ISSN: 0000-0000 (print), 0000-0000 (online)

AIMS Journal

Journal homepage: www.jdconline.net/aims

Regular Paper

Intelligent Software Cost Estimation in Centralized and Distributed Environment

Mahwish Anwar[a], Rizwan Anjum[b]

[a]*First Point Group, Diera, 25570, Dubai*
[b]*University College of Engineering, The Islamia University of Bahawalpur, 63100, Pakistan*

ARTICLE INFO

Article history:
Received 23 January, 2017
Revised 17 April, 2017
Accepted 03 Mary 2017

Keywords:
Software Cost Estimation Models,
Use Case Point Model,
Centralized & Distributed Environment

ABSTRACT

Software cost estimation is a critical process which foretells the amount of efforts and time needed to develop a software system. It helps the software industries to analyze the feasibility and workability of a project in a particular environment. Making cost estimation of a project is necessary; therefore, realizing its importance we have put the comparative study of different environment of cost estimation for a medium size project. This study involves the quantitative analysis of the project using distributive and centralized approach. We have used Use Case Point Model and Function Point Model for centralized software development environment and Advance COCOMO model is used for distributed environment. The calculated cost obtained from these environments is then compared to the original cost and then it is considered that which of these costs is much closer to the original cost. From this comparative study it is observed that which environment is better for the systems like web based application.This study depicted the results favoring the distributive environment. It provides us information about the suitability of software cost estimation models in either environment.

1. Introduction

The Cost Estimation rests on the information which is available at the time for the development of software. In the literature of software management, many models for cost estimation were presented but most of them had some drawbacks and became inapplicable due to the quick advancement of technology. It is really difficult to get an exact estimation on which one can relay because software systems lacks the detailed data about the upcoming software at such initial stages also because of the un localized placement of various constituents of the software which is developed. A model for software cost estimation which works on ideality should give firm confidence, brief description and accuracy for its estimation. There are some cost estimation models e.g. COCOMO [10, 11] and some sizing procedures e.g. Function Point analysis is known to the professionals and are used in a wide range in software engineering. Proper software cost estimation gives the foundation for planning and control of the software and good and effective planning and control can results the useful data for the study of software cost estimation model. Purpose of this research is to carry out a comparative study on different software cost estimation models in centralized and distributed environments. For this purpose a web based application system has been developed and its cost is estimated for both environments. We have used Use Case Point Model and Function Point Model for centralized software development environment and Advance COCOMO model is used for distributed environment. The calculated cost obtained from these environments is then compared to the original cost and then it is considered that which of these costs is much closer to the original cost. From this comparative study it is observed that which environment is better for the systems like web based application.

* *Corresponding author.* Tel.: +0-000-000-0000 ; fax: +0-000-000-0000.
E-mail address: *mahwish.anwer@hotmail.com*
http://dx.doi.org/.....

Overall, main contributions of this research are: First, we develop a system for which the cost has to be determined by using different cost estimation models. Second, different models have been considered for centralized and distributed model. Cost estimation has been performed on different modules of the project. Third, the both estimated costs are then compared to the original cost of the system. Lastly, the comparison of cost shows that which environment applies less cost for the system like web based application system which is considered here in this comparative study. Although there is much difference in calculated costs and original cost, but the study shows the suitable environment for considered system.

2. Related Work in Software Cost Estimation

Subhasis et al., has worked on the cost estimation of Distributed Systems. In Software Engineering, the models such as COCOMO (Constructive Cost Model) and Sizing methods such as Function Point Analysis are widely known and are mostly used. [4] E.N.C Nanjangud et al., have worked on the requirement of an integrated formal model for the analysis of GSD (global software development). [3] [13] [14] As the trend of adapting the approach of Global development for cost cutting a less development efforts is increasing rapidly but the fact is that there are many hidden costs like hands off between different sites, integration of software modules, synchronization and distribution factors which is also an overhead. To that end, the Nanjangud et al., have worked and present a model of integrated formal methods for the analysis of global software development method [3] Arshid et al., proposed the ER model estimating the cost of a software projects by using pathway density. [1]

Iman et al., presented a model in which they focused on the neural networks approach to determine to cost of software by keeping the merits of COCOMO model and according to them it can be noticed that the accuracy of cost estimation increases effectively by using this neural network model. [2] [8] [9]

3. Web based Application (Wasaib) Whose Cost Has to Be Estimated: A Case Study

Wasaib is an online community designed to make your social life more active and stimulating. New and existing contacts can be maintained by the social network of Wasaib. The people who are never met before can also reach out to each other and can establish relationships here. We can send messages, upload pictures, write notes, and post arbitrary advertisements on each other's home. It is easy to find the people of similar hobbies on the plate form of Wasaib. It is very easy to join Wasaib community portal. The only thing to join the network is valid email address. Project scope includes the following features:

a) Admin Panel b) User Panel c) User Panel

- **Home**

Home is the main page of an account. Here 6 of all friends are shown. A search point for finding the friends is present here. The comments are posted here. And on the left side of the home page there are more links to get access for the different forum of the current user's account like pictures profile etc.

