier) then (EFFORT is Medium) (0.57)
9. If (function_point is 4-tier) and (use_case_point is 3-tier) and (object_point is standalone) and (feature_Points is stanalone) and (SSU is 4_tier) then (EFFORT is Medium) (0.65)

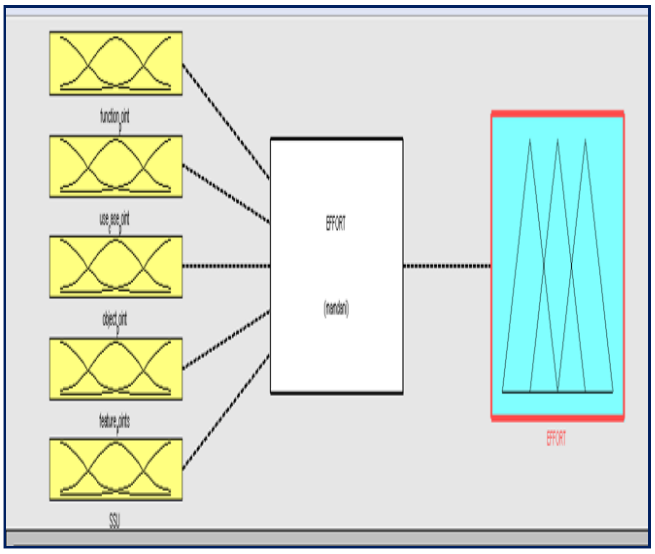


Fig 2: Mamdani fuzzy logic system for effort estimation

Rule Viewer for inference System:
Figure below used for inference the test case effort estimations.

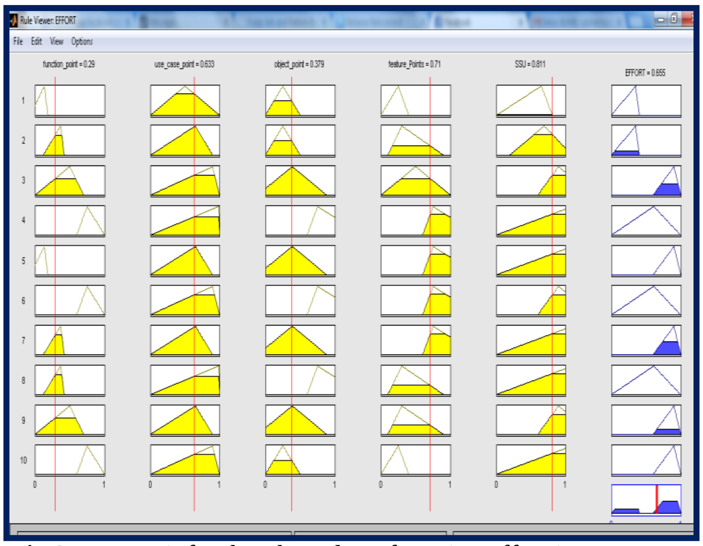


Fig 3: we can infer that the value of testing effort is around 65% when software size is large and use case point is around 63%.

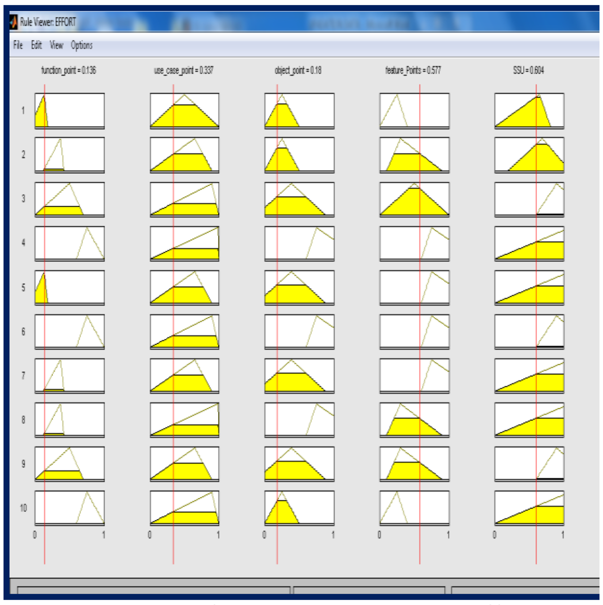


Fig 4: when the functional point testing effort are low also use case are low even the object point and feature point and size of software is moderate then effort is around 32%

