DOI: 10.21917/ijsc.2013.0078

AN ENHANCED MODEL TO ESTIMATE EFFORT, PERFORMANCE AND COST OF THE SOFTWARE PROJECTS

M. Pauline[1], P. Aruna[2] and B. Shadaksharappa[3]
[1, 2]Department of Computer Science and Engineering, Annamalai University, India
Email: [1]pmariasundaram@yahoo.com and [2]arunapuvi@yahoo.co.in
[3]Department of Computer Science and Engineering, Shirdi Sai Engineering College, India
Email: bichagal@yahoo.com

Abstract
The Authors have proposed a model that first captures the fundamentals of software metrics in the phase 1 consisting of three primitive primary software engineering metrics; they are person-months (PM), function-points (FP), and lines of code (LOC). The phase 2 consists of the proposed function point which is obtained by grouping the adjustment factors to simplify the process of adjustment and to ensure more consistency in the adjustments. In the proposed method fuzzy logic is used for quantifying the quality of requirements and is added as one of the adjustment factor, thus a fuzzy based approach for the Enhanced General System Characteristics to Estimate Effort of the Software Projects using productivity has been obtained. The phase 3 takes the calculated function point from our work and is given as input to the static single variable model (i.e. to the Intermediate COCOMO and COCOMO II) for cost estimation. The Authors have tailored the cost factors in intermediate COCOMO and both; cost and scale factors are tailored in COCOMO II to suite to the individual development environment, which is very important for the accuracy of the cost estimates. The software performance indicators are project duration, schedule predictability, requirements completion ratio and post-release defect density, are also measured for the software projects in my work. A comparative study for effort, performance measurement and cost estimation of the software project is done between the existing model and the authors proposed work. Thus our work analyzes the interactional process through which the estimation tasks were collectively accomplished.

Keywords:
Effort Adjustment Factor (EAF), Kilo Lines of Code (KLOC), Total Effort Multiplier (TEM), Scale Factors, Cost Drivers, Value Added Factors

1. INTRODUCTION

In this paper, we will be discussing our proposed model for effort estimation. The paper presents a model that first captures the fundamental of software metrics. In the first phase, LOC is presented as primarily a measurement technique for quantifying the size of a software product [5]. Function points are an indirect measure of software size based on external and internal application characteristics, as well as application performance [8]. Function Points have a significant cost estimating relationship (CER) to software costs. Once determined, function points can be input into empirical statistical parametric software cost estimation equations and models in order to estimate software costs [9]. Person month metric are used to express the effort a personnel devote to a specific project. Software size estimates are converted to software effort estimations to arrive at effort for all work elements, and then the total cost of the whole software project is calculated. Estimating size and effort are the most important topics in the area of software project management.

In the Second phase, while discussing a proposed model for effort estimation, a number of enhancements to adjustment factors of functional size measurements have been introduced. One of the enhancements proposed in this work is grouping the available 14 GSCs into three groups. They are "*System complexity*", "*I/O complexity*" and "*Application complexity*". Another important enhancement in this proposed Effort Estimation model is the consideration of the quality of requirements as an adjustment factor and this "Quality complexity" is added as the fourth group to the adjustment factor. There are several approaches for estimating such efforts, in this work a fuzzy logic based approach for quality selection is proposed.

The obtained function point is given as input to the next phase; the phase 3 consists of Intermediate COCOMO and COCOMO II model. In this work, former computes effort as a function of program size and analysis has been done to define rating for the cost drivers and by adding the new rating the developmental effort is obtained. While for the latter, it gets the value for function point from our proposed work as input and computes effort as a function of program size, set of cost drivers, scale factors, Baseline Effort Constants and Baseline Schedule Constants. Thus this phase explains the Empirical validation for software development effort multipliers, analyses to define the ratings for the cost drivers and scale factors of Intermediate COCOMO model and COCOMO II. Cost estimation must be done more diligently throughout the project life cycle so that in the future there are fewer surprises and unforeseen delays in the release of a product.

Performance of the software projects are also measured in this phase, the measurement indicators are project duration, schedule predictability, requirements completion ratio and post-release defect density which are also calculated. By adding the new rating the developmental effort obtained is very much nearer to the planned effort and also a comparative study is done between the existing and our method.

2. RELATED WORK

One of the popular functional sizing units is function points [1]. In function point sizing, visible external aspects of software that can be counted consist of five items; each of the functions that are assigned one of the five items is further classified as complex, average, or simple. The complexity weights are applied to the initial function point count to arrive at an unadjusted function point. Second, Function point counting passes through an adjustment phase. This phase consists of scoring a group of general systems characteristics (GSC) that rate the general functionality of the application being counted, from the GSC, the value adjustment factor (VAF) is determined, The last step is to calculate the final adjusted function point count by multiplying the VAF times the unadjusted function point [2][3][4]. One of the enhancements proposed in the model is grouping the 14 GSCs into groups. [6][7][10]. The count total is the summation of all the Information domain value and weighing factor. The fourteen GSC is based on responses to the following involving a scale from 0 to 5. The scores for these characteristics are then summed based on the following formula to arrive at the value adjustment factor (VAF) [11][12][13]. Incomplete requirements and changing requirements rank as the second and third main causes of project failures [14].

