Software Engineering Cost Estimation Using COCOMO II Model

Hana Rashied Ismaeel
Al-Nahrain University

Abeer Salim Jamil
Mansour University College

ABSTRACT

In this paper we discuss the use of **COCOMO II** (Constructive Cost Model) to estimate the cost of software engineering . The COCOMO II which allow us estimate the cost, effort and scheduling when planning new software development . We use the effort equation guidance to find the number of person / months which is needed to complete the project and duration equation to specified the numbers of months which is needed to complete this project.

This paper present how implemented **COCOMO II** model equation about the same data in different language we choose **C** and **OOP(C++)**,we make the analysis of these two program and we conclude that relation between effort and duration was forward relation, we mean that when effort increasing duration will increasing in the author side.

Key Words : COCOMO II , Estimation Model, Cost Estimation Model , Effort Equation, Schedule Equation .

1. Introduction

COCOMO II (Constructive Cost Model) is a model that allows one to estimate the cost , effort, and schedule when planning a new software development activity . It consists of three sub models, each one offering increased integrity the further a long one is in the project planning and design process. Listed increasing fidelity, these sub models are called the applications composition, early design. and post – architecture models.[1]

COCOMO II is useful for a much wider collection of techniques and technologies. COCOMO II provides up –to-date support for business software , object – oriented software , software created via evolutionary development models and software developed using commercial- off-the-shelf application composition utilities (Boehm [1]) .[1,2]

COCOMO II includes the application composition model (for early prototyping efforts) and the more detailed Early Design and Post- Architecture models (for subsequent portions of the life cycle).[10]

2. Objective of Software Cost Estimation with COCOMO II

The most fundamental calculation in the COCOMO II model is the use of the effort equation to estimate the number of person/ months required to develop a project. Most of the other COCOMO II results including the estimates for requirements and maintenance , are derived from this quantity.[1,2]

Software cost estimation is important for making good management decisions. It is also connected to determining how much effort and time a software project requires . Cost estimation has several uses : [1,3,10]

1. It establishes a firm, reliable budget for an –in house project.
2. It facilitate competitive contract bids.
3. It determines what is most effective for the organization.

Software cost estimation provides the important link between the general concepts and techniques of economic analysis and the particular world of software engineering .

3. Estimation Models

Estimating models have been generated by measuring certain properties and characteristics (duration, cost, team size, disk usage, etc...) of past successful projects. Curves are then fit to the data points to get descriptive equations which are the models. We hope these equations can be used for prediction. The fact that they came from measurements of past projects is what gives the hope.[3]

1. Small Projects:[3,4]

Measurements of small to moderate projects resulted in a model of the following form:

$$EFFORT = a * SIZE + b \quad \text{… (1)}$$

Here, the magnitude of the effort is a linear function of the size of the project (Number of Lines of Code, usually). This model holds up until a certain point, usually for projects that can be reasonably accomplished by small teams of two or three people. By increasing the size of the project, the above model becomes less and less accurate and the need for a new model increases.

2. Large Projects:

Projects requiring teams of more than three or so people tend to behave in the following way:

$$EFFORT = a * SIZE^{b} \quad \text{... (2)}$$

Here, the size of the project is scaled exponentially, therefore as a product increases in size, the effort to produce the product grows more than linearly (for $b >= 1$). It means that if we try to develop a larger product, our productivity (*SIZE / EFFORT*) Decreases.

This decrease in productivity on larger projects is called a *diseconomy of scale*. The main reasons why larger software products incur diseconomies of scale are the following:

1.Relatively more product design is required to develop the thorough unit-level specifications required to support the parallel activity of a larger number of programmers.

2.Relatively more effort is required to verify and validate the larger requirements and design specifications.

3.Even with a thoroughly defined specification, programmers on a large project will spend relatively more time communicating and resolving interface issues.

4.Relatively more integration activity is required to put the units together.

