INFOTEH-JAHORINA Vol. 13, March 2014.

Challenges in Estimating Software Testing Effort

Ljubomir Lazić, Ivan Đokić
State University of Novi Pazar
Vuka Karadžića bb
36 300 Novi Pazar, SERBIA
llazic@np.ac.rs, idjokic@np.ac.rs

Stevan Milinković
The School of Computing
Union University
Belgrade, Serbia
smilinkovic@raf.edu.rs

Abstract— **Effort estimation is one of the critical challenges in Software Testing Life Cycle (STLC). It is the basis for the project's effort estimation, planning, scheduling and budget planning. This paper illustrates few models with an objective to depict the accuracy and bias variation of an organization's estimates of software testing effort using two historical datasets of 50 finished software projects. In this research, a statistical study is performed. A multiple regression analysis model has been built to predict the Testing Time Effort for software development projects based on several Independent or Predictor variables. The independent variables under consideration are: total number of test cases created per module, experience of the developer of the module in years, experience of the tester in years and the size of the module. Besides, the software documentation size (#Pages of requirements, design, project plans, test plans, requirements changes, training materials, HELP text, and user manuals) was selected and used for the software testing effort estimation. The proposed models' estimation figures are accurate enough to be appropriate techniques for estimating effort for software testing.**

Key words - software testing; testing effort estimation; multiple regression analysis, estimation accuracy

I. INTRODUCTION

Software Testing is an important process of software development which is performed to support and enhance reliability and quality of the software. It consists of estimating testing effort, selecting suitable test team, designing test cases, executing the software with those test cases and examining the results produced by those executions. Test Estimation is the estimation of the testing size, effort and schedule for a specified software project in a specified environment using defined methods, tools and techniques. Studies indicate that 40-50 percent of the cost of software development is devoted to testing, with the percentage for testing critical software being even higher.

Today many approaches are available to estimate the overall software size and effort, however there is a lack of structured and scientific methods for estimating software testing size and effort [1,2]. One of the most well-known overall software size and effort estimation model is cost and quality estimation model COnstructive QUALity MOdel (COQUALMO) [5] which is an extension to the COnstructive COst MOdel (COCOMO) [4]. The COCOMO is an algorithmic software cost estimation model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics. Some of the techniques used for estimating test effort range from percentage of the development effort to more refined approaches based on Use Case and Function Point (FP) - depending on functional and technological complexity[1,2,7,10] .

Some of the commonly used test estimation techniques are:

1) Use of Development Size estimates - Source Lines of Code (SLOC), use case, function points
2) Experience Based - Analogies and experts: rule of thumb
3) A percentage of development effort
4) Risk-based methods - determine what to test and how much
5) Use of Historical Data + Parametric model design
6) Work breakdown structure – WBS method

To ensure project/program success, increase overall product quality and improve time to market, it is an imperative that the approach to estimating software test effort is as accurate as possible.

Data from few large projects was collected, although different metrics were used for the two projects. A multi linear model was used, which was reduced by factor analysis to remove some of the metrics that were linearly related to one another. Application of a stepwise multilinear regression technique selected independent variables from the metrics that remained to determine the coefficients for the equation which represents the Testing Time Effort model (TTEM) for Software Development and Software Testing Life Cycle.

The independent variables under consideration are: total number of test cases created per module, experience of the developer of the module in years, experience of the tester in years and the size of the module. Besides, the software documentation size (#Pages of requirements, design, project plans, test plans, requirements changes, training materials, HELP text, and user manuals) was selected and used for the software testing effort estimation. The proposed models' estimation figures are typical of those studies using regression analysis – the 'goodness of fit' may be reasonable and accurate enough to be appropriate techniques for estimating effort for software testing [6,7,10].

II. SOFTWARE TESTING EFFORT MODEL

A. Parametric Modeling Process

Parametric modeling is a statistical technique whereby a dependent variable is estimated based on the values of and the relationships between the independent variable(s). The nature of the dependent variable can vary greatly based on one's domain of interest.

