International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 7 (2015) pp. 18529-18550
© Research India Publications
http://www.ripublication.com

A Method to Effort Estimation for XP Projects in Clients Perspective

E.Karunakaran and Dr.N.Sreenath

Department of CSE,
Pondicherry Engineering College, Pillaichavady, Puducherry-605014, India;
ekaruna@pec.edu

Abstract:

All engineering fields have definite methods to calculate the cost of the product whereas in software engineering although many methods are exists, but no two methods will give same value neither for the size of the product nor for the effort and cost of the product. Also, most of the existing methods and models to find the cost of a project are based on previous experience or using the history of the project. But using past experience cannot predict the future works due to the complexity as well as the new projects may not match with the earlier projects. So it is a research problem to get an optimal solution satisfying the objective function. The COCOMO model can help to find the effort of the project based on the previously finished project data for the traditional projects by multiplying the effort multiplier with the number of lines as size of the project. Nowadays the project development methodology has been changing to overcome the problems of project failures, to improve the efficiency, and makes the development methodology much easier. Agile software development methods have various forms like e**X**treme **P**rogramming (XP) (Kent Beck -1999), SCRUM (Ken Schwaber and Mike Beedle – 2001) etc is introduced since 1999. The aim is to find the optimal size, effort and cost on both client and stack holder perspective for XP projects. The transparency is required for the client in estimation of the project. Based on the previous experience a method is proposed which is easy and accurate in sizing, effort estimation of the project. The sizing is done by a modified COSMIC full function points concepts. The effort estimation is the product of size and effort multipliers of the project, where effort multiplier are found using the finished XP projects of a leading software company from Chennai, India who were practicing XP methodology since 1999. The effort multiplier values are tuned by training the twenty finished projects data. The results are validated by using MMRE and different regression methods. These effort multipliers are applied to five different new projects (these projects are already finished and reserved for validation). These results of the projects are evaluated using different regression methods. Finally a tool is developed which takes the finished project as input and produce the output of the size, effort and cost of the project, which is more transparent and trust worthy to the client.

Key Words: Estimation, Effort Estimation, Sizing the software, COSMIC full function, Extreme Programming, Software Measurement, eXtreme software size Unit.

NOMENCLATURE

COSMIC	– Common Software Measurement International Consortium
XP	– e**X**treme **P**rogramming
COCOMO	– **CO**nstructive **CO**st **MO**del
CMSE	– Constrained Minimum Sum of Squared Error
CMAE	– Constrained Minimum Sum of Absolute Error
CMRE	– Constrained Minimum Sum of Relative Error
XU	– eXtreme software sizing Unit
F_i	– Fields in user Interface
F_t	– Fields in Table
F_c	– Fields in Coding
W_i	– Weight to create Interface Field
W_t	– Weight to create Table Field
W_c	– Weight to create Coding Field
MIS	– Management Information System
TCP	– Telecommunication Project
MMRE	– Mean Magnitude Relative Error
PRED	– Prediction
PM	– Person Month
KSLOC	– Source Lines of Codes in Kilo bytes.

1. Introduction

Most of the engineering fields have definite method for finding the size of the product and fixing the cost. In civil engineering, the sizing and cost estimation is almost easily understandable by the consumer, so that they can purchase the building with various size and rate according to their budget. The rate per square feet varies like Rs.1300, 1400, 1700, 2000 and 2500 based on the quality of work. For most of the products, workmanship takes 30% and quality of materials takes 70% of the cost. Whereas in software development, more than 70% takes workmanship (programmers skill) and less than 30% takes material cost i.e., the software, tools and hardware to develop the project. In civil, people compromise the quality based on their budget but in software development always prefers only quality deliverables, otherwise the software become unusable. Here the quality indicates bugs free, secured, robustness, understandability, reusability, extendability, modifiability, adaptability, nice looking graphic user interface, good database design, various output reports including statistical and graphical display etc. This quality in turn depends on the design and development skills of the programmer.

A software development methodology is a framework that is used to plan and control the process of developing. The existing common methodologies include waterfall, prototyping, iterative & incremental development, spiral development, rapid application development, and extreme programming. To reduce the failure and improve the quality and efficiency of the product, the software development methodology has been changing since the year 1999. Agile software development methods has various forms like Extreme Programming (Kent Beck -1999), Feature Driven Development (Jeff De Luca – 1999), Scrum (Ken Schwaber and Mike Beedle – 2001), [1] Agile implementation of RUP (Carig Larmann – 2002), Crystal – as family of agile methodologies (Alistair Cockburn – 2004) and so on. eXtreme Programming (XP) software development methods are nowadays wide spread and also accepted one. The use of Extreme programming practices emerged as a discipline because of the simplicity, error free, communication flow, focus on programming and unit testing. The actual development consists of requirement gathering, estimation, iteration planning, standup meeting and release plan.

Measurements in the physical world can be categorized in two ways namely a direct measure (for example: the length of bolt) and indirect measurement (for example: the quality of the bolts). Software metrics can be categorized similarly. Direct measures of the software engineering process include cost and effort applied namely lines of code (LOC) produced, execution speed, memory size and defects reported over a period of time. Indirect measures of the product include functionality, quality, complexity, efficiency, reliability, maintainability and many other abilities. The success or failure of contract is determined by the cost estimation’s deliverables such as size, effort, cost and schedule of delivery. Inaccurate and unrealistic cost estimation may lead to loss or rejection of the projects. So, accurate size of the project, effort to develop the project and cost of the project and scheduling should be precise and transparent to the customer. Many models are developed to find the size, effort and cost estimation for development of the project. In the proposed method, the sizing is done by a modified COSMIC full function point. The parameters are fields of user interface, fields of table and the memory variables in the coding. Effort estimation is done based on the research results of the past XP projects data. For this purpose, the data of 25 projects already finished using the XP methodology are considered, of which the data of 20 projects are used for creating the proposed model and the remaining 5 projects data are used for validation. The rest of the paper is arranged as follows. Section 2 discusses about the background details. Section 3 gives an overview about regression techniques. Section 4 and 5 describe the proposed work, research methods and validation. Results and comparison technique is discussed in Section 6.