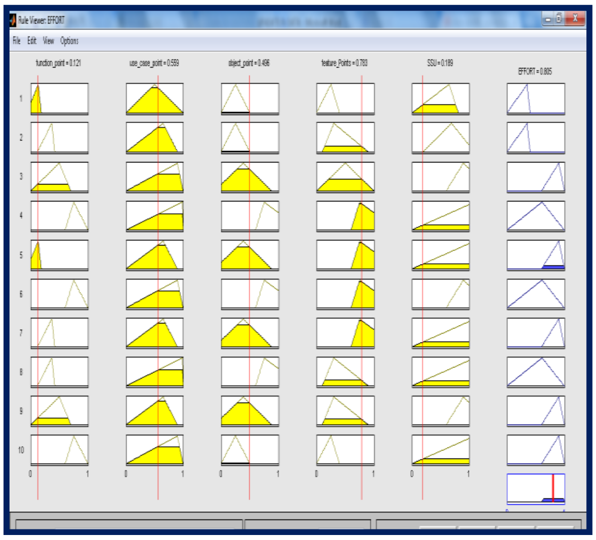


fig5: From above figure we can infer that even the software size is small but the efforts are more influenced by use case point, object point and feature points here in this case effort is maxi around 80%.

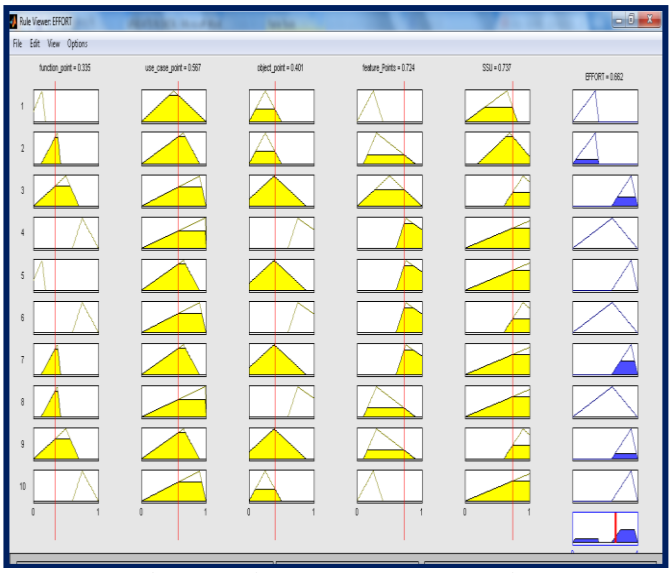


Fig 6: the influence of parameters on the software test effort estimation with variation of different fuzzy inputs values.

Enumeration Based Test effort Estimation:

In this technique we classify all the test cases under the consideration. Then we estimate the effort in each test case in person-hours that is further classified in to best case, worst case and normal case. And then we calculate the expected effort using the Beta distribution[12].

