I.J.Modern Education and Computer Science, **2013, 8, 1-7**
Published Online October 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2013.08.01

Evaluation of the Cost Estimation Models: Case Study of Task Manager Application

Mohammed Mugahed Al_Qmase, M. Rizwan Jameel Qureshi
Faculty of Computing and Information Technology, King Abdulaziz University, Saudi Arabia
Qumasi@hotmail.com, anriz@hotmail.com

***Abstract* —** The need to accurately estimate time and cost for effective planning of software projects is becoming crucial driven by the escalating demands of the software market. Several models proposed in the history of Software Engineering discipline to estimate time, costs associated with planning and managing software projects as Line of Code (LOC), Function Point (FP) and Constructive Cost Model (COCOMO). This paper focuses upon the COCOMO Model. It is further consisted of its two sub models called COCOMO I and COCOMO II. The primary objective of this research is to use an appropriate case study to evaluate the accuracy of the sub models COCOMO I and II and ascertain the variation of the realistic resource effort, staff and time. The findings to date show that the Application Composition Model of COCOMO II is more accurate in determining time and cost for the successful conclusion of a software project than the other two COCOMO I and II Models for a similar application for example Task Manager.

***Index Terms*—** COCOMO I, COCOMO II, Software Cost Estimation, Case Study, Sizing Methods

I. Introduction

Cost estimation is one of more challenging requirements of project management procedures. Basically it is a prediction methodology towards fine tuning the cost estimates for a successful conclusion of a project. For appropriate resource allocation, the accuracy and the efficiency in cost estimation are extremely important imperatives for keeping the development costs as within the budget envelope. Several models proposed for software cost estimation such as LOC, FP, COCOMO and Use Case estimations. These models use sophisticated mathematical methods towards cost evaluation convergence.

COCOMO is one of the more ubiquitous techniques available for investigating cost, effort, deployment of staff and ascertaining an accurate road map of precise time lines of the entire project. COCOMO I was first published in 1981 [1]. It is known to consist of two sub-models structured as COCOMO I (also referred as COCOMO'81) and of course COCOMO II. COCOMO II was introduced in 1995, featuring such attributes as cost estimation aimed at object oriented software development [2]. COCOMO I, in essence is formulated as a hierarchy of three sub-models geared towards the Basic, Intermediate and Advanced. These three sub-models in turn address the Organic, semi-detached and embedded modes of precise simulation. COCOMO II on the other hand comprises of a sequential assimilation of four sub-models individually addressing the Application Composition, Early Design, Reuse and Post-Architecture [1]. This paper contains the results of COCOMO I and COCOMO II applications a specific case study. For this type of higher level synthetic based simulation successive use of Java, JSP, JavaScript and Oracle are exploited at will.

The remainder of this paper is organized as follows: Section 2 describes the related work. Section 3 covers the details of COCOMO I and I. Section 4 illustrates sizing methods and cost drivers. Section 5 presents the research setting and analyzes the requirements of the project. Section 6 provides experiment and analysis. Conclusion is given in the final section.

II. Related Work

Boehm et al. [1] proposed evaluation criteria for the validity of the process models and they provided effective results. This article also explained the strengths and weaknesses of various cost estimation techniques for the period of 1965 to 2005 (40 years). COCOMO-II [2] was an excellent model up to 2005 but it was not equipped for the new requirements and development styles of the current software market to estimate project costs. COCOMO-II directed the software experts to create and designed new models such as the Chinese government version of COCOMO (COGOMO) and the Constructive Commercial-off-the-Shelf Cost Model (COCOTS) etc.

Boehm [3] discussed different software cost estimation techniques and highlighted various hot areas and challenges of research in the field of software cost estimation. In [3], it is emphasized that there was a need to research more in this field to open the new horizons for upcoming researchers. Nasir [4] discussed the strengths and weaknesses of various software estimation techniques to provide the basis for the exactness of software cost estimation. Basic Project Estimation Process was presented. The different types of models (derived from COCOMO I&II) were also discussed [4].