2. BACKGROUND

2.1 SCRUM :

[2] Scrum assigns work to an entire team, not to an individual. Instead of a manager, estimating time on behalf of other individuals and assigning tasks based on conjecture, team members in Scrum use effort and degree of difficulty to estimate their own work. Teams often call estimation as "planning poker". This is a major break from waterfall which emphasis on collective effort. In the Sprint Planning Meeting, the team sits down to estimate its effort for the stories in the backlog. The Product Owner needs these estimates, so that he or she is empowered to effectively prioritize items in the backlog and, as a result, forecast releases based on velocity.

2.2 SLIM :

SLIM Estimate helps to estimate the cost, time, and effort required to satisfy a given set of system requirements and determine the best strategy for designing and implementing the software or systems project. In addition to software cost estimation, this powerful systems and software project estimation tool provides a high level of configurability to accommodate the different design processes being used by developers today.

2.3 COCOMO:

COnstructive **CO**st **MO**del II (COCOMO II) is a model that allows one to estimate the cost, effort, and schedule when planning a new software development activity. The original COCOMO model [3] was first published by Dr. Barry Boehm in 1981, and reflected the software development practices of the day. COCOMO II is an effective method for traditional projects and it requires as many as 22 parameters to be given by the evaluator. The formulae for finding the effort estimation is give by

$$PM = A * SIZE^{B} * \prod_{i=1}^{p} EM_{i} \quad \ldots (2.1)$$

Where
PM = effort estimation in person months
A = multiplicative constant
SIZE = estimated size of the software, measured in KSLOC
B = scale factors
EM = effort multipliers

In COCOMO II, B is defined as a function of scale factors, in form of

$$B = b_{0} + \sum_{i=1}^{5} b_{i} SF_{i} \quad \ldots (2.2)$$

In COCOMO II [4] uses 22 (5-scale factors and 17-effort multipliers) in its Post-Architecture model which is given in the Table: 2.1.

Table 2.1 - COCOMO II cost driver values

Cost Driver	Very Low	Low	Normal	High	Very High	Extra High
	6.20	4.96	3.72			0.00
PREC	5.0	4.05	3.04	2.48	1.24	0.00
FLEX	7.07	5.65	4.24	2.03	1.01	0.00
RESL	5.48	4.38	3.29	2.83	1.41	0.00
TEAM	7.80	6.24	4.68	2.19	1.10	0.00
PMAT	0.82	0.92	1.00	3.12	1.56	
RELY		0.90	1.00	1.10	1.26	
DATA	0.73	0.87	1.00	1.14	1.28	1.74
CPLX		0.95	1.00	1.17	1.34	1.24
RUSE	0.81	0.91	1.00	1.07	1.15	
DOCU			1.00	1.11	1.23	1.63
TIME			1.00	1.11	1.29	1.46
STOR		0.87	1.00	1.05	1.17	
PVOL	1.42	1.19	1.00	1.15	1.30	
ACAP	1.34	1.15	1.00	0.85	0.71	
PCAP	1.29	1.12	1.00	0.85	0.76	
PCON	1.22	1.10	1.00	0.90	0.81	
AEXP	1.19	1.09	1.00	0.88	0.81	
PEXP	1.20	1.09	1.00	0.91	0.85	
LTEX	1.17	1.09	1.00	0.91	0.84	
TOOL	1.22	1.09	1.00	0.90	0.78	1.80
SITE	1.43	1.14	1.00	0.93	0.86	
SCED				1.00	1.00	

Where PREC, FLEX, RESL, TEAM, PMAT, RELY, DATA, CPLX, RUSE, DOCU, TIME, STOR, PVOL TOOL, SITE, SCED, ACAP, PCAP, PON, APEX, PLEXM, and LTEX, are scale factor and effort multiplier. The same are abbreviated as Precedentedness, Development Flexibility, Architecture / Risk Resolution, Team Cohesion, Process Maturity, Required Software Reliability, Database Size, Product Complexity, Development for Reusability, Documentation Match of Life-Cycle needs, Execution Time Constraint, Main Storage Constraint, Platform Volatility, Use of Software Tools, Multisite Development, Required Development Schedule, Analyst Capability, Programmer Capability, Personnel Continuity, Application Experience, Platform Experience, and Language and Tool Experience.

It is very difficult to give the parameter more correctly to the new project for estimation. To give values for the 22 parameters in advance and that may lead to wrong estimation even a few parameters input fed wrongly. These parameters are company's environment dependent so it varies from one company to another. Therefore, it clearly indicates the effort estimated is not accurate and hence the client cannot trust this methodology. In addition, in terms of client perspective it is very difficult to provide more parameters that too about the company. Therefore, only the important factors, which affect the software development, are taken as multipliers. Moreover, it is clear that the multipliers value of the past finished projects is contained among these important factors alone. Even if the estimation accuracy is slightly less than best existing methods, this method is preferred because of its simplicity and the ease of evaluation by the client himself.

2.4 COSMIC FULL FUNCTION POINTS (CFFP):

The COSMIC [5] Method defines a standardized measure of software Functional Size expressed in CFFU units. The measurement is carried out by mapping the Function User Requirement (FUR) of the software onto the COSMIC Generic Software Model (shown in Figure 1). The purpose of the measurement and scope of the software to be measured based on the level of decomposition and level of granularity of the software. As can be seen in Fig.1, there are four different data movement types as give below:
E-Entry type move data across the boundary from the user to the functional process.
X-eXit type move data across the boundary to the user.
R-Read type move data from persistent storage in the functional process.
W-Write type move data from the functional process to persistent storage.