Software development environment consists of series of phases [2,7] and in all the phases of Software Development Life Cycle (SDLC) software testing is one of the major phases. As such software testing is the process of validation and verification of the software product. Effective software testing will contribute to the delivery of reliable and quality oriented software product, more satisfied users, lower maintenance cost, and more accurate and reliable result in day to day working environment of software professionals.

In this paper, we discuss the use of parametric modeling for Estimating Software Testing Effort and present a nine step parametric modeling process, adopted from [11].

The overall purpose of parametric models is to make an estimation or prediction based on current information. In the general sense, a function y = f(*x1, x2, x3, …*) is created such that *xi* is an input to the function and *y* is the variable being estimated. Some examples of the types of relationships in parametric modeling are [4,5]:

• *Analogy*: Outcome = f(previous outcome, difference) – used to make a prediction based on what happened before and then taking into account the differences in the scenario. *Examples*: development time prediction; traffic patterns.

• *Unit Cost*: Outcome = f(unit costs, unit quantities) – used to make a prediction based on known production values. *Examples*: potential profit based on software units available to be sold.

• *Activity-Based*: Outcome = f(activity levels, duration) – used to make a prediction based on time spent performing a specified activity. *Examples*: training personnel costs.

• *Relationship-Based*: Outcome = f(parametric relationships) – used to make a prediction based on the relations and interactions of inter-dependent variables. *Examples*: predicting defects introduction based on software complexity, technology used, and team performances; software size/cost models.

Parametric models may be calibrated for use in a particular situation, organization, or even particular project.

B. Proposed Approach for Estimating Software Testing Effort

In this section, we present the nine-step process, adopted from [11], for parametric modeling, as shown in Figure 1. The steps are shown in a general waterfall order. However, feedback and concurrency between steps can and should occur, as the dashed arrows indicate.

Hence, software testing is a necessary and important activity of software development process. However, it affects overall software life cycle, because quality of software life cycle depend upon testing technique demanding adequate test case preparation, modeling, and documentation which make the process complicated and challenging. These impending challenges have to be addressed by researchers and practitioners working closely together by estimating the amount of effort that is required to develop user-friendly software [10].

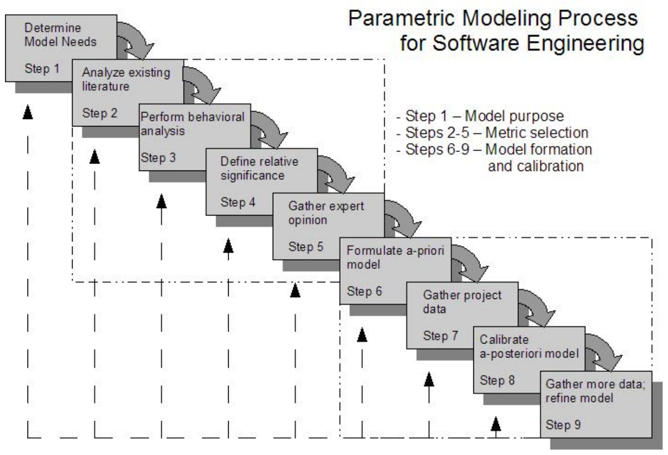
Parametric Modeling Process for Software Engineering
Determine Model Needs
Step 1
Analyze existing literature
Step 2
Perform behavioral analysis
Step 3
Define relative significance
Step 4
Gather expert opinion
Step 5
Formulate a-priori model
Step 6
Gather project data
Step 7
Calibrate a-posteriori model
Step 8
Gather more data; refine model
Step 9
- Step 1 – Model purpose
- Steps 2-5 – Metric selection
- Steps 6-9 – Model formation and calibration

Fig.1 Parametric Modeling Process for Software Engineering [11]

An effective metrics program [8] must be tightly coupled to the software development process. The metrics program and the development process are mutually supportive. It is essential for test professionals to know how their testing project is progressing and what the quality of the product they are testing is.

