International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 5, July - 2012

Software Cost Estimation Model based on Proposed Function Point and Trimmed Cost Drivers Using Cocomo II

M.Pauline[1],
*1. Asst.Professor,
Dept. of CSE,
M.V.Jayaraman College of Engineering,
Bangalore
pmariasundaram@yahoo.com*,
Ph: 09849151779.

P.Aruna[2]
*2.Professor,
Dept. of CSE,
Annamalai University,
Annamalainagar,
Chidambaram
Tamil Nadu, South India.*

B.Shadaksharappa[3]
*3.Professor and Head,
Dept. of CSE,
ShirdiSai College of Engineering,
Bangalore
Karnataka*

Abstract

The software cost estimation model, constructive cost model, in its last update (constructive cost model II) has a set of seventeen cost drivers and a set of five scale factors. Reliable effort estimation remains an ongoing challenge to software engineers. The credibility of the client to the business enterprise increases with the accurate estimation. Project planning is one of the most important activities in software projects. If cost and effort are determined pessimistic in software projects, suitable occasions can be missed; whereas optimistic predictions can be caused to some resource losing. The main reason for this problem is imprecision of the estimation. COCOMO II is an objective cost model for planning and executing software projects. A cost model provides a framework for communicating business decisions among the stakeholders of a software effort. In this paper, a model for effort estimation is discussed, which focuses on minimizing the effort by enhancing the adjustments made to the functional sizing techniques. The idea of grouping is introduced to the adjustment factors to simplify the process of adjustment and to ensure more consistency in the adjustments. The proposed method uses fuzzy logic for quantifying the quality of requirements and this quality factor is added as one of the adjustment factor. The calculated function point from the model is given as input to the popular COCOMO II model for cost estimation whose cost factors can be tailored to the individual development environment, which is important for the accuracy of the cost estimates. Cost estimation must be done more diligently throughout the project life cycle so that in the future there are fewer surprises and unforseen delays in the release of a product.

Keyword: Effort Adjustment Factor (EAF), Kilo Source Lines of Code (KSLOC), Total effort multiplier (TEM), Scale Factors, Cost Drivers.

1. INTRODUCTION

Software project failures have been an important subject in the last decade. Software Cost estimation is a prediction of the cost of the resources that will be required to complete all of the work of the software project. COCOMO II is a tailorable family of software sizing models, involving Object Points, Function Points, and Source Lines of Code. In this paper, 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 model 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 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, this work proposes a fuzzy logic based approach for quality selection. The obtained function point is given as input to the COCOMO II model which computes effort as a function of program size, set of cost drivers, scale factors, Baseline Effort Constants and Baseline Schedule Constants. Thus the paper explains the

Empirical validation for software development effort of COCOMO II model and analysis has been done to define the ratings for the cost drivers and scale factors and by adding the new rating the developmental effort is very much nearer to the planned effort.

2. RELATED WORK

Software Sizing is critical for accurately estimating and managing projects and for determining productivity, cost effectiveness etc. 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: 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 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][5]. One of the enhancements proposed in the model is grouping the 14 GSCs into groups. The grouping not only simplifies the counting process, but also reduces the probability of errors while counting. In addition, it improves the correlation of adjusted function point (AFP) with actual effort and decreases the effort variance. [6][7][8][9][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 (ranging from 0 to 5) 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]. This paper presents a Mamdani fuzzy modeling scheme where rules are derived from multiple knowledge sources such as previously published databases and models, existing literature, intuition and solicitation of expert opinion to verify the gathered information [15]. Fuzzy logic proposed by Zadeh is a mathematical tool for dealing with uncertainty and also it provides a technique to deal with imprecision and information granularity. A keen mapping between input and output spaces may be developed with the help of fuzzy logic [16] [17]. Fuzzy logic models can be easily constructed without any data whatsoever, or with a small sample used to validate the model [18]. Improving software effort estimation does not necessarily require adopting sophisticated formal estimation models or expensive project experience databases. 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 in order to assess the relative conciseness of implementation per function point.[21].A study accomplished by presents the conclusion that the most critical input to the COCOMO II model is size, so a good size estimate is very important for any good model estimation [22].