- **Profile** (key details like your name, photo and location.)
  - Edit Profile
  - View Friend's Profile
- **Friends** (List of friends is shown here)
  - Delete Friends
  - Search Friends
- **Messages** (Private messages can be sent to any of friends)
  - Inbox
- **Scraps/Comments** (Scraps can be written on self-home as well as friend's home.)
- **Photos** (Pictures can be posted on home)
- **Videos** (Videos can be uploaded on user's home)
- **Logout** (Logout of the system)

This system has been developed by using PHP and MySQL. For Web based systems MySQL is a very good databases and its combination with PHP is even better because PHP is a free source.

4. Implementation of Software Cost Estimation Models in Centralized and Distributed System

Time calculated = 6310 minutes = 105.2 hours

Extra time given = 5 hours

Total Time =110.2hours 1)

Total modules (use-cases) = 9

Time for each use-case = 110.2/9 = 12.25 hours ~ 13 hours

Complete Project

File Count=61, File Size=111k, KLOC Rating=3.387, Average K-LOC=0.056

5. Centralized Environment

5.1. Software Cost Estimation using Use Case Model

While working on Object based Systems, which afterwards became associated with Rational Software and then IBM, Gustav Karner developed the technique of UCP in 1993. This method was employed to find solutions of sizes for object oriented software. This method was designed specifically for Use Case based object oriented systems, which basically is similar to system concept as the function Point (FP) estimation procedure [5] [6] [7].

First of all we have to estimate the size of code (size estimation). The basis of software cost estimation lies in the following contents [17] [18] [19] [20].

- Unadjusted Use Case Weight (UUCW) – It is the point size of software which results in number and density of Use Cases.
- Unadjusted Actor Weight (UAW) – It is the point size of software that gives multiple and complicated actors.
- Technical Complexity Factor (TCF) – It is used to accommodate the size according to Technical consideration.
- Environmental Complexity Factor (ECF) – It is used to accommodate the size according to Environmental consideration.

If we estimate and calculate the above mentioned factors the ultimate size estimate can be made. This final value is called Use Case Point (UCP) for development of a software project. The following Use Case Diagram of the System for which Cost estimation is required.

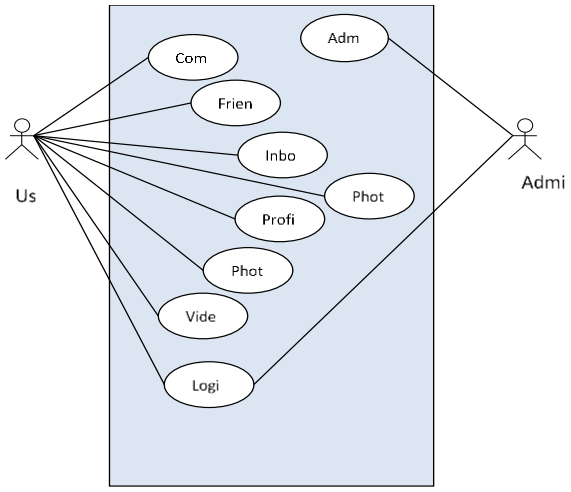

Fig 1. System Use Case

5.2. Unadjusted Use Case Weight (UUCW)

The UUCW is a factor that adds to the size of the under developing software. It is estimated according to the number and intricacy of the Use Cases for the system. Finding a UUCW for a system involves identification a categorization of Use Cases as Simple, Average or Complex. This classification is based on number of transactions included in Use Case. There is a fore defined weight of every classification. [16]

When we have classified these cases as simple average or complex, the overall weighted UUCW is calculated by adding the respective weights of each case. Following table depicts the various categorizations of Use Cases depending on number of transactions and weight values given to each Use Case within the classes.

Table 1: Use Case Classification

Use Case Class	No. of Transactions	Weight	No. Of use Cases (in Proposed Model) out of total 9
Simple	1 to 3	5	4
Average	4 to 7	10	4
Complex	8 or more	15	1

Table 1 shows the classification of the system Use Cases. According to the classification, weight is assigned to the Use Cases and the total weight UUCW is calculated. To measure the UUCW, the identification and definition of number of transactions for each Use Case is required. The online Community portal Use Case Representation diagram is showing that there are 9 Use Cases for the system considering 4 of these Use Case 4 are Simple, 4 are average and is Complex.

