Modification of Software Constructive Cost Model II (COCOMO II) to Suit Nigerian Development Environment

[1]Odion P. O. and [2]Onibere E. A.

[1]Department of Computer Science, Nigerian Defence Academy, Kaduna, Nigeria
[2] Department of Computer Science, University of Benin, Benin-city, Nigeria
Corresponding Author's e-mail address: podion2012@gmail.com

Abstract

The extended fashion of Barry Beohm constructive cost model II (COCOMO II) has been adapted to determine the suitability of a developed software constructive programme to suit Nigerian Development Environment. It consists of three sub models which are the applications composition, early design and post-architecture design. One of the findings made was that 90 % of software developers in Nigeria use non-parametric methods for software development estimation, which usually lead to under-budgeting or over-budgeting. The Visual Basic programming language was used to code the modified model, because of its graphical interfaces. Sample results show a minimal difference between what was obtained in COCOMO II and the new model in order to ascertain the accuracy of the derived model. The comparison shows that the new model is environmentally friendly and can be adopted in any environment with little modification.

Keywords: Modification, Software, Constructive Cost Model, Suit Nigerian, Development Environment.

1. INTRODUCTION

Between 1965 and 1975, there was serious software crisis in developed countries as a result of more demands on software to service their cheaper and more powerful machines that came into be in that era. In 1967, the North Atlantic Treaty Organization (NATO) formed the Science Committee to provide solutions to the crisis (Donelson, 1976). The problems were under- and over-budgeting (software cost), late delivery of milestones and deliverables, poor software security, so many errors in delivered software, etc.

In 1967, when NATO Science Committee was discussing 'The Software Crisis' in one of their conferences, Friedrich Bauer a German representative suggested the term 'Software Engineering' as the best way to conceive of both the problem and the solution to software crisis. In 1972, Bauer published the following definition of software engineering: “The establishment and use of sound engineering principles to obtain economically software that is reliable and works on real machines efficiently” (Bauer, 1972).

Some of the Engineering principles are: Requirements elicitation/analysis, Feasibility study, Planning, Design, Estimation (Costing), Testability, Implementation, Maintainability, Management, etc. (Albert and Jayesh, 1992).

1.1 Estimating Software Cost

The process of estimating a software cost is not different from the general process for estimating any other element of cost. There are, however, aspects of the process that are peculiar to software estimation. Some of the unique aspects of software estimation are driven by the nature of software as a product. Software cost estimation is a continuing activity, which starts at the proposal stage and continues through the life-time of a project. Continual cost estimation is to ensure that the spending is in line with the budget (Bernard, 1987).

Software cost estimation is one of the most challenging tasks in software project management, since it is not a physical artifact like engineering projects. One of the first steps in any estimate is to understand and define the system to be estimated.

Liming (2009) stated that the software estimation process includes:

- Estimating the size of the software product to be produced.
- Estimating the effort required.
- Project schedules (Duration)
- Finally, estimating overall project cost.

1.2 Reasons for Software Estimation

Estimation involves answering the following questions:

a. How much effort is required to complete each activity?
b. How much calendar time is needed to complete each activity?
c. What is the total cost of each activity?

2. DATA COLLECTIONS AND METHODOLOGY

A survey was conducted in Nigeria to find out the following:

a. How software development estimate is carried out.
b. The appropriate parameters considered for estimation.

The findings of the survey, together with the study of existing COCOMO II cost estimate model were used to develop a new model that is appropriate for Nigerian environment.

2.1 Methods of Data Collection

A structured non-disguised questionnaire was the main instrument used in gathering data for this study. The instrument contains fifteen (15) opened- and closed-ended questions. Most of the respondents are computer professionals and ICT institutions. The instrument was designed to assess the respondents’ opinion on the following:

a. Knowledge of COCOMO II.
b. The type of estimation method(s) being used by them.
c. The attributes considered when estimating the cost and duration of software development.
d. Sought their opinion for the need to calibrate software estimation model for the Nigerian environment.

Interview method (Telephone and Direct contact), Journals and Secondary sources (Books, internet browsing, etc.) were consulted as data collected avenue.

2.2 Population, Samples and Sampling Procedures of the Study

a. Population

While a total of 120 respondents were contacted, but only 65 responded and that made-up the sampling population. They include: Software Developers, Computer Scientists, System Analysts, ICT Professionals and End-Users.

b. Samples and Sampling Procedures

The population was divided into strata.

- The existing six zones of the federation formed the strata.
- Then random sampling was applied within each stratum to select a state to represent the stratum.
- Cities were considered during the random sampling, because software developers and organizations are mostly found in major cities in Nigeria.

Table-1 shows the analysis of Respondents according to their profession**.**

Table 1: Analysis of Respondents according to their Profession

Profession	Developers	Users	Developers/Users	Total
Software Engineers	6	4	4	14
Programmers	12	3	5	20
System Analysts	8	4	-	12
Computer Science /ICT Lecturers	2	3	5	10
End-Users	2	3	4	09
Total	28	18	19	**65**

2.3 Data Analysis and Findings

The following findings were made on analyzing the data:

a) 70 % of the software developers in Nigeria used water-fall life-cycle model, because of the following reasons:
   - ➢ Easy to understand and implement.
   - ➢ Widely used and known.
   - ➢ Reinforces good habits: define-before-design, design-before-code.