Reusability of components in Component Based Development (CBD) is illustrated in [5]. The research in [5] also discussed and compared different architectures of CBD. The detailed explanation of advantages and disadvantages of CBD is also elaborated. A comparison, of component based development (CBD) with other traditional software development practices, is also provided. Succi and Baruchelli [6] highlighted the importance of standardization of components for the software reusability. The major finding in [5] is how much total development cost of a software system affected due to usage of component-based software engineering. The main two factors those were affecting the standardization cost of a component have been explained. According to them, the cost of the standardization of component(s) must be included during the cost-benefit analysis of a software system. Gill [7] highlighted the pertinent issues of software reusability for component based development on the basis of CBSE, highlighted the important issues of software reusability and high level reusability guidelines. Gill [7] outlined the aspects of reusability from product reliability improvement and reduction in software development costs.

The problem of crosscutting produced during component development was elaborated in [8]. This problem was solved by the extension with Aspect oriented methodology. It was demonstrated by an example as to how new business rules resulted in the more adaptable and reusable components. Aspect Component Based Software Engineering developed with success in the CORBA Component Model domain [9]. Dolado [10] provides the validation of component-based method (CBM) by analyzing 46 projects. A relationship is also established (based on the analysis of 46 projects) between Kilo Line of Code (KLOC) and number of Component (NOC) by providing examples [10].

III. The Details of COCOMO I & II

COCOMO I (COCOMO'81) consists of three models [1].

1. **Basic COCOMO** is a static single-valued model that computes software development effort (and cost) as a function of program size expressed in estimated lines of code.
2. **Intermediate COCOMO** computes software development effort as a function of program size and a set of "cost drivers" that include subjective assessments of product, hardware, personnel, and project attribute.
3. **Detailed COCOMO** incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process.

Equations

COCOMO'81 models depend on the two main equations [9]:

$$\text{Effort (MM)} = a * (\text{KDSI})^{\wedge} b \qquad (1)$$

$$\text{Schedule (TDEV)} = 2.5 * (\text{MM})^{\wedge} c \qquad (2)$$

Coefficients a, b and c depend upon the mode of the development to determine the size and the complexity of the project. Following are the three modes that are applied to each of the models discussed previously [1].

1. **Organic Mode:** covers relatively small and simple software projects and be conducted by small teams with good application experience work for a set of less than rigid requirements.
2. **Semi-detached Mode:** is applicable for software projects must be carried out by teams with mixed levels of experience and deal with mixed requirements (set of rigid and less rigid requirements.
3. **Embedded projects:** covers software projects to be developed from a set of tight hardware, software and operational constraints.

COCOMO II is composed of the following four sub-models [2].

1. **The Application Composition model** involves prototyping efforts to resolve potential high-risk issues such as user interfaces, software/system interaction, performance, or technology maturity. This model used Application Points sizing method. It gathers application perspective consisting of a number of screens, reports and third generation language (GL) components.
2. **The Early Design model** involves exploration of alternative software/system architectures and concepts of operation. At this stage, requirements are not enough to support fine-grain cost estimation. The corresponding COCOMO 2.0 capability involves the use of function points and a small number of additional cost drivers [2]. It used FP sizing method. FP used number of inputs, outputs, inquiries, files and interfaces.
3. **The Reuse model** computes the effort of integrating reusable components. It uses how many of LOC reused or generated.
4. **The Post-Architecture model** is applied once the system architecture is designed and ample information is available about the system. This model works most effectively if software life-cycle architecture has been developed. It is

also based on LOC reused or generated. Figure 1 presents the four sub-models.

IV. Sizing Methods, Cost Drivers and Cost Estimations

COCOMO software cost estimation model requires sizing information (as input of estimation cost models). Three different sizing options are available as part of the model hierarchy: object points, function points, and lines of source code. The COCOMO II application composition model used object points. The object point is an indirect software measure computed using counts of the number of screens (at the user interface), reports, and 3GL components likely to be required to build the application.