Metrics & Measurements is a key aspect in both project and test management. Gone are those days when metrics was considered a CMM&TMM Level-4 or Level-5 requirement in the quality Levels journey that is to say, only matured companies need to follow metrics. Metrics have become the backbone of every organization and have become a de-facto requirement for periodical project reviews of all companies now. The metrics program, by adding quantitative measurement (Number of Use cases, Number of Test cases, Tester Years of experience, Developer Years of experience, Testing Time Effort in Days) [1,7,8], makes the development process more visible and understandable to the software development managers and team. At the same time, the development process defines integral points of data collection in support of the metrics program.

The software development process is made up of phases. For our purposes, the development process (W-model) will be divided into four phases: 1. Requirements (software systems engineering); 2. Design (analysis models and designs); 3. Implementation (coding, unit testing, subsystem testing) and 4. Testing (integration testing, system testing, acceptance testing). Within each of these phases, unique development products are produced, like, source lines of code, Number of Pages produced for: requirements, design, project plans, test plans, requirements changes, training materials, HELP text, and user manuals. Collected data is statistically analyzed and their actual production time efforts, average values, standard deviation etc. are calculated. Jones [2] estimated the number of test cases which can be determined by the function point's estimate for the corresponding effort. The actual effort in person-hours was then calculated with a conversion factor obtained from previous project data. The main disadvantage of using function points is that they require detailed requirements in advance. In another study on Test Point Analysis, a method for estimating the effort has been emphasized to perform all functional test activities based on use case points [9]. This model estimates the effort required for all test activities together, such as defining, implementing and executing all the tests.

III. EMPIRICAL METHOD

This section will present our empirical method of establishing Testing Time Effort (TTE) estimation model for testing process. The two phases of the method are to: (1) identify the STLC objectives to be managed quantitatively and construct data samples from which initial linear regression estimation model will be designed; (2) establish new TTE estimation model according testing process artifacts performance baseline for the identified metrics and cross validate all models using new dataset.

A. Building TTE Regression Models

This research is driven towards achieving several objectives as follows:

• To analyse existing metrics, techniques and approaches used in building prediction model for TTE

• To formulate prediction model for TTE using statistical approach based on metrics in software testing process

• To evaluate the accuracy of proposed prediction model based on acceptable criteria for final selection of defect prediction model for TTE

This section describes the model-building strategies that were used for predicting Testing Time Effort. Multiple linear regression (MLR) analysis was used to model the relationship between quality, testing and software metrics [1,8] based on two data samples: Dataset_1 and Dataset_2. In the first phase we used collected metrics of a large project (Dataset_1, Table I) consisting on 31 module (component) to preliminary build few candidates of TTE regression model. There are many variables that define a software development project. These authors have extensive experience in the area of software development and software quality. Based on that experience and after analyzing literature on testing effort estimation, we selected four independent variables-metrics X1 (Number of Use cases), X2 (Tester Years of experience), X3 (Developer Years of experience) and X4 (Number of Test cases). In our work, the dependent variable that is to be measured is the TTE i.e.Y [Day] and the independent variables are the metrics presented in Table I.

We applied Surface Response Modeling (SRM) method, a case of Design of Experiment (DOE) method in MINITAB ver.16 statistical software tool (see Fig. 2, MINITAB 16 screenshots) in order to find regression equation for Y as functions of X1,X2,X3 and X4 in the form:

$$Y = b_0 + \sum_{i=1}^{n} b_i X_i + \sum_{i \neq j,1}^{n} b_{ij} X_i X_j \quad (1)$$

where : Y - is the estimated (dependent) output variable i.e. TTE i.e.Y [Day] in our case, n - number of independent variables-metrics (4 in our case), b_0 , b_i – linear regression coefficients without variable interaction, b_{ij} – variable interaction regression coefficients, and X_i – real i^{th} metrics values in the experiment.