The main disadvantages of this method is some user interface have less than 10 fields for Enter or eXit whereas some other user interface may have more than 50 fields. Similarly, many reports involing the Read and Write using same data will give an increased size of the project. Therefore, it is proposed to introduce a modification in this method to find the size of the project, by overcoming the above disadvantage.

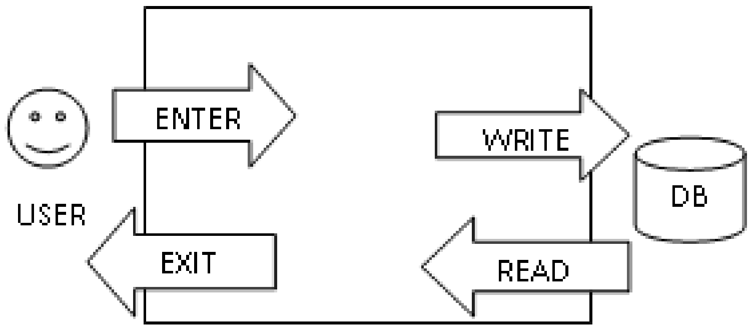


Fig 2.1: Functional Process

2.5 OTHER METHODS:

[6] Research work on the quality of the documentation using a functional size method is recent. In this work, documentation quality rating scale a, b, c, d, e is fixed based on facts, some of which can be listed as: The presence or absence of a data model. The presence or absence of information to identify the data movements (entry, read, write, exit). The presence or absence of documentation enabling identification of each functional process. In this process, three iterations are made for documentation quality

and the quality rating is never decreasing from iteration to iteration. In the third iteration, the documentation quality of a functional process is always equal to or higher than the second and first iteration. At the end of third iteration only the highest rating like a or b will exist.

[7] Karel Dejaeger noted that by selecting a subset of highly predictive attributes such as project size, development, and environment related attributes, typically a significant increase in estimation accuracy might obtain. [8] Jose Javier Dolado states that the component based method behaves reasonably, although not as well as expected for global methods such as Mark II function points for software size prediction. [9] Vu Nguyen and Bert Steece in their research results noted that the regression model that minimize the sum of relative errors and imposes non-negative coefficients is a favourable technique for calibrating the COCOMO model parameters. [10] Jongmoon Baik used Bayesian calibration with the three-dimensional tool rating scales namely TCOV (Completeness of Activity Coverage), TINT (Degree of Tool Integration), and TMAT (Tool Maturity and User Support) shows better prediction accuracy than that of COCOMO II.2000 Bayesian calibration with just one-dimensional tool rating scale over 15 project data points.

The questions examined are the classic methods of the procedure like full function points or COCOMO is applicable. Using accumulated experience, a method is proposed which uses a modified COSMIC Full Function Points to size the software and the size unit named as eXtreme size Unit (XU). The concept of COCOMO model is used for finding the multiplicative factors and the same is multiplied with XU to get the effort estimation. For effort multipliers, the important four parameters alone are taken into account, which contains all the driving factors to develop a project. These parameter values have been found using the twenty developed XP project results from a leading software company situated at Chennai which developing their projects using Extreme Programming concepts since 1999. It is also found that the parameter values are accurately working to this company for their new projects as well. This paper proposes a robust regression technique to find the exact value of the parameters, which will play a major role in fixing the effort estimation. In this paper some of the common techniques, such as Minimum Sum of Squared Error (MSE), Minimum Sum of Absolute Errors (MAE), and Minimum Sum of Relative Error (MRE) are used to fix up the errors. Also compared the performance of this approach with other regression techniques like Ridge, Lasso, Least Square using cross validation and found this method is better than others are.

3. Techniques:

In statistics, regression analysis is a statistical process for estimating the relationships among variables. It includes many techniques for modelling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables.

Regression analysis is widely used for prediction and forecasting, where its uses has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are closely related to the dependent variable, and to explore the forms of these relationships. More specifically, regression analysis helps one to understand how the typical value of the dependent variable changes when any one of the independent variables is varied.

3.1 Model Accuracy Measures:
MMRE and PRED [11] are the most widely used metrics for evaluation of the accuracy of cost estimation models. These metrics are calculated based on a number of actual observed and estimates generated by the model. They are derived from the basic magnitude of the relative error MRE, which is defined as:

$$MRE_i = \left| \frac{y_i - \bar{y}_i}{y_i} \right| \ldots (3.1)$$

where y_i and $\bar{y}_i$ are the actual and the etimated values.

Since y_i is the log transformed, MRE_i is calculated in the Eq.3.2.

$$MRE_i = \left| 1 - e^{(y_i - \bar{y}_i)} \right| \ldots (3.2)$$

The mean of MRE of N stories is defined as

$$MMRE_i = \frac{1}{N} \sum_{i=1}^{N} MRE_i \ \ldots \ (3.3)$$

As every estimate is included in calculating MMRE, extreme values of MRE can significantly affect MMRE. To handle this problem, another important criterion used for model evaluation is PRED. PRED(p) is defined as the percentage of estimates where MRE is not greater than p, that is PRED(p) = k/n, where k is the number of estimates with MRE falling within p, and n is the total number of estimates. Conte et al. [15] considered MMRE≤0.25 as an acceptable level for effort prediction models. PRED(x) considers the average fraction of the MRE's off by not more than x as defined by

$$PRED(x) = \frac{1}{N} \sum_{i=1}^{N} \begin{cases} 1 & if\ MRE_i \le x \\ 0 & otherwise \end{cases} \ \ldots \ (3.4)$$

3.2 Constrained Multiple Regression Measure:
For more accuracy, three constrained regression models are considered. These models are based on Minimum Sum of Squared Error (MSE), Minimum Sum of Absolute Errors (MAE), Minimum Sum of Relative Error (MRE) techniques that have been applied to build or evaluate the cost estimation model. These models are

CMSE (Constrained Minimum Sum of Squared Error) :

$$Minimize \sum_{i=1}^{N} (y_i - \bar{y}_i)^2 \ldots (3.5)$$

subject to $MRE_i \leq c$

CMAE (Constrained Minimum Sum of Absolute Error) :

$$Minimize \sum_{i=1}^{N} |y_i - \bar{y}_i| \ldots (3.6)$$

subject to $MRE_i \leq c$

CMRE (Constrained Minimum Sum of Relative Error):

$$Minimize \sum_{i=1}^{N} \left| \frac{y_i - \bar{y}_i}{y_i} \right| \ldots (3.7)$$

subject to $MRE_i \leq c$ where the tuning parameter $c \geq 0$, which controls the upper bound of MRE for each estimate.