Keshwani [15][16] has presented a Mamdani fuzzy modeling scheme where rules are derived from multiple knowledge sources such as previously published databases and models. A keen mapping between input and output spaces may be developed with the help of fuzzy logic [17]. Fuzzy logic models can be easily constructed without any data whatsoever, or with a small sample used to validate the model [18]. Estimation using expert judgements is better than models [19]. This model is serving as a framework for an extensive current data collection and analysis effort to further refine and calibrate the model's estimation capabilities [20]. To determine the nominal person months for the Early Design model, the unadjusted function points have to be converted to source lines of code in the implementation language [21]. A study accomplished by, presents the conclusion that the most critical input to the COCOMO II model is size [22]. Improving software effort estimation does not necessarily require adopting sophisticated formal estimation models or expensive project experience databases [23]. Existence of a consistently applied process is an important and a prerequisite for a successful measurement program in case of different environments [24]. In traditional software cost models, costs are derived from effort. Empirical estimation models provide computational formulae for calculating the effort based on statistical approach by referring the past data of more or less similar projects executed [27][26] The Intermediate COCOMO model computes effort as a function of program size and a set of cost drivers [28]. Software organizations, whether they are just starting a measurement program or have a well-developed program, want a way to gauge the performance of their software projects against other organizations in their industry [30]. Performance measurement might be referred to as performance monitoring or performance auditing [31]. An effective set of performance measures will provide actionable information, on a focused set of metrics, to provide a balanced view of project performance to improve the project management process [33]. Organization will be interested in monitoring and comparing the projects and project performances. Based on the performances, appropriate rewards or incentives need to be given to the better achieving project teams [34] [35]. Function point from the proposed method as input, and gives to the static single variable model (Intermediate COCOMO and COCOMO II) for cost estimation whose cost factors are tailored in intermediate COCOMO and both, cost and scale factors are tailored in COCOMO II to suite to the individual development environment, which is very important for the accuracy of the cost estimates [36].

3. SYSTEM OVERVIEW

In that respect our method proposes a notion of primary metrics and the mode to calculate the Lines of code, Function point and Person month are discussed in phase 1. In the phase 2 a fuzzy based proposed model for effort estimation is discussed, In our model the enhancements proposed is grouping the fourteen GSCs into groups, first group is "*System complexity*" which consist of Data communication Complexity, Distributed Data Processing Complexity, Performance Complexity and Heavily used configuration Complexity, the average of the four weighted scores together gives the System complexity. Second group is "*I/O complexity*" which consist of Transaction rate Complexity, Online data entry Complexity, End user efficiency Complexity and Online update Complexity , and the third group is "*Application complexity*" which consists of Complex processing Complexity, Reusability Complexity, Installation Ease Complexity, Operational Ease Complexity, Multiple Sites Complexity, Facilitate Change Complexity.

- To investigate how the cost and effort estimation task is concentrated on the development of software systems and not much on the quality coverage, our paper focus on the Quality assurance for effort estimation work. The questions we raise are as follows,
  - Why grouping of General System characteristic for software estimation as a collaborative activity is needed?
  - What types of Quality assurance are needed to accomplish the estimation task?
  - What type of techniques can be considered for building our quality models?
  - Which type will overcome all the potential problems?
  - Does trimming of scale factors and cost drivers improve the estimation and how our model benefits by trimming?

The grouping of the 14 GSC into groups is needed to simplify the counting process and reduces the probability of errors while counting; this enhanced system focuses on minimizing the effort by enhancing the adjustments made to the functional sizing techniques.

In the existing systems, the effort and cost estimation are more concentrated on the development of software systems and not much on the quality coverage. Hence, the proposed model ensures the quality assurance for the effort estimation.

This paper presents fuzzy classification techniques as a basis for constructing quality models that can identify outlying software components that might cause potential quality problems and this "Quality complexity" is added as the fourth group in the enhancement process. From the four groups, proposed value adjustment factor is calculated. The total adjustment function point is the product of unadjusted function point and the proposed value adjustment factor.

In phase 3, COCOMO II model computes effort as a function of program size (function point got from phase 2 of our model is converted to Lines of code), set of trimmed cost drivers, trimmed scale factors, Baseline Effort Constants and Baseline Schedule Constants. Empirical validation for software development effort multipliers of COCOMO II model is analyzed and the ratings for the cost drivers are defined. By adding new ratings to the cost drivers and scale factors and seeing that the characteristic behaviour is not altered, the developmental person month of our proposed model is obtained, also in phase 3 Intermediate COCOMO model computes effort as a function of program size, got from the phase 2 and a set of trimmed cost drivers, also the effort multipliers of Intermediate COCOMO model is analyzed and the ratings for the cost drivers are defined. By adding new ratings to the cost drivers and seeing that the characteristic behaviour is not altered, the developmental person month of our proposed model is obtained. It is observed that the effort estimated with COCOMO II and Intermediate COCOMO are very much nearer to their respective planned efforts; with our proposed cost model minimal effort variance can be achieved by predicting the cost drivers for computing the EAF and the last component of this phase in our proposed model measures the performance of software projects with its measurement indicators. Thus our proposed model computes Effort, Cost and measures the performance of the software projects, also a comparative study is done between the existing model and our model taking samples data's of HR application and Hospital application.

4. MODELING PROCEDURE

The proposed modeling procedure clearly describes the steps to build the estimation models. The three layer in this procedure are displayed in the below Fig.1. The tasks and their importance are also explained in detail in their respective sections.