TABLE I. HISTORICAL METRICS DATASET_1 FROM LARGE SOFTWARE PROJECT

Modu-les	Y TTE [Day]	*X1* *#Use cases*	*X2* *#Tester Years of experience*	*X3* *#Developer Years of experience*	*X4* *#Test cases*
Mod.1	48.29	3	4	9	31
Mod.2	66.04	17	3	6	18
Mod.3	58.35	12	3	11	38
Mod.4	36.00	3	6	6	4
Mod.5	39.82	5	7	13	30
Mod.6	31.19	13	6	6	15
Mod.7	6.92	11	4	15	9
Mod.8	40.66	8	3	8	53
Mod.9	15.30	3	1	9	12
Mod.10	20.73	2	4	8	11
Mod.11	75.36	16	1	8	41
Mod.12	17.30	3	6	9	18
Mod.13	55.16	7	4	8	44
Mod.14	170.19	41	6	11	89
Mod.15	340.29	63	4	13	260
Mod.16	193.23	7	6	13	55
Mod.17	34.43	2	6	6	7
Mod.18	91.62	10	6	6	30
Mod.19	5.40	7	4	8	16
Mod.20	22.17	9	8	6	54
Mod.21	38.91	7	6	5	29
Mod.22	38.51	29	4	8	48
Mod.23	35.98	7	8	21	21
Mod.24	118.82	19	6	6	60
Mod.25	45.23	7	3	6	48
Mod.26	45.41	12	6	9	72
Mod.27	37.69	15	4	8	66
Mod.28	37.01	10	6	6	14
Mod.29	41.62	3	5	6	12
Mod.30	43.93	9	4	6	8
Mod.31	128.48	35	3.6	7.75	136

Further, the model building strategies have the following associated factors to compare TTE estimation models characteristics of the goodness:

• The coefficient of determination, R^2 - it is the amount of variation explained by the regression equation which is used to predict future outcomes on the basis of other related information. It is a statistical term saying how good the particular generated equation is at predicting functional defects. R-Sq. value must be above 50%.
• The F-ratio is used to test the hypothesis that all regression coefficients are zero at statistically significant levels.

• Where parametric testing is appropriate, a significance level of $\alpha = 0.05$ was adopted for statistical inference. For example, all interval estimates are reported using confidence level 95%.

Large deviations in size estimation area require model performance comparison using some heuristic rejection rules that compare more than just mean performance data [12]. The results for each validated method were compared using each treatment of MMRE, SD_{MRE}, and larger R^2 correlation. MMRE comes from the mean magnitude of the relative error, or MRE, the absolute value of the relative error:

$$MRE = \left| predicted - actual \right| / actual \tag{2}$$

The mean magnitude of the relative error, or MMRE, is the average percentage of the absolute values of the relative errors over an entire data set. Given *n* tests (estimators), the MMRE is:

$$MMRE = \frac{100}{n} \sum_{i}^{n} \frac{\left| predicted_i - actual_i \right|}{actual_i} \tag{3}$$

The standard deviation, or SD_{MRE}, is root mean square of MRE. Given a *n* tests (estimators), the SD_{MRE} is:

$$SD_{MRE} = \frac{100}{T} \sum_{i}^{n} (MRE_i - MMRE)^{1/2} \tag{4}$$

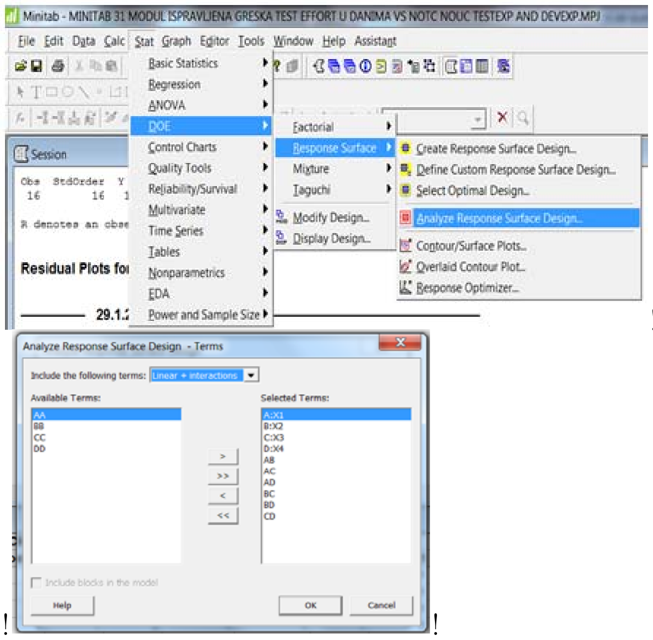

Fig. 2 DOE->SRM MINITAB ver.16 statistical software screenshots

!