Expected Effort= Best Case + Worst Case + (4 * Normal Case) / 6

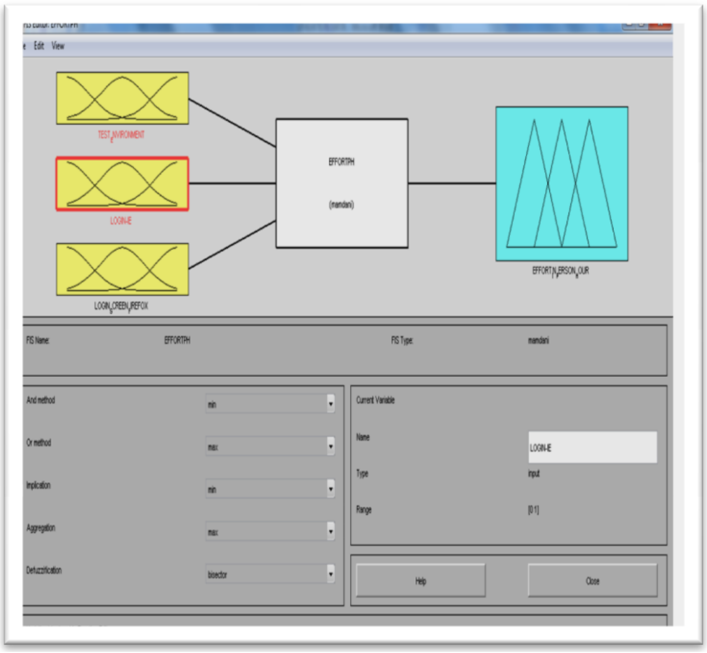
If we sum up all we will get the total effort in each case.

	Test Case Description	←------Effort in PH------------→			
		Best case	Worst Case	Normal case	Expected
Setup Test Enviro-nment	Check Test Environment	1	2	1.5	1.500
	Install Screen Recorder	0.75	1.5	1	1.042
	Ensure Defect Reporting Mechanism	1.25	3	3	2.042
Login screen on IE	Correct Login	0.05	0.2	0.1	0.108
	Wrongid and Correct Password	0.07	0.2	0.1	0.112
	Correct id and wrong password	0.07	0.2	0.1	0.122
Login Screen on Firefox	Correct Login	0.04	0.2	0.1	0.108
	Wrong id and Correct Password	0.07	0.2	0.1	0.112
	Correct id and wrong password	0.07	0.2	0.1	0.122

Table 2: Test case description for browsers

Here below in inference rules I used the B_CTE for Best case of check test environment and B_CL for best case of correct Login.W_CL for worst case of correct login N_WI_CP is the normal case of wrong id and correct password and so on.
Inference rules for table 2. given below.

1. If (TEST_ENVIRONMENT is B_CTE) and (LOGIN-IE is B__CL) and (LOGIN_SCREEN_FIREFOX is not B_CL) then (EFFORT_IN_PERSON_HOUR is EXPECTED-STE) (1)
2. If (TEST_ENVIRONMENT is B_ISR) and (LOGIN-IE is B__CL) and (LOGIN_SCREEN_FIREFOX is not B_CL) then (EFFORT_IN_PERSON_HOUR is EXPECTED-IE) (0.18)
3. If (TEST_ENVIRONMENT is B_CTE) and (LOGIN-IE is N_WI_CP) and (LOGIN_SCREEN_FIREFOX is not B_CL) then (EFFORT_IN_PERSON_HOUR is EXPECTED-IE) (1)
4. If (TEST_ENVIRONMENT is B_CTE) and (LOGIN-IE is not N_WI_CP) and (LOGIN_SCREEN_FIREFOX is B_CL) then (EFFORT_IN_PERSON_HOUR is EXPECTED_LI_FF) (0.108)
5. If (TEST_ENVIRONMENT is B_CTE) and (LOGIN-IE is not N_WI_CP) and (LOGIN_SCREEN_FIREFOX is B_WI_CP) then (EFFORT_IN_PERSON_HOUR is EXPECTED_LI_FF) (0.806)
6. If (TEST_ENVIRONMENT is B_ISR) and (LOGIN-IE is W_WI_CP) and (LOGIN_SCREEN_FIREFOX is not B_WI_CP) then (EFFORT_IN_PERSON_HOUR is EXPECTED_LI_FF) (0.806)
7. If (TEST_ENVIRONMENT is w_CTE) and (LOGIN-IE is W_CL) and (LOGIN_SCREEN_FIREFOX is not W_WI_CP) then (EFFORT_IN_PERSON_HOUR is EXPECTED_LI_FF) (0.806)
8. If (TEST_ENVIRONMENT is W_EDR) and (LOGIN-IE is N_WI_CP) and (LOGIN_SCREEN_FIREFOX is not B_FPF) then (EFFORT_IN_PERSON_HOUR is EXPECTED-STE) (0.806)
9. If (TEST_ENVIRONMENT is w_CTE) and (LOGIN-IE is W_WI_CP) and (LOGIN_SCREEN_FIREFOX is not W_WI_CP) then (EFFORT_IN_PERSON_HOUR is EXPECTED-STE) (0.806)
10. If (TEST_ENVIRONMENT is B_ISR) and (LOGIN-IE is W_CL) and (LOGIN_SCREEN_FIREFOX is not B_WI_CP) then (EFFORT_IN_PERSON_HOUR is EXPECTED-STE) (1)