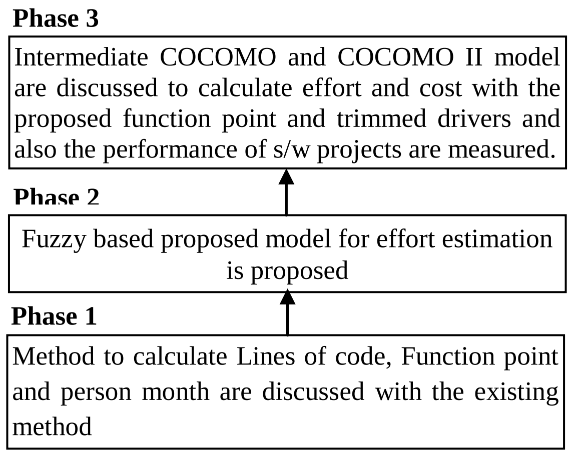

Phase 3
Intermediate COCOMO and COCOMO II model are discussed to calculate effort and cost with the proposed function point and trimmed drivers and also the performance of s/w projects are measured.
Phase 2
Fuzzy based proposed model for effort estimation is proposed
Phase 1
Method to calculate Lines of code, Function point and person month are discussed with the existing method

Fig.1. Block diagram of the Proposed Model

5. FIRST PHASE

The first phase consists primitive software engineering metrics called as primary metrics, they are person-months PM), function-points (FP) and lines of code (LOC). The notion of primary software metrics is introduced in the lower layer. Other metrics are presented in higher layers using the primary metrics as foundations. The three metrics, PM, LOC and FP represent measures of *personnel effort, programmer productivity,* and *software functionality*.

5.1 LINES OF CODES

LOC is presented as a measurement technique for quantifying the size of a software product. LOC is more of a measurement technique than a counting technique. There are many ways of obtaining the LOC of a program without actually counting program lines of code.

The steps for calculating Lines of codes are:

- Each Statement (executable or declarative) is counted as one line.
- Comments are excluded from the count.
- For languages that use delimiters each delimiter corresponds to one statement.

5.2 FUNCTION POINT

The function point metric (FP) proposed by Albrecht can be used effectively as a means for measuring the functionality delivered by a system using historical data. FP can then be used to estimate the cost or effort required to design, code and test the software, predict the number of errors that will be encountered during testing and forecast the number of components and/or the number of projected source lines in the implemented system.

The steps for Calculating Function point metric is:

- Count total is calculated using Information domain and the weighting factor.
- The Value added factor is based on the responses to the following 14 characteristics, each involving a scale from 0 to 5 and the empirical constants
- Function point is the product of Count total and the Value added factor.

Thus Function points (FP) provide a measure of the functionality of a software product and is obtained using the following equation:

$$FP = \text{count-total} \times [0.65 + 0.01 \times \Sigma\, Fi]$$

where, the count-total is a summation of weighted input/output characteristics, and Fi is the summation of fourteen ranked factors.

5.3 PERSON MONTHS

One PM is normally defined as the output of one person in one month, working 40 hours/week, with one month defined as four weeks.

The steps for calculating person months are,

For Basic COCOMO model are static single variable with format,

- PM = f (LOC) hence
- Hence person months for organic, semidetached and embedded are
- PM = 2.4 KLOC 1.05 (organic)
- PM = 3.0 KLOC 1.12 (semidetached)
- PM = 3.6 KLOC 1.20 (embedded)

For intermediate COCOMO, the cost driver multiplier, F, can be factored with the constant to give,

- PM = 3.2 F * KLOC 1.05 (organic)
- PM = 3.0 F * KLOC 1.12 (semidetached)
- PM = 2.8 F * KLOC 1.20 (embedded)

6. SECOND PHASE

In the middle layer our model proposes a fuzzy based model for effort estimation, the enhancements proposed in our model is grouping the fourteen GSCs into three groups and quality is added as the fourth group.

6.1 PROPOSED FUNCTION POINT

In function point sizing, external aspects of software that is counted consist of five items; they are outputs, inquiries, inputs, files, and interfaces. Each of the functions that are assigned one of the five items is further classified as complex, average, or simple. The complexity weights are applied to the initial function point count in the same way as Albrecht's function point metric to arrive at an unadjusted function point. Our model proposes the enhancements to adjustment factors of functional size measurements. The enhancements proposed in this model are grouping the 14 GSCs into three groups which simplify the counting process and reduce the probability of errors while counting.

6.2 QUALITY OF EFFORT

The quality of requirements is rated and this *Quality complexity* is added as the fourth group among the adjustment factors in our model. Quality of requirements takes as inputs a set of stated or implied needs, relevant technical documentation and the ISO Standard itself and produces a quality requirement specification. The standard identifies six key quality attributes.

Functionality is the group of attributes that refer to the functions and their specific estates, the functions is the degree to which the software satisfies the stated needs as indicated by the following sub-attributes namely suitability, accuracy, interoperability, compliance and security. Reliability is the amount of time the software is available for use as indicated by the following sub-attributes namely maturity, fault tolerance, and recoverability. Usability is the degree to which the software is easy to use as indicated by the following sub-attributes namely understandability, learnability, and operability. Efficiency is the degree to which the software makes optimal use of system resources as indicated by the following sub-attributes namely time behavior and resource behavior. Maintainability is the ease with which repair may be made to software as indicated by the following sub-attributes namely analyzability, changeability, stability, and testability. Portability is the ease with which the software can be moved from one environment to another as indicted by the following sub-attributes namely adaptability, installability, conformance and replaceability.

The above six key quality attributes are taken to quantify the quality of requirements using fuzzy logic and is added as the fourth group to the enhancement of the adjustment factor The scores (ranging from 0 to 5) for these characteristics in each group are then summed based on the following formula to arrive at the Enhanced value adjustment factor.

Thus in our model, the proposed VAF = 0.65 + 0.01 $\sum$ proposed four groups, where 0.65 and 0.01 are empirically derived constants.

6.3 FUZZIFICATION OF INPUTS

Our model considers all the six key quality attributes (for Quality Complexity), they are *Functionality*, *Reliability*, *Usability*, *Efficiency*, *Maintainability* and *Portability* as inputs and provides a crisp value of Quality efforts using the Rule Base. All the six quality attributes, which is taken as inputs can be classified into fuzzy sets viz. Low, Medium and High. The output Quality Efforts is classified as Very High, High, Medium, and Low. In our proposed model to fuzzify the inputs, the triangular membership functions are chosen namely Low, Medium and High. Also the quality effort which is the output variable in our model has four membership functions they are very high, high, medium and low. All the inputs and outputs are fuzzified and all possible combination of inputs were considered in our model which leads to 34 i.e. 81sets. Quality Effort in case of all 81 combinations is classified as Very High, High, Medium, and Low by expert opinion in our proposed model.