Each object instance (e.g., a screen or report) is classified into one of three complexity levels (i.e., simple, medium, or difficult) using criteria suggested by Boehm [2]. In essence, complexity is a function of the number and source of the client and server data tables that are required to generate the screen or report and the number of views or sections presented as part of the screen or report. Drivers are particularly helpful to the estimator in order to understand the impact of different factors that affect project costs COCOMO has 7 to 17 multiplicative factors that determine the effort required to complete a software project. All cost drivers have qualitative rating levels ('extra low' to 'extra high') that express the impact of the driver and a corresponding set of effort multiplier.

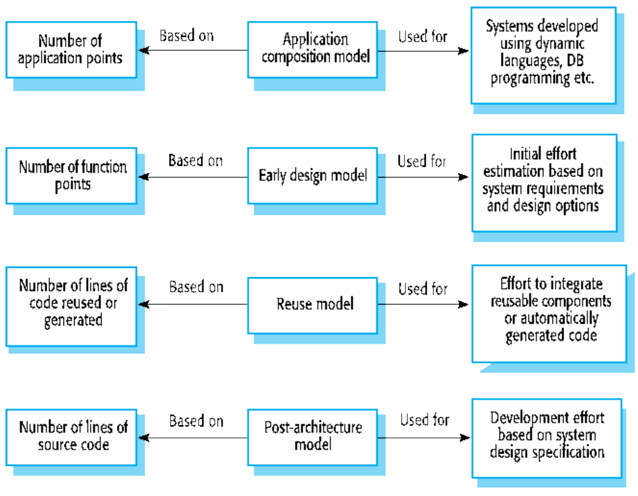

Number of application points
Based on
Application composition model
Used for
Systems developed using dynamic languages, DB programming etc.
Number of function points
Based on
Early design model
Used for
Initial effort estimation based on system requirements and design options
Number of lines of code reused or generated
Based on
Reuse model
Used for
Effort to integrate reusable components or automatically generated code
Number of lines of source code
Based on
Post-architecture model
Used for
Development effort based on system design specification

Fig. 1: COCOMO II Models [2]

V. Research Setting

The case study is a course project to develop a task manager (web application) development. The application provides facilitation to a supervisor to track the progress of job tasks assigned to his/her team. Team inserts the completed tasks associated with time spent for each task. The supervisor will check these tasks and give some notes on these tasks. This project has three actors: administrator, supervisor and team members. Each actor has his/her own perspective as shown in Table 1. More than one programming Languages is used during the case study to develop Task Manager Application. Therefore, size for each one of the language is required to measure as shown in the Table 4.

Table 1 : Details of Task Manager Application

Actors	Job in the system
Team Members	- At the end of day, the system allows the member to insert the tasks that have been completed by him association with the time spent to complete them. - The member can receive a report from supervisor as a feedback of these tasks.
Supervisor	-The supervisor can receive the tasks completed by his/her employees. He can reject or pass or give some notes on any task to improve depending on three criteria: 1) it is on or out the scope; 2) it has any effect on the project or not; 3) it is completed on realistic time or not.
Administrator	- The system generates different types of reports for administrator about each employee such as: 1- number of task reject and number of task pass. 2- calculate the active hours (which are the total number of hours that spent for pass task) and present them as chart (this chart will present active hours for each employee that allows the administrator) to evaluate the employee.

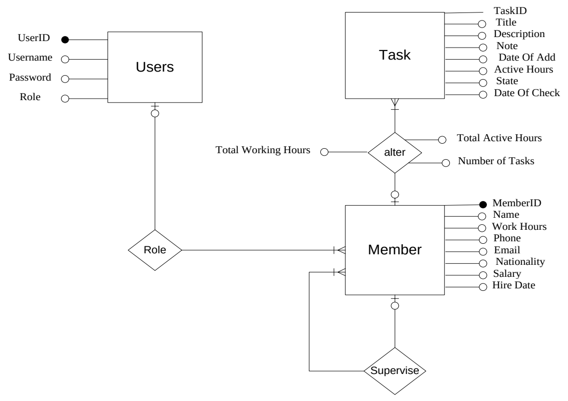


Fig. 2: ERD of Task Manager Application

Table 2: Complexity Parameters

Description	Low	Medium	High	Total
Inputs	**20*3**			60
outputs		**3*5**		15
Queries	**12*3**			36
Files	**3*7**			21
Interfaces	**3*5**	**2*7**	**1*10**	39
Total Unadjusted Function Points				**171**

