An Approach to Find Maintenance Costs Using Cost Drivers of Cocomo Intermediate Model

C.V.S.R SYAVASYA
[1,]M.Tech, GITAM UNIVERSITY

Abstract:
Maintenance of software under several cost drivers is as sort of assigning a range of ratings instead of a single point rating to get range of estimates. Several parameters come into use like Annual change traffic and development costs in man-months. Software is being used worldwide, in the same way maintenance of software is equally important as of development of software. Here, we have taken particular range values of lines of code to evaluate costs pertaining to development which intern evaluates final maintenance costs. Every multiplier is taken randomly according to characteristics of product and made a product with development costs to obtain maintenance costs of software. This procedure is done for all the three modes namely organic, semidetached and embedded modes in order to obtain comparison statements of results. The main aim to carry out this review is to explore the changes in maintenance of software when new cost drivers are introduced to the product.

Keywords: Eaf, Act, Cocomo, Ks, Sloc, Nnl, Nml, Nol

1.Introduction

As COCOMO is a non-proprietary model, its details are available in the public domain [1], encouraging researchers and practitioners in the software engineering community to independently evaluate the model. There have been many extensions independently reported; use machine learning techniques to generate effort models from the original COCOMO model. Gulezian proposed a calibration method by transforming the model equation into a linear form and estimating the model parameters using standard linear regression techniques.

Source Lines of Code [2]

The COCOMO calculations are based on your estimates of a project's size in Source Lines of Code (SLOC). SLOC is defined such that only Source lines that are DELIVERED as part of the product are included, test drivers and other support software is excluded, SOURCE lines are created by the project staff , code created by applications generators is excluded. The original COCOMO 81 model was defined in terms of Delivered Source Instructions, which are very similar to SLOC. The major difference between DSI and SLOC is that a single Source Line of Code may be several physical lines. For example, an "if-then-else" statement would be counted as one SLOC, but might be counted as several DSI. Glance of cocomo model: The COCOMO cost estimation model is used by thousands of software project managers, and is based on a study of hundreds of software projects. Unlike other cost estimation models, it is an open model, so all of the details are published, including: The underlying cost estimation equations are obtained by software maintenance cost formulated. It is well defined, and because it doesn't rely upon proprietary estimation algorithms, Costar offers these advantages to its users: Its estimates are more objective and repeatable than estimates made by methods relying on proprietary models it can be calibrated to reflect your software development environment, and to produce more accurate estimates Costar is a faithful implementation of this model that is easy to use on small projects, and yet powerful enough to plan and control large projects. Typically, you'll start with only a rough description of the software system that you'll be developing, and you'll use Costar to give you early estimates about the proper schedule and staffing levels. As you refine your knowledge of the problem, and as you design more of the system, you can use Costar to produce more and more refined estimates.Costar allows defining a software structure to meet your needs. Your initial estimate might be made on the basis of a system containing 3,000 lines of code. Your second estimate might be more refined so that you now understand that your system will consist of two subsystems. There are some sizing approaches for estimating the software maintenance efforts such as source lines of code (SLOC). COCOMO is used to compute the effort and calendar time based on the size and several characteristics of the software product.

Effort Adjustment Factor [2]

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.

2. Cost Drivers

COCOMO II has 17 cost drivers to assess 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 a 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. *Co*nstructive *Co*st *Mo*del is used for estimating costs of software projects [3]. It was created by Barry W. Boehm and published in 1981 using data collected from 63 projects. The quantity of projects studied and the care in its articulation have made the model popular and encouraged its use. The model offers empirical formulas for estimating software costs. We have taken the COCOMO as the starting point for our research. After application of the first version of his model to a wide range of situations, Boehm concluded that coverage for only one mode of developing software was insufficient, so he provided for three modes: organic, semidetached and embedded. In this way, he recognized the influence of various characteristics such as size, communication needs, experience in similar projects, etc. Furthermore, Boehm provides the COCOMO in three versions: basic, intermediate, and detailed. The basic version is useful for quick estimations, although without fine precision. The intermediate version deals with 15 attributes of the project (reliability required, database size, memory restrictions, response time required, etc.), all of whose valuation acts as a multiplying factor in the model.

3. Estimating Maintenance Cost using Cost Model

The first version of COCOMO is COCOMO 81. This model, extensively described The Software Engineering Economics is still an interesting model to understand. The detailed version deals with estimates in each of the phases in the life cycle of the project. The basic version of the model offers the following formulas to calculate the cost of software development measured in *MM* (i.e., man-months):
The COCOMO 81 model is a static model with a hierarchy of three estimation models.

