An Effective Model of Effort Estimation for Cleanroom Software Development Approach

Manan Jindal
University of Petroleum and Energy Studies, Dehardun, India Uttarakhand, India

Komal Munjal
University of Petroleum and Energy Studies, Dehardun, India Uttarakhand, India

Anurag Jain
University of Petroleum and Energy Studies, Dehardun, India Uttarakhand, India

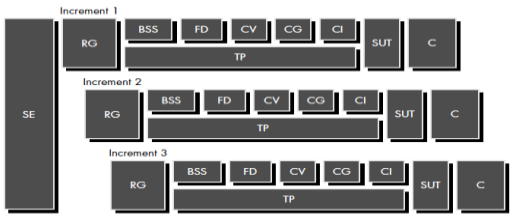
Hitesh Kumar Sharma
University of Petroleum and Energy Studies, Dehardun, India Uttarakhand, India

ABSTRACT
The integration of mathematical modelling, proof of correctness and statistical software quality assurance lead to extremely high-quality software. The integration was named as cleanroom software engineering. It proofs the correctness of the deliverables of each phase, instead of the classic analysis, design, code, test, and debug cycle, the cleanroom approach suggests a different point of view. Due to the evolution in development methodology there is a strong need of evolution in estimation models also. In this work, proposed the new cost estimation model. The evolved model is proposed for the new development methodologies and includes some more factors for estimation used in these new approaches.

Keywords
Cleanroom Software Engineering, COCOMO, Effort Estimation, Cost Drivers, SDLC.

1. INTRODUCTION
Effort estimation is one of the critical tasks in software project management. The estimation for a software projects is hard because of the uniqueness of each and every project. The unavailability of the past data cause for the inaccurate results in effort estimation. Getting 100% accurate estimation is the myth for these kinds of project. As you can see in the graph shown in Figure 1, it has found that there are 400% chances for wrong estimation at the very first stage of the project development. As move to the next phases, the estimation will match with the actual efforts given to the project. As new approaches have been evolved in this decade for software development process, the estimated methods should also evolve. The traditional COCOMO needs some extended feature for accurate calculation of efforts in these new approaches. To estimate the accurate efforts for a project B. Bohem proposed the COCOMO (Cost Constructive Model). He proposed some single variable and multi-variable equation to calculate the efforts for a project. The multi-variable model includes 15 cost drivers involved in traditional development approach. The clean room approach for the project development involves some more cost driver factors.

2. CLEANROOM SOFTWARE ENGINEERING
The cleanroom approach took the software engineering on another level. It mainly emphasizes on specification and design and rigorously test the design before move to the development phase. It uses various box structure specification for proof of correctness. The different phase of cleanroom process for each iteration is shown in figure 1. Once functionality has been assigned to the software element of the system, the continuous pipeline of cleanroom iteration or increment is started [1-5].



SE- System Engineering
RG- Requirements Gathering
BSS- Box structure specification
FD- Formal Design
CV- Correctness Verification
CG- Code generation
CI- Code inspection
SUT- statistical use testing
C- Certification
TP- Test planning

Fig 1: Cleanroom Development Model (Ref: Software Engg.: A practitioner Approach, Roger Pressman)

The following tasks occur:

Increment planning: In this phase the project plan for each increment is developed. The functionality, size and development schedule is created. Special attention is needed to take care that the increments are integrated in timely manner.

Requirements gathering: more detailed description is generated in this phase for each increment

Box structure specification: A box structure is used to specify the increment. It show the main functionality of each increment in connected boxes.

3. COST CONSTRUCTIVE MODEL (COCOMO)
The Cost Constructive Model is the most popular model used for effort and duration estimation. It was proposed in year 1981 by B. Bohem [6]. It was proposed in three different versions.

- Basic COCOMO
- Intermediate COCOMO
- Detailed COCOMO

3.1 COCOMO Basic
This is a single variable effort estimation method. It takes only the numbers of KLOC and results into the efforts in P-M (person month). The formula is given below the table. It also considers the range of the project size in its calculation. As shown in the following tables (Table 1 and 2). The multiplier is changed for the different category of the project.