Table 3: Complexity Factors (14 Questions)

14 Questions	**Scales**
Does the system require reliable backup and recovery?	1
Are there distributed processing functions?	0
Is performance critical?	2
Will the system run in an existing heavily utilized operational environment?	0
Does the system require online data entry?	5
Does the online data entry require the input transactions to be built over multiple screens or operations?	1
Are the master files updated online?	3
Are the inputs, outputs, files, and inquiries complex?	1
Is the code designed to be reusable?	1
Are the conversion and installation included in the design?	0
Is the system designed for multiple installations in different organizations?	0
Is the internal processing complex?	2
Are data communications required?	1
Is the application designed to facilitate change and ease of use by the user?	4
Project Complexity (PC)	21

Table 4: Calculate Function Point

Un-adjustable Function Count**(UFC)**	**171**
Technical Complexity Factors **(TCF)**	***0.65 + 0.01 * ∑Fi (F1 to F14)*** 0.65 + (0.01 * 21) = **0.86**
Function Points **(FP)**	**UFC*TCF** 171*0.86 = ~ **147 FPs**

Table 5: Calculate Function Point

Un-adjustable Function Count**(UFC)**	**171**
Technical Complexity Factors **(TCF)**	***0.65 + 0.01 * ∑ Fi (F1 to F14)*** 0.65 + (0.01 * 21) = **0.86**
Function Points **(FP)**	**UFC*TCF** 171*0.86 = ~ **147 FPs**

VI. Experiment and Analysis

COCOMO I & II are applied prospectively in sub sections 6.1 and 6.2.

6.1 COCOMO I Application

Table 6: Basic COCOMO

MODE	Effort		Schedule	
	A	B	a	b
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	**3.6**	**1.2**	**2.5**	**0.32**

Effort (MM) = A * (KDSI) ^ B (3)
Schedule (TDEV) = a * (MM) ^ b (4)

The size of the project is small and the complexity is simple, we categorize this project as Organic and the number of lines of code is 6762.1 DSI (6.7621KDSI) (See Table 4).

Table 7: Apply Equations of Basic COCOMO I model

MM(Man Month)= 2.4*6.7621^ 1.05= **17.86**
TDEV(Time)= 2.5*17.86^0.38= **7.5 (~8 months)**
People =MM/TDEV17.86/7.5= **~2 members**

Table 8: Intermediate COCOMO

MODE	Effort		Schedule	
	Ai	Bi	a	b
Organic	3.2	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	**2.8**	**1.2**	**2.5**	**0.32**

E = Ai(KLOC)^Bi * EAF (5)
Schedule (TDEV) = a * (MM) ^ b (6)

Table 9: Intermediate COCOMO I Cost Drivers [8]

Cost Drivers	**Ratings**					
	Very	Low	Nominal	High	Very	Extr
Product attributes						
RELY	0.75	**0.88**	1.00	1.15	1.40	
DATA		**0.94**	1.00	1.08	1.16	
CPLX	0.70	**0.85**	1.00	1.15	1.30	1.65
Hardware attributes						
TIME			**1.00**	1.11	1.30	1.66
STOR			**1.00**	1.06	1.21	1.56
VIRT		0.87	**1.00**	1.15	1.30	
TURN		0.87	**1.00**	1.07	1.15	
Personnel attributes						
ACAP	1.46	1.19	**1.00**	0.86	0.71	
AEXP	1.29	**1.13**	1.00	0.91	0.82	
PCAP	1.42	1.17	**1.00**	0.86	0.70	
VEXP	1.21	1.10	**1.00**	0.90		
LEXP	1.14	1.07	**1.00**	0.95		
Project attributes						
MODP	1.24	1.10	**1.00**	0.91	0.82	
TOOL	1.24	1.10	**1.00**	0.91	0.83	
SCED	1.23	1.08	**1.00**	1.04	1.10	

Table 10: Apply Equations of Intermediate COCOMO I model [9]

