Software efforts estimation using Use Case Point approach by increasing Technical Complexity and Experience Factors

Chetan Nagar[1]

[1]Ph.D Student, Mewar University (Gangrar) Chittodgarh Rajasthan India

callchetan_nagar@yahoo.com

***Abstract*— An IT industry wants a simple and accurate method of efforts estimation. Estimation of efforts before starting of work is a prediction and prediction always not accurate. Intermediate COCOMO considered 17 factor that affecting the efforts, UCP considered 13 Technical Complexity Factors and 05 Experience factors. There is a lot factors that can affect efforts estimation .Most of the parameter are covered by COCOMO and UCP, but some parameters which are included in COCOMO left by UCP. UCP is one of the popular approaches of effort estimation. This paper is increasing the Technical complexity and Experience factors used in traditional UCP approach.**

***Keywords*—.UCP (Use Case Point: it is one of the approach of efforts estimation), COCOMO (it is one of the approach of efforts estimation). FP (Function Point), TCP (Technical Complexity Factors), EF (Experience Factor), Efforts Estimation, EAF (Efforts Adjustment Factors) Cost Drivers,**

I. INTRODUCTION

Several estimating models have been developed over the years. Those preceding Use Case Point (UCP) and forming the basis for the UCP model include Function Point Analysis and the Constructive Cost Model. Function Point Analysis (FPA) was a valuable technique developed by A. J. Albrecht, in 1977. FPA assigns a point to each function in an application. Various modifiers then act on the function points in order to adjust for the product environment. Modifiers typically included applying weighted percentages or multipliers that would directly increase or decrease the point values. Environment factors included modifiers for complexity of technical issues, developer skill level, and risk. One problem organizations attempting to use this method would run into was consistent definition of a function and consistent definition of environmental factors across multiple projects and multiple development languages. To produce reliably accurate estimates, FPA relies heavily on historical data to derive weighting values and modifiers. Software efforts estimation is one of important activity of software development. The Constructive Cost Model, also known as COCOMO, was created by Barry Boehm, in 1981. COCOMO used statistical returns to calculate project cost and duration within a given probability. The model sought to provide a tool for predictably estimating cost, and continues to evolve today under the sponsorship of the University of Southern California. The model was/is interesting and produced worthy merits in applying statistical analysis to the problem of cost estimating. However, a major defining point in statistics is sample set size. The underlying assumption for COCOMO (like FPA) is that a statistically significant historical database exists to drive the statistical factoring. This will become a common theme through many attempts to create estimating models. Software engineering teams are typically very good at collecting lists of bugs, but notoriously bad at gathering meaningful historical or statistically significant metrics useful in predicting future projects.

One after one three models of COCOMO given by Barry Boehm:

A. Simple COCOMO.
B. Intermediate COCOMO.
C. Advance COCOMO

A. Simple COCOMO: - It was the first model suggested by Barry Boehm, which Follows following formula:

$$\text{Efforts} = a*(\text{KLOC})^{b}$$

Here a and b are complexity factor.

TABLE I
Complexity Factors

Model	A	B
Organic (simple in terms of size and complexity	3.2	1.05
Semi-ditched (average in terms of size and complexity	3.0	1.12
Embedded (Complex)	2.8	1.20

B. *Intermediate COCOMO:*-Previous model does not include the factors which can affect the efforts. Intermediate COCOMO includes 17 factors that can affect the efforts estimation.

Efforts= $a*(KLOC)^{b}*EAF$
Here a and b are complexity factor.

TABLE III
Complexity Factors

Model	A	B
Organic (simple in terms of size and complexity	3.2	1.05
Semi-ditched (average in terms of size and complexity	3.0	1.12
Embedded (Complex)	2.8	1.20

Following are Efforts Adjustment Factors used in Intermediate COCOMO. Typical values for EAF range from 0.9 to 1.4.

TABLE IIIII
Efforts Adjustment Factors used in Intermediate COCOMO

Cost Driver	**Sample Project Value**	**Description**
DATA		Database size.
CPLX		Product complexity.
TIME		Execution time constraint.
STOR		Main storage constraint.
RUSE		Required reusability.
DOCU		Documentation match to life-cycle needs.
PVOL		Platform volatility.
SCED		Scheduling factor.
RELY		Required reliability.
TOOL		Use of software tools.
APEX		Application experience.
ACAP		Analyst capability.
PCAP		Programmer capability.
PLEX		Platform experience.
LTEX		Language and tools experience.
PCON		Personnel continuity.
SITE		Multisite development.