UUCW = (Total No. of Simple Use Cases x 5) + (Total No. Average Use Cases x 10) + (Total No. Complex Use Cases x 15)

For the Online Community Portal, the UUCW = (4 x 5) + (4 x 10) + (1 x 15) = 75

UUCW = 75

5.3. Unadjusted Actor Weight (UAW)

It is another factor which adds to the size of the software being developed. It is estimated depending on the frequency and complexity of the actors for the system. Like UUCW every actor should be recognized and categorized as simple average or Complex based on the type of actor. The Weight of every class is to be defined first. UAW is the Sum of Contributions of each of the actors. The given Chart tells the Various Categorizations of Actors and the Assigned Weight age Value.

Table 2: Actors Classification

Actors Class	Type Of Actors	Weight
Simple	Simple Actors are External system that must interact with the system using a well-defined Application Programming Interface	1
Average	External system that must interact/communicate with the system using some communication protocols (e.g FTP, TCP/IP, HTTP, database)	2
Complex	Complex actors are Human actors who use a GUI application interface	3

Table 2 show the classification of the Actors involved in system. According to the classification, weight is assigned to each type of actors and the total un-adjusted actor weight (UAW) is calculated. In our system, there are 2 Complex Actors, one is administrator and the other is the user who creates the account and links with online community.

UAW = (Total No. of Simple actors x 1) + (Total No. Average actors x 2) + (Total No. Complex actors x 3)

For the Online Community Portal, UAW = (0x 1) + (0 x 2) + (2 x 3) = 6, **UAW = 6**

5.4. Technical Complexity Factor (TCF)

To state the technical importance of a system, it is applied to the approximate size of the software. Then it is ranked between 0 and 5.rank zero states that the factor is of no use, while rank 5 states that it is inevitable. There are, 13 such technical factors provided in the table, this rank the s then multiplied by the already defined value for each factor. The total of all the estimated values is the TF. We then use this TF to obtain the TCF by the give formula

TCF = 0.6 + (TF/100)

Table 3: Technical Factor

Factors	Description	Weight	Assigns Value	Weight x Assigned Value
T1	Distributed system	2.0	0	2
T2	Response time/performance objectives	1.0	3	3
T3	End-user efficiency	1.0	2	2
T4	Internal processing complexity	1.0	2	2
T5	Code reusability	1.0	3	3
T6	Easy to install	0.5	0	0
T7	Easy to use	0.5	4	2
T8	Portability to other platforms	2.0	0	0
T9	System maintenance	1.0	3	3
T10	Concurrent/parallel processing	1.0	2	2
T11	Security features	1.0	4	4
T12	Access for third parties	1.0	2	2
T13	End user training	1.0	1	1
			Total TF	**26**

Table 3 calculates the value of different technical Factors involved in the system. A value ranging from **0 to 5** is assigned to each factor and this value is then multiplied to the weight of each factor. Total of their product results the value of Technical Factor (TF). Next, the TCF is calculated

TCF = 0.6 + (TF/100)

For the Online Community Portal System, TCF = 0.6 + (26/100) = 0.86, **TCF = 0.86**

5.5. Environmental Complexity Factor (ECF)

It is a factor which is applied to the estimated size of the software to give the information about environmental assumptions of the system. It is estimated by giving a score between 0 [No experience] and 5[Expert]. To each of the eight environmental factors given in the table, this score is then

multiplied by the already known weighted value for each factor. The sum of all estimated values is EF. The EF is then implemented to calculate the ECF with the following method.

ECF = 1.4 + (-0.03 x EF)

For the calculation of ECF a value is given to every environmental factor depending on the experience level of the team. The following diagram depicts the values given for Online Community Portal. The values are then multiplied by weighted values and total ECF is calculated.

Table 4: Environmental Factor

Factors	Description	Weight	Assigned Value	Weight x Assigned Value
E1	Familiarity with development process used	1.5	2	3
E2	Application experience	0.5	5	2.5
E3	Object-oriented experience of team	1.0	2	2
E4	Lead analyst capability	0.5	2	1
E5	Motivation of the team	1.0	5	5
E6	Stability of requirements	2.0	5	10
E7	Part-time staff	-1.0	0	0
E8	Difficult programming Language	-1.0	2	-2
	Total EF			**21.5**

Table 4 calculates the value of different Environmental Factors involved in the system. A value ranging from 0 to 5 is assigned to each factor which depends on the expertise of the software engineer. Value is assigned 0 if the software engineer is not experienced and 5 if the software engineer is expert. This value is then multiplied to the weight of each factor. Total of their product results the value of Environmental Factor (EF). Next, the ECF is calculated:

ECF = 1.4 + (-0.03 x EF)

For the Online Community Portal, ECF = 1.4 + (-0.03 * 21.5) = 0.755, **ECF = 0.755**

5.6. Use Case Points (UCP)

At last the Use Case Point (UCP) can be estimated if the unaccommodated project size (UUCW and UAW) Technical Factor (TCF) and environmental Factors (ECF) have been calculated. The UCP is calculated depending on the formulae given bellow. [17] [18] [19] [20]. If we have calculated above four factors the UCP can be calculated by the given formula.