The initial linear regression model was built using DOE -> SRM options in MINITAB ver.16 statistical software, without X1, X2, X3, X4 independent variables interaction to estimate test effort Y [Day], as the dependent variable, using the proposed metrics values from Table I. Then we developed linear regression model, with independent variables interaction, to estimate test effort Y [Day] using all the proposed metrics values from Table I. All regression model candidates are expressed with equation (5) to (8).

$$Y[Day] = 16.237 + 0.50712 * X1 - 0.775 * X2 - 0.367 * X3 + 0.3498 * X4 \tag{5}$$

$$\begin{aligned} Y[Day] &= 22.962 + 4.606 * X1 - 0.376 * X2 - 0.478 * X3 - 1.776 * X4 + 0.72 * X1 * X2 \\ &- 0.707 * X1 * X3 - 0.015 * X1 * X4 + 0.364 * X2 * X3 \\ &- 0.074 * X2 * X4 - 0.273 * X3 * X4 \end{aligned} \tag{6}$$

$$Y[Day] = 0.8773 * X4 \tag{7}$$

$$Y[Day] = 0.4341 * X4 \tag{8}$$

In second phase new regression models candidates were designed and cross validated, as well as previous TTE models designed in the first phase, using new Dataset_2 presented in Table II, which is adapted from published work [3]. As for the type of software which data is collected, it comprises of web-based and component-based developed in Java, PHP or Hypertext Preprocessor and .NET. During this phase, further refinement is done to the previous equations. Realizing the need to generate prediction model that is both practical and makes sense from the software engineering practitioners, the predictors have been revised so that only logical predictors selected by filtering metrics that contain only valid data and reducing the model in order to have logical correlation with TTE.

TABLE II. DATASET_2 FROM 14 PROJECTS

Project	**Y TTE [Day]**	***X4 #Test cases***	***X5 Size [KLOC]***	***X6 #Requir. Pages***	***X7 #Design Pages***
A	16.79	224	28.8	81	121
B	45.69	17	6.8	171	14
C	13.44	24	5.4	23	42
D	4.9	25	1.1	23	42
E	4.72	28	1.2	23	54
F	32.69	88	6.8	20	70
G	64	149	4	38	131
H	5.63	24	0.2	26	26
I	9.13	13	1.4	15	28
J	89.42	306	36	57	156
K	17	142	12.3	162	384
L	8.86	40	3.8	35	33
M	30.99	151	26.1	88	211
N	41.13	157	24.2	102	11

Note, please, that we included in Dataset_2 new software metrics for independent variables X5 (software Size in KLOC), X6 (#Requirement document Pages) and X7 (#Design documents Pages), as well, old X4 (#Test cases) in order to ivestigate their influence on Testing Time Effort i.e.Y [Day] estimation model for all 14 projects presented in Table II. All regression model candidates are expressed with equation (9) and (10) formed by the Dataset_2.

$$Y[Day] = 3.3448 + 0.4341 * X4 - 1.633 * X5 + 0.1852 * X6 - 0.1261 * X7 \quad (9)$$

$$Y[Day] = 0.4341 * X4 - 0.1261 * X7 \quad (10)$$

We did small investigation to build TTE estimation model based on Work breakdown structure – WBS method i.e. WBS using software artifacts: documentation and program SLOC produced in SDLC and STLC.