3.3 Model Validation:
The model can be accepted only if the predicted value for new projects comes closer to the actual value. The percentage of errors in the model can be calculated using the new project. It will be a time consuming job to check with the new project and also it is practically difficult. Hence, the 5 finished projects (reserved out of the 25 projects) are used for cross validating their results of the proposed model. Many cross-validations are existing in the field, the most common and simple holdout strategy is K-fold error validation. In this method, the dataset is splitted into two sets namely training set and testing set. The training set is used to fit the model and test set is used to predict the error. By randomly dividing the set into training & testing and repeat for many times to make the model more accurate. The variance of the estimation is reduced by increasing the number of repetition. And also to compare our model with other regression model like Stepwise Subset Selection, Ridge and Lasso are taken and calculation procedure of these methods are explained below.

Stepwise Subset Selection: Stepwise Subset Selection is used to predict parameter that is most relevant to the response value. Even only four variables are used with each variable having 3 different values the number of combination for selecting a subset is $3^4 = 81$. Among the 81 sets, the best one is chosen by comparison. The step involved in selection of the parameter is given in the following procedure.

Step: 1 Initialize Best Estimated Value (BEV) to zero and set first set as best set.
Step: 2 Select one set of parameters from the set and find the estimated value (EV) for this set of parameter using the multipliers and size.
Step: 3 If value (EV) is less than BEV then store EV to BEV and make that set as best set.
Step: 4 Repeat steps: 2 & step: 3 till all the set are over.
Step: 5 Now the best set and its value (BEV) are available.
Ridge: Ridge regression, proposed by Hoerl and Kennard address the issue of collinearity. The model estimates coefficients by minimizing

$$S(\overline{\beta}) = \sum_{i=1}^{N} \left(y_i - \beta_o - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 \quad \text{subject to} \sum_{j=1}^{p} \beta_j^2 \le s \ldots \ (3.8)$$

Lasso:
Lasso regression, proposed by Tibshirani provides a method to reduce the bias-variance. Lassso estimates the regression coefficients β = (β1, β2, …, βn) by minimizing

$$S(\overline{\beta}) = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 \ \text{subject to} \sum_{j=1}^{p} \beta_j^2 \le t \ldots \ (3.9)$$

4. Methods

4.1 Dataset :

The main aim of this work is to find a method, which should be easy to understand by the client to find the size, the effort in mandays and cost of the project. Also it is intended to create a tool which should get minimum inputs to find the above three estimates namely size, effort estimation and cost estimation. Extreme Programming is selected, as the method of development methodology due to smaller modules (i.e. stories) and less risk in development, testing, integration etc. For sizing the project, modified COSMIC Full Function Points is used which is more suitable for XP projects as well as in the clients perspective it is easy to understand. Effort multipliers are restricted to only four variables which are the key factors for development of the projects. These effort multipliers values are found using the twenty five finished XP projects from a reputed software company situated at Chennai, India which applies XP methodologies since 1999.

4.2 Sizing the software:

The size is found by the using Eq.4.1.

$$XU_k = (Fi_k * Wi + Ft_k * Wt + Fm_k * Wm)/5 \ldots(4.1)$$

Where
XU_k - eXtreme software size Unit of the kth story,

k = 1 to n, n is the number of stories in the project.
Fi - Number of fields in the interface of the story,
Ft - Number of fields in the table created by the story
Fm - Number of variables handled by the program of the story and
Wi=0.562, Wt=0.325, Wm=1.625 are the weight of the Fi, Ft, Fm respectively averaged value given by 10 expert member of the different software companies for creation of the fields.

The following five different projects out of twenty five projects are shown as example to measure the size of the software and the result are given in the Tables: 4.1 to 4.5.

TABLE 4.1 Project-1: L-Sales

Functional Process	XU
Estimation	22
Sales	32
Credit	12
Cash	48
File	67
User	15
Invoice	25
Rate	87
Total	308

TABLE 4.2 Project-2: Direct Communication

Functional Process	XU
OPCGROUP	58
EnumOPCItemAttributes	6
OPCServer Object	24
OPCBrowser Object	14
OPCGroups Object	09
OPCGroup Object	08
OPCItems Object	15
OPCItem Object	4
IOPCCommon	11
IOPCEventServer	41
IOPCConnectionPointContainer	4
OPCHDAServer	21
OPCHDABrower Object	12
OPCHDAItems Object	68
Total	295

TABLE 4. 3 Project-3 Ex-Cargo

Functional Process	XU
Shipment Details	25
Consignor / Consignee Details	14
Inbound/Outbound	19
Report-Finance	18
Report-Planning	07
Booking Details	10
Shipment Status	08
Planning by Truck	08
Status Report	18
Shipment Sold	16
Cosignee Reverse	10
Account Details	10
Total	163

TABLE 4.4 Project-4 Mobile Park

Functional Process	XU
MobilePark Owner	14
City Entry	14
Zone Entry	15
Policy Entry	15
Loading Credits	15
Effected Parking	4
Client Account	4
Parking Policy Database	4
Town Parking Tax	4
Credit Purchase	4
Upload	4
Download	4
Total	101