7. THIRD PHASE

In the last Phase, the Intermediate COCOMO model computes effort as a function of program size and a set of cost drivers, COCOMO II has some special features, which distinguish it from other ones. The usage of this method is very wide and its results usually are accurate.

7.1 INTERMEDIATE COCOMO

The Intermediate COCOMO equation is given by E = aKLOC^ b * EAF, where, a and b are the domain constants of the intermediate COCOMO model. These formulae link the size of the system, domain constants and Effort Multipliers (EM) to find the effort to develop a software system. The effort adjustment factor/ Total adjustment factor has been calculated using 15 cost drivers. Cost drivers are grouped into four categories; they are Product, Computer, Personnel and Project. Each cost driver has been rated on a six-point ordinal scale ranging from low to high importance. Based on the rating, an effort multiplier is determined, Product of all effort multipliers leads to EAF. Cost drives have a rating level that expresses the impact of the driver on development effort, PM. These rating can range from Extra Low to Extra High. For the purpose of quantitative analysis, each rating level of each cost driver has a weight associated with it. The weight is called Effort Multiplier.

The steps involved in the proposed model for calculating the Effort are:

- Count Total is calculated using Information domain and the weighting factor. The complexity weights are applied to the initial function point count to arrive at an unadjusted point total.
- The Value adjustment factor is based on the responses to the following 14 general system characteristics, each involving a scale from 0 to 5 and the empirical constants. Grouping the fourteen general system characteristics into three groups are used instead of the 14 general system characteristics in the function point original methodology.

- The fourth group is the quality factor, which is the set off quality characteristics, they are Functionality, Reliability, Usability, Efficiency Maintainability and Portability
- Total degree of influence = Σ system Complexity + Σ I/O Complexity + Σ Application Complexity + Σ quality Complexity
- Proposed Value adjustment factor is [(TDI * 0.01) + 0.65], where TDI is the total degree of influence and, 0.01 and 0.65 are the empirical constants.
- Total adjustment function point is the product of unadjusted function point and the proposed Value adjustment factor.
- From the Function point, the lines of code is calculated, which is the product of function point and the multiplication language factor.
- Intermediate COCOMO model computes effort as a function of program size and a set of cost drivers.
- The cost drivers are assigned new ratings in such a way that the existing characteristic behavior of the intermediate model is not altered.
- Total Effort multiplier is the product of the ratings of the assigned cost drivers.
- From the obtained TEM, the developmental person month is calculated, which is very much nearer to the planned effort (Table.7).

7.2 COCOMO II

In COCOMO II effort is expressed as a function of program size, set of cost drivers, scale factors, Baseline Effort Constants and Baseline Schedule Constants. COCOMO II has some special features, which distinguish it from other ones. The Usage of this method is very wide and its results usually are accurate. In COCOMO II effort is expressed as a function of program size, set of cost drivers, scale factors, Baseline Effort Constants and Baseline Schedule Constants.

$$\text{PM} = \text{A} \times \text{size E} \times \prod_{i=1}^{n} EM$$

where, $\text{E} = \text{B} + 0.01 \text{ x } \sum_{j=1}^{5} SF_j$

The application size is exponent, is aggregated of five scale factors that describe relative economies or diseconomies of scale that are encountered for software projects of dissimilar magnitude. They are Precedentedness (PREC), Development Flexibility (FLEX), Architecture / Risk Resolution (RESL), Team Cohesion (TEAM) and Process Maturity (PMAT)

These are the 17 effort multipliers/ cost drivers used in COCOMO II Post-Architecture model to adjust the nominal effort, Person Months, to reflect the software product under development. They are grouped into four categories: product (*Required Software Reliability, Data Base Size, Developed for Reusability, Product Complexity and Documentation Match to Life-Cycle Needs*), *platform* (*Execution Time Constraint, Main Storage Constraint, Platform Volatility*), *personnel* (*Analyst Capability, Programmer Capability, Personnel Continuity, Application Experience, Platform Experience, Language and tool experience*), *and project*(*Use of Software Tools, Multisite Development and Required Development Schedule*). The EM values are selected appropriately and tailored and used to estimate the development projects. The Driver symbol are grouped into four category, they are Product drivers (consists of RELY, DATA, CPLX, RUSE and DOCU), Platform drivers (consists of TIME, STOR, PVOL), Personnel (consists of ACAP, PCAP, PCON, APEX, PLEX, LTEX) and Project drivers (consists of TOOL, SITE and SCED).

7.3 PERFORMANCE MEASURES

Performance measurement is a process of assessing the results of a company, organization, project, or individual. Software Organization wants a way to gauge the performance of their software projects against other organization. In our document a set of defined software project performance measures are defined which can be used by software development projects to make valid comparisons of performance is made.

7.3.1 Project Duration:

Project duration is a measure of the length of a project in work days, excluding times when the project is not active due to work stoppages. Project duration does not include non-work days such as weekend days and holidays. Project start is the date when user requirements have been baselined. Project end is the date of the first installation of the software application.

Project Duration = (num _days - stoppage_days)

7.3.2 Schedule Predictability:

Schedule predictability is a measure of how much the original project duration estimate differs from the actual project duration that was achieved. Schedule predictability is a positive value when there is a schedule overrun and a negative value when there is a schedule underrun.

$$\text{SP} = \frac{(\text{Project Dur}) - (\text{Est Proj Dur})}{\text{Estimated Project Duration}} * 100$$

Schedule predictability is a positive value when there is a schedule overrun and a negative value when there is a schedule underrun.