- Identifies deliverables and milestones.
- Document driven and documentation standards.
- Works well on mature products and weak teams.

Waterfall Model

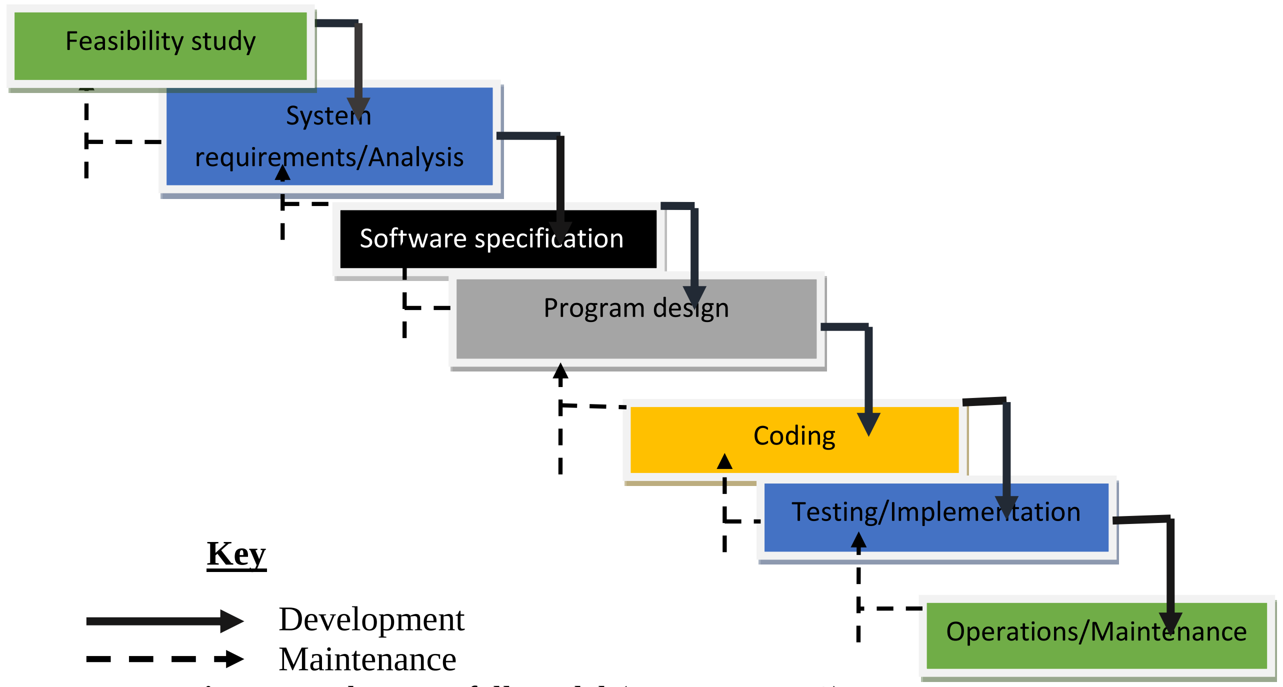


Figure 1: The waterfall model (Winston, 1970)

b) Eight (8) software attributes were commonly considered by the majority of software developers. These eight attributes were summarized after the examination of the attributes collected during the questionnaire, interview and observation methods.

They are:

i) Complexity of the software
ii) Required reliability
iii) Size of database
iv) Required efficiency (memory and execution time)
v) Analyst and Programmer capability
vi) Experience of team in the application area

vii) Experience of team with the programming language
viii) Use of tools and software engineering practices.

c) Ninety percent (90 %) of software cost estimate methods used were non-algorithmic methods. These methods are Price-to-win, Expert judgment, Estimation by Analogy, Bottom-up and Top-down methods.

d) 80 % of the respondents have no knowledge of COCOMO II cost estimate.

2.4 An Overview of COCOMO II Equations

The first version of COCOMO model was originally developed in 1981 (Boehm, 1981). Due to its difficulties in estimating the cost of software developed to new life-cycle processes and object-oriented approaches, a newest version named COCOMO II was developed in 2000 (Boehm, 2000). The stages of COCOMO II are application composition and early design and post-architecture.

In the COCOMO II model, some of the most important parameters contributing to a project's effort, cost and duration are; the Scale drivers and the Cost drivers (software attributes). The Scale drivers determine the exponent used in the effort equation, while the cost drivers are multiplicative factors that determine the effort required to complete a software project.

The Scale drivers are:

- Precedence
- Development Flexibility
- Architecture / Risk Resolution
- Team Cohesion
- Process Maturity

The Cost drivers are:

- **Product Factors**
  - Reliability (RELY)
  - Data (DATA)
  - Complexity (CPLX)
  - Reusability (RUSE)
  - Documentation (DOCU)
- **Platform Factors**
  - Time constraint (TIME)
  - Storage constraint (STOR)
  - Platform volatility (PVOL)
- **Personnel Factors**
  - Analyst capability (ACAP)
  - Program capability (PCAP)
  - Applications experience (APEX)
  - Platform experience (PLEX)
  - Language and tool experience (LTEX)
  - Personnel continuity (PCON)
- **Project Factors**
  - Software tools (TOOL)
  - Multisite development (SITE)
  - Required schedule (SCED)

The COCOMO II model makes its estimates of required effort (measured in Person-Months (PM)) based primarily on an estimate of the software project's size (KSLOC)) (Boehm, 2000):

$$\mathbf{Effort = a \times EAF \times (KSLOC)^{E}} \qquad (1)$$

$$\mathbf{Cost - Estimate = Effort + g} \qquad (2)$$

'a' has a constant value of 2.94

where:

EAF is the Effort Adjustment Factor derived from the Cost Drivers

E is an exponent derived from the five Scale Drivers

g is the additional fixed cost of the project.