MM(Man Month) = 2.4*6.7621^ 1.05 = **17.86**
MM korr = (0.88*0.94*0.85*1.13)* 17.86 = **14.2**
TDEV(Time)= 2.5*14.2^0.38 = **6.9 (~7 months)**
People = MM/TDEV= 14.2/6.9 = **~2 members**

6.2 COCOMO II Application

Application Point

The Task manager application has five screens (three of them are simple and two of them are medium in terms of complexity) *(See Table 11)* and three report (two of them are simple and one is medium in terms of complexity) *(see Table 10)* and one 3GL component and three data tables (User Role, Member and Task) *(See Figure 2)*. The project has 20% reused component development. The developer experience and environment maturity are low which is seven 7 *(See Table 16)*.

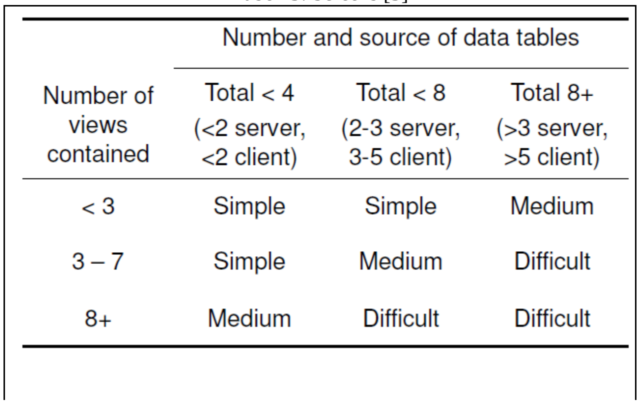
Table 11: Screens

Screen Name	Data table	Views (data items)	Complexity (*see table11*)
Personal Information	Needs **(1) data table** which is (member table see figures 2)	5	simple
Check Tasks	Needs **(2) data table s** which are (member and task tables see figures 2)	9	medium
Add task	Needs **(2) data table s** which are (member and task tables see figures2)	6	simple
My tasks repository	Needs **(2) data table s** which are (member and task tables see figures 2)	6	simple
Add Employee	Needs **(1) data table** which is (member table see figures 2)	11	medium

Table 12: Reports

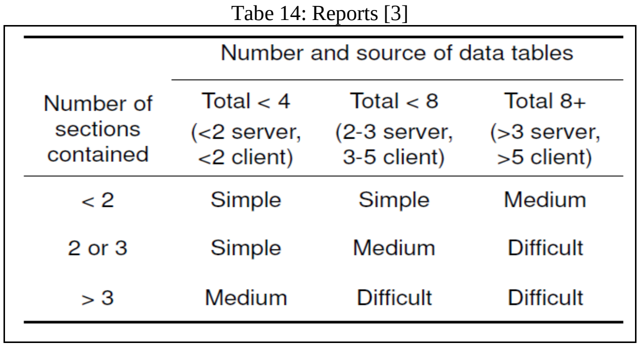
Reports Name	Data table	sections	Complexity (see table12)
Total time was working for each member	Needs **(2) data table s** which are (member and task tables see figures 2)	3	simple
Total tasks completed for each member	Needs **(2) data table s** which are (member and task tables see figures 2)	2	simple
Total time against Total tasks	Needs **(2) data table s** which are (member and task tables see figures 2)	6	medium

Tabe 13: Screens [3]



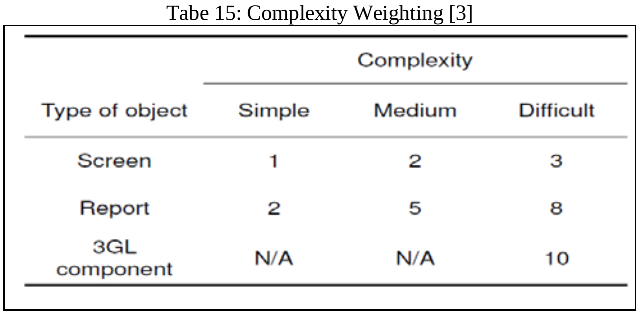
Number of views contained	Number and source of data tables		
	Total < 4 (<2 server, <2 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server, >5 client)
< 3	Simple	Simple	Medium
3 – 7	Simple	Medium	Difficult
8+	Medium	Difficult	Difficult