7.3.3 Requirements Completion Ratio:

The requirements completion ratio measures the extent to which planned functional requirements were satisfied in the final product implementation.

The requirements completion ratio (RCR) is expressed as a percentage as,

$$\text{RCR} = \frac{\text{Satisfied reqs}}{\text{Planned reqs}} * 100\%$$

7.3.4 Post Release Defect Density:

Post-release defect density is the number of unique defects per unit size discovered during the first six months after initial deployment of the software.

$$\text{PRDD} = \frac{\sum \text{D}}{\text{Size}}$$

Table.1. Count Total values of Hospital and HR application

	Hospital Application data			**HR Application data**		
Information Domain Value	Count	Weighing factor		Count	Weighing factor	
External Inputs (EIs)	33	03(Simple)	99	11	03(Simple)	33
External Outputs (EOs)	03	04(Simple)	12	01	07(Complex)	07
External Inquiries (EQs)	-	-	-	04	03(Simple)	12
Internal Logical Files (ILFs)	02	07(Simple)	14	03	07(Simple)	21
External Interface Files (EIFs)	-	-	-	03	05(Simple)	15
Count Total -------------------------------→			125	Count Total -----------→		88

8. EXPERIMENTAL RESEARCH SETUP AND RESULTS

8.1 EFFORT ESTIMATION

Function Points and the effort in person-months are computed for the HR application and Hospital application. Below is our proposed model Factor Value for Hospital application and HR application are given,

Table.2. Proposed model Factor Value for Hospital and HR application

System Complexity:

Data Communication	0	3
Distributed Data Processing	0	1
Performance	1	3
Heavily used configuration	0	2

I/O Complexity:

Transaction rate	2	3
On-line data entry	5	3
End User Efficiency	3	4
On-line update	0	3

Application Complexity:

Complex Processing	0	2
Reusability	0	3
Installation Ease	2	3
Operational Ease	5	3
Multiple sites	0	3
Facilitate Change	0	4

Quality Complexity:

Quality of requirements (for our model)	1	0.5

Table.3. Quality Effort for fuzzification of the Quality Complexity

System Name	**Type**	**Version**	**No. of Inputs**	**No. of Outputs**	**No. of Rules**	**AND Method**	**OR Method**	**Imp Method**	**Aggregation Method**	**Defuzzy Method**
Quality Effort	mamdani	2.0	6	1	81	min	max	min	max	centroid

Table.4. Inputs for fuzzification of the Quality Complexity

	Name	**Range**	**Num MFs**	**MF1**	**MF2**	**MF3**
Input 1	Functionality	[0,1]	3	‘Low’: ‘trimf’, [0 0.16 0.33]	‘Medium’: ‘trimf’, [0.3 0.45 0.62]	‘high’: ‘trimf’, [0.57 0.85 1]
Input 2	Reliability	[0,1]	3	‘Low’: ‘trimf’, [0 0.16 0.34]	‘Medium’: ‘trimf’, [0.3 0.45 0.62]	‘high’: ‘trimf’, [0.56 0.85 1]
Input 3	Usability	[0,1]	3	‘Low’: ‘trimf’, [0 0.16 0.35]	‘Medium’: ‘trimf’, [0.3 0.45 0.62]	‘Low’: ‘trimf’, [0.56 0.8 1]
Input 4	Efficiency	[0,1]	3	‘Low’: ‘trimf’, [0 0.16 0.34]	‘Medium’: ‘trimf’, [0.3 0.4 0.65]	‘high’: ‘trimf’, [0.6 0.85 1]

Input 5	Maintainability	[0,1]	3	'Low': 'trimf', [0 0.16 0.33]	'Medium': 'trimf', [0.3 0.45 0.62]	'high': 'trimf', [0.58 0.85 1]
Input 6	Portability	[0,1]	3	'Low': 'trimf', [0 0.16 0.35]	'Medium': 'trimf', [0.3 0.4 0.65]	'high': 'trimf', [0.58 0.8 1]

Table.5. Output of the Quality Complexity

	Name	**Range**	**Num MFs**	**MF1**	**MF2**	**MF3**	**MF4**
Output	Quality Effort	[0 1]	4	'Low': 'trimf', [0 0.12 0.23]	'Medium': 'trimf', [0.4 0.51 0.62]	'high': 'trimf', [0.60 0.75 0.82]	'V.high': 'trimf', [0.8 0.91 1.0]

FP Estimated = Count total × [0.65 + 0.01 × Σ(Fi)]

- FP Estimated for Existing (Hospital application) = 125 x [0.65 + 0.01*103.75] = 103.75 FP
- FP Estimated for Existing (HR) = 88 x [0.65 + 0.01 * 40] = 92.4 FP
- FP for the Proposed model (Hospital application) = 125 x [0.65 + 0.01*4.91] = 87.39 FP
- FP for the Proposed model (HR) = 88 x [0.65 + 0.01 * 9.0] = 65.12

Assuming Productivity for VB/Oracle is 15hrs/Function Point

- Effort for the Existing model (Using Hospital application) is 1556.25 person hours
- Effort for the Existing model (HR) is 1386 person hours
- Effort for the proposed model (Hospital application) is 1310 person hours
- Effort for the proposed model (HR) is 976.8 person hours

Assuming a person works for 8.5hrs/day and 22 days a month, the effort obtained for the existing and proposed are,

- Effort for Hospital application in person month is 8 approximately.
- Effort for HR in person month is 8 approximately.
- Effort from the proposed model for Hospital application in person month is 7 approximately.
- Effort from the proposed model for HR in person month is 7 approximately.

8.2 PERFORMANCE MEASURES FOR HOSPITAL AND HR APPLICATION ARE:

For Hospital application the project started on Nov 3rd and ended on Sep 14th, For HR application the projects started on Nov 5th and ended on Sept 16th.