1) Basic Model: It provides a rough estimate of effort based on size and mode of software development.
2) Intermediate Model: It refines the basic model by use of 15 cost drivers-these are subjective attributes. The impact of cost drivers is considered at the project level, Effort is further adjusted to take into account any schedule constraint.
3) Detailed Model: This model further refines the estimation by considering the phase-wise impact of cost-drivers, and the computations in this are for various sub-systems/modules.
   All the three COCOMO 81 models use size as the main input. Another factor used in all three models is the "mode" of project development- the way the project is going to be handled. The three modes considered are:

1) Organic Mode: Project developed by a small, experienced team working in a familiar environment. The team communicates easily. Projects done like these are small or medium sizes with minimal need for innovations. Environment, hardware and tools are stable.
2) Semi-detached Mode: This mode lies between organic and embedded mode.
3) Embedded Mode: Projects developed by large teams where communication is not achieved easily. Projects are medium to large, being developed under tight constraints, possibly with instable hardware and software environment and requiring complex, innovate processing.
   We propose and review under the intermediate model basing on some multipliers taken randomly according to requirement of project.

4. Intermediate Model:

The Intermediate model refines the Basic Model by introducing cost drivers" [3]. It first uses a nominal cost equation for computation and then applies a factor based on the cost driver to it.
Here, cost driver is the product of a *(size^b) implies:

$$\mathbf{MM_{dev} = a * (size^b) * Multiplier}$$

Where MM_{dev} is the development costs in man-months, a and b are factors to calibrate using past data for each development mode.
There are totally 15 multipliers in Intermediate Model which are rated in different value starting from very low to extra high.

5. Results and Discussion:

According to our review done starting from organic mode, where lines of code starting from 80 to 120, where *KS* is an estimate of the delivered program size in thousands of instructions (or roughly, lines of source code).

To estimate the maintenance cost, another parameter is needed: the annual change traffic (*ACT*) which consists of the proportion of original instructions that undergo a change during a year by addition or modification [5].

$$ACT = (NNL + NML)/(NOL) \quad (4)$$

Where:
NNL = number of new lines, *NML* = number of modified lines, and *NOL* = number of original lines

$$MM_{MAIN} = ACT * MM_{DEV}$$

WHERE **ACT = (*NNL + NML)/ (NOL)***
NNL = number of new lines, *NML* = number of modified lines, and *NOL* = number of original lines *and* MM_{DEV}:
AT Organic mode: $MM_{DEV} = 3.2\ KS^{1.05}$ * **Multiplier value**
Semidetached mode: $MM_{DEV} = 3{\cdot}0\ KS^{1.12}$ * **Multiplier value**
Embedded mode: $MM_{DEV} = 2.8\ KS^{1.20}$ * **Multiplier value**

Finding Act Values

Where, KS is thousand source lines of code. Here the range of KS started from 80 to 120 In first step, we have calculated Annual change traffic values and formulated in the form of table as follows: In the below table, Number of lines added is taken (NNL) as 15, Number of lines deleted (NML) as 20, Total Number of original lines (NOL) as 80. Therefore, According to above formula, ACT is resulted as 0.4375. Number of lines added is taken (NNL) as 25, Number of lines deleted (NML) as 30, Total Number of original lines (NOL) as 90. Therefore, According to above formula, ACT is resulted as 0.611. Number of lines added is taken (NNL) as 35, Number of lines deleted (NML) as 40, Total Number of original lines (NOL) as 100. Therefore, According to above formula, ACT is resulted as 0.75. Number of lines added is taken (NNL) as 45, Number of lines deleted (NML) as 50, Total Number of original lines (NOL) as 110. Therefore, According to above formula, ACT is resulted as 0.8636. Number of lines added is taken (NNL) as 55, Number of lines deleted (NML) as 60, Total Number of original lines (NOL) as 120. Therefore, According to above formula, ACT is resulted as 0.9583. Here, the table is shown between the Total Number of lines taken and obtained Annual Change traffic values.