TABLE 4.5 Project-4.5: Forging

Functional Process	XU
Receipts	14
Issues	14
Stock	09
Ledge & Opening Balance	18
RM details	15
Queries – Ledger details	33
Inventory Maintenance	8
RM, TR Entry Details	10
TDC & TC Entry	12
Forging Entry & Backup	15
Queries – Forging details	18
Transfer Heatcode, ICIN, DN Modification & ReIndexing	15
IOPCEventServer	40
Reports	45
Total	266

4.3 Finding the Value of the Effort Parameters:
For effort estimation, only four factors are considered which influence the complete project development process namely **Programmer's skill, Software type, Software & Tool Used, and Database Used.** The other factors also influence the software development but those factors are contained in the 4 factors considered and also have only normal values. Apart from this, the paper is focusing on the **client's perspective;** it means that the client need not to bother about company's parameters which affects the effort of the project development. A company can finish the project at the earliest by using more number of expert programmers and some company may use inexperienced programmers which will take more time than the required one. So, it is required to find the parameter values which should give the effort estimation in terms of actual mandays to deliver the valuable product which includes design, code, integrate, test and documentation by a set of skilled programmers.

4.3.1 Method followed to find the value of the Effort Multipliers:
The effort multipliers taken are Skill (PS), Software Type (ST), Software and tool Used (SU) and Database Used (DU). The values of PS are Beginner, Skilled & Expert are named as x (x1, x2, x3). The values of ST are MIS, Webbased, Tele Communication are made as y (y1, y2, y3). The values of SU are VB/PHP, Java/C#, TC/C++/VC++ are named as z (z1, x2, z3) and the values of DU are MS-Access, My-SQL, Oracle are named as w (w1, w2, w3). Each of the project is belongs to any one of the combination ($3^4 = 81$) values containing different x, y, z, w values.

All the training sets (twenty projects) are splitted into N stories and each actual values are averaged to 5 XUs. Now out of N stories L stories are removed randomly and mean of remaining stories for each variables are found as follows:

Step: 1 Remove L stories from N stories randomly.
Step: 2 Each of 3^4= 81 sets totals & counters variables are initialized to 0.
Step: 3 Take a story from the available N-L stories.
Step: 4 Add the value to the correct set and increase the counter of the respective set
Step: 5 Repeat 3 & 4 till all the N-L stories are over.
Step: 6 Find average of each set value using total and counter.
Step: 7 Once again 3^4 set's total & counter variables are initialized to 0.
Step: 8 Take a story from the available N-L stories one by one and estimate.
Step: 9 If actual effort value > (estimated average value *1.2) then skip (go to step: 7)
Step: 10 Add the value to the correct set and increase the counter respective set
Step: 11 Repeat 8 to 10 till all the N-L stories are over.
Step: 12 Find average of each set value using total and counter.
Step: 13 Remove L stories from N stories randomly and repeat Step 7 to step 12 for 10 times.

This gives us maximum of 3^4 (81) values with various combination of effort multipliers. By mapping to the concerned parameter and solving the same (using eq. 4.1) gives us the effort multiplier values. These values are purely based on the previous 20 projects. To reduce the errors of the project, the stories having extreme value are removed from the training set. Also comparative study using the data between different effort multipliers and within the effort multiplier is made to get tune the value fine tuned. Some samples for programmer's skill are furnished in the Tables 4.6 & 4.7 and the same shown in the Fig. 4.1 & 4.2. Using these values tuned effort multiplier values are derived between programmer's skill and software used and the same is tabulated in the Table 4.8.

TABLE 4.6 Man days requirement – Comparison between Expert and Skilled teams

Functional Process	Actual Size (XU)	Allotted Size (XU)		Actual Man Days (5 XU)	
		E-TEAM	S-TEAM	E-TEAM	S-TEAM
MobilePark Owner	14	7	7	7	10
City Entry	14	7	7	7	10
Zone Entry	15	8	7	8	9
Policy Entry	15	8	7	9	9
Loading Credits	15	8	7	8	9
Parking details *	20	10	10	11	14
Upload & Download	8	4	4	5	6
AVERAGE	14.42	7.42	7	7.86	9.57

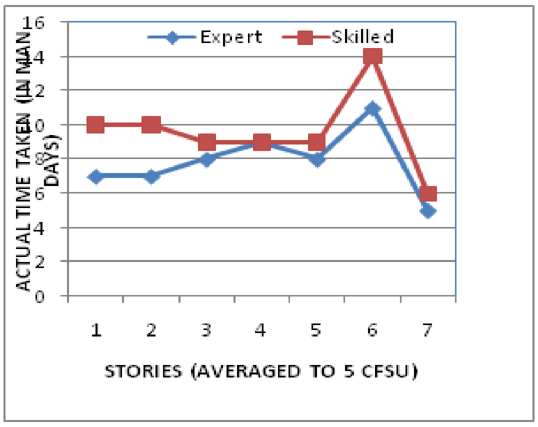
Project Name: Mobile Parking,
Layer: Tele. Communication
Software Used: Java

(* The stories Effected Parking, Client Account, Parking Policy Database, Town Parking Tax, Credit Purchase are merged and made as one story namely Parking details)

TABLE 4.7 Man days requirement – Comparison between Expert and Skilled teams

Functional Process	Actual Size (XU)	Allotted Size (XU)		Actual Man Days (5 XU)	
		E-TEAM	S-TEAM	E-TEAM	S-TEAM
Estimation	22	12	10	1.7	3.0
Sales	32	17	15	1.8	2.8
Credit	12	07	05	1.2	2.0
Cash	48	26	22	1.9	2.7
File	67	37	30	2.1	2.6
User	15	09	06	1.3	2.2
Invoice	25	15	10	1.4	2.0
Rate	87	47	40	1.6	2.5
AVERAGE	38.5	21.25	17.25	1.625	2.475