UCP = (UUCW + UAW) x TCF x ECF = (75 + 6) x 0.86 x 0.755 = 52.5933

For the Online Comunity Portal, the overall estimated size for the development of software is 52.6 Use Case Points. The Project size is determined, now the total labour for the project can be calculated. For the Online Community Portal example, 38.34 man hours per use case point will be used as calculated above.

Estimated Effort = UCP x Hours/UCP

For the Online Community Portal System, Estimated Effort = 52.5933x 38.34

Estimated Effort = 2016.42 ~ 2016 hrs

Effort is estimated in person hours so result is

Effort = 2016 hours

Function Point Analysis: For centralized software development environment, we have used Use-Case point model before, but the result are considerably different from which are expected. So we use anther model i.e. Function Point Analysis Model to measure the effort for centralized environment of software development.

Step 1: The Value of Counting Function Points (FP)

Function point analysis have many benefits. Once when the history of the application under consideration we can also have FP counts for those applications, we can now add these capabilities to our software development arsenal: [21],[22],[23]

1. The ability to accurately estimate:
    A. Cost of project
    B. Duration of Project
    C. Best size of project staffing
2. The capacity to determine other metrics, such as:
    A. Project defect rate
    B. Cost per FP
    C. Function Point's per hour (a productivity rate)
    D. The productivity benefits of using new or different tools

Step 2: Count Data Element Type (DET)

Relational Database Tables are used to store the data for Wasaib Application. Wasaib (Web Based Application) contain the following tables

1. Users *(Table 5)*

2. Album view *(Table 6)*
3. Photos *(Table 7)*
4. Friend request *(Table 8)*
5. Email *(Table 9)*
6. Scraps *(Table 10)*
7. Videos *(Table 11)*

The table contain the full list of the fields that are stored in each table. The data provided by the tables include Field name, Comments, Notes and the indicators for DET. Last Row of each table gives the total number DETs.

Table 5: DET count for "Users"

Field	Count as DET	Notes
Id	No	This is a technical Object. User can not recognize it so it is not counted.
Username	Yes	
Userid	Yes	
Passwd	Yes	
Email	Yes	
Sex	Yes	
Dob	Yes	
Sunsign	Yes	
Rstatus	Yes	
College	Yes	
Country	Yes	
City	Yes	
Picture	Yes	
Status	Yes	
Total DETs	**13**	

Table 6: DET count for Album view

Field	Count as DET	Notes
Id	No	This is a technical Object. User can not recognize it so it is not counted.
Userid	Yes	Counting a DET for data that are required by user to establish relationship with other ILF/EIF. Foreign keys fit this definition.
Aid	Yes	To establish relationship with other ILF/EIF user requires DETs. Foreign keys are suitable to define this.
Pics	Yes	
Picdescription	Yes	
Total DETs	**4**	

Table 7: DET count for "Photos"

Field	Count as DET	Notes
Aid	No	This is a technical Object. User can not recognize it so it is not counted.
Userid	Yes	To establish relationship with other ILF/EIF user requires DETs. Foreign keys are suitable to define this.
Albumpic	Yes	
Albumname	Yes	
Description	Yes	
Total DETs	**4**	

Table 8: DET count for "Friendrequest"

Field	Count as DET	Notes
id	No	This is a technical Object. User can not recognize it so it is not counted.
Userid	Yes	To establish relationship with other ILF/EIF user requires DETs. Foreign keys are suitable to define this.
Frienduserid	Yes	To establish relationship with other ILF/EIF user requires DETs. Foreign keys are suitable to define this.
Status	Yes	

Total DETs	**3**

Table 9: DET count for "Email"

Field	Count as DET	Notes
ID	No	This is a technical Object. User can not recognize it so it is not counted.
Message	Yes	
Subject	Yes	
From	Yes	
To	Yes	
Date	Yes	
Status	Yes	
Total DETs	**6**	

Table 10: DET count for "Scraps"

Field	Count as DET	Notes
id	No	This is a technical Object. User can not recognize it so it is not counted.
Userid	Yes	To establish relationship with other ILF/EIF user requires DETs. Foreign keys are suitable to define this.
Touserid	Yes	To establish relationship with other ILF/EIF user requires DETs. Foreign keys are suitable to define this.
Comment	Yes	
Status	Yes	
Total DETs	**4**	