3. SYSTEM OVERVIEW

In this fuzzy based proposed model for effort estimation, 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 consist of Complex processing Complexity , Reusability Complexity , Installation Ease Complexity, Operational Ease Complexity, Multiple Sites Complexity, Facilitate Change Complexity . The grouping of the 14 GSC into groups simplifies the counting process and reduces the probability of errors while counting.

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 quality assurance for the effort estimation is proposed in this paper. This paper discusses 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. These models are using software complexity metrics that are available early in the development process and can provide support during the design and the code phase. Experimental results based on real project data are presented to underline the suggested approach and its advantages compared to crisp classification and decision techniques. 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. The COCOMO II model computes effort as a function of program size, set of cost drivers, 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 very much nearer to the planned efforts, with our proposed cost model minimal effort variance can be achieved by predicting the cost drivers for computing the EAF.

4. PROPOSED FUNCTION POINT

In this proposed model the Enhancements to adjustment factors of functional size measurements is introduced. 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.

The three groups are:

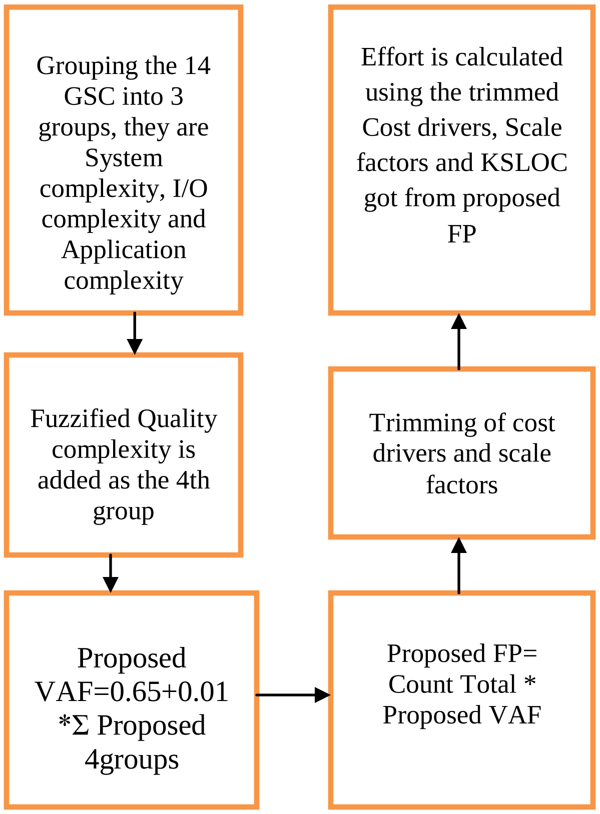
- The first group is *System complexity* which includes data communication, distributed data processing, performance, heavily used configuration and complexity.

- The Second group is *I/O complexity* which includes transaction rate, on-line data entry, end user efficiency and online-line update complexity.

- The third group is *Application complexity* which includes complex processing, reusability, installation ease, operational ease, multiple sites and facilitates change complexity.

5. MODELING PROCEDURE

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


Grouping the 14 GSC into 3 groups, they are System complexity, I/O complexity and Application complexity
Fuzzified Quality complexity is added as the 4th group
Proposed VAF=0.65+0.01 *Σ Proposed 4groups
Proposed FP= Count Total * Proposed VAF
Trimming of cost drivers and scale factors
Effort is calculated using the trimmed Cost drivers, Scale factors and KSLOC got from proposed FP

6. 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 proposed model. The ISO 9126 standard was developed in an attempt to identify quality attributes for computer software. 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.

Proposed VAF = 0.65 + 0.01 Σ proposed four groups, Where 0.65 and 0.01 are empirically derived constants.

7. FUZZIFICATION OF INPUTS

Our proposed 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 3^4 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.