Project Name: LIPS,
Layer: MIS,
Software Used: VB6



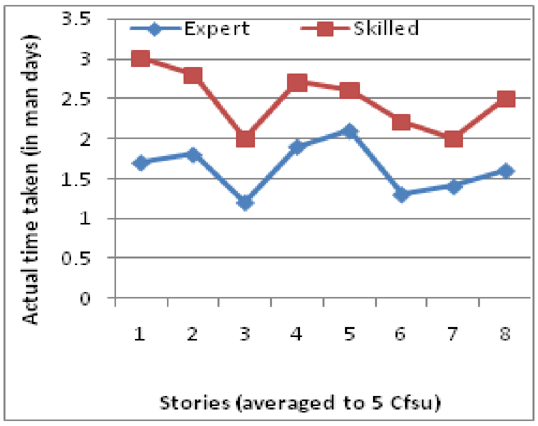


Fig. 4.1: Expert Vs Skilled (TC) **Fig.4.2: Expert Vs Skilled (MIS)**

TABLE 4.8 Effort multiplier values (Software Type Vs Programmer's Skill)

Developer \ Software Used	**Beginner**	**Skilled**	**Expert**
MIS	**2.0**	**1.1**	**0.6**
WEB	**3.2**	**1.5**	**1.0**
TELE	**3.5**	**2.1**	**1.3**

As an example, the comparison within software type data is furnished in the

Table 4.9. Also a comparison between software type and software used are listed in the Table 4.10 and content of Table 4.9 and4.10 are shown in the Fig. 4.3 & 4.4.

TABLE: 4.9 Man days requirement – Comparison between Software Type

SOFTWARE	Actual Size (XU)		Actual Man Days (5 XU)		Actual Time Taken (in Man Days)	
	MIS	T.C.	MIS	T.C.	MIS	T.C.
STORY – 1	22	14	2.95	10.71	13	30
STORY - 2	37	14	2.43	10.00	18	28
STORY - 3	12	15	2.92	9.33	7	28
STORY - 4	48	15	2.81	8.67	27	26
STORY - 5	67	15	2.76	8.67	37	26
STORY - 6	15	20	4.33	9.00	13	36
STORY - 7	25	8	2.60	8.75	13	14
STORY - 8	87		2.76		48	
AVERAGE	38.5		3.37	9.30	22	26.86

Project Name: LIPS & MOBI LE PARK
Layer: MIS & T.C.
Software Used: VB6 & Java

TABLE: 4.10 Time taken – Comparison between different Software / Layers

SOFTWARE	Actual Size (XU)				Actual Time Taken (Man Days for 5 XU)			
	VB	JAVA	JAVA	C++	VB	JAVA	JAVA	C++
	MIS	MIS	T.C.	T.C.	MIS	MIS	T.C.	T.C.
STORY - 1	22	16	14	16	2.95	5.94	10.71	10.00
STORY - 2	37	27	14	15	2.43	5.19	10.00	9.33
STORY - 3	12	36	15	14	2.92	5.14	9.33	10.00
STORY - 4	48	38	15	14	2.81	4.74	8.67	9.29
STORY - 5	67	19	15	17	2.76	5.53	8.67	10.00
STORY - 6	15	32	20	10	4.33	5.00	9.00	11.00
STORY - 7	25	39	8	16	2.60	4.87	8.75	9.38
STORY - 8	87	50			2.76	4.00	9.31	9.80
AVERAGE	44.71	36.71	14.43	14.57	2.46	4.27	9.31	9.80

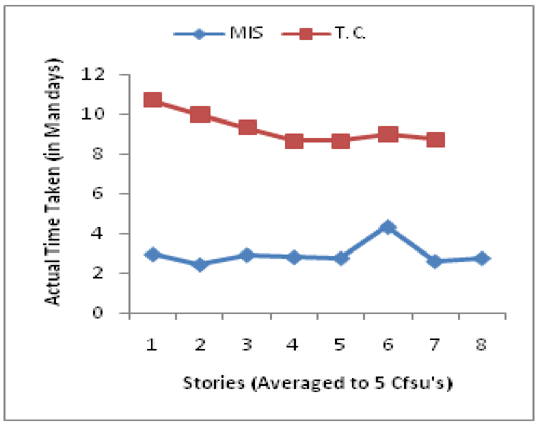


Fig. 4.3 MIS Vs Telecommunication

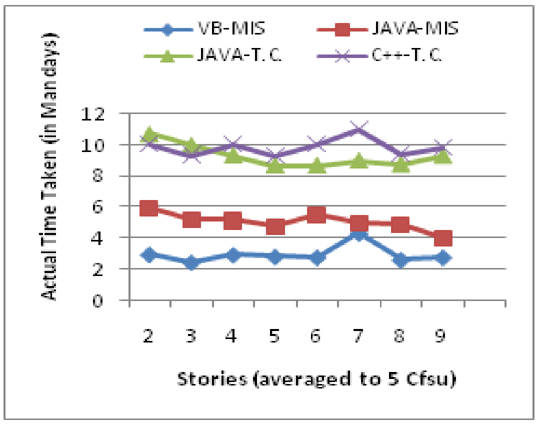


Fig. 4.4 MIS Vs Software Used

The derived effort multiplier values between software type & software used and between software type & database used are listed in the tables: 4.11 & 4.12 respectively.

TABLE 4.11 Effort multiplier values – (Software Type Vs **Software & tool Used)**

Software Used	VB	JAVA	C	PHP	VC++	.NET
MIS	**1.0**	**1.3**	**1.5**	**1.4**	**1.7**	**1.2**
Web	**1.2**	**1.5**	**2.1**	**1.4**	**2.2**	**1.7**
Tele Comm.	**2.2**	**1.9**	**2.0**	**1.7**	**2.0**	**1.9**

TABLE 4.12 Effort multiplier values – (Software Type Vs **Database Used)**

Database Used	Oracle	Access	MySql
MIS	1.0	0.75	0.9
Tele Comm.	1.3	1.4	1.2
Web	1.25	1.35	1.2

4.4 Effort Estimation:

Effort estimation is the product of size and the effort multipliers. The proposed method uses 4 multipliers namely $\beta_1, \beta_2, \beta_3,$ and extreme software size Unit (XU) of the project which contains a set of stories. Effort estimation is found using the Eq.4.2.

$$Effort\ Estimation = \sum_{i=1}^{N} \left(XU_i * \prod_{j=1}^{3} \beta_j\right)\ mandays \ldots (4.2)$$

Where XU – eXtreme software sixe Unit, β_1= PS – Programmer's Skill, $\beta 2$=SU – Software Used and β_3=DU – Database Used.