4. COCOMO II MODEL [3,4]

COCOMO II has three different models : [3]

1. The Application Composition Model

Suitable for projects built with modern GUI- builder tools.[4]

2. The Early Design Model

This model is used to make rough estimates of a project's cost and duration before it is entire architecture is not determined. It uses a small set of new Cost Drivers, and new estimating equations. For the Early Design and Post Architecture model :- [3,4]

$$\text{Effort} = \frac{\text{ASLOC}\left(\frac{\text{AT}}{100}\right)}{\text{ATPROD}} + a\left[\text{size}\left(1+\frac{\text{BRAK}}{100}\right)\right]^{b}\prod_{i}\text{EM}_{i}$$

… (3)

$$b = 1.01 + \frac{1}{100} \sum SF_j$$

… (4)

$$Size = KSLOC + KASLOC \left(\frac{100 - AT}{100}\right) AAM$$

… (5)

Where :

a = 2.5 $\mathbf{SF_j}$ = scale factor, $\mathbf{EM_i}$ = effort multiplier

BRAK = Percentage code discarded due to requirement volatility.
ASLOC = size of adapted components.
AT = percents of components adapted.
ATPROD = Automatic Translation Productivity.
AAM = Adaptation Adjustment Multiplier .

3. The Post-Architecture Model [8]

This is the most detailed COCOMO II model. It is used after project's overall architecture is developed. It has new cost drivers, new line counting rules, and new equations.

5. Cost Estimation Metrics

Cost estimates are need throughout software life cycle. Preliminary are required to determine the feasibility of a project. Detailed estimates are needed to assist with project planning .The actual effort for individual task is compared with estimated and planned values, enabling managers to reallocated resources when

necessary . The metrics used to compute the cost estimation with COCOMO II is :-[2,3]

5.1. Source Lines of Code[4]

The COCOMO calculations are based on your estimates of a project's size in Source Lines of Code (SLOC). SLOC is defined such that:

1. Only Source lines that are DELIVERED as part of the product are included -- test drivers and other support software is excluded.

2. SOURCE lines are created by the project staff -- code created by applications generators is excluded .

3. One SLOC is one logical line of code .

4. Declarations are counted as SLOC.

5. Comments are not counted as SLOC .

5.2. Function Point [6,9]

Function points are computed by counting the following software characteristics :

1- External inputs and outputs .
2- User interactions .
3- External interfaces.
4- Files used by the system .

Each of this is then individually assessed for complexity and given a weighting value varies from 3 (for simple external inputs) to 15 (for complex internal files).

The function point count is computed by multiplying each raw count by the estimated weight and summing all values, then multiplied by the project complexity factors which consider the overall complexity of the project according to range of factors such as the degree of distributed processing , the mount of reuse , the performance , show the table (1) and table (2).

	1-5 Data Element Types	**6-19 Data Element Types**	**20+ Data Element Types**
0-1 File Type Referenced	Low	Low	Average
2-3 File Type Referenced	Low	Average	High
4+ File Type Referenced	Average	High	High

Table (1) : Represent the relation between weight value and complexity factor.

	Low	Average	High
External Input	x3	x4	x6
External Output	x4	x5	x7
Logical Internal File	x7	x10	x15
External Interface File	x5	x7	x10
External Inquiry	x3	x4	x6

Table(2) :Represent how count the external inputs and outputs, user interactions and external interfaces.

Function point counts can be used in conjunction with lines of code estimation techniques. The number of function points is used to estimate the final code size.

Based on historical data analysis, the average number of lines of code in a particular language required to implement a function point can be estimated (AVC). The estimated code size for a new application is computed as follows:

Code size = AVC x Number of function points

… (6)

The advantage of this approach is that the number of function points can often be estimated from the requirements specification so an early code size prediction can be made, show the table (3).

Language	Ratio-Source :Executable
Assembler	1:1
Macro-Assembler	1:1.5
C	1:1.25
ALGOL	1:3
COBOL	1:3
FORTRAN	1:3
Pascal	1:3.5
RPG	1:4
PL1	1:4
MODULA-2	1:4.5
Ada	1:5
PROLOG	1:5
LISP	1:5
FORTH	1:5
BASIC	1:5
LOGO	1:6
4-GLs	1:8
APL	1:9
Objective-C	1:12
SmallTalk	1:15
Query Languages	1:20
Speadsheet	1:50

Table(3) :Levels of selected software language relative to Assembler language

5.3. The Scale Drivers

In the COCOMO II model, some of the most important factors contributing to a project's duration and cost are the Scale Drivers. You set each Scale Driver to describe your project; these Scale Drivers determine the exponent used in the Effort Equation.[5,8]

The 5 Scale Drivers are:

- Precedent ness
- Development Flexibility
- Architecture / Risk Resolution
- Team Cohesion
- Process Maturity

5.4. Cost Drivers

COCOMO II has 17 cost drivers – you assess your project, development environment, and team to set each cost driver. The cost drivers are multiplicative factors that determine the effort required to complete your software project. For example, if your project will develop software that controls an airplane's flight, you would set the Required Software Reliability (RELY) cost driver to Very High. That rating corresponds to an effort multiplier of 1.26, meaning that your project will require 26% more effort than a typical software project. COCOMO II defines each of the cost drivers, and the Effort Multiplier associated with each rating. [8,9]

5.5. COCOMO II Effort Equation[5,6]

The COCOMO II model makes its estimates of required effort (measured in Person-Months – PM) based primarily on your estimate of the software project's size (as measured in thousands of SLOC, KSLOC):

$$\textbf{Effort} = 2.94 * \textbf{EAF} * (\textbf{KSLOC})^{\textbf{E}} \qquad \textbf{... (7)}$$

Where

EAF Is the Effort Adjustment Factor derived from the Cost Drivers.

E Is an exponent derived from the five Scale Drivers .

As an example, a project with all Nominal Cost Drivers and Scale Drivers would have an EAF of 1.00 and exponent, E, of 1.0997. Assuming that the project is projected to consist of 8,000 source lines of code, COCOMO II estimates that 28.9 Person-Months of effort is required to complete it:

$$\textbf{Effort} = 2.94 * (1.0) * (8)^{1.0997}$$

$$= \textbf{28.9 Person-Months}$$

5.6. Effort Adjustment Factor [4,6]

The Effort Adjustment Factor in the effort equation is simply the product of the effort multipliers corresponding to each of the cost drivers for your project.

For example, if your project is rated Very High for Complexity (effort multiplier of 1.34), and Low for Language & Tools Experience (effort multiplier of 1.09), and all of the other cost drivers are rated to be Nominal (effort multiplier of 1.00), the EAF is the product of 1.34 and 1.09.

$$\text{Effort Adjustment Factor} = EAF = 1.34 * 1.09 = 1.46 \quad \dots (8)$$

$$\text{Effort} = 2.94 * (1.46) * (8)^{1.0997}$$

$$= 42.3 \text{ Person-Months}$$

5.7. COCOMO II Schedule Equation [7,9]

The COCOMO II schedule equation predicts the number of months required to complete your software project. The duration of a project is based on the effort predicted by the effort equation:

$$\text{Duration} = 3.67 * (\text{Effort})^{SE} \quad \dots (9)$$

Where:

Effort Is the effort from the COCOMO II effort equation .
SE Is the schedule equation exponent derived from the five Scale Drivers .

Continuing the example, and substituting the exponent of 0.3179 that is calculated from the scale drivers, yields an estimate of just over a year, and an average staffing of between 3 and 4 people:

Duration = 3.67 * (42.3)$^{0.3179}$

= 12.1 months

Average staffing=Effort/Duration … (10)

Average staffing = (42.3 Person-Months) / (12.1 Months)

= 3.5 people

5.8.The SCED Cost Driver [7,9]

The SCED cost driver is used to account for the observation that a project developed on an accelerated schedule will require more effort than a project developed on its optimum schedule. A SCED rating of Very Low corresponds to an Effort Multiplier of 1.43 (in the COCOMO II.2000 model) and means that you intend to finish your project in 75% of the optimum schedule (as determined by a previous COCOMO estimate). Continuing the example used earlier, but assuming that SCED has a rating of Very Low, COCOMO produces these estimates:

Duration = 75% * 12.1 Months = 9.1 Months

Effort Adjustment Factor = EAF

= 1.34 * 1.09 * 1.43

= 2.09

$$\text{Effort} = 2.94 * (2.09) * (8)^{1.0997}$$
$$= 60.4 \text{ Person-Months}$$

$$\text{Average staffing} = (60.4 \text{ Person-Months}) / (9.1 \text{ Months})$$
$$= 6.7 \text{ people}$$

Notice that the calculation of duration isn't based directly on the effort (number of Person-Months) – instead it's based on the schedule that would have been required for the project assuming it had been developed on the nominal schedule. Remember that the SCED cost driver means "accelerated from the nominal schedule".

6. Project Cost

Software project managers are responsible for controlling project budgets so , they must be able to make estimates of how much a software development is going to cost . The principal components of project costs are : [4]

1- **Hardware costs .**
2- **Travel and training costs.**
3- **Effort costs (the costs of paying software engineering).**

The dominant cost is the effort cost. This is the most difficult to estimate and control, and has the most significant effect on overall costs.

Software costing should be carried out objectively with the aim of accurately predicting the cost to the contractor of developing the software . [6]

software cost estimation is a continuing activity which starts at the proposal stage and continues throughout the lifetime of project. Projects normally have a budget , and continual cost estimation is necessary to ensure that spending is in line with the budget. Effort can be measured in staff- hours or staff-months (used to be known as man-hours or man – month).Boehm (1981) [1] discusses seven techniques of software cost estimation:[5]

1- Algorithmic cost modeling

A model is developed using historical cost information which relates some software metric (usually its size) to the project cost. An estimate is made of that metric and the model predicts the effort required.[7]

2- Expert judgment

One or more experts on the software development techniques to be used and on the application domain are consulted. They each estimate the project cost and the final cost estimate is arrived at by consensus.[2]

3- Estimation by analogy

This technique is applicable when other projects in the same application domain have been completed. The cost of a new project is estimated by analogy with these completed projects.[1]

4- Parkinson's Law

Parkinson's Law states that work expands to fill the time available. In software costing, this means that the cost is determined by available resources rather than by objective assessment. If the software has to be delivered in 12 months and 5 people are available, the effort required is estimated to be 60 person-months.[2]

5- Pricing to win

The software cost is estimated to be whatever the customer has available to spend on the project. The estimated effort depends on the customer's budget and not on the software functionality.[3]

6- Top-down estimation

A cost estimate is established by considering the overall functionality of the product and how that functionality is provided by interacting sub-functions. Cost estimates are made on the basis of the logical function rather than the components implementing that function.[7]

7- Bottom-up estimation[8]

The cost of each component is estimated. All these costs are added to produce a final cost estimate.

Each technique has advantages and disadvantages. For large projects, several cost estimation techniques should be used in parallel and their results compared. If these predict radically different costs, more information should be sought and the costing process repeated. The process should continue until the estimates converge.

Cost models are based on the fact that a firm set of requirements has been drawn up and costing is carried out using these requirements as a basis. However , sometimes the requirements may be changed so that a fixed cost is not exceeded.

7. Execution Cost Estimation

The objective of this is to determine the cost estimation of the software after this software is complete design by apply the cost estimation metrics then the result should be recorded during this step. This operation of test involves the performing the following tasks .:-

1- Build Test Data

In this step the software system should be design and the metrics of cost estimation should be specified . This information should be specified before the operation of test is begin.

2- Execute Cost Estimation Metrics

There are many methods of metrics testing an apply over the project .These metrics are :-

1- COCOMO II Effort Equation.
2- Effort Adjustment Factor.
3- COCOMO II Schedule Equation .

3- Record The Result

The test report should indicate the numbers of person should be work to complete this software system and the numbers of months are needed to complete this system .

The executing testing and test result are illustrated in figure (1) :-

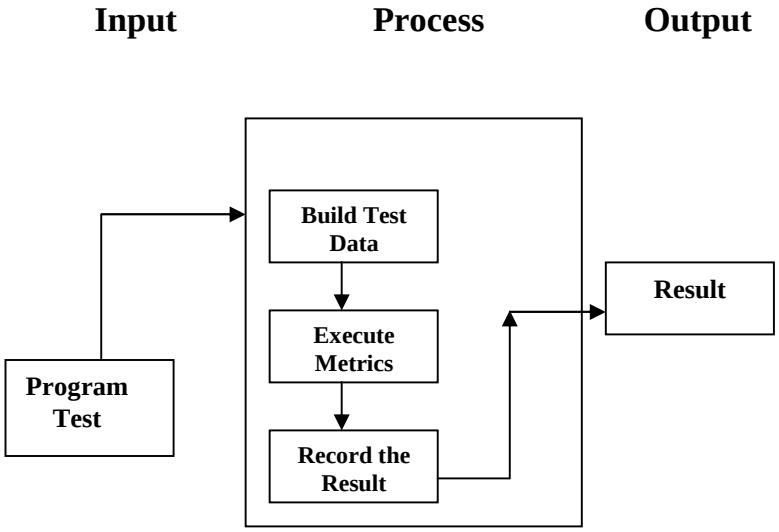

Figure (1) Diagram Execution Cost Estimation

8. System Implementation

This paper suggest to use COCOMO II Model to Estimate the cost of project by applying the metrics of cost estimation until the operation is complete , the result is represent the volume of effort is needed to estimate the number of person / months required to develop a project. This is explaining in the general algorithm :-

Step 1 : Input develop software .

Step 2 : Compute the value of **EAF**.

Step 3 : Find the value of **E** and **SE**.

Step 4 : Find the size of Software **(KSLOC).**

Step 5 : Calculate the **Effort Equation** by :
Effort = 2.94 * EAF * $(KSLOC)^{E}$

Step 6 : Calculate the **Duration Equation** by :
Duration = 3.67 * $(Effort)^{SE}$

Step 7 : Calculate the **Average staffing** by :
Average staffing=Effort/Duration

Step 8 : Analysis the result and draw the output is represented the volume of effort to complete the software.

Step 9 : End.

An important advantage of applying the COCOMO II model to estimate the cost , effort, and schedule of project after the user complete the design of project, then the result is a good guide to show the average staff required to develop a project .

9. Analysis Result

after complete the operation of metrics COCOMO II model over the two , the analysis of the result of metrics is :-

1- The first project build by **C** – Language with all normal cost drivers and scale drivers would have an EAF of 1.00 and exponent , E of 1.0997 and consist of 16,000 source lines of code , COCOMO II estimates that 90.543 person – months of effort is required to complete it :-

$$\text{Effort}=2.94*(1.0)*(16)^{1.0997}$$
$$=62.018 \text{ Person - Months}$$

and

$$\text{Duration}=3.67*(62.018)^{0.3179}$$
$$=13.630 \text{ months}$$

and

$$\text{Average Staffing} = \text{Effort}/\text{Duration}$$
$$=(62.018)/(13.630)$$
$$=4.549 \text{ Person}$$

After complete the operation of COCOMO II model estimation can be specifying the average staffing of between 4 and 5 people is needed to develop this project .

2- The second build by **OOP(C++)**with all very high cost driver and scale drivers would have an EAF of 1.26 and exponent , E of 1.3856 and consist of 20,000 source lines of code , COCOMO II estimates that 235.1933 Person – Months of effort is required to complete it :-

$$\text{Effort} = 2.94*(1.26)*(20)^{1.3856}$$
$$= 235.1933 \text{ Person-Month}$$

and

$$\text{Duration} =3.67*(235.1933)^{0.3179}$$
$$=20.8289 \text{ Month}$$

and

$$\text{Average Staffing} = \text{Effort}/\text{Duration}$$
$$= 235.1933/20.8229$$
$$= 11.294 \text{ Person}$$

After complete the operation of COCOMO II model estimation can be specified the average staffing of between 11 and 12 people is needed to develop this project.

3- From step one and two can be analysis the result of estimation by show the degree of effort and schedule are increase with increase the size of project , cost drivers and scale drivers , then the average staffing increase linearly. Show in figure (**2**),Figure (**3**) and Figure(**4**).

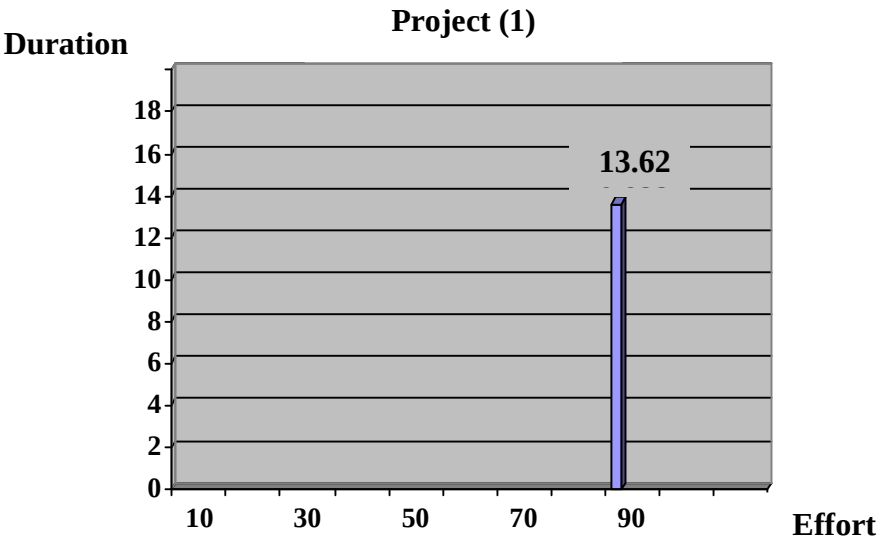

Figure (2) Project(1) Relation between Effort And Duration

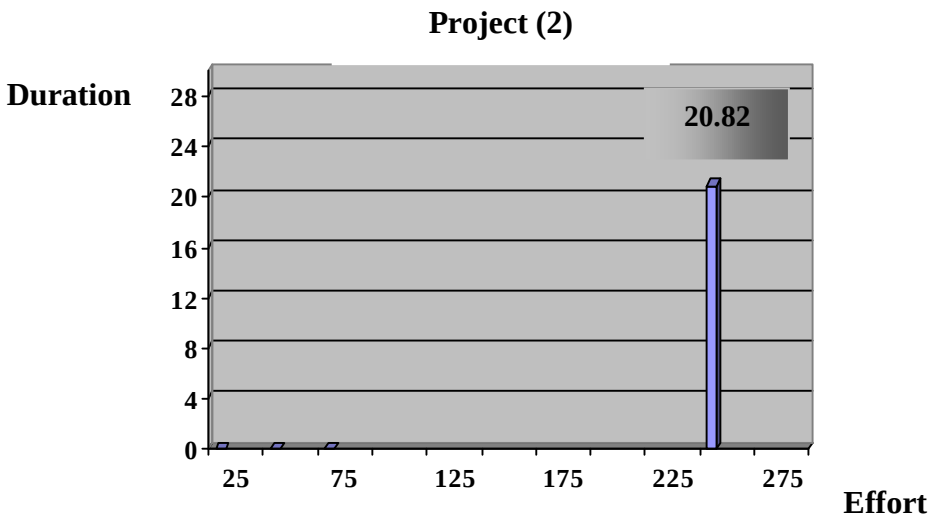

Figure (3) Project(2) Relation between Effort And Duration

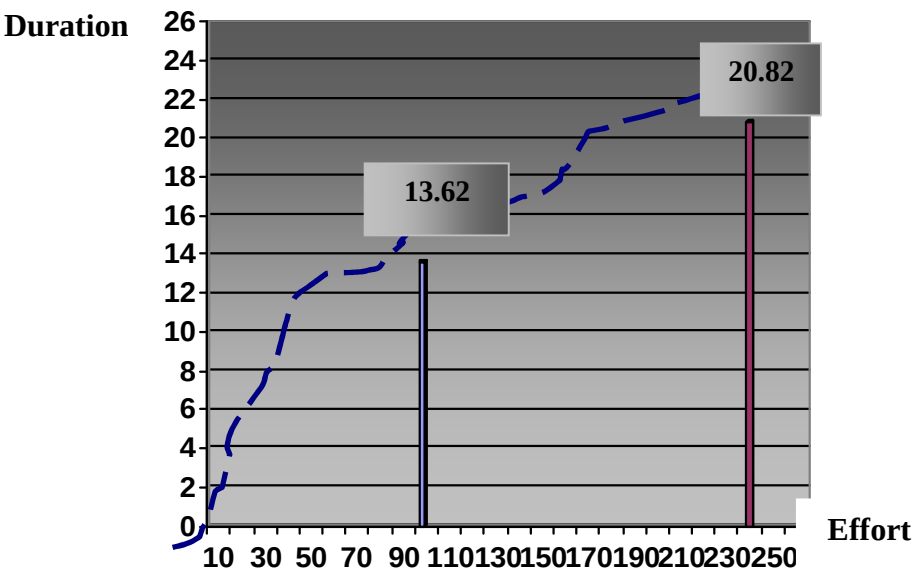

Figure (4) Relation between Effort and Duration in Project 1 and 2

10- Conclusion

1- COCOMO II model is used to estimate the cost, effort and schedule when planning a new software development.

2 - The effort Equation is used to estimate the number of person / month and the schedule equation is used to specified the number of months is needed to develop software.

3- The degree of effort and duration are increase with increase the size of project, cost drivers and scale drivers , then the average staffing increase linearly.

4 - The COCOMO II model is a good guide to estimate the requirements and maintenance of software.

11- Reference

1- Boehm , B. W (1981) ," Software Engineering Economics ",Englewood Cliffs, N. J . Prentice-Hall.

2- Pressman , Roger S. , " Software Engineering ", Mcgraw-Hill, Singapore , 1997.

3- Suns more, H. E. , Conte , S. D. , "Software Engineering Metrics and Models ",The Benjamin / Cummings , 1986 .

4- Perrly , William e. , "Effective Methods For Software Testing ", john Wiley and Sons, Canada, 2000.

5- Sommerville , Lan , " Software Engineering " , Addison - Wesley, Harlow, 2001.

6- Pfleeger , Shari lawernce , " Software Engineering theory and Practice ", Second Edition , Preice-Halll , Inc, New Jersey,2001.

7- Hamlet , Dick , Maybee , Joe , "The Engineering Of Software ", Addison - Wesley ,Inc,USA,2001.

8- Shooman , Martin L. , "Software Engineering Design , Reliability, And Management" , cgraw – Hill,Singapore,1988.

9-Jonas Verlodt , "Software Cost Estimation Using COCOMO II model " ,2004.
Available at :

http://www.jsc.nasa.gov/bu2/COCOMO.html

Software Engineering Cost Estimation Using COCOMO II Model

عبير سالم جميل هناء رشيد اسماعيل

كلية المنصور الجامعة جامعة النهريــن

المستخلص

نناقش في هذا البحث كيفية استخدام نموذج الـ COCOMO II لتخمين كلفة البرامج في مرحلة التصميم . حيث ان هذا النموذج (COCOMO II) يستخدم لتخمين كل من الكلفة , الجهد والوقت المطلوب لبناء البرمجيات في مرحلة التصميم وذلك من خلال تطبيق كل من معادلات الجهد (Effort) لحساب عدد الاشخاص على الزمن المطلوب لبناء البرنامج وايضا تطبيق معادلات الوقت (Duration) لحساب عدد الاشهر المطلوبة لاكمال هذا البرنامج . في هذا البحث تم توضيح كيفية تطبيق نموذج الـ COCOMO II على نوعين مختلفين من البرامج الاول تم استخدام لغة الـ C لبنائه والثاني تم استخدام لغة (C++) OOP لبنائه ، ثم بعد الانتهاء من تطبيق معادلات الـ COCOMO II على البرنامجين تم اجراء تحليل النتائج والمقارنة ما بين البرنامجين ومن ثم تم التوصل الى وجود علاقة طردية ما بين الجهد (Effort) والوقت (Duration) حيث انه كلما ازداد الجهد سوف يزداد الوقت في نفس الوقت .