Table 11: DET count for "Videos"

Field	Count as DET	Notes
id	No	This is a technical Object. User can not recognize it so it is not counted.
Userid	Yes	To establish relationship with other ILF/EIF user requires DETs. Foreign keys are suitable to define this.
Video	Yes	
Descrip	Yes	
Total DETs	**3**	

Step 3a: Count the ILF

Let think of these seven tables as separate ILFs, because of the number of DETs they contain, each table would be considered to be a "Low" complexity table. Because each Low table is worth 7 FPs, the FP count for these six tables would be 49 FPs (7x7=49)

Table 12: ILF Complexity Matrix

Record Element Type (RETs)	Data Element Type (DETs)		
	19-Jan	**20-50**	**50+**
1	L	L	A
2 to 5	L	A	H
6 or more	A	H	H

Table 13: ILF Weights

Complexity	Points
Low	7
Average	10
High	15

Table 14: Total FP counts due to ILF

ILF	No. of RETs	No. of DETs	Complexity	Function Points
Users	1	13	Low	7
Albumview	2	4	Low	7
Photos	1	4	Low	7

Friendrequest	1	3	Low	7
Email	1	6	Low	7
Scraps	1	4	Low	7
Videos	1	3	Low	7
Total Function Points Counts				**49**

5.7. 3b: Determine the count resulting from EIF

In this application there are no EIFs, so EIFs contribute zero FPs to the overall FP count.

3c: Determine the count resulting from EI

Table 15: Count resulting from EI

FTR's	**Data Element Types (DET's)**		
	1-4	**5-15**	**16+**
0-1	L	L	A
2	L	A	H
3 or more	A	H	H

Table 15 describes EI complexity matrix to carry our example forward, as we can see from the following table, an Average complexity EI is worth 4 FPs.

Table 16: EI Weights

Complexity	**Points/Weight**
Low	3
Average	4
High	6

Table 17: External Inputs

Function	**Page**	**EI**	**DET-FTR** (Resulting complexity) by using above table	**EO**	**DET-FTR** (Resulting complexity) by using above table	**EQ**	**DET-FTR** (Resulting complexity) by using above table
			Comments				
Add Comment	Insertcomment.php					yes	Low
View home page comments	Home.php			Yes	Low		
Full View of Comment	Fullview.php			Yes	Low		
Delete Comment	Delscrap.php					yes	Low
Friends							
Search Friend	Searchfriend.php					yes	Low
Accept Friend Request	Acceptfriend.php					yes	Low
Deny Friend Request	Denyfriend.php					yes	Low
Add Friend	Requestfriend.php	Yes	Low				
View Friends	Friendlist.php			Yes	Low		
Delete Friends	Delfriend.php			Yes	Low		
Inbox (Personal Message)							
Compose Message	Email.php	Yes	Low				
Send Inbox message	Insertemail. p					yes	Low
View Sent Messages	Viewsentemail.php			Yes	Low		
View Received Messages	Viewemail.php			Yes	Low		
Delete Messages	Delemail.com					yes	Low
Photo Album							
Add Photo Album	Insertalbum.php					yes	Low
View Photo Album	Photosnew.php			Yes	Low		
Delete Photo Album	Delphotoalbum.php					yes	Low
Photos							

Add Photo	Insertpic.php					yes	Low
Upload Photo	Uploadpic.php					yes	Low
View Photos	Albumpicz.php			Yes	Low		
Edit Photo	Editphoto.php					yes	Low
Delete Photos	Delpic.php					yes	Low
Videos							
Add Video	Insertvideos.php					yes	Low
View Videos	Videos.php			Yes	Low		
Delete Videos						yes	Low
Login/ Logout							
Sign In/ Sign Up Index.php	Signin	Yes	Low				
Enter Login Details to Database	Enterdata.php	Yes	Low				
Session	Connection.php						
Sign Out	Signout.php						
Profile							
View Profile	Profile.php			yes	Low		
Edit Profile	Editpprofile.php	Yes	Low				
Update Profile	Updateprofile.php					yes	Low
Edit Profile Picture	Editpicture.php	Yes	Low				
Admin Panel							
Delete Records	delrec.php					yes	Low
Show Records	showrecord.php			yes	Low		
Update Records update.php	update.php					yes	Low
Delete Comments	deletecomments.php					yes	Low
Edit Comments	editcomments.php	Yes	low				
Update Comments	Updatecomments.php	Yes	low				
Show Comments	Showcomments.php			yes	Low		
Edit Users	Edituser.php	Yes	low				
Total		**10**	**9 x 3 = 27**	**12**	**12 x 4 = 48**	**19**	**19 x 3 = 57**

Step 3d: Determine the count resulting from EO's

Allocating FP's to EO's is very similar to the process for EI's. Again, we perform our lookup using DET's and FTR's, with a resulting Low/Average/High complexity.