Tabe 14: Reports [3]



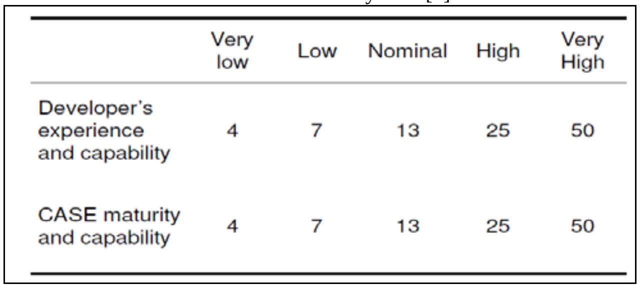
Number of sections contained	Number and source of data tables		
	Total < 4 (<2 server, <2 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (>3 server, >5 client)
< 2	Simple	Simple	Medium
2 or 3	Simple	Medium	Difficult
> 3	Medium	Difficult	Difficult

Tabe 15: Complexity Weighting [3]



Type of object	Complexity		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component	N/A	N/A	10

Tabe 16: Productivity Rate [3]



	Very low	Low	Nominal	High	Very High
Developer's experience and capability	4	7	13	25	50
CASE maturity and capability	4	7	13	25	50

According to the Information above (See Tables 11, 12 and 15):

1) **AP** =(3*1) +(2*2)+(2*2)+(1*5)*(1*10)= 26 (7)
Where **AP** is Application point and **NAP** New Application points

2) **NAP = AP * [(100-%Reuse)/100]** (8)
NOP =26 * [(100 – 20) / 100] = **20.8 OPs**

3) **Effort =NAP/productivity**
Productivity = 7
Effort = 20.8 / 7 **= 2.9 person month.**

4) **Time = 3*Effort^(0.33+0.2*(B-1.01))** (9)
Time = 3*2.9 ^(0.33+0.2*(1.17-1.01))= **4.4 Months**

5) **People=Effort/time** (10)
People =2.9/4.4 = **0.66 (~ one staff)**

Early design Model

The case study project has 6.7621 KLOC (See Table 4), B=1.17 (B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.), A= 2.94 (A could take several values depending on the estimation phase.) and M = PERS * RCPX * RUSE * PDIF * PREX * FCIL * SCED which is show below.

Table 17 : Early Design Model's drivers [5]

Driver	Meaning
RCPX	Product reliability and complexity.
RUSE	Required reuse.
PDIF	Platform difficulty.
PERS	Personnel capability.
PREX	Personnel experience.
FCIL	Facilities.
SCED	Required schedule.

Table 18 : Early Design Model's drivers with their weights

Driver	XLO	VLO	LO	NOM	HI	VHI	XHI
RCPX	0.73	0.81	**0.98**	1.00	1.30	1.74	2.38
RUSE	xxxx	xxxx	**0.95**	1.00	1.07	1.15	1.24
PDIF	xxxx	xxxx	0.87	**1.00**	1.29	1.81	2.61
PERS	2.12	1.62	**1.26**	1.00	0.83	0.63	0.50
PREX	1.59	1.33	**1.12**	1.00	0.87	0.71	0.62
FCIL	1.43	1.30	1.10	**1.00**	0.87	0.73	0.62
SCED	xxx	1.43	1.14	**1.00**	1.00	1.00	xxxx

1) **PM = A *Size^B * M** (11)
M =0.98*0.95*1*1.26*1*1.12*1*1=1.31
PM (Effort)=2.94* 6.7621^1.17 *1.31=**36.04**

2) **Time = 3*Effort^(0.33+0.2*(B-1.01))** (12)
Time = 3***36.04**^ (0.33+0.2*(1.17-1.01)) = **11 Months**

3) **People=Effort/time**
People =30.62/11= 3.27 (~ 3 staff)

Post Architecture Model

Table 19 : Post Architecture Model's drivers with their weights [5]