FIS Name: EFFORTPH
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: bisector
Current Variable
Name: LOGIN-E
Type: input
Range: [0 1]

Fig 7: inference system for enumeration based test effort

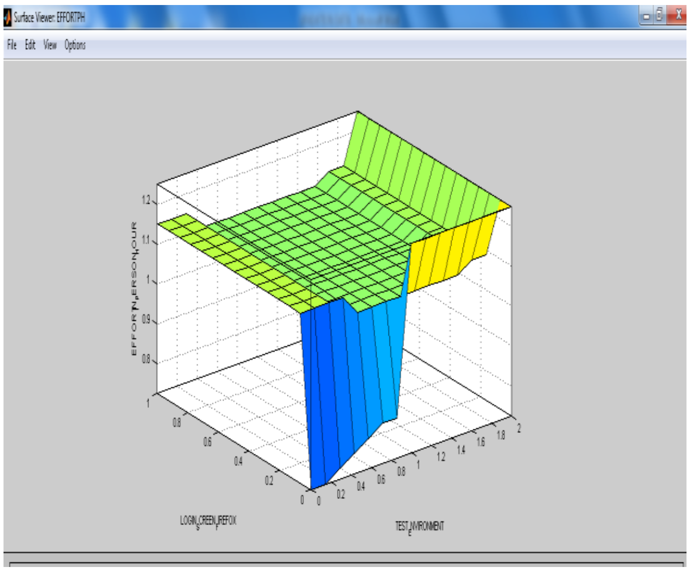


Fig 8: Login on Firefox browser, test environment and effort person per hours.

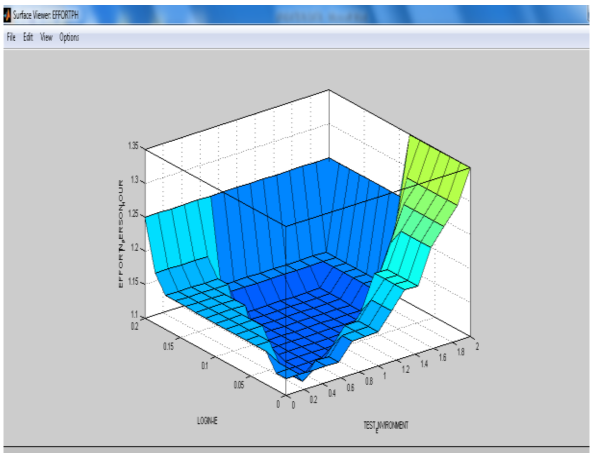


Fig 9: Login on IE browser, test environment and effort.