Table 18: DET-FTR

FTR	**Data Element Types (DET)**		
	1-5	**6-19**	**20+**
0-1	L	L	A
2-3	L	A	H
4 or more	A	H	H

"To carry our example forward, using the table that follows, we will see that an Average complexity EO has a value of 5 FPs."

Table 19: EO Weights

Complexity	**Points/Weight**
Low	4
Average	5
High	7

Step 3e: Determine the count resulting from EQ's

Table 20: EQ Complexity Matrix Weights

FTRs	Data Element Types (DETs)		
	1-5	6-19	20+
0-1	L	L	A

2-3	L	A	H
4 or more	A	H	H

"Carrying our EQ example forward, from the table below it can be found that a High complexity EQ is worth 6 FPs."

Table 21: EQ Weights

Complexity	Points/Weight
Low	3
Average	4
High	6

As the values of the ILFs, EIFs, EIs, EOs, and EQs in the application have been determined, we add up each of the individual counts to get a total unadjusted function point count for the application. This is shown in the table that follows.

Table 22: Un-adjusted Function Point Count

Function Type	Section Total
ILF	49
EIF	0
EI	27
EO	48
EQ	57
Un Adjusted Function Point Count	**181**

Table 22 describes the unadjusted function point count for the Wasaib Community portal application the total unadjusted function point count for this application (at this date and time) is 99 function points.

Step 4: Determine the Value Adjustment Factor (VAF)

Here are a few definitions and facts:

- "There are 14 "General System Characteristics", or GSCs in Value Adjustment Factor (VAF)."
- "These GSCs represent characteristics of the application under consideration. Each is weighted on a scale from 0 (low) to 5 (high)."
- "When you sum up the values of these 14 GSCs you get a value named `` Total Degree of Influence", or TDI. As you can see from the math the TDI can vary from 0 (when all GSCs are low) to 35 (when all GSCs are high)."

Below is the list of 14 GSCs:

Table 23: Value Adjustment Factor Count

General System Characteristics GSCs	Assigned Value from 0 (Low) to 5 (High)
Data Communication	0
Distributed Data Processing	0
Performance	1
Heavily used configuration	0
Transaction rate	1
Online data entry	4
End User Efficiency	1
Online Update	1
Complex Processing	1
Reusability	1
Installation Ease	0
Operational Ease	4
Multiple Sites	0
Facilitate Change	0
Total Degree of Influence TDI	**14**

Given this background information, you can see with the following formula:

$$\mathbf{VAF = (TDI \times 0.01) + 0.65}$$

That VAF can vary in range from 0.65 (when all GSCs are low) to 1.35 (when all GSCs are high).

$$VAF = (14 \times 0.01) + 0.65 = VAF = 0.79$$

Step 5: Calculate the Adjusted Function Point Count

The final step in our five-step process is to determine the Adjusted Function Point Count which can be easily determined by the equation given below:

"Adjusted FP Count = Unadjusted FP Count * VAF"

Adjusted FP Count = 181 * 0.79 = 142.99 ~ 143

"As we saw in the previous section, the VAF can vary from 0.65 to 1.35, so the VAF exerts an influence of +/- 35% on the final Adjusted FP Count."

Step 6: Calculate the Effort

Let in this particular case study, a programmer works on 30 functions per month.

Effort in Person Month = FP divided by no. of FP's per month

Effort in person-month = 143/35 = 4.08 person month
Convert then in hours as:
1person-month = 176 hours of Effort
Effort = 4.08 x 176 = 718.08 ~ 718 hours

5.8. Software Cost Estimation using COCOMO Models

Boehm [10] proposed this widely used family of models. In the COCOMOs, the code-size *S* is given in Thousand Line of Code (KLOC) and Effort is in person-months. **COCOMO** relates the software development cost and effort to the program Size as its function. Program Sizes expressed in various lines of codes. [Source Line of Code Kilo Lines of Code [10] [11] [12] [15] [20]. COCOMO applies to three degrees of software projects: [10] [11] [12] [15] [20]

- Organic projects – ""small" teams with "good" experience working with "less than rigid" requirements"
- Semi-detached projects – ""medium" teams with mixed experience working with a mix of rigid and less than rigid requirements"
- Embedded projects – "developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects (hardware, software, operational) Basic COCOMO is beneficial for the accurate and quick estimation of software cost. But, COCOMO does not give the draw backs and differences in the techniques, usage of personnel quality and experience".