Input1 Name='Functionality', Range=[0 1], NumMFs=3, MF1='Low':'trimf',[0 0.16 0.33], MF2='medium':'trimf',[0.28 0.46 0.65], MF3='high':'trimf',[0.59 0.80 1]

Output1 Name='Qualityeffort', Range=[0 1], NumMFs=4, MF1='Low':'trimf',[0 0.12 0.23], MF2='Medium':'trimf',[0.40 .51 0.62], MF3='High':'trimf',[0.60 0.75 0.82], MF4='Very_High':'trimf',[0.80 0.91 1.0]

Figure 2. Input and Output for fuzzification

8. COCOMO II

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

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

$$\text{Where } E = B + 0.01 \text{ x } \sum_{j=1}^{5} SFj$$

8.1 SCALE FACTORS

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)

Table 1: Scale Factors that are encountered for the software projects

Scale factors	VL	L	N	H	VH	EH
PREC	6.2	4.96	3.72	2.48	1.24	0
FLEX	5.07	4.05	3.04	2.03	1.01	0
RESL	7.07	5.62	4.24	2.83	1.41	0
TEAM	5.48	4.38	3.29	2.19	1.1	0
PMAT	7.8	6.24	4.68	3.12	1.56	0

8.2 TOTAL EFFORT MULTIPLIER

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*). Table below lists the different cost drivers with their rating criterion.

Baseline Effort Constants: A = 2.94; B = 0.91
Baseline Schedule Constants: C = 3.67; D = 0.28

Table 2: Effort multipliers that are used in COCOMO II

Driver symbol value		L			H	VH	XH
RELY	EM1	0.82	.92	.00	1.10	1.26	0.82
DATA	EM2		.90	.00	1.14	1.28	
CPLX	EM3	.73	.87	.00	1.17	1.34	1.74
RUSE	EM4		.95	.00	1.07	1.15	1.24
DOCU	EM5	.81	.91	.00	1.11	1.23	
TIME	EM6			.00	1.11	1.29	1.63
STOR	EM7			.00	1.05	1.17	1.46
PVOL	EM8		.87	.00	1.15	1.30	
ACAP	EM9	.42	.19	.00	0.85	0.71	
PCAP	EM10	.34	.15	.00	0.88	0.76	
PCON	EM11	.29	.12	.00	0.90	0.81	
APEX	EM12	.22	.10	.00	0.88	0.81	
PLEX	EM13	.19	.09	.00	0.91	0.85	
LTEX	EM14	.20	.09	.00	0.91	0.84	
TOOL	EM15	.17	.09	.00	0.90	0.78	
SITE	EM16	.22	.09	.00	0.93	0.86	0.80
SCED	EM17	.43	.14	.00	1.00	1.00	

The above 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 (consist of RELY,DATA,CPLX,RUSE and DOCU),Platform drivers (consist of TIME,STOR,PVOL), Personnel (consist of ACAP,PCAP,PCON,APEX,PLEX,LTEX) and Project drivers (consist of TOOL,SITE and SCED)

8.3 PROJECT DISTRIBUTION

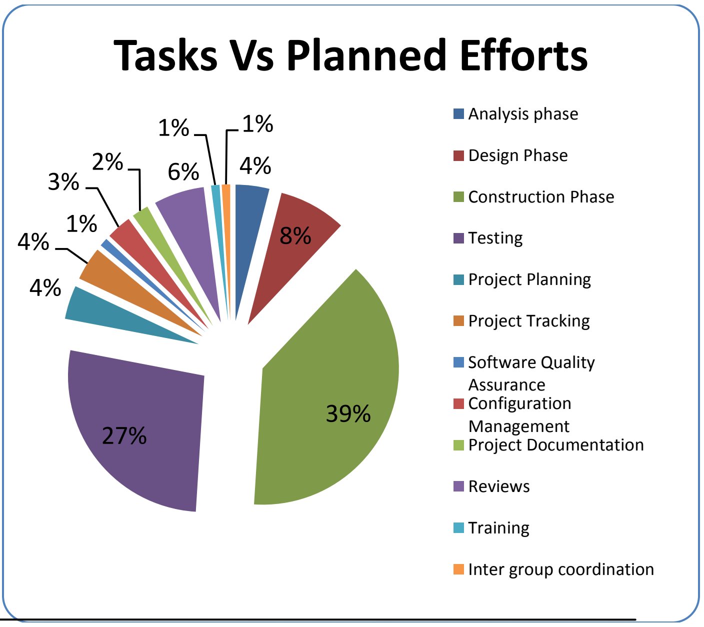


Figure 3 shows the graphical representation using Pie chart the planned effort for Project efforts distribution.