Drivers	VLO	LO	NOM	HI	VHI	EHI
RELY	0.75	**0.88**	1.00	1.15	1.39	xxxx
DATA	xxxx	**0.93**	1.00	1.09	1.09	xxxx
CPLX	0.75	**0.88**	1.00	1.15	1.30	1.66
RUSE	xxxx	**0.91**	1.00	1.14	1.29	1.49
DOCU	0.89	**0.95**	1.00	1.06	1.13	xxxx
TIME	xxxx	xxxx	**1.00**	1.11	1.31	1.67
STOR	xxxx	xxxx	**1.00**	1.06	1.12	1.57
PVOL	xxxx	0.87	**1.00**	1.15	1.30	xxxx
ACAP	1.50	1.22	**1.00**	0.83	0.67	xxxx
PCAP	1.37	1.16	**1.00**	0.87	0.74	xxxx
PCON	1.24	1.10	**1.00**	0.92	0.84	xxxx
AEXP	1.22	**1.10**	1.00	0.89	0.81	xxxx
PEXP	1.25	**1.12**	1.00	0.88	0.81	xxxx
LTEX	1.22	1.10	**1.00**	0.91	0.84	xxxx
TOOL	1.24	1.12	**1.00**	0.86	0.72	xxxx
SITE	1.25	1.10	**1.00**	0.92	0.84	0.78
SCED	1.29	1.10	**1.00**	1.00	1.00	xxxx

The case study project has 6.7621 KLOC (See Table 4), B=1.17 (B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.), A= 2.94 (A could take several values depending on the estimation phase.) and M = (Multiply 17 Drivers) which show below.

1) **PM = A *Size^B * M**
M =0.93 * 0.88 * 0.91 * 0.95 * 1.1 * 1.12*0.88=**0.77 PM**
PM (Effort) =2.94* 6.7621^1.17 *0.77=**21.19**

2) **Time = 3*Effort^(0.33+0.2*(B-1.01))**
Time = 3*21.19^(0.33+0.2*(1.17-1.01))= **9.06 Months**

3) **People=Effort/time**
People =21.19/9.06 = **2.33 (~2 staffs**

Table 20 : COCOMO I Comparing Models

Model	Mode	Effort (MP)	Time (Month)	People
Basic	Organic	17.86	7.5	~2
intermediate	Organic	14.2	7	~2

Table 21 : COCOMO II Comparing Models

Model	Effort(MP)	Time(Month)	People
Application point	2.9	4.4	~1
Early Design	36.04	11	~ 3
Post-Architecture	21.19	9	~2