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 will be required to complete it: (Boehm, 1981)

$$\boldsymbol{Effort = 2.94 \times (1.0) \times (8)^{1.0997} = 28.9\ Person - months} \qquad (3)$$

a) Effort Adjustment Factor

The Effort Adjustment Factor in the effort equation is simply the product of the effort multipliers corresponding to each of the cost drivers for a 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 (Boehm, 2000).

$$\boldsymbol{Effort\ Adjustment\ Factor = 1.34 \times 1.0 = 1.46} \qquad (4)$$

$$\boldsymbol{Effort = 2.94 \times (1.46) \times (8)^{1.0997} = 42.4\ Person - months} \qquad (5)$$

b) COCOMO II Schedule Equation

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

$$\boldsymbol{Duration = 3.67 \times (Effort)^{SE}} \qquad (6)$$

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:

$$\boldsymbol{Duration = 3.67 \times (42.3)^{0.3179} = 12.1 months} \qquad (7)$$

c) Important Notes

The following points are to be taking into consideration about COCOMO II: (Boehm, 2000)

- The COCOMO II cost estimation model was calibrated for USA environment.
- Unlike other cost estimation models, COCOMO II is an open model, because it can be recalibrated to reflect your software environment.
- The COCOMO II calculations are based on the estimates of a project's size in thousand of Source Lines of Code (KSLOC) (Boehm, 2000 and Touesnard, 2004).

A KSLOC estimate of a software system can be obtained from experience, the size of previous systems, the size of a competitor’s system, and breaking down the system into smaller pieces and estimating the SLOC of each piece (David, 2002).

d) Descriptions of COCOMO Suite of Models

Table 2 shows the different COMOCO models.

Table 2: Description of evolution of COCOMO Suite of Models (Valerdi et al, 2005)

Model	**Description**
COCOMO 81	Constructive Cost Model I (1981)
COCOMO II	Constructive Cost Model II (2000)
COINCOMO	Constructive Incremental Cost Model (2004)
DBA COCOMO	DataBase Access COCOMO (2004)
COQUALMO	Constructive Quality Model (1998)
iDAVE	Information Dependability Attribute Value Estimation (2003)
COPLIMO	Constructive Product Line Investment Model (2003)
COPSEMO	Constructive Phased Schedule and Effort Model (1998)
CORADMO	Constructive Rapid Application Development Model (1999)
COPROMO	Constructive Productivity-Improvement Model (1998)
COCOTS	Constructive Commercial Off-the-Shelf Cost Model (2000)

COSYSMO	Constructive Systems-Engineering Cost Model (2002)
COSOSIMO	Constructive System-of-Systems Integration Cost Model (2004)

3. THE DERIVATION OF THE NEW MODEL

In the derivation of the new model, we considered two aspects of programming:

(a) Structured Programming

(b) Object Oriented Programming (OOP).

3.1 The Model Parameters

The parameters used for the new model are Source Lines of Codes (SLOC) for structured programming or Modified Object Points (MOP) for object oriented programming, software life-cycle (waterfall model), software attributes and the scale factors.

a) The Source Lines of Codes

Putnam suggests that for each piece, three distinct estimates should be made (Putnam, 1980):

- Smallest possible SLOC = *a*
- Most likely SLOC = *m*
- Largest possible SLOC = *b*

Then the expected SLOC for piece s_i can be estimated by adding the smallest estimate, largest estimate, and four times the most likely estimate and dividing the sum by 6. This calculation is represented by the following formula:

$$\boldsymbol{s_i = \frac{a + 4m + b}{6}} \qquad (8)$$

The expected SLOC for the entire software system E is simply the sum of the expected SLOC of each piece:

$$\boldsymbol{S = \sum_{i=1}^{n} s_i} \qquad (9)$$

where n is the total number of pieces (Putnam, 1980).

b) The Object Points (OP)

(i) Calculating Number of Object Points (NOP)

An initial size measure is determined by counting the number of Screens, Reports and Third-generation language (3-GL) components used (Assassa, 2010).
Each object is then classified as *simple, medium, or difficult* using the guidelines shown in Tables 3 and 4 (Fenton, 1997). The number of screens and reports is then weighted according to Table 5. The weights represent the relative effort required to implement an instance of that complexity level (Fenton 1997).

Table 3: Object Point Complexity Levels for Screens

	Number and source of data tables		
Number of view contained	Total < 4	Total < 8	Total 8+
<3	simple	simple	medium
3-7	simple	medium	difficult
8 and above	medium	difficult	difficult

Table 4: Object Point Complexity Levels for Reports

	Number and source of data tables		
Number of sections contained	Total < 4	Total < 8	Total 8+
0-1	simple	simple	medium
2-3	simple	medium	difficult
4 and above	medium	difficult	difficult

Table 5: Complexity Weights for Object Points

Object Types	**Simple**	**Medium**	**Difficult**
Screen	1	2	3
Report	2	5	8
3-GL component	10	10	10

(ii) Summarizing Object Points Counts

- Assess object-counts in the system: number of screens, reports, & 3GL.
- Assess complexity level for each object : simple, medium and difficult.
- Calculate the Number of Object-Point (NOP) count of the system: add all weights for all object instance.