The effort estimation for five project which are reserved for validation are carried out using the Eq.4.2 and the results are listed in the Table 4.13 of the 3rd column. The algorithm for Effort Estimation is as follows:

Step: 1 Take the Project and store number of stories as N.
Step: 2 Find the XUi of the story for i = 1 to N.
Step: 3Using the result tables (4.8, 4.11 & 4.12) values and equation 4.2 find the Effort Estimation.
Step: 4 Repeat step: 1 to step: 3 for each project.

4.5 Validation Process

4.5.1 Error Measurement using MMRE

Using the Eq.3.1 the Magnitude Relative Error (MRE) is found and listed in the 4th column of the Table 4.13.

Table: 4.13 MRE of the new 5 Projects (in mandays)

Project	Actual Value	Effort Value	MRE
Project-1	375	345	0.08000
Project-2	958	901	0.059499
Project-3	657	590	0.101979
Project-4	905	910	0.00552
Project-5	465	445	0.043011
Mean	672	638.2	0.058002

The mean Magnitude Relative Error is calculated using the Eq.3.3 and the result is furnished below:

MMRE = 0.058002

4.5.2 Error Measurement using PRED (0.2)

The PRED (0.2) is found using the Eq.3.4 for the five projects and the results are furnished in the Table 4.14

Table: 4.14 PRED(0.20) of the new 5 Projects (in mandays)

Project	Actual Value	Effort Value	MRE
Project-1	325	345	0.061538
Project-2	928	850	0.084052
Project-3	627	520	0.170654
Project-4	905	910	0.005525
Project-5	430	395	0.081395
Mean	643	604	0.080633

4.8 Cross Validation Results

The constraint values are used to generate MMRE and PRED in the cross-validation procedure. Tables 4.15 and 4.16 furnish the performance measure of MMRE and PRED(0.20) obtained on each technique using ten-fold cross-validation. The mean, median and standard deviation obtained by Subset Selection, Ridge, and Lasso are almost same. Whereas CMRE and CMSE are having lesser standard deviation than all other methods. Therefore, CMRE and CMSE are better regression method to evaluate the cost estimation for XP projects.

Table: 4.15 MMRE for the techniques

Method	Mean	Median	Standard Deviation
Subset Selection	0.29	0.19	0.33
Ridge	0.28	0.18	0.29
Lasso	0.29	0.19	0.29
CMSE	0.28	0.18	0.28
CMAE	0.27	0.17	0.28
CMRE	0.23	0.16	0.25

Table: 4.16 PRED(0.20) for the techniques

Method	Mean	Median	Standard Deviation
Subset Selection	0.29	0.19	0.33
Ridge	0.28	0.18	0.29
Lasso	0.29	0.19	0.29
CMSE	0.28	0.18	0.28
CMAE	0.27	0.17	0.28
CMRE	0.23	0.16	0.26

5. RESULTS AND DISCUSSION

The results show that the VB requires half the time to complete the project of size 5 XUs than the time required for Java. Also it is very clear that the time required for Java in MIS is almost half of the time required in telecommunication system. Fig.4.1 and Fig. 4.2 clearly indicate Expert Programmers out performed over Skilled Programmers. The time effort for 1 XU for different software and different software type are estimated and furnished in the Table: 4.11 and it is evident that the VB software requires very less effort time to develop a Management Information System followed by Java. In case of Tele Communication project C++ and VC++ may be effective software language to optimize time effort. The effort value for different software, software type and databases have been calculated and furnished in the Tables: 4.8, 4.11 & 4.12. The effort values given in the table are based on the effort valued of the twenty projects used for the present study. The sizing of the software using the proposed method is much easy to understand so; the client will trust the method. Since the tool will calculate the size and effort estimation using the finished projects, the calculation becomes much simpler. The effort multipliers are calculated using twenty finished XP projects by removing the extremed valued stories gives reasonably accurate results. Also these values are tuned by training using those finished projects data. The validation results using 5 projects of different applications and lesser in errors. CMRE method is better method compared to all other methods like Ridge, Lasso, Subset Selection, and CMSE as it is evident from the Tables 4.15 & 4.16.

6. CONCLUSION AND FUTURE WORK

This paper has described finding the size of the project by means of stories using the factors namely user interface fields, table fields and coding memory variables which will be unique and same size whoever find the size. Also a tool is developed which will calculate the size, effort and cost of the project by giving the finished project as input data. This paper describes to find the effort estimation using minimum multiplicative factors and the determination of values of the factors by using the finished XP projects. The factors are first tuned to minimize the error. The XU and factors are validated using five different projects and found the error is minimal. Further the approach is compared with other regression techniques. By imposing the constraint on the regression function, problem associated like extreme sizing stories are removed to make the size more accurate in the client perspective. An advantage of this approach in addition to finding the actual size and efforts, companies can also fix the real effort need for a future project by computing the effort multipliers based on the finished projects. The company can find the difference in percentage of mandays between our data and their results obtained using this proposed method and the actual results obtained by them and use the same as a multiplier to find the actual mandays required by easily computing the mandays using the proposed method. So that they can know the effort required by their teams, using this tool and can add the percentage to the calculated one.

7. Acknowledgments

We would like to thank Dr. Vivekanandhan and Dr.A.V. Ravi Prakash for review and providing comments on the early versions of this paper.

8. References

[1] XiaoFeng. “The Combination of Agile and Lean in Software Development - An Experience Report Analysis,” IEEE Conference Publications. 2011, pp: 1-9.