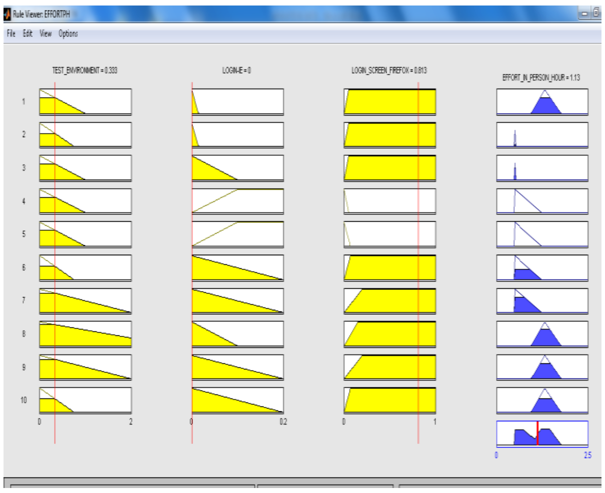


Fig 10: Test environment and login on Firefox and effort person per hours

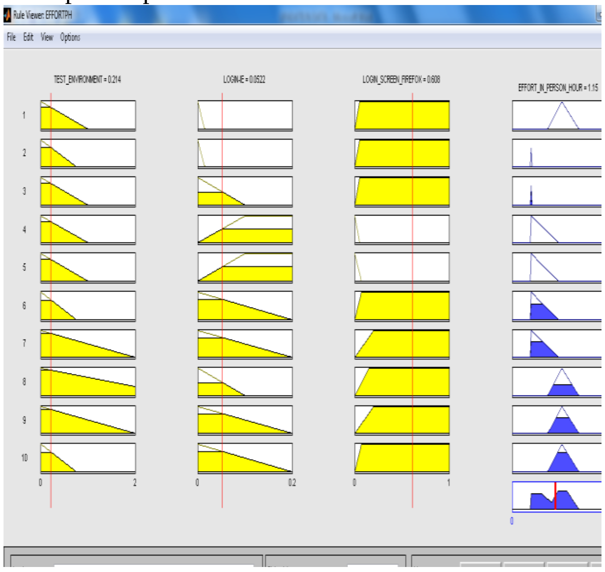


Fig 11: Test environment and login on IE and effort person per hours

Conclusion

Test effort estimation using the well established tools is highly influence our approach of software developments .but with the inventions of new mathematical models like fuzzy logic we can infer the correct estimate and the development of software become more reliable. in this paper I just given the ideas based on approximate values but on actual test case parameters how this model varies will have to be described and tested.

REFERENCES

[1]. Design of a Fuzzy Logic Software Estimation ProcessMotivations Département de Génie Logiciel et des Technologies de l'Information Laboratoire de Recherche en Génie logiciel (GÉLOG) Francisco Valdés Alain Abran,2009.

[2]. Andreas Spillner, Tilo Linz, Hans Schäfer. (2006). *Software Testing Foundations - A Study Guide for the Certified Tester Exam - Foundation Level - ISTQB compliant*, 1st print. dpunkt.verlag GmbH, Heidelberg, Germany. ISBN 3-89864-363-8

[3]. Atkinson, K. and M.J. Shepperd. 'The use of function points to find cost analogies', in Proc.European Software Cost Modelling Conference. Ivrea, Italy: 1994.

[4]. K. Srinivasan, and D. Fisher, "Machine learning approaches to estimating software development effort", IEEE Transactions on Software Engineering, 21(2) 1995.

[5]. Z. Fei, and X. Liu, "f-COCOMO: fuzzy constructive cost model in software engineering", Proceedings of the IEEE International Conference on Fuzzy Systems, IEEE Press, New York, 1992 pp. 331–337..

[6]. J. Ryder, "Fuzzy modeling of software effort prediction", Proceedings of IEEE Information Technology Conference, Syracuse, NY, 1998.

[7]. Agarwal, M. (2007). Fuzzy logic control of washingachines.http://softcomputing.tripod.com/

[8]. G. D. Boetticher, "An assessment of metric contribution in the construction of a neural network-based effort estimator", Proceedings of Second International Workshop on Soft Computing Applied to Software Engineering, 2001.

[9]. W. Pedrycz, H. F. Peters, and S. Ramanna, "A fuzzy set approach to cost estimation of software projects", Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering, Alberta, Canada, 1999.

[10]. M. W. Nisar, and Y.-J. Wang, and M. Elahi, "Software Development Effort Estimation Using Fuzzy Logic – A Survey", in 5th

International Conference on Fuzzy Systems and Knowledge Discovery, 2008, pp. 421-427.

[11]. Geoffery K.Gill and chris F.Kemeren Cyclomatic Density and software maintenance productivity,IEEE transaction on software engineering.Vol 17,No.12 1284-1288.december 1991.

[12]. Software estimation best practices, tools & techniques: a complete guide for software project estimators, J. Ross Pub., 2009