We propose for software artifacts: #Requir. Pages, #Design Pages and estimated [1,2,4,7] application Size [KLOC], according to good estimation accuracy of the regression model based on independent variables X4, X5 (Size [KLOC]), and X6 (#Requirement document Pages) and X7 (#Design documents Pages) in Dataset_2. Using mentioned software artefacts we build new test effort estimation model:

$$Y[Day] = \sum_{P=1}^{N} TTE(Phase\ \ TTA) \quad (11)$$

Where, N=6 phases, TTE(Phase TTA) - is average (norm, standard) effort rate to test developed (produced) software artifacts (TTA -Testing Time of Artifacts) in phase P=1, 2, 3…6. For average (norm, standard) testing time rate we used data from literature survey done by Wagner [13].

For development Phase:

P=1, average rate to test Requirements is 8 Pages/Day, and TTE(Req.) = #Req. Pages / 8 ;

P=2, average rate to test Design documents is 30 Pages/Day, and TTE(Des.) = #Des. Pages / 30;

P=3, average rate to test program code is 600 LOC/Day, and TTE(Size[KLOC]) = 0.6 Size(KLOC);

P=4, average rate to test program module (UT - Unit test) is 309.3 LOC/Day, and TTE(UT) = 0.3093 Size(KLOC);

P=5, average rate to test software application during integration phase (IT) is 185.6 LOC/Day, and TTE(IT) = 0. 1856 Size(KLOC);

P=6, average rate to test software system (ST) is 154.6 LOC/Day, and TTE(IT) = 0. 1546 Size(KLOC);

Finally, practical WBS based TTE model is expressed in a form:

$$Y[Day] = \frac{\#\operatorname{Re} q.Pages}{8} + \frac{\#Des.Pages}{30} + 1.2495 Size(KLOC) \quad (12)$$

Table III provides an overview of the accuracy of 7 proposed testing effort estimation models. In the first column coded name TTEMn Di_Vk (n is model number, i=1 or 2, k=1 or 2) for TTE model is given, where D1 or D2 means that TTE model is designed using Dataset_1 or Dataset_2 and V1 or V2 means which dataset (Dataset_1 or Dataset_2) is used for model verification. In the second column, a list of independent variables used in TTE model is given. In the third column we provided equation number for TTE calculation, and in 4, 5 and 6 column, data for R^2, MRE and SD_{MRE} are presented.

One can observe from this table that TTEM3 D1_V1, TTEM6 D2_V2, and TTEM7 D2_V2 provides best estimation accuracy: MMRE and $SD_{MRE} < 50\%$.

TABLE III. TESTING TIME EFFORT (TTE) ESTIMATION MODEL ACCURACY COMPERISON

Model Name	Independ. variables included in model	Eq. No.	R^2	*MMRE [%]*	*SD_{MRE} [%]*
TTEM1 D1_V1	X1,X2,X3,X4	(5)	89.6	54.9	45.3
TTEM2 D1_V1	X1,X2,X3,X4, X1X2,X1X3, X1X4,X2X3, X3X4	(6)	78.8	63.7	94.7
TTEM3 D1_V1	**X4**	**(7)**	**74.9**	**52.2**	**33.1**
TTEM3 D1_V2	X4	(7)	74.9	240	164
TTEM4 D2_V2	X4	(8)	64.5	85.4	67
TTEM4 D2_V1	X4	(8)	64.5	65.7	22
TTEM5 D1_V1	X4,X5,X6,X7	(9)	64.5	63.1	51.2
TTEM6 D1_V1	**X7,X4**	**(10)**	**64.5**	**38**	**25**
TTEM7 D1_V1	**X5,X6,X7**	**(12)**	**-**	**47.1**	**44.8**

I. CONCLUSIONS AND NEXT STEPS

It has been clearly demonstrated that regression analysis has been successfully applied to formulate a effort prediction model for software testing process. By using statistical approach such as regression analysis, the research can justify the reasons and significance of metrics: Number of Use cases, Tester Years of experience, Developer Years of experience, Number of Test cases, software Size in KLOC, #Requirement document Pages and #Design documents Pages, from requirement, design and coding phase in predicting Testing Time Effort.