Table 22 : Actual time and people after the project finish

Time(Month)	People
2.5	1

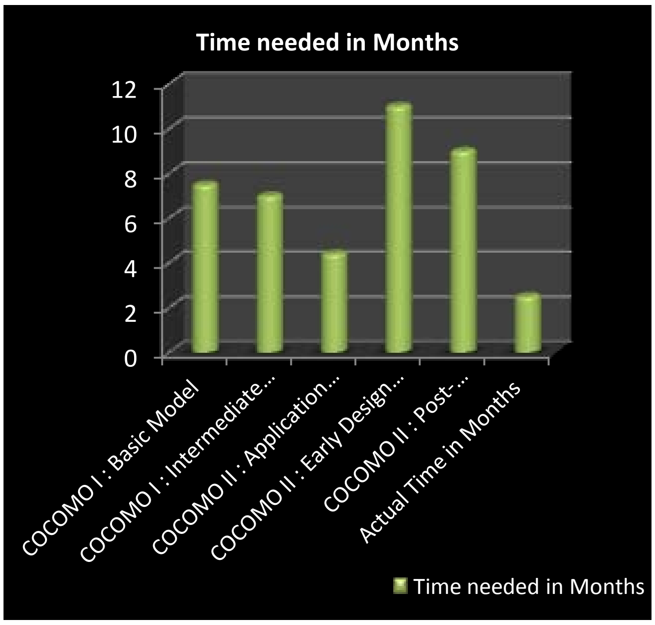


Fig. 3: Comparing Modules' result with Actual time needed

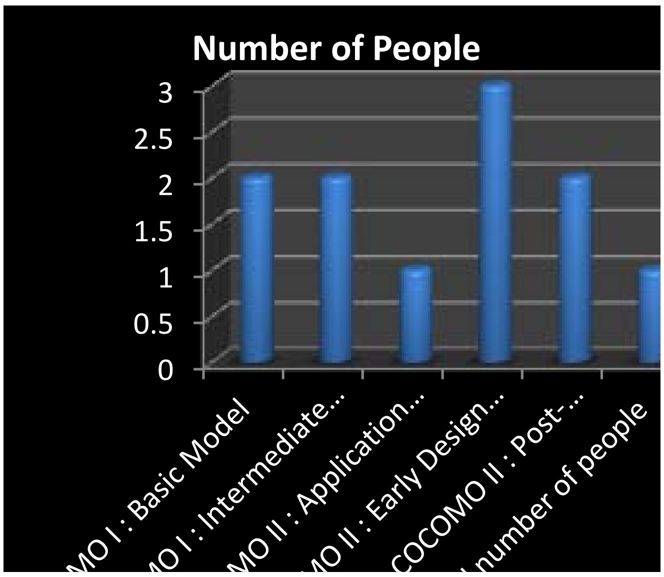


Fig. 4: Comparing Modules' result with Actual staffs needed

Figure 3 & 4 illustrates the implementation of COCOMO I and II on modules (of case study result) with actual time and staff needed. It is clearly depicted from the results that the Application Composition Model is more accurate, for a project such as Task Manager Application, as compared to other sub models of COCOMO I & II.

VII. Conclusion

Converted case study on COCOMO I & II confirms that the Application Composition Model (one of COCOMO II sub-Models) is relatively more accurate than the remaining COCOMO I & II sub-models as shown in Figures 3 and 4. On the other hand, at the constraint of some limitations related to cost and time estimates. Such models are those included in COCOMO must have entered in terms of KLOC. Such methods as FP, OP, and LOC are primary sizing procedures adopted to compute the model inputs. The accuracy of the models depends on: 1) the accuracy and fidelity of input data; 2) estimate the size for each programming language incorporated in the project to achieve the accuracy related to time and costs.

References

[1] Boehm, B. W. and R. Valerdi. Achievements and Challenges in Cocomo-Based Software Resource Estimation published by IEEE Computer Society. 74-83 (2008).

[2] Boehm, B. W. An Overview of the COCOMO 2.0 Software Cost Model (1999).

[3] Zaid, A., M. H. Selamat, A. A. A. Ghani, R. Atan and K. T. Wei. Issues in Software Cost Estimation, IJCSNS Int J of Computer Science and Network Security, 8(11): 350-356 (2008).

[4] Nasir, M. A Survey of Software Estimation Techniques and Project Planning Practices, Proceedings of the Seventh ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06), (2006).

[5] Qureshi, M. R. J. and S. A. Hussain. A Reusable Software Component-Based Development Process Model Int. J of Advances in Engineering Software, 39(2): 88-94 (2008).

[6] Succi, G. and F. Baruchelli. The Cost of Standardizing Components for Software Reuse, Standard View 5(2) (1997).

[7] Gill, N. S. Reusability Issues in Component-Based Development, ACM SIGSOFT Software Engineering Notes, 28(4): 4 – 4 (2003).

[8] Clemente, P. J. and J. Hernández. Aspect Component Based Software Engineering, University Extremadura. 1-4 Spain (2001).

[9] Frakes, W. B. and K. Kang. Software Reuse Research: Status and Future, IEEE Transactions on Software Engineering, 31(7): 529-536 (2005).

[10] Dolado, J. J. A Validation of the Component-Based Method for Software Size Estimation, IEEE Transactions on Software Engineering, 26(10): 1006-1021 (2000).

Authors' Profiles

M. Rizwan Jameel Qureshi: Assistant Professor of Information Technology, Faculty of Computing & Information Technology, King Abdulaziz University, major in software engineering and database management systems

Mohammed Mugahed Al_Qmase: Under-graduated student of Information Technology in King Abdulaziz University, major in software engineering