Table 1: Classes of Projects

Project Class	Project Size KLOC	Deadline	Development Environment
Organic (O)	2-50	Not tight	Simple/Familiar/In-house
Semi-Detached (S)	50-300	Medium	Medium
Embedded (E)	>300 KLOC	Tight	Complex

Table 2: Coefficients a_b, b_b, c_b and d_b values

Project	a_b	b_b	c_b	d_b
O	2.4	1.05	2.5	0.38
S	3.0	1.12	2.5	0.35
E	3.6	1.20	2.5	0.32

where *E* is effort applied in Person-Months, and *D* is the development time in months. The coefficients a_b, b_b, c_b and d_b are given in table 2.The basic COCOMO does not include the other factors involved in developments process. It considers only the size of the project. Bohem evolve the new version of COCOMO in which he considered some more factors involved in effort estimation. This version is explained in the next section.

$$E = a_b\,(KLOC)^{b_b}$$
$$D = c_b(E)^{d_b}$$

3.2 Intermediate COCOMO

Intermediate COCOMO model uses the multivariable approach. It considers and effort adjustment factor (EAF) in calculation of actual effort. The EAF is the multiplication of the values of 15 cost drivers. The formula for calculation is given below.

Table 3: Coefficients a_i, b_i, c_i and d_i values

Project	a_i	b_i	c_i	d_i
O	3.2	1.05	2.5	0.38
S	3.0	1.12	2.5	0.35
E	2.8	1.20	2.5	0.32

There will be six levels for each cost drivers, which ranges from "very low" to "ultra-high". The value for each level has been defined in the table (Table 5). EAF (Effort Adj. Factor) is the multiplication of all efforts ranges from 0.9 to 1.4. The same *E* is used for calculation of time *D*.

Table 4: 15 Cost Drivers

S. No.	Attribute	Cost Driver
1	Product Related	RELY
		DATA
		CPLX
2	Hardware Related	TURN
		VIRT
		STOR
		TIME
3	Personal Related	ACAP
		LEXP
		VEXP
		PCAP
		AEXP
4	Project Related	MODP
		SCED
		TOOL

3.3 Detailed COCOMO

Detailed COCOMO focuses on phase wise effort calculation for different phases in SDLC. This model introduced two new constants. The values of these two constants given in table 5 and 6. The formula is:

$$E_p = \mu_p E$$
$$D_p = \tau_p D$$
$$E = a_i(KLOC)^{b_i} \times EAF$$
$$D = C_i(E)^{d_i}$$

4. E-COCOMO (EXTENDED COST CONSTRUCTIVE MODEL)

As this study has adapted new approaches in software development, there is a strong need of adding some more cost drivers in the list of 15 cost drivers. In this study, worked on two new approaches.

- Cleanroom Software Engineering
- Formal methods for specification

This study has identified that there is a need of inclusion of the one cost driver *i.e.*, **Formal Method Knowledge Capability (FMKC).** This cost driver should involve in intermediate COCOMO. The detailed COCOMO should also evolve with some updated phases. IPRG (Increment planning and Requirement gathering), BSSFD (Box structure specification and Formal Design), CVCG (Correctness verification and code generation), STPUT (Statistical Test planning and Use Testing). The tables (Table 5 and Table 6) is given below for the values of effort and time constants for these updated phases [7, 8].

Table 5: Table for E-COOCMO μ_p used for Cleanroom Engineering Phases

Mode & code size	IRPG	BSSFD	CVCG	STPUT
Small Org	0.150	0.650	0.170	0.030
Medium org	0.150	0.640	0.170	0.040
Medium S	0.160	0.640	0.160	0.040
Large S	0160	0.630	0.150	0.060
Large E	0.180	0.620	0.140	0.060
Large extra	0.180	0.610	0.140	0.070

Table 6: Table for E-COOCMO τ_p used for Cleanroom Engineering Phases

Mode & code size	IRPG	BSSFD	CVCG	STPUT
Small Org	0.140	0.660	0.170	0.030
Medium org	0.140	0.650	0.170	0.040
Mediun S	0.150	0.650	0.160	0.040
Large S	0.150	0.640	0.150	0.060
Large E	0.170	0.630	0.140	0.060
Large extra	0.170	0.620	0.140	0.070

5. ALGORITHMS TO IMPLEMENT E-COCOMO

5.1 Algorithm to Implement Basic COCOMO

Basic COCOMO follows one variable function as defined in section 3.1. To implement these functions two algorithms have been written. Algorithm 5.1.1 to calculate the efforts in basic COCOMO.

5.1.1 Calculate_efforts (int, float[ ][ ])
This algorithm takes the input of KLOC and the basic COCOMO constant values array and returns the calculated efforts in P-M (person-month).

```
Calculate_efforts( int kloc, float basic[][])

            {
         double e = 0.0
         if (kloc > 2 && kloc < 50)
         {
            e = Math.Round(basic[0, 0] *
Math.Pow(kloc, basic[0, 1]), 2);
         }
         if (kloc > 50 && kloc < 300)
         {
            e = Math.Round(basic[1, 0] *
Math.Pow(kloc, basic[1, 1]), 2);
         }
         if (kloc > 300)
         {
            e = Math.Round(basic[2, 0] *
Math.Pow(kloc, basic[2, 1]), 2);
         }
         return e;
      }
```

```
public double calculate_duration(int kloc, float eff)
 {
         double d =0.0;
         if (kloc > 2 && kloc < 50)
         {
            d = Math.Round(basic[0, 2] *
Math.Pow(eff, basic[0, 3]), 2);
         }
         if (kloc > 50 && kloc < 300)
         {
            d = Math.Round(basic[1, 2] *
Math.Pow(eff, basic[1, 3]), 2);
         }
         if (kloc > 300)
         {
            d = Math.Round(basic[2, 2] *
Math.Pow(eff, basic[2, 3]), 2);
         }
          return d;
}
```

5.1.2 Calculate_duration (int, float)
This algorithm takes the input of KLOC and efforts from algorithm 5.1.1. and return the calculated duration in month.

5.2 Algorithm to Implement Basic COCOMO

E-COCOMO follows multi variable function as defined in equation 2. To implement these functions, three algorithms have been written.

5.2.1 Calculate_efforts (int, float[ ][ ])
This algorithm takes the input of KLOC and the array of basic COCOMO constant values and return the calculated efforts in P-M (person-month).

```
Calculate_efforts( int kloc, float basic[][])
            {
         double e = 0.0
         if (kloc > 2 && kloc < 50)
         {
            e = Math.Round(inter[0, 0] *
Math.Pow(kloc, inter[0, 1]), 2);
         }
         if (kloc > 50 && kloc < 300)
         {
            e = Math.Round(inter[1, 0] *
Math.Pow(kloc, inter[1, 1]), 2);
}
         if (kloc > 300)
         {
            e = Math.Round(inter[2, 0] *
Math.Pow(kloc, inter[2, 1]), 2);
         }
        return e;
      }
```

5.2.2 Calculate_duration (int, float)
This algorithm takes the input of KLOC and efforts from algorithm 5.2.1 and return the calculated duration in month.

```
public double calculate_duration(int kloc, float eff)
      {
         double d =0.0;
         if (kloc > 2 && kloc < 50)
         {
            d = Math.Round(inter[0, 2] * Math.Pow(eff,
inter[0, 3]), 2);
         }
         if (kloc > 50 && kloc < 300)
         {
            d = Math.Round(inter[1, 2] * Math.Pow(eff,
inter[1, 3]), 2);
         }
         if (kloc > 300)
         {
            d = Math.Round(inter[2, 2] * Math.Pow(eff,
inter[2, 3]), 2);
         }
          return d;
              }
 public double Calculate_EAF()
          {
         double eaf=0.0;
         double c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,
                  c12,c13,c14,c15,c16;
         if (DATA.Equals("Very Low"))
         {
            c1 = .75;
         }
         if (DATA.SelectedItem.Equals("Low"))
         {
            c1 = .88;
         }
         if (DATA.SelectedItem.Equals("Nominal"))
         {
            c1 = 1.00;
         }
         if (DATA.SelectedItem.Equals("High"))
         {
            c1 = 1.15;
         }
         if (DATA.SelectedItem.Equals("Very High"))
         {
            c1 = 1.40;
         }
         if (DATA.SelectedItem.Equals("Ultra High"))
         {
            c1 = 1.00;
         }
    ///for Second Cost Driver ////////////////////////////////////
         if (STOR.SelectedItem.Equals("Very Low"))
         {
            c2 = .75;
}
```

5.2.3 Calculate_EAF ( )

This algorithm is used to calculate the EAF (Effort Adjustment Factor). It uses the values of 16 Cost drivers and returns the calculated value of EAF.

```
if (STOR.SelectedItem.Equals("Low"))
{
            c2 = .88;
          }
         if (STOR.SelectedItem.Equals("Nominal"))
          {
            c2 = 1.00;
          }
         if (STOR.SelectedItem.Equals("High"))
          {
            c2 = 1.15;
          }
         if (STOR.SelectedItem.Equals("Very High"))
          {
            c2 = 1.40;
          }
         if (STOR.SelectedItem.Equals("Ultra High"))
          {
            c2 = 1.00;
          }
         eaf = c1*c2*c3*c4*c5* c6*c7*c8*c9*c10
                  *c11*c12*c13*c14*c15*c16;
          return eaf;
}
```

The values of the cost drivers are taken from a predefined array as defined in Table 4. The user only enters the level of the cost driver.

6. IMPLEMENTATION OF E-COCOMO

The implementation of E-COCOMO is done in .net framework using C# programming language. The tool is based on the algorithms defined in previous section. It has 7 Main panels from Figure 2 to Figure 7.

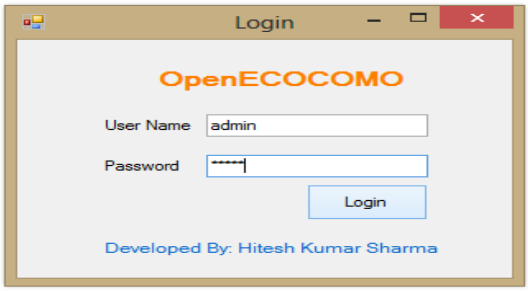


Fig 2: Login Panel

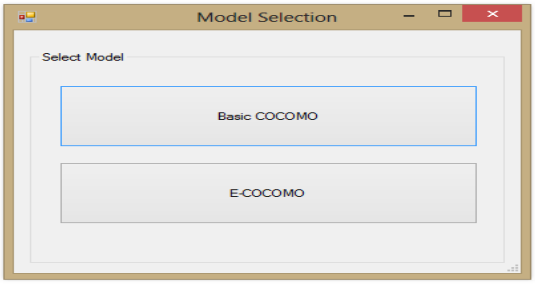


Fig 3: Model Selection Panel

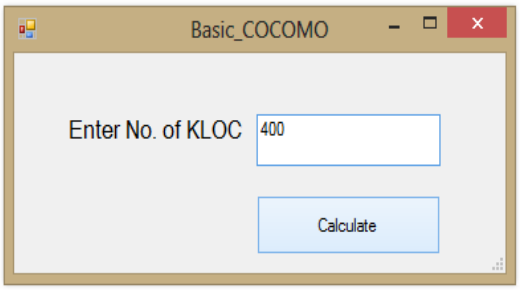


Fig 4: Basic COCOMO input Panel