A number of multiple regression models were developed with various combinations and transformations of the independent variables. The models were then analyzed for their ability to accurately predict the dependent variable.

In carrying out the research, the activities were subjected to several limitations. First, the research only produced two general models based on: Use of Historical Data + Parametric model design and Work breakdown structure –

WBS method, due to limited number of data points. Second, data collected is only limited to software development projects in which their metrics are rigorously collected and tracked.

Projects that were not involved in metrics collection are no part of the data collected. Third limitation is that this research only focuses on V-shaped development model since that is the process model being adopted by the organization selected for this research. Fourth, data sets used in this research is a mix of metrics from web-based and component-based software. Therefore, findings of the research are the final result of using metrics from both software types.

More variants of the model could be developed by improving the model to predict test effort of projects with high number of intensive test activities (more than 6) including: non-functional test activities such as security testing, usability testings and performance testing. To achieve this, related metrics affecting these nonfunctional testing must be well defined, collected and tracked. It is hoped that the outcome of this research has been able to contribute and expand existing knowledge in software engineering estimation domain, particularly in the area of software testing, software quality management and software process planing. With the continuous effort in improving such prediction, more high quality software product can be developed in the future.

Acknowledgements

This work has been done within the project 'Optimal Software Quality Management Framework', supported in part by the Ministry of Science and Technological Development of the Republic of Serbia under Project No.TR-35026.

References

[1] Jensen. R, Putnam. L, Roetzheim. W, Software Estimating Models: Three Viewpoints, *CrossTalk*, pp.23-29, February 2006.

[2] Jones. C, "Software Assessments, Benchmarks, and Best Practices", Addison-Wesley Professional: Boston, MA. 2nd ed. McGraw-Hill, New York, 2000.

[3] Suffian. M, Suhaimi. I, A Prediction Model for Functional Defects in System Testing using Six Sigma, *ARPN Journal of Systems and Software*, Volume 1 No. 6, September 2011. pp. 219-224.

[4] Boehm BW, Horowitz E, Madachy R, Reifer D, Clark BK, Steece B, Brown AW, Chulani S, Abts C. 2000. Software Cost Estimation with COCOMO II. Prentice Hall PTR: Upper Saddle River, NJ.

[5] Chulani.S, Boehm.B, Modeling Software Defect Introduction Removal: COQUALMO (Constructive QUALity MOdel), USC-CSE-99-510, The Center for software Engineering, University of Southern California, Los Angeles, CA, 1999.

[6] Yooichi. Y, Mitsuhiko. K, Software Cost and Quality Analysis by Statistical Approaches, The 20th International Conference on Software Engineering , Kyoto, Japan ,April 19 - 25, 1998 .

[7] S. H. Kan, "Metrics and Models in Software Quality Engineering," Second Edition, Addison-Wesley, 2003.

[8] Lj. Lazić and N. Mastorakis, "Cost Effective Software Test Metrics," WSEAS TRANSACTIONS on COMPUTERS, Issue 6, Vol. 7, no 6, 2008, pp. 599-619.

[9] Srivastava. R, Kumar. S, Singh. P, Raghurama. G, Software Testing Effort: An Assessment Through Fuzzy Criteria Approach, Journal of Uncertain Systems, Vol.5, No.3, pp.183-201, 2011

[10] Glass, R.L., Software Testing and Industry Needs, *IEEE Software*, 2006.

[11] Sherriff, M., Boehm, B. W., Williams, L., and Nagappan, N., An Empirical Process for Building and Validating Software Engineering Parametric Models, North Carolina State Univeristy CSC-TR-2005-45, October 19 2005.

[12] Menzies, T., Chen,Z., Hihn, J., Lum, K., Selecting best practices for effort estimation", IEEE Trans. on Soft. Eng. 32(11), 2006.

[13] Wagner. S, A literature survey of the quality economics of defect-detection techniques, In Proc. 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE '06), 2006.