Table.6. The values of the performance Indicators using existing method and the Enhanced method

	Using the existing method		**Using our Enhanced method**	
Performance Indicators	**Hospital application**	**HR application**	**Hospital application**	**HR application**
Effort Estimation	104	92.4	89	64.68
Productivity	8	8	7	7
Project Duration	158 days	163 days	136 days	141 days
Schedule Predictability	–10.2% (underrun)	–7.4% (underrun)	–11.6%(Underrun)	–8.4%(Underrun)
Requirements Completion Ratio	75%	87.5%	75%	87.5%
Post-Release Defect Density	3.8 per 100 FP	4.3 per 100 FP	3.3 per l00 FP	3.1 per 100 FP

8.3 COST ESTIMATION FOR HR APPLICATION USING INTERMEDIATE COCOMO

Table.7. The planned effort for HR application

	% of total	**Person days**
Analysis Phase	3	3.648
Design Phase	9	10.944
Construction Phase	39	47.424
Testing	27	32.832
Project Planning	4	4.864
Project tracking	4	4.864
Software Quality Assurance	1	1.216
Configuration Management	3	3.648
Project Documentation	2	2.432
Reviews	6	7.296
Training	1	1.216
Inter group coordinal	1	1.216
	100	121.6

8.4 COST ESTIMATION USING INTERMEDIATE COCOMO FOR HR APPLICATION

KSLOC = FP * Multiplication Language Factor

- KSLOC (Using Albrecht method) = 92.4 * 29
  = 2679.6/1000 = 2.6 KSLOC
- KSOLC (Our proposed model) = 65.12 * 29
  = 1888.48/1000 = 1.8 KSLOC

Nominal Person Month = Effort Factor * KSLOC ^ Effort Exponent (project belong to Semi-detached Mode)

Nominal Person Month = 3 * KSLOC ^ 1.12

- Nominal Person Month (Existing) = 3 * 2.6 ^ 1.12 = 8.7 PM
- Nominal Person Month (our proposed model) = 3 * 1.8 ^ 1.12 = 5.7 PM

Total Planned Efforts, interms of Person Month for HR application is 121.6/170 = 0.72

By selecting minimal ratings for product and computer attributes and maximum ratings for Personnel and Project attributes, Effort Multilier is (selecting values from cost drivers) 0.75 * 0.7 * 0.7 * 1 * 1 * 0.87 * 0.87 * 0.71 * 0.82 * 0.7 * 0.9 * 0.95 * 0.82 * 0.83 * 1.1

Developmental PM = Nominal Person Month * TEM

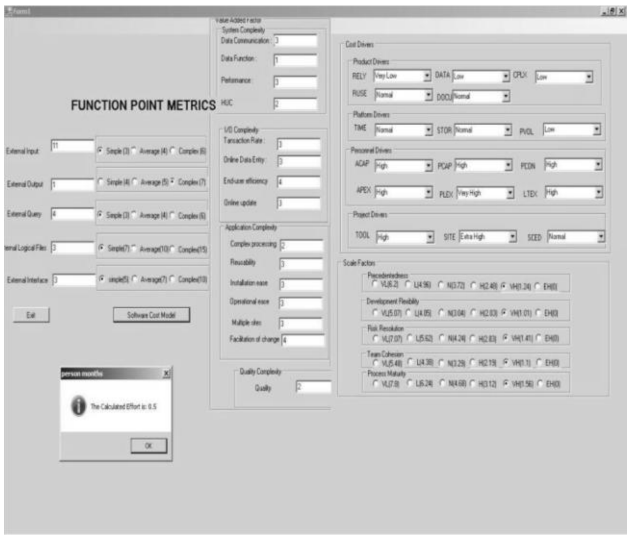
- Developmental PM (Albrecht) = 8.7 * 0.2 = 1.74
- Developmental PM (our Proposed model) = 5.7 * 0.2 = 1.14

After trimming the cost drivers of Intermediate COCOMO for existing and proposed, TEM is 0.025, hence

- Developmental PM = 8.7 * 0.025 = 0.5
- Developmental PM = 5.7 * 0.025 = 0.6

From the above result it shows with the trimming of cost driver the developmental person for both existing and proposed is nearer to planned effort than the Nominal person month. Also we find that our proposed model value is much nearer to planned effort than the existing method.

9. EXPERIMENTAL RESULTS



Form1

FUNCTION POINT METRICS

External Input: 11 [✓ Simple (3)] Average (4) Complex (6)
External Output: 1 Simple (4) Average (5) [✓ Complex (7)]
External Query: 4 [✓ Simple (3)] Average (4) Complex (6)
ernal Logical Files: 3 [✓ Simple(7)] Average(10) Complex(15)
External Interface: 3 [✓ simple(5)] Average(7) Complex(10)

Exit | Software Cost Model

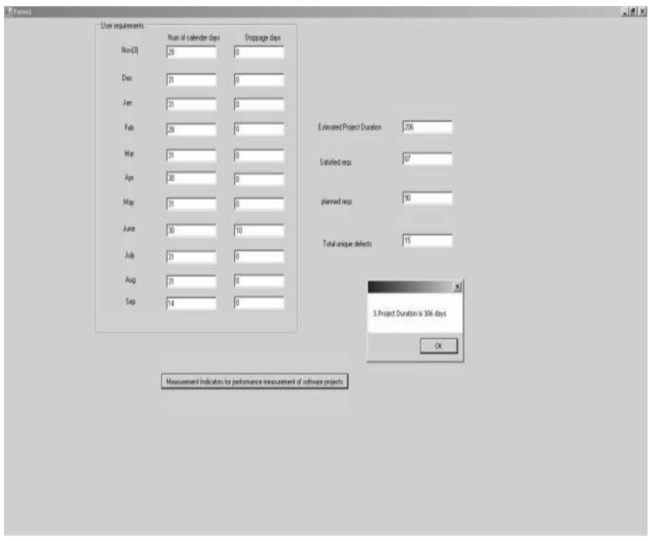
Value Added Factor
System Complexity: Data Communication: 3; Data Function: 1; Performance: 3; HUC: 2
I/O Complexity: Transaction Rate: 3; Online Data Entry: 3; End-user efficiency: 4; Online update: 3
Application Complexity: Complex processing: 2; Reusability: 3; Installation ease: 3; Operational ease: 3; Multiple sites: 3; Facilitation of change: 4
Quality Complexity: Quality: 2