[2] Shalaby.M, El-Kassas. “Applying Scrum Framework in the IT Service Support Domain.,” IEEE Conference Publications. 2011, pp.9-15.

[3] B.Boehm, C.Abts, A.W.Brown, S.Chulani, B.K.Clark, E.Horowitz, R.Madachy, D.Refer, and B.Steece. “Software Cost Estimation with COCOMO II,” Prentice Hall. 2000.
[4] Vu Nguyen, Bert Steece, Barry Boehm. “A Constrained Regression Technique for COCOMO Calibration”.
[5] COSMIC – Common Software Measurement International Consortium 2007. “The COSMIC Functional Size Measurement Method – version 3.0 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761:2003),” September 2007.
[6] Jean-Marc Desharnais, Computer Engineering Department, Bogazici University, Istanbul, Turkey. “Using the COSMIC Method to Evaluate the Quality of the Documentation of Agile User Stories,” Joint Conference of the 21st International Workshop on Software Measurement. 2011 pp. 269-272.
[7] Karel Dejaeger, Wouter Verbeke et. “Data Mining Techniques for Software Effort Estimation. A Comparative Study,” IEEE Transaction on Software Engineering. vol.38, no.2, March2012, pp:375-397.
[8] Jose Javier Dolado. “A Validation of the Component-Based Method for Software Size Estimation,” IEEE Transaction on Software Engineering. Vol..26, No.10, Oct.2000, pg.1006-1021,.
[9] Jongmoon Baik. “Disaggregating and Calibrating the CASE Tool Variable in COCOMO II,” IEEE Transactions on Software Engineering. November 2002, vol.28 n11. p.1009-1022.
[10] M.Jorgensen. “Experience with the Accuracy of Software Maintenance Task Effort Prediction Models,” IEEE Transaction on Software Engineering. Aug.1995, vol.21, no.8, pp.674-681.
[11] S. Chulani, B.Boehm, and B.Steece. “Bayesian analysis of empirical software engineering cost models,” IEEE Transaction on Software Engineering. July/August 1999, vol.25 n.4, pp: 573-583.
[12] M.Jorgensen, M.Shepperd. “A Systematic Review of Software Development Cost Estimation Studies,” IEEE Transactions on Software Engineering. January 2007, vol.33 no.1. p.33-53.
[13] Kent Beck, Marti Fowler. “Extreme Programming Explained: Embrace Change,” Addison-Wesley, 2000.
[14] Kent Beck et al. “Planning Extreme Programming ,” Addison-Wesley, 2000.
[15] Ron Jeffries, et al. “Extreme Programming Installed, “ Addison-Wesley, 2001.
[16] Paul Rodrigues, E.Karunakaran. “Extreme Programming on PWAP (ficticious) a wireless application,” Agile International Conference, Itally, 2001, pp. 73.
[17] Ekrem Kocaguneli, Jacky Keung, Ray Madachy. “Active Learning and Effort Estimation: Finding the Essential Content of Software Effort Estimation Data,” IEEE Transactions on Software Engineering, August 2013, vol.39 n8. p.1040-1053.
[18] C.G.Bai, Q.P. Hu, M.Xie, and S.H. Ng. “Software failure predictions based on a Markov Bayesian network model,” Journal of System Software, vol.74, no.3, pp.275-282, Feb.2005.

[19] X.Teng, H.Pham. “A new methodology for predicting software reliability in the random field environments,” IEEE Transaction on Software Engineering, Sep. 2006, vol.55, no.3, pp.458-468.

[20] Cheng-Gang Bai, Kai-Yuan Cai, Qing-Pei Hu, and Szu-Hui Ng. “On the Trend of Remaining Software Defect Estimation,” IEEE Transaction on Systems, Man, and Cybernetics – Part A : Systes and Humans, Sep. 2008, vol.38, no.5,pp.

[21] I.Pardoe. “An Introduction to Bootstrap Methods Using Arc.,” Technical Report 631, Univeristy Of Minnesota, St. Paul, Feb.2000.

[22] T.Bollinger. “Building Tech-Savvy Organizations,” IEEE software, Aug.2000, pp.73-75.

[23] A.Miller. “Subset Selection in Regression,” 2nd ed., Chapman Hall, 2002.

[24] C.Lokan. “What Should You Optimize When Building an Estimation Model,” Proc. 11th IEEE Int’l Software Metrics Symp. , 2005, pp.34.

[25] Xiao Xiao and Tadashi Dohi. “Wavelet Shrinkage Estimation for Non-Homogeneous Poisson Process Based Software Reliability Models,” IEEE Transaction on Reliability vol.62, no.1, March 2013.

[26] S.Basu and N.Ebrahimi. ”Bayesian software reliability models based on martingale processes. Technometrics,” 2003, vol.45, no.2 pp.150-158.

[27] AGandy, U.Jensen. “A non-parametric approach to software reliability,” Appl. Stochasti Models Business Ind. , 2004 vol.20, no.1, pp3-15.

[28] S.P. Wilson and F.J.Samaniego. “Nonparametric analysis of the order-statistic model in software reliability,” IEEE Transactions on Software Engineering, 2007, vol.33, no.3, pp 198-208,.

[29] K.Srinivasan and D.Fisher. “Machine Learning Approaches to Estimating Software Development Effort,” IEEE Transaction on Software Engineering. Feb.1995, vol.21, no.2, pp.126-137.

[30] T. Menzies, Z. Chen, J. Hihn, K. Lum. “Selecting Best Practices for Effort Estimation,” IEEE Transactions on Software Engineering. 2006, vol. 32, no.11.

[31] T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit. “A simulation study of the Model Evaluation Criterium MMRE,” IEEE Transactions on Software Engineering. 2003, Vol. 20, no. 11.

[32] M. Shepperd. “Evaluating Software Project Prediction Systems,” 11th IEEE International Software Metrics Symposium. 2005, Como, Italy.