NOP = Total (weighted screens + weighted reports + weighted 3GL components)

For example, assuming an accounting software system has:

- 2 simple screens
- 4 medium screens
- 2 difficult screens

Total = 8 screens

The same system has:

- 1 simple report
- 2 medium reports
- 2 difficult reports

Total = 5 reports

And 3GL components (**Total = 3 components**)
Using the Complexity Weights for Object Points in Table 5 above, the number of object points (NOP) is calculated as follows:

2 simple screens	x 1 =	2
4 medium screens	x 2 =	8
2 difficult screens	x 3 =	6
1 simple report	x 2 =	2
2 medium reports	x 5 =	10
2 difficult reports	x 8 =	16
3 3GL components	x 10 =	30

Total (NOP) = 74

c) Software Life-cycle Parameter
The software life cycle considered here is the 'Waterfall model' by Winston (Darren and Lindsey, 2007).

i) Feasibility study
ii) System requirements/Analysis

iii) Software specification
iv) Program design
v) Coding
vi) Testing/implementation
vii) Operations/Maintenance

d) The Scale Drivers Parameter
The scale drivers determine the exponent used in the Effort Equation. The Barry Boehm (Boehm, 2000) five scale drivers were considered, because of their general applications to software development. They are:

i) Precedence
ii) Development flexibility
iii) Architecture/Risk Resolution
iv) Process Maturity
v) Team cohesion

e) Software Attributes Parameter

- The eight major software attributes already mentioned during data collection were used.
- They are multiplicative factors that determine the effort required to complete a software project.

3.2 Rating of the Modified Model Parameters

(a) Life-cycle Parameter

The rating of the life-cycle model depends on the number of stages passed through during development of the software. E.g., if all the stages are used, the value seven (7) is automatically taken; otherwise, the value is equal to the number of stages used.

(b) The Scale Drivers Parameter

The five scale drivers were rated as follows:

0.1 = Very low
0.2 = Low
0.3 = Average
0.4 = High
0.5 = Very high

(c) The software attributes Parameter

i)	Very low	= 1
ii)	Low	= 2
iii)	Average	= 3
iv)	High	= 4
v)	Very High	= 5

The new model was then derived by using the parameters discussed above. The Boehm's COCOMO II models in equations 1 & 2 were modified to arrive at the models in equations 10 to 13 below:

$$e_i = NSLIC \times (KSLOC)^{ssf} \times ASAT$$

Or

$$\boldsymbol{e_i = NSLIC \times (NOP)^{ssf} \times ASAT} \qquad (\mathbf{10})$$

$$\boldsymbol{E = \sum_{i=1}^{n} e_i} \qquad (\mathbf{11})$$

$$\boldsymbol{COST\ ESTIMATION = N(E + M)} \qquad (\mathbf{12})$$

$$D = ASAT \times (E)^{\frac{ssf}{10}} \qquad (13)$$

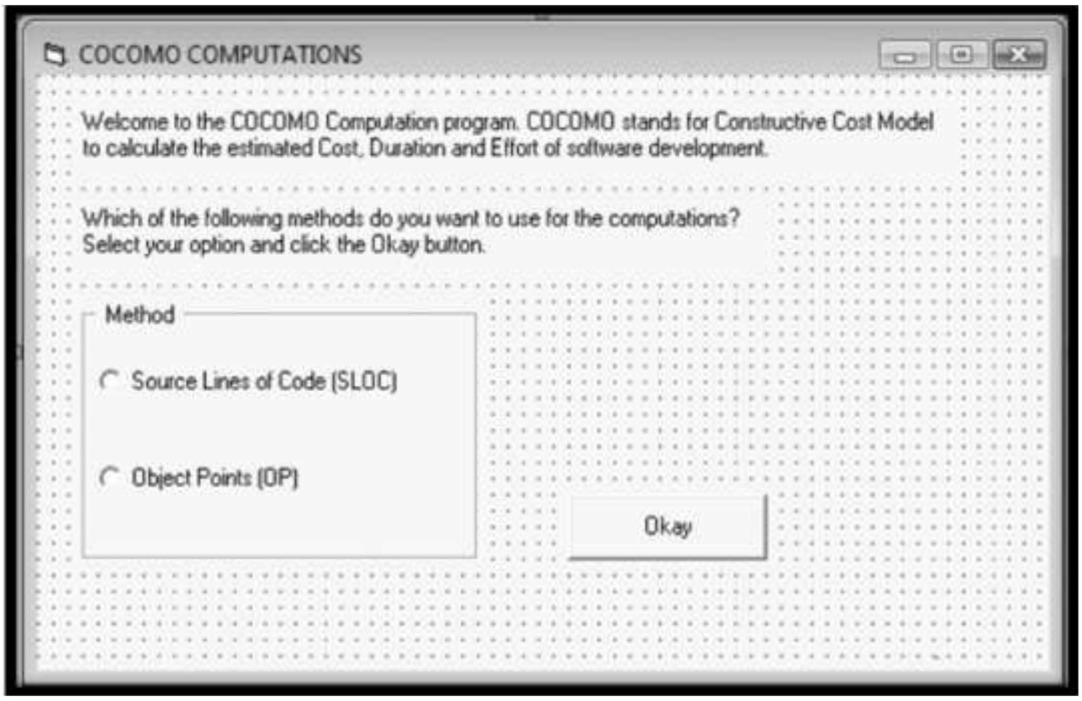
where:

e – Effort of each sub-program
NSLIC - No-of-software-life-cycle used.
SLOC – Source Lines of Codes
NOP - Number Object Point.
ssf – Sum-of-the-rate of scale drivers
ASAT – Average Software Attributes taken
E – Effort of the entire software development
n – Number of sub-programs
M – Miscellaneous cost of the project.
D – Duration or timing to complete the project
ND – Number of Developers.

3.3 Program Design and Implementation

A program was developed to implement the modified cost estimation model using Visual Basic (VB) programming language. The program has three interfaces for the users to interact with.

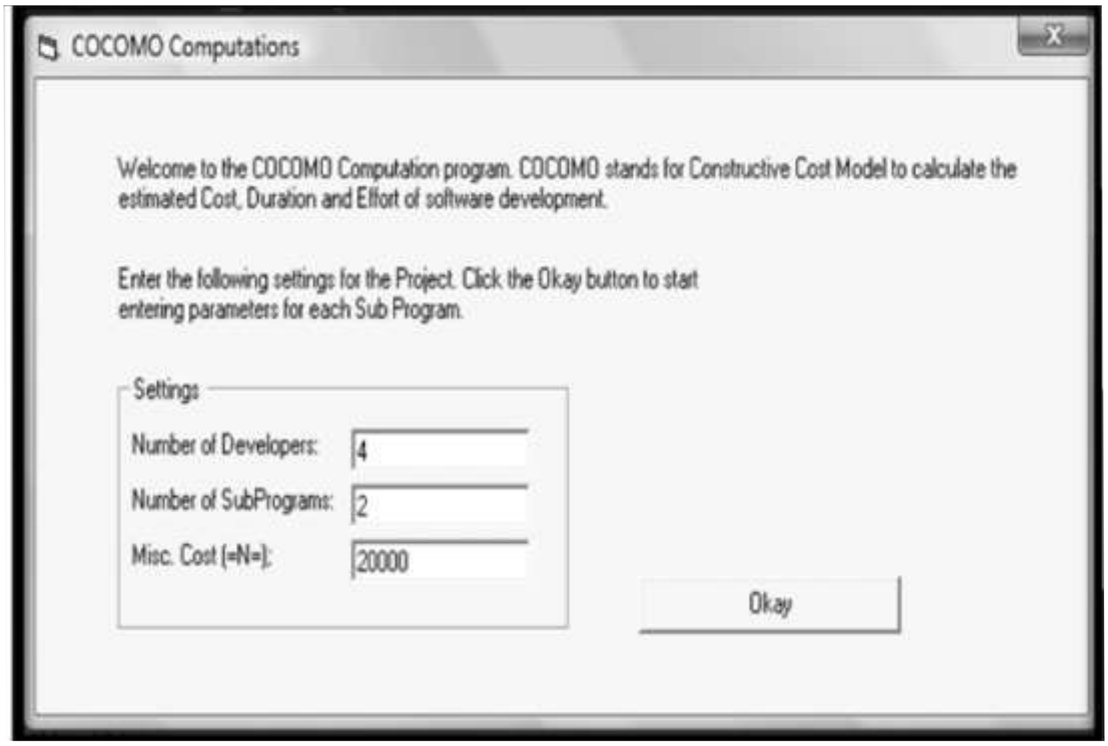
The first interface shows where to indicate whether the user wants to run the program using Source Lines of Codes or Object Points.


COCOMO COMPUTATIONS

Welcome to the COCOMO Computation program. COCOMO stands for Constructive Cost Model to calculate the estimated Cost, Duration and Effort of software development.

Which of the following methods do you want to use for the computations? Select your option and click the Okay button.

Figure 2: First Interface Module



COCOMO Computations

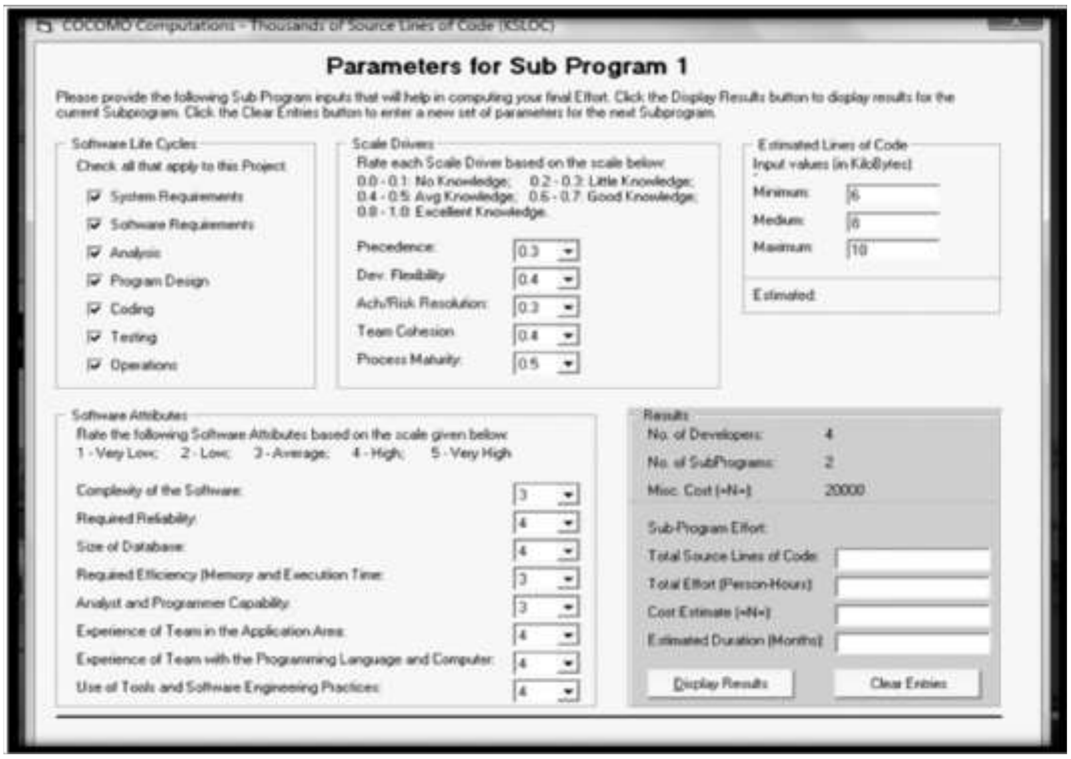
Welcome to the COCOMO Computation program. COCOMO stands for Constructive Cost Model to calculate the estimated Cost, Duration and Effort of software development.

Enter the following settings for the Project. Click the Okay button to start entering parameters for each Sub Program.

Settings

Number of Developers:	4
Number of SubPrograms:	2
Misc. Cost (=N=):	20000

Figure 3: Second Interface Module



COCOMO Computations - Thousands of Source Lines of Code (KSLOC)

Parameters for Sub Program 1

Please provide the following Sub Program inputs that will help in computing your final Effort. Click the Display Results button to display results for the current Subprogram. Click the Clear Entries button to enter a new set of parameters for the next Subprogram.

Software Life Cycles

Check all that apply to this Project

- [x] System Requirements
- [x] Software Requirements
- [x] Analysis
- [x] Program Design
- [x] Coding
- [x] Testing
- [x] Operations

Scale Drivers

Rate each Scale Driver based on the scale below:
0.0 - 0.1: No Knowledge; 0.2 - 0.3: Little Knowledge;
0.4 - 0.5: Avg Knowledge; 0.6 - 0.7: Good Knowledge;
0.8 - 1.0: Excellent Knowledge.

Precedence:	0.3
Dev. Flexibility:	0.4
Ach/Risk Resolution:	0.3
Team Cohesion:	0.4
Process Maturity:	0.5

Estimated Lines of Code

Input values (in KiloBytes)

Minimum:	6
Medium:	8
Maximum:	10

Estimated:

Software Attributes

Rate the following Software Attributes based on the scale given below:
1 - Very Low; 2 - Low; 3 - Average; 4 - High; 5 - Very High

Complexity of the Software:	3
Required Reliability:	4
Size of Database:	4
Required Efficiency (Memory and Execution Time:	3
Analyst and Programmer Capability:	3
Experience of Team in the Application Area:	4
Experience of Team with the Programming Language and Computer:	4
Use of Tools and Software Engineering Practices:	4

Results

No. of Developers:	4
No. of SubPrograms:	2
Misc. Cost (=N=):	20000

Sub-Program Effort:

Total Source Lines of Code:	
Total Effort (Person-Hours):	
Cost Estimate (=N=):	
Estimated Duration (Months):	

Figure 4: Third Interface Module

The third interface is the New Model computations interface that leads to the required results. It consists of the following:

- Software life-cycles selection
- Scale drivers entries
- Estimated size of codes entries
- Software attributes entries.

Entries are entered on the third interface based on the number of sub-programs indicated at the second interface. For example, if the number of sub-programs is two, the third interface will appear two times for input entries.

4.0 MODEL IMPLEMENTATION AND RESULTS

4.1 The Modified Model Estimation Examples

Example 1: In an assessment of an airline sales system built in C by Hassan, 2010 using Object Point estimation technique, the detailed estimates are shown as follows:

The application system has 3 screens and will produce 1 report:

- ➢ **A booking screen:** records a new sale booking.
- ➢ **A pricing screen:** shows the rate for each day and each flight.
- ➢ **An availability screen:** shows available flights.
- ➢ **A sales report:** shows total sale figures for the month and year, and compares figures with previous months and years.

a) Rating of System

- ➢ Booking screen:
  - o Needs three (3) data tables (customer info, customer history table, available seats).
  - o Only 1 view of the screen is enough. So, the booking screen is classified as simple.
- ➢ Similarly, the levels of difficulty of the pricing screen, the availability screen and the sales report are classified as simple, simple and medium, respectively.
- ➢ There is no 3GL component.
- ➢

Again, using the Complexity Weights for Object Points in Tables 3 to 5 above, the number of object points (NOP) is calculated as follows:

That is:

2 simple screens - 2 x 1 = 2
1 medium screen - 1 x 2 = 2
1 medium report - 1 x 5 = 5
0 3GL component - 0 x 10 = 0

NOP = 9

b) Rating of the Cost Drivers (SSF)

The cost drivers contribute to the cost of any software development. The rating was done as follows:

Precedent	= 0.5
Development flexibility	= 0.3
Architecture/Risk Resolution	= 0.4
Team cohesion	= 0.5
Process Maturity	= 0.4
Sum total (ssf)	**= 2.1**

c) Rating of the Software Attribute (SAT)

Complexity of the software	= 3
Required reliability	= 2
Size of data base	= 3
Required efficiency	= 4
Analyst and Programmer capability	= 3
Experience of team in the application area	= 3
Experience of team with the programming language and computer	= 3
Use of tools and software engineering practices.	= 4
Sum total (SAT)	**= 25**

Average Software Attribute **(ASAT)** = 25/8 = **3.125**

Recall, e_i= NSLIC * $(NOP)^{ssf}$ * ASAT is:

$e_i = 7 *(9)^{2.1} *3.125 = 2207.28$

Number of sub-programs: 2 simple screens
1 medium screen
1 medium report

The number of sub-programs (n) is 4:

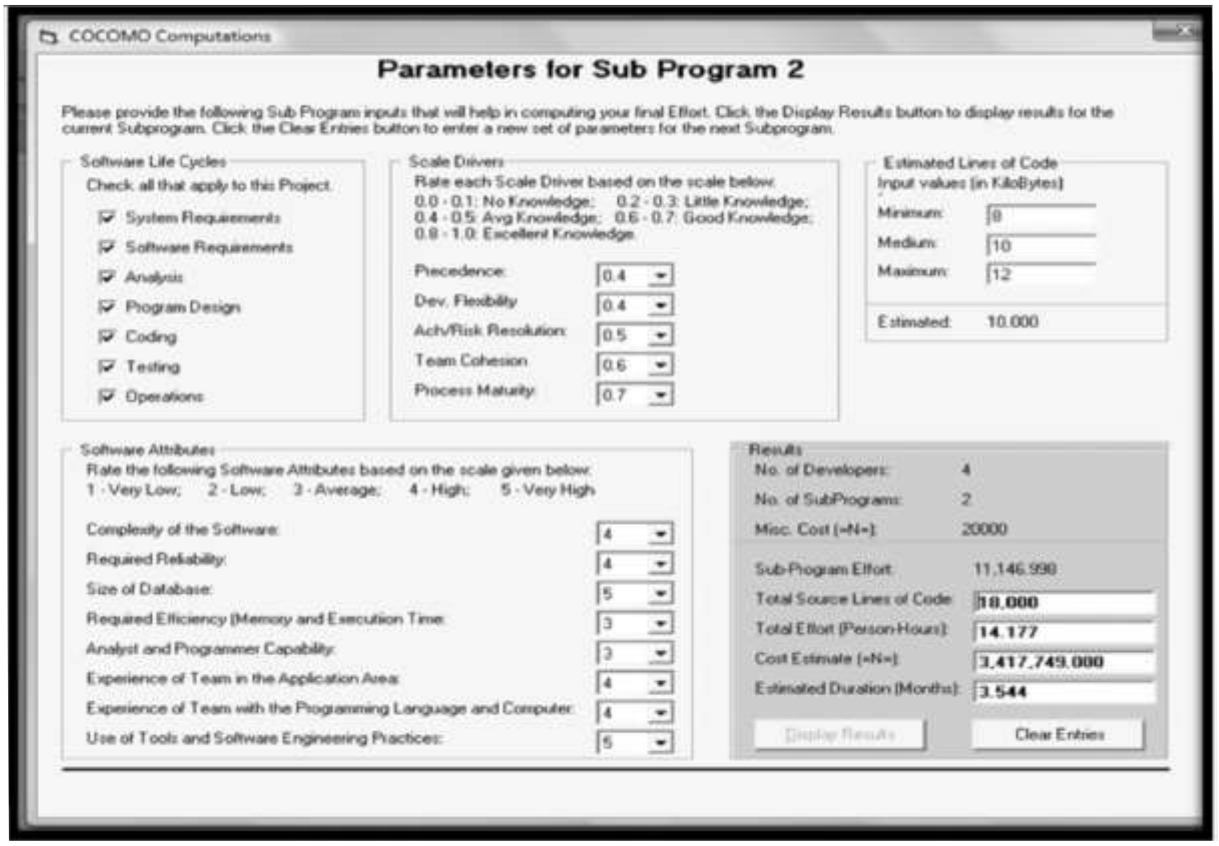
$$E = (2207.28 * 4) = 8829.12$$

Therefore, the **Cost-Estimate = ₦882,912**

The duration of the project is:

$$D = 3.125 \times (22)^{0.21}$$
$$D = 3.65 = 4\ months$$

Example 2:
A project with a Nominal Software Attributes that passed through all the stages of Water-fall Life cycle model and assuming that the project is broken into two (2) sub-programs, number of developers four (4), Miscellaneous of twenty thousand naira (N20,000) and the values of other parameters as shown in the figures below. The new model estimates; Cost and Duration (months) are as follows:


COCOMO Computations

Parameters for Sub Program 2

Please provide the following Sub Program inputs that will help in computing your final Effort. Click the Display Results button to display results for the current Subprogram. Click the Clear Entries button to enter a new set of parameters for the next Subprogram.

Software Life Cycles
Check all that apply to this Project.

- [✓] System Requirements
- [✓] Software Requirements
- [✓] Analysis
- [✓] Program Design
- [✓] Coding
- [✓] Testing
- [✓] Operations

Scale Drivers
Rate each Scale Driver based on the scale below.
0.0 - 0.1: No Knowledge; 0.2 - 0.3: Little Knowledge; 0.4 - 0.5: Avg Knowledge; 0.6 - 0.7: Good Knowledge; 0.8 - 1.0: Excellent Knowledge

Precedence:	0.4
Dev. Flexibility	0.4
Arch/Risk Resolution:	0.5
Team Cohesion	0.6
Process Maturity	0.7

Estimated Lines of Code
Input values (in KiloBytes)

Minimum:	8
Medium:	10
Maximum:	12
Estimated:	10.000

Software Attributes
Rate the following Software Attributes based on the scale given below:
1 - Very Low; 2 - Low; 3 - Average; 4 - High; 5 - Very High

Complexity of the Software:	4
Required Reliability:	4
Size of Database:	5
Required Efficiency (Memory and Execution Time:	3
Analyst and Programmer Capability:	3
Experience of Team in the Application Area:	4
Experience of Team with the Programming Language and Computer:	4
Use of Tools and Software Engineering Practices:	5

Results

No. of Developers:	4
No. of SubPrograms:	2
Misc. Cost (=N=):	20000
Sub-Program Effort:	11,146.998
Total Source Lines of Code:	18,000
Total Effort (Person-Hours):	14.177
Cost Estimate (=N=):	3,417,749.000
Estimated Duration (Months):	3.544

Figure 5: Results interface module of Example 2

The total SLOCs is 18,000 (eighteen thousand) and the total estimated cost is N3,417,749.00 (three million, four hundred and seventeen thousand, seven hundred and forty-nine naira). The estimated duration is 4 months.

5. CONCLUSION

In this work, eight estimation attributes were used to modify the COCOMO II, because they were found to be common among the majority of the respondents during data survey. It is recommended that more attributes should be used to expand the scope of the new model. The Boehm (2000) software scale drivers used have global influence on the power of the KSLOC^{s} and OP^{s} when estimating software cost. Also, apart from the waterfall model used, other existing models (i.e., rapid prototype, spiral model, dynamic systems development model, V-process model) can be tested on the new model

Conclusively, we view this research work as an initial step to indigenous algorithmic model for estimating software cost, which is also friendly to other environments. We state categorically that this research work will provide a good foundation, which can be built upon by intending researchers on this work.

References

Albert, L and Jayesh, P (1992), "Nine Management Guidelines for Better Cost Estimating", CACM, Vol. 35, No. 2.
Assassa, G. (2010), *Software Cost Estimate*, Software Engineering, Pressman Publisher, Sommerville, pp 12-22.
Bauer, F.L. (1972), "Software Engineering in Information Processing", Proceedings of the IFIP Congress held in Ljubljana, Yugoslavia, pp 71.
Bernard, L. (1987) "Cost Estimation for Software Development", Addision_Wesley.
Boehm, B.W. (1981), Software Engineering Economics, Prentice-Hall, Englewood Cliffs, USA. pp 767-776.
Boehm, B.W. (2000), Software Cost Estimate with COCOMO II, Prentice-Hall, Englewood Cliffs, USA.
David, G. (2002), "Schaum's Outline of Software Engineering", New York: McGraw-Hill Trade, USA.
Darren, D. and Lindsey, B. (2007), "Successful IT Projects", Thomson Learning High Holborn Row, Middlesex University Press, London WC1R 4LR, UK, pp 12-13.
Donelson, W.S. (1976), *"General Overview of Software Cost Estimation Methods"*, Project planning and control, Datamation, Vol. 22, No. 6, pp. 73-80.
Fenton, N.E. and Pfeeger, S.L. (1997), *Software Metrics: A Rigorous and Practical Approach*, International Thomson Computer Press.
Hassan, A.E. (2010), Software Architecture, Examples of COCOMO II, CISC 322, pp 1-17.
Liming, W (2009 & 2012), The Comparison of the Software Cost Estimating Methods, University of Calgary, wul@cpsc.ucalgary.ca & www.compapp.dcu.ie/LWu1.html
Putnam, L.H. (1980), "Example of an Early Sizing, Cost and Schedule Estimate for an Application Software System", *A Tutorial on Software Cost Estimation and Life-Cycle Control*, IEEE Computer Society, New York: Computer Society Press, pp 102-127.
Touesnard, B. (2004), "Software Cost Estimation: SLOC-based Models and the Function Points Model", Version 1.1, http://www.ifpug.org
Valerdi R, Lane J.A. and Brown A.W. (2005), *COCOMO Suite Methodology and Evolution*, The Journal of Defense Software Engineering, Vol. 5, No. 1, pp 20-25.
Winston W. R. (1970). "Managing the Development of Large Software Systems" in: *Technical Papers of Western Electronic Show and Convention* (WesCon) August 25–28, Los Angeles, USA.