Cost Drivers
Product Drivers: RELY Very Low; DATA Low; CPLX Low; RUSE Normal; DOCU Normal
Platform Drivers: TIME Normal; STOR Normal; PVOL Low
Personnel Drivers: ACAP High; PCAP High; PCON High; APEX High; PLEX Very High; LTEX High
Project Drivers: TOOL High; SITE Extra High; SCED Normal

Scale Factors
Precedentedness: VL(6.2) L(4.96) N(3.72) H(2.48) [✓ VH(1.24)] EH(0)
Development Flexibility: VL(5.07) L(4.05) N(3.04) H(2.03) [✓ VH(1.01)] EH(0)
Risk Resolution: VL(7.07) L(5.62) N(4.24) H(2.83) [✓ VH(1.41)] EH(0)
Team Cohesion: VL(5.48) L(4.38) N(3.29) H(2.19) [✓ VH(1.1)] EH(0)
Process Maturity: VL(7.8) L(6.24) N(4.68) H(3.12) [✓ VH(1.56)] EH(0)

person months
The Calculated Effort is: 0.5
OK

Fig.2. The input and output of the Effort Estimation



Form1

User requirements

	Num of calender days	Stoppage days
Nov(3)	29	0
Dec	31	0
Jan	31	0
Feb	29	0
Mar	31	0
Apr	30	0
May	31	0
June	30	10
July	31	0
Aug	31	0
Sep	14	0

Estimated Project Duration: 206
Satisfied reqs: 87
planned reqs: 90
Total unique defects: 15

3.Project Duration is 306 days
OK

Measurement Indicators for performance measurement of software projects

Fig.3. The input and output of the project duration measurement indicator

Table.8. Effort Estimation using existing COCOMO II and the proposed model

	Results obtained Using Albrecht's Method	**Results obtained Using Proposed Method**
FP	480	366.1
KLOC	43.68	33.31
Scale Factor	6.32	6.32
PM	8.8	6.9
TDEV	10.9	10.1

Table.9. Effort Estimation using existing COCOMO and the proposed model

	Values obtained Using Existing methods	**Values obtained using the method Proposed**
VAF	1.05	0.74
FP	92.4	64.68
Planned Effort	0.72	0.72
Nominal PM	8.74	5.7
Developmental PM	1.74	1.14
With Proposed rating, PM (trim)	0.5	0.6

10. CONCLUSION & FUTURE SCOPE

The Authors have proposed a model for effort, cost and performance measure of the software projects. The primary metrics of function points, person-months, and lines of code are presented as Convertible primary metrics upon which static single variable model to estimate cost and project performance measures are built which can be used by software development projects to make valid comparisons of the performance. An approach for grouping the available value adjustment factor into three groups and the quality factor got from the fuzzy rule based approach is added as an another group. From the four groups, enhanced adjustment factor is obtained and the effort is calculated taking HR application and hospital application as case studies. The Experimental research setup shows the factor values for hospital and HR application, also the quality attributes for fuzzification of the quality attributes is shown, from which Function point is calculated for the existing and the authors model. Performance measure using existing method and enhanced method are measured. Planned effort for HR application is shown, Cost estimation for intermediate COCOMO with the existing method and the author's method with altered rating is calculated. An analysis is done between the Albrecht's method and Authors method. Based on the above results, the proposed method for effort estimation is nearer to the result of other estimation models. Hence this type of Estimation may be recommended for the software development. The unique difference between the proposed and existing estimation of effort for the software system development is the level of quality consideration. It is also found that the obtained person month with the altered rating is very much nearer to the planned effort.

REFERENCES

[1] R. Agarwal, M. Kumar, S. Malick, R. M. Bharadwaj and D. Anantwar, "Estimating Software projects", *Software Engineering Notes*, *ACM SIGSOFT*, Vol. 26, No. 4, pp. 60–67, 2001.

[2] H. Azath and R. S. D. Wahidabanu, "Function Point: A Quality Loom for the Effort Assessment of Software Systems", *International Journal of Computer Science and Network Security*, Vol. 8, No. 12, pp. 321 – 328, 2008.

[3] Charles R. Symons, "Function Point Analysis: Difficulties and Improvements", *IEEE Transactions on Software Engineering*, Vol. 14, No. 1, pp. 2 – 11, 1988.

[4] M. J. Basavaraj and K. C. Shet, "Software Estimation using Function Point Analysis Difficulties and Research Challenges", *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*, pp. 111 – 116, 2007.

[5] Kiumi Akingbehin and Bruce Maxim, "A Three-Layer Model for Software Engineering Metrics", *Proceedings of the Seventh ACIS International Conference on Software Engineering Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pp. 17 – 20, 2006.

[6] Galal H. Galal-Edeen, Amr Kamel and Hanan Moussa, "Lessons Learned from Building an Effort Estimation Model for Software Projects", *International Journal of Software Engineering*, Vol. 3, No. 2, pp. 71 – 86, 2010.

[7] Valerie Marthaler, "Function Point Counting Practices Manual (Release 4.1)", *International Function Point User's Group*, 1999.