Table 24: Type of System under Consideration

Software project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Table 24 values are specified for the coefficients for each degree of software project. Values are chosen and applied in calculations according to the nature of system. As Our Proposed System is **Organic** and the **KLOC** calculated for this system is **0.056.**

Table 25: Type of System under Consideration

Cost Drivers		**Ratings**					
		Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
	Product attributes						
RELY	Required software reliability	0.75	0.88	1	1.15	1.40	
DATA	Size of application database		0.94	1	1.08	1.16	
CPLX	Complexity of the product	0.7	0.85	1	1.15	1.3	1.65
	Hardware attributes						
TIME	Runtime performance Constraint			1	1.11	1.3	1.66
STOR	Memory constraints/Main Storage			1	1.06	1.21	1.56
VIRT	Volatility of the virtual machine environment		0.87	1	1.15	1.3	
TURN	Required turnabout time		0.87	1	1.07	1.15	
	Personnel attributes						
ACAP	st capability	1.46	1.19	1	0.86	0.71	
AEXP	Applications experience	1.29	1.13	1	0.91	0.82	
PCAP	Software engineer capability	1.42	1.17	1	0.86	0.7	
VEXP	Virtual machine experience	1.21	1.1	1	0.9		
LEXP	Programming language Experience	1.14	1.07	1	0.95		
	Project attributes						

MODP	Application of software engineering methods	1.24	1.1	1	0.91	0.82
TOOL	Use of software tools	1.24	1.1	1	0.91	0.83
SCED	Required development Schedule	1.23	1.08	1	1.04	1.1

5.9. Advance COCOMOs

It is among the most famous and old techniques of effort estimation. In this technique, the line of code is predicted we are to predict the line of code. When we are facing and handling a new and unfamiliar project, the most difficult task is the counting of Line of Code. In the above mentioned situation we need to divide the project into various modules and the modules into sub modules in order to minimize the complexity of the problem. The most experienced person of team should take the responsibility to count KLOC because we need a correct estimation of KLOC before writing. Among all the 15 characteristics, each of them gets a rating on a 6-point scale that ranges from "very low" to "extra high".

An effort multiplier from the table below applies to the rating. The product of all effort multipliers results in an effort adjustment factor (EAF). Typical values for EAF range from 0.9 to 1.4. Now the Intermediate Constructive Cost Model (COCOMO) formula will be of the form:

$$\mathbf{E=}a_i\mathbf{(KLOC)}^{(b_i)}\mathbf{.EAF}$$

Where E = effort applied [person-months],

KLOC = estimated number of thousands of delivered lines of code for the project,

EAF is the factor which we have calculated above.

The coefficient $\mathbf{a_i}$ and the exponent $\mathbf{b_i}$ are given in the next Table 7". "E is used in the development time D calculation exactly in the same way as it was used in Basic Constructive Cost Model." [10] [11] [12] [15] [20].

5.10. Detailed COCOMO

In detailed COCOMO, all the characteristics are access at each step of software development process like Analysis, Design etc. It uses multiple multipliers for each attribute of cost driver.

Different effort multipliers are used by the detailed model for each cost driver attribute. So the amount of effort at each step of development can be calculated. To calculate the Effort of complete software, the software system is divided into multiple modules and then we estimate the cost and effort for each module by using COCOMO and then sum the efforts. Detailed COCOMO uses Function of program size and cost to calculate the efforts in accordance with the Software development life cycle. [10] [11] [12] [15] [20]

EAF= 1.0117 (where normal value ranges from 0.9 to 1.4

$E=a_i(KLOC)^{(b_i)}.EAF$

$E= 3.2 \times (0.053)^{1.05} \times 1.0117$

Table 26: Effort Calculation

Module No.	Module Name	Avg KLOC	Calculated Effort E [person-month]	Calculated Effort [E] hrs [person-hour]
1	Comments	0.053	0.148	26.048
2	Friends	0.034	0.093	16.368
3	Inbox	0.039	0.107	18.832
4	Photo Album	0.06	0.169	29.744
5	Photos	0.068	0.192	33.792
6	Videos	0.084	0.24	42.24
7	Login/Logout	0.022	0.059	10.384
8	Profile	0.064	0.181	31.856
9	Admin Side	0.037	0.148	26.048
Total			**1.337**	**235.312**

E= 1.337 [person-months]

Development Time (D) = $c_b(\text{Effort Applied})^{d}_{b}$ **[months]**

$= 2.5 \times (\mathbf{1.337})^{0.38}$

$= 2.8$

Convert it in hours as

1person-month = 176 hours of Effort

As 1person-day = 8 hours of effort by an average person

As calculated above, time in hours is

2.8x (8x22) = 492 hrs

6. Conclusion

To increase the chances of success of project many models have been introduced on the platform of software cost estimation. However, there is still a room for improvement in the current models of cost estimation. We have used the Use Case Point Model and Function Point Model for Centralized and COCOMO II for Distributed Software Development Environment in order to compare the accuracy of either model. This paper is based on implementation of above mentioned models on a Medium scale online web-based application and consists of some facts and figures which prove that COCOMO Models is more closely related to the Actual Cost. Now the practitioners can implement COCOMO Model on the basis of mathematical study showed in this paper as it is much more beneficial and is more relevant to the actual cost.