9. EFFORT ESTIMATION

The steps involved in the proposed model for calculating proposed 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

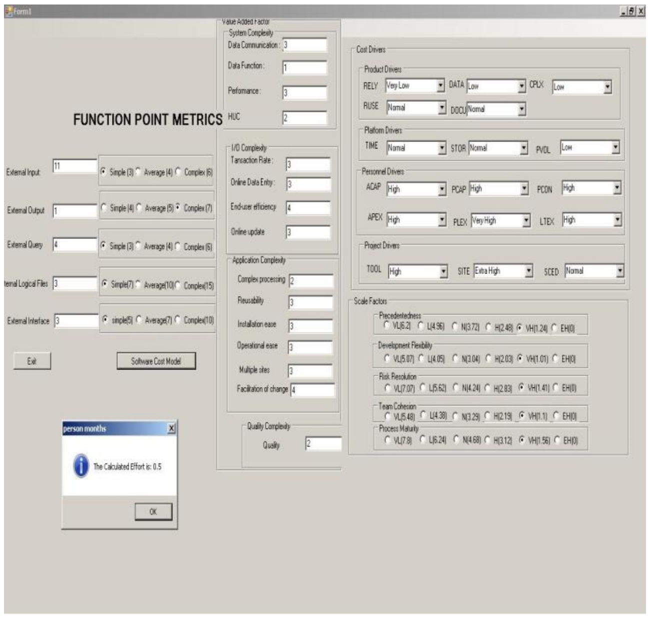
Application Complexity + Σ quality Complexity

- 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.
- Using the Total adjustment function point and multiplication language factor, lines of code is calculated.
- The Model computes effort as a function of program size, Scale factors 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 (shown in the Fig 3 above).

10. EXPERIMENTAL RESULTS

Table 3 Effort Estimation using existing cocomo II and the proposed model

	Albrecht's Method	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


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)
Internal Logical Files: 3 — [✓ Simple(7)] Average(10) Complex(15)
External Interface: 3 — [✓ simple(5)] Average(7) Complex(10)

Exit | Software Cost Model

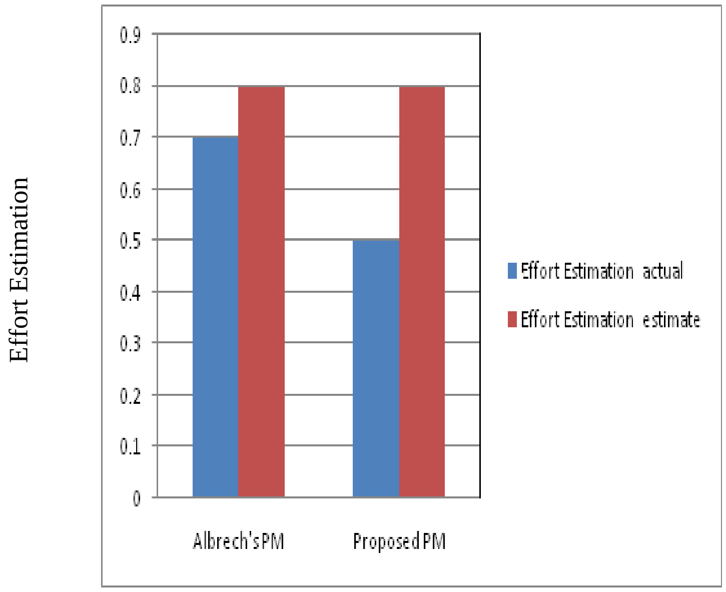
person months
The Calculated Effort is: 0.5
OK

Value Added Factor
System Complexity: Data Communication: 3; Data Function: 1; Perfomance: 3; HUC: 2
I/O Complexity: Tansaction 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)

Figure 4: The input and output of the Effort Estimation



SOFTWARE APPLICATION

Figure 5 shows the graphical representation of Albrect's actual and planned effort and the proposed actual and planned effort