C. *Advance COCOMO*:-In advance COCOMO model no of efforts adjustment factors are increases, now it become 22.

Efforts= $a*(KLOC)^{b}*EAF$
Here a and b are complexity factor.

TABLE IVV
USE CASE CALCULATION

Model	A	B
Organic (simple in terms of size and complexity	3.2	1.05
Semi-ditched (average in terms of size and complexity	3.0	1.12
Embedded (Complex)	2.8	1.20

Following parameters not included in intermediate COCOMO:

TABLE V
Efforts Adjustment Factors used in Advance COCOMO other than Intermediate COCOMO

Scale Factor	**Sample Project Value**	**Description**
PREC	**nominal**	**Precedence.**
PMAT	**CMM Level I (upper)**	**Process maturity.**
TEAM	**nominal**	**Team cohesion.**
FLEX	**nominal**	**Development flexibility.**
RESL	**little (20%)**	**Architecture and risk resolution.**

In the mid-1990s, Jim Rum Baugh, Grady Booch and Ivar Jacobson of Rational Software Corporation developed the Unified Modeling Language (UML) as notation and methodology for developing object-oriented software. UML was incorporated into the Rational Unified Process (RUP) by Rational Software. Within UML is the concept of defining the requirements for software products with Use Cases. Around the same time, Gustav Karner, also of Rational Software Corporation, created a software project estimating technique based on Use Case Points, much the way that FPA assigns points to functions, and including statistical and weighted modifiers. Karner's technique is now incorporated into RUP. Use Cases, as defined by UML, describe the things actors want the system to do and have proven to be an easy method for capturing the scope of a project early in its lifecycle. For their use, the case study team liked being able to create estimates early in the project lifecycle as a way to respond to the needs of their customers. Additionally, they find Use Cases to be a more consistent artifact then functions upon which to base an early project estimate. However, like FPA and COCOMO, the accuracy of estimates created using the RUP UCP estimating technique is largely dependent on a sizable volume of relevant historical data.

In UCP approach estimation divided into three parts

A. Calculate no of Actors.
B. Calculate no of Use Cases
C. Calculate TCF and EF

A. Calculate no of Actors:-We use following table to calculate no of Actors used in project

TABLE VV
Actor Calculation

Actor Type	Description	Quantity	Weight Factor	Subtotal
Simple	Defined API		1	

Average	Interactive or protocol driven interface		2	
Complex	Graphical user interface		3	
Total Actor Points				

B. Calculate no of Use Cases:-We use following table to calculate no of Use Cases used in project

TABLE VIVI
Use Case Calculation

Use Case Type	Description	Quant ity	Weight Factor	Subtotal
Simple	Up to 3 transactions		5	
Average	4 to 7 transactions		10	
Complex	More than 7 transactions		15	
Total Use Cases				

UUCP =Weighted Actors + Weighted Use Cases
UCP=UUCP*TCF*EF
Calculate TCF (Technical Complexity Factor)
List of Technical factors where weight factor rate from 0-2 and project rating rate from 0-5

TABLE VIIVII
Technical Complexity Factors

Technic al Factor	Factor Description	Wight Factor	Project Rating	Sub Total
T1	Must have a	2		

	distributed solution			
T2	Must Respond to specific performance objective	1		
T3	Must meet end user efficiency desired	1		
T4	Complex internal processing	1		
T5	Code must reusable	1		
T6	Must be easy to install	0.5		
T7	Must be easy to use	0.5		
T8	Must be portable	2		
T9	Must be easy to change	1		
T10	Include special security feature	1		
T11	Must provide direct access to third parties	1		
T12	Requires special user training facilities	1		
T13	Must allow concurrent user	1		
TOTAL				

TCF= (0.01 * TC factor) + 0.6
Calculate EF (EXPERIENCE FACTOR)

TABLE VIIIX
Experience Factors

Experience factor	Factor Description	Wight Factor	Project Rating	Sub Total
E1	Familiar with FTP software	1		

	Process			
E2	Application Experience	0.5		
E3	Paradigm Experience	1		
E4	Lead analyst capability	0.5		
E5	Motivation	0		
E6	Stable Requirements	2		
E7	Part time workers	-1		
E8	Difficulty of programming Language	-1		
TOTAL				

EF= (-0.03 *E factor) + 1.4

II. RESEARCH WORK

In our research we are including all factors of COCOMO in to UCP .Some parameters which is included in COCOMO not included in UCP. These are the parameter that can be including in UCP in our search

1. Database Size.
2. Documentation
3. Scheduling Factors
4. Use of software tools
5. Multi site Development.
6. Programmer Capability
7. Platform Experience
8. Personnel Continuity.