The System under consideration is divided into various modules based on functionality. Existing Software Cost estimation models have been focused in this paper. Software effort and Cost estimation models and metrics which have been used in this paper have also been presented in it. The quantitative analysis of these modules has been performed according to the corresponding models. The Facts and Figures of the quantitative analysis showed that those values which were obtained through Distributed Environment Models are more closely related to the Actual Cost of project.

REFERENCES

[1] Arshid Ali, Salman Qadri, Syed Shah Muhammad , Jalil Abbas, Muhammad Tariq Pervaiz, Sarfaraz Awan, "Software Cost Estimation through Entity Relationship Model", Report and Opinion, Vol. 2, no. 5, pp.36-40, 2010.

[2] Iman Attarzadeh, Siew Hock Ow, "A Novel Soft Computing Model to Increase the Accuracy of Software Development Cost Estimation", IEEE, Vol 3, Pp 603-607, 2010.

[3] Nanjangud C. Narendra, Karthikeyan Ponnalagu, Nianjun Zhou and Wesley M. Gifford, "Towards a Formal Model for Optimal Task-Site Allocation and Effort Estimation in Global Software Development", Service Research and Innovation Institute Global Conference, pp. 470-477, 2012.

[4] Subhasis Dash, Arup Abhinna Acharya, "Cost Estimation for Distributed Systems Using Synthesized Use Case Point Model", Advances in Computing, Communication and Control Communications in Computer and Information Science Volume 125, 2011, pp 161-169.

[5] Murali Chemuturi, Software Estimation Best Practices, Tools and Techniques for Software Project Estimators, J.Ross Publishing, 2009, pp. 84-87

[6] Dennis, Alan R., Barbara Haley Wixom, and David Tegarden. Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach, Third Edition, John Wiley & Sons, Chapter 5 - Functional Modeling, 2009,

[7] Dennis, Alan R., Barbara Haley Wixom, and David Tegarden. Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach, Fourth Edition, John Wiley & Sons, Chapter 2 - Project Management, 2012.

[8] Witting, G., Finnie, G., "Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development Effort", Journal of Information Systems, Vol.1, no.2, pp.87-94, 1994.

[9] N. Karunanitthi, D.Whitely, and Y.K.Malaiya, "Using Neural Networks in Reliability Prediction," IEEE Software Engineering,Vol.9, no.4, pp.53-59, 1992

[10] Boehm, B., An Overview of the COCOMO 2.0 Software Cost Model, 1999

[11] Boehm B., Abts, C., and Chulani, S., Software Development Cost Estimation Approaches – A Survey, University of Southern California Center for Software Engineering, Technical Reports, USC-CSE-2000- 505, 2000.

[12] Boehm B. W. "Software Engineering Economics", Englewood Cliffs, NJ, Prentice-Hall, 1981.

[13] James D. Herbsleb. Global software engineering: The future of sociotechnical coordination. In 2007 Future of Software Engineering, FOSE '07, pages 188–198, Washington, DC, USA, 2007. IEEE Computer Society.

[14] James D. Herbsleb and Audris Mockus. An empirical study of speed and communication in globally distributed software development. IEEE Trans. Software Eng., 29(6):481–494, 2003.

[15] Boehm, B.W. "Software engineering economics." IEEE transactions on software engineering, volume se-10, nr. 1, januari 1984.

[16] Hareton Leung and Zhang Fan, "Software Cost Estimation".

[17] Chetan Nagar, Software efforts estimation using Use Case Point approach by increasing Technical Complexity and Experience Factors, International Journal on Computer Science and Engineering, Vol. 3. No. 10, Page-3377-3345, Oct 2011.

[18] Matthias Kerstner," Software Test Effort estimation Methods", 2 February 2011.

[19] Bogdan Stepien,"Software Development Cost Estimation Meth- ods and Research Trends",Computer Science, Vol.5, 2003.

[20] Jyoti G. Borade , Vikas R. Khalkar ," Software Project Effort and Cost Estimation Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X, Vol. 3, Issue 8, pp. 730-739 ,August 2013.

[21] http://alvinalexander.com/FunctionPoints/

[22] http://www.fprecorder.com