11. CONCLUSION & FUTURE SCOPE

In this paper an approach for grouping the available value adjustment factor into three groups is proposed and the quality requirements got from the fuzzy rule based approach is added as an another group. From the four groups, enhanced adjustment factor is obtained. In this paper we have also altered the ratings of the cost drivers of the Cocomo II and by adding the new rating the existing characteristic of the model is not altered. By tailoring the value of the cost drivers, the total effort multiplier is obtained. From the enhanced adjustment factor, the altered rating of the cost driver, Scale Factors, Effort and Schedule Constants, the effort of the software project in person month is obtained. It is found that the obtained person month is very much nearer to the planned effort. 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 work further requires validation, for this purpose more data has to be collected from projects. SLOC can be given as input, and last, the model can be extended to be an estimation model not just for effort, but also for other metrics such as cost, resources, and others. Other metrics may be used to estimate the effort and substituting other quality factors can be explored as a future scope.

REFERENCES

1) Agarwal.R, Kumar.M, Malick.S, Bharadwaj.R.M, and Anantwar.D, “Estimating Software projects”, ACM SIGSOFT, Vol 26, 2001.
2) Azath.H, Wahidabanu.R.S.D, “Function Point: A Quality Loom for the Effort Assessment of Software Systems”, International Journal of Computer Science and Network Security, IJCSNS Vol.8 No12, Dec 2008.
3) Charles R. Symons, “Function Point Analysis: Difficulties and Improvements”, IEEE Trans., Software Eng., Vol. 14, no. 1, pp.2-11, Jan 1988.
4) Basavaraj.M.J, Dr. Shet.K.C, “ Software Estimation using Function Point Analysis Difficulties and Research Challenges”, T.Sobh (ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, 111-116. © 2007 Springer.
5) KiumiAkingehin 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 (SNPD’06).
6) Galal. H. Galal-Edeen, AmrKamel, and HananMoussa, “ Lessons Learned from Building an Effort Estimation Model for Software Projects”, Int.J. of Software Engineering, IJSE Vol.3 No.2 July 2010.
7) Function Point Counting Practices Manual (Release 4.1), International Function Point User’s Group (IFPUG), May 1999.
8) Albrecht, Allan.J, “Measuring Application Development Productivity”, Proceeding SHARE/GUIDE IBM Application Development Symposium, October 1979.
9) Lokan C.J, “An Empirical Analysis of Function Point Adjustment Factors”, Information and Software Technology, Vol.42, pp.649-659, 2000.
10) Longstreet.D, “Function Points Step by Step”, Longstreet Consulting, Inc., January 1999.
11) Pauline.M, Aruna.P, Shadaksharappa.B,”A Layered Model For Software Metrics”, International Conference on Intelligent Design and Analysis of Engineering products, system and computation, IDAPSC10, pp.63-65.
12) Pauline.M, Aruna.P, Shadaksharappa.B, “A Cost Model for Estimation of the Software Developed”, International Conference on Communication, Computation, Control and Nanotechnology, ICN-2010, pp. 762-764.
13) ZhendongLun, “Software Cost Estimation”, Department of Computer Science, Sothern Illinois University Edwardsville.
14) Standish Group, CHAOS Report”, Standish Group, 1994.
15) Keshwani ET AL., “Rule-based Mamdani-type fuzzy modeling of skin permeability”, Applied Soft Computing (2008), pp 285-294, doi: 10.1016/j.asoc.2007.01.007 Copyright © 2007 Elsevier B.V.
16) Kirti Seth, Arun Sharma and Ashish Seth, “Component Selection Efforts Estimation – A Fuzzy Logic Based Approach”, International Journal of Computer Science and Security, (IJCSS) Vol.3 No 3.
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, “FULLSOME: Fuzzy Logic for Software Metric Practitioners and Researchers”, IEEE, 0-7803-5871-6/99.
19) SaleemBasha and Dhavachelvan.P, “Analysis of Empirical Software Effort Estimation Models”, (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No.3, 2010.
20) Sharma.T.N, “Analysis of Software Cost Estimation Using COCOMO II”, International Journal of Scientific & Engineering Research Volume 2, Issue 6, June-2011, ISSN 2229-5518.
21) JongmoonBaik, “COCOMO II Model Definition Manual”.
22) Majed Al YahyaRodina 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, April 2010.