NOL(Original Lines of Code)	ACT
80	0.4375
90	0.611
100	0.75
110	0.8636
120	0.9583

Table1: Value of ACT after Inserting Values in Equation

Finding Multiplier Values

Now, After Annual change traffic values are obtained, we calculate Development costs in man-months. According to formula to calculate development costs in man-months, we first calculate Multipliers, which are used as a product value for all the three modes namely organic, semidetached, embedded modes.
Now, we assign range of cost driver and calculate Multiplier values as follows:

KS(NOL)	MULTIPLIER EQUATION	MULTIPLIER VALUE
80	RELY(LOW) * SCED(HIGH)	0.915
90	DATA(LOW) * TOOL(HIGH)	0.8554
100	CPLX(LOW)*MODP(HIGH)	0.7735
110	TIME(HIGH)*PCAP(VERY HIGH)	0.777
120	STOR(NOMINAL)*ACAP(VERY HIGH)	0.71

Table2: Showing the Results of Multiplier Values

In the above table, we observe that, at Thousand lines of source code (KS) as 80, by taking RELY (Required Software Reliability) as low and SCED (Required Development Schedule) as high, the product of both parameters resulted as 0.915. At thousand lines of code (KS) as 90, by taking DATA (Database size) as low and TOOL (Usage of software tools) as high, the product of both parameters resulted as 0.8554. At thousand lines of code (KS) as 100, by taking CPLX (Project complexity) as low and MODP (Modem programming practices) as

high, the product of both resulted as 0.7735. In the same way, we have calculated for KS 110 and 120 and obtained values are 0.777 and 0.71.

Now, after knowing the Multiplier values from the above table, we calculate Development costs in man-months by using the formula of MM_{DEV} mentioned as equation1:

AT ORGANIC MODE:

By taking KS values ranging from 80 to 120 and a value is 3.2 and b value is 1.05 in intermediate model of organic mode. Initially, at KS as 80 and substituted in organic mode equation and multiplied by Multiplier value of 80, development cost is obtained as 291.61. In this way, remaining values of development costs are obtained and formulated in form of table as follows:

KS VALUE(NOL)	MM_{DEV}(with multipliers)
80	291.61
90	308.51
100	311.60
110	345.96
120	346.37

Table3: Showing the Results of Development Costs for Its Ks Values

Final Maintenance Costs In Man-Months:

Here, we have obtained values of MMdev and ACT. Hence the product of both Development costs in man-months and Annual Change Traffic gives final Maintenance cost in man-months as follows:

KS	MM_{MAIN}(ACT * MM_{DEV})
80	127.57
90	188.49
100	233.7
110	298.77
120	331.92

Table4: Showing Results Of Final Maintenance Cost

Therefore, as thousand lines of code (KS) increases, Maintenance costs in Man-months also increases in organic mode.

In the same way,
At Semi-Detached Mode:

By taking KS values ranging from 80 to 120 and a value is 3.0 and b value is 1.12 in intermediate model of organic mode. After the product made between each value of cost driver to development cost equation, and the obtained result is multiplied to annual change traffic, finally Maintenance cost in man-months is obtained. The values obtained in calculating Maintenance cost in man-months in Organic mode are used here in calculating maintenance cost in man-months in semidetached mode. The purpose of same values imposing on equation of semidetached mode is to make comparisons between Line of code and maintenance cost in man-months.

KS VALUE(NOL)	MM_{MAIN}
80	162.54 (NNL-15,NML-20)
90	242.1 (NNL-25,NML-30)
100	355.8 (NNL-35,NML-40)
110	389.2 (NNL-45,NML-50)
120	435 (NNL-55, NML-60)

TABLE5: Showing the Final Results of Maintenance Costs for Its KS Values

The above table shows that, for each lines of code taken as same in both organic and semidetached modes, Maintenance costs moves from low to high from each KS point of view.
In the same way Embedded mode values are calculated and are observed that, for every KS value increasing, Maintenance cost in man-months is also increasing for all the three modes. Increasing factor is more by introducing cost drivers multipliers in Development costs.

6. Conclusion:

Multiplier factor is used in the equation to get the total cost. The development cost computed uses the same equation as the basic model. Here, we have taken range of values from multipliers and finally maintenance cost is summarised according to each source lines of code. The aim of taking up this task of exploring new results in maintenance costs by introducing new cost driver multipliers is fulfilled by figuring out the result tables and conclusions are made according to the obtained results. We conclude that, as the multiplier is introduced in a project and as lines of code increases, maintenance cost of project also increases. As part of Future work, Estimation can also be done using a range of new cost drivers based on new requirement to get a range of estimations.

References:

1) http://csse.usc.edu/csse/TECHRPTS/PhD_Dissertations/files/Nguyen_Dissertation.pdf
2) http://www.softstarsystems.com/overview.htm
3) "Software Requirements and Estimation" by Swapna Kishore, Rajesh Naik published by TATA McGRAW HILL
4) Pressman RS. Software Engineering A Practitioner's Approach, ch. 20. McGraw-Hill: New York NY, 1992; 677.