Parameter 1-5 considered as TCF and 6-8 considered as EF.

New Technical Complexity Factors with weight factors is follows:

TABLE X
Extended Technical Factors

Technical Factor	Factor Description	Wight Factor	Project Rating	Sub Total
T1	Must have a distributed solution	2		

T2	Must Respond to specific performance objective	1		
T3	Must meet end user efficiency desired	1		
T4	Complex internal processing	1		
T5	Code must reusable	1		
T6	Must be easy to install	0.5		
T7	Must be easy to use	0.5		
T8	Must be portable	2		
T9	Must be easy to change	1		
T10	Include special security feature	1		
T11	Must provide direct access to third parties	1		
T12	Requires special user training facilities	1		
T13	Must allow concurrent user	1		
T14	Database Size.	1		
T15	Documentation	1		
T16	Scheduling Factors	-1		
T17	Use of software tools	1		
T18	Multi site Development	-1		

New Experience Factors with weight factors is follows:

TABLE XIX
Extended Experience factor

Experience factor	Factor Description	Wight Factor	Project Rating	Sub Total
E1	Familiar with FTP software Process	1		
E2	Application	0.5		

	Experience			
E3	Paradigm Experience	1		
E4	Lead analyst capability	0.5		
E5	Motivation	0		
E6	Stable Requirements	2		
E7	Part time workers	-1		
E8	Difficulty of programming Language	-1		
E9	Programmer Capability	1		
E10	Platform Experience	1		
E11	Personnel Continuity	1		

III. RESULT

We have taken data from a small software development company. First we have estimated efforts by using old UCP method and estimated the efforts required to build the project. We have seen as usual estimated efforts were less than actual efforts and deviation (average deviation)) was % Result Shown in below table:

TABLE XIX
Case Study

PROJECT NO	ESTIMATED EFFORTS	ACTUAL EFFORTS	DEVIA TION %
A	1320	1584	20
B	880	1039	18
C	1080	1221	13
D	720	800	11

Now we taken same projects and estimated the efforts using our approach. We have seen deviation is fall and it is % only. So have seen a improvement of % in estimation.

TABLE XIIXI
Case Study

PROJECT NO	ESTIMATE D EFFORTS	ACTUAL EFFORTS	DEVIATI ON %
A	1426	1584	11
B	938	1039	11
C	1130	1221	08
D	758	800	06

IV. CONCLUSION

In this paper we have tried to reduce the % of deviation by using some extra factors. Although efforts estimation can never become a exact science, but we tried to minimize error of effort estimation. We cane see from result around 7% deviation is reduces. We have tried to include eight extra factors. We have assigned a weight factor this parameters, this values are assign on trail and error basis, you can change this values for your project than you may change.

REFERENCES

[1] Vahid Khatibi, Dayang N. A. Jawawi "Software Cost Estimation Methods: Review", Journal of Emerging Trends in Computing and Information Sciences Volume 2 No. 1 January 2011..

[2] Boehm," Software Engineering Economics", Prentice Hall, 1981.

[3] Chiu , N.H., Huang, S.J., "The adjusted analogy-based software effort estimation based on similarity distances", Journal of Systems and Software 80 (4), 628–640.2007

[4] Cuadrado-Gallego, J. J., Rodri, et al. "Analogies and Differences between Machine Learning and Expert Based Software Project Effort Estimation". Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 11th ACIS International Conference

[5] Roger E Masse "An Analysis of the Evolution of COCOMO and Function Point" July 8 1997. http://www.rogermasse.com/papers/software-metrics/

[6] *Edward R Carroll "Estimating Software Based on Use Case Point" October 2005 Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications.*

[7] Rum Baugh, J, Jacobson, I., and Booch, G, the Unified Modeling Language Reference Manual, Addison Wesley,Boston, MA, 1999.