[8] Allan J. Albrecht, "Measuring Application Development Productivity", *Proceeding of the Joint SHARE/GUIDE/IBM Applications Development Symposium*, pp. 83 – 92, 1979.

[9] C. J. Lokan, "An Empirical Analysis of Function Point Adjustment Factors", *Information and Software Technology Journal*, Vol. 42, No. 9, pp. 649 – 659, 2000.

[10] David Longstreet, "*Function Points Step by Step*", Longstreet Consulting, Inc., 1999.

[11] M. Pauline, P. Aruna and B. Shadaksharappa, "A Layered Model for Software Metrics", *International Conference on Intelligent Design and Analysis of Engineering Products, System and Computation*, pp. 63 – 65, 2010.

[12] M. Pauline, P. Aruna and B. Shadaksharappa, "A Cost Model for Estimation of the Software Developed", *International Conference on Communication, Computation, Control and Nanotechnology*, pp. 762 – 764, 2010.

[13] Zhendong Lun, "Software Cost Estimation", Department of Computer Science, Southern Illinois University Edwardsville, 2008.

[14] Standish Group, "*CHAOS Report*", Standish Group International, 1994.

[15] Keshwani et al., "Rule-based Mamdani-type fuzzy modeling of skin permeability", *Applied Soft Computing*, Vol. 8, pp. 285 – 294, 2007.

[16] Kirti Seth, Arun Sharma and Ashish Seth, "Component Selection Efforts Estimation – a Fuzzy Logic Based Approach", *International Journal of Computer Science and Security*, Vol. 3, No. 3, pp. 210 – 214, 2009.

[17] Roger Jang and Ned Gulley, "Fuzzy Logic Toolbox for MATLAB", User's Guide. The Math Works Inc, USA, 1995.

[18] Stephen G. MacDonell, Andrew R. Gray and James M. Calvert, "FULSOME: Fuzzy Logic for Software Metric Practitioners and Researchers", *Proceedings of the IEEE 6th International Conference on Neural Information Processing*, Vol. 1, pp. 308 – 313, 1999.

[19] Saleem Basha and P. Dhavachelvan, "Analysis of Empirical Software Effort Estimation Models", *International Journal of Computer Science and Information Security*, Vol. 7, No. 3, pp. 68 – 77, 2010.

[20] T. N. Sharma, "Analysis of Software Cost Estimation Using COCOMO II", *International Journal of Scientific & Engineering Research*, Vol. 2, No. 6, pp. 1 – 5, 2011.

[21] Jongmoon Baik, "COCOMO II Model Definition Manual", Center for Software Engineering, USC, 2000.

[22] Majed Al Yahya, Rodina Ahmad and Sai Lee, "Impact of CMMI Based Software Process Maturity on COCOMO II's Effort Estimation", *The International Arab Journal of Information Technology*, Vol. 7, No. 2, pp. 129 – 137, 2010.

[23] Magne Jorgensen, "Practical Guidelines for Expert-Judgment-Based Software Effort Estimation", *IEEE Software*, Vol. 22, No. 3, pp. 57 – 63, 2005.

[24] Frank Niessink and Hans van Vliet, "Two Case Studies in Measuring Software Maintenance Effort", *Proceedings of the International Conference on Software Maintenance*, pp. 76 - 85, 1998.

[25] R. S. Pressman, "*Software Engineering A Practitioner's Approach*", McGraw-Hill, 1997.

[26] Barry W. Boehm, Ellis Horowitz, Ray Madachy, Donald J. Reifer, Bradford K. Clark, Bert Steece, A. Winsor Brown, Sunita Chulani and Chris Abts, "*Software Cost Estimation with COCOMO II*", Prentice Hall, pp. 544, 2000.

[27] B. Boehm and P. Papaccio, "Understanding and controlling software costs", *IEEE Transactions on Software Engineering*, Vol. 14, No. 10, pp. 1462 – 1477,1988.

[28] M. J. Basavaraj and K. C. Shet, "Empirical Validation of Software development effort multipliers of Intermediate COCOMO Model", *Journal of software*, Vol. 3, No. 5, pp. 65 – 71, 2008.

[29] Samuel Lee, Lance Titchkosky and Seth Bowen, "Software Cost Estimation", Department of Computer Science, University of Calgary, 2002.

[30] Mark Kasunic, "A Data Specification for Software Project performance Measures: Results of a Collaboration on Performance Measurement", *Software Engineering Institute, Technical Report*, 2008.

[31] Patricia Lichiello, "*Guidebook for Performance Measurement*", University of Washington Health Policy Analysis Program, 1999.

[32] Guidance white paper, "Basic Performance Measures for Information Technology Project", Department of Energy *Software Quality and Systems Engineering*, pp. 2 – 8, 2002.

[33] Manjul Sahay and R. S. Susheer, "Objective Based Performance Measurement [OBPM]", *White Paper, Transversal e Networks Pvt Ltd*, 2004.

[34] M. Pauline, P. Aruna and B. Shadaksharappa, "Fuzzy-Based Approach Using Enhanced Function Point to Evaluate the Performance of Software Project", *The IUP Journal of Computer Sciences*, Vol. VI, No. 2, pp. 20 – 33, 2012.

[35] M. Pauline, P. Aruna and B. Shadaksharappa, "Software Cost Estimation Model based on Proposed Function Point and Trimmed Cost Drivers Using COCOMO II", *International Journal of Engineering Research & Technology*, Vol. 1, No. 5, pp. 1 – 7, 2012.

[36] M. Pauline, P. Aruna and B. Shadaksharappa, "Layered model to Estimate Effort, Performance and cost of the Software projects", *International Journal of Computer Applications*, Vol. 63, No. 13, pp. 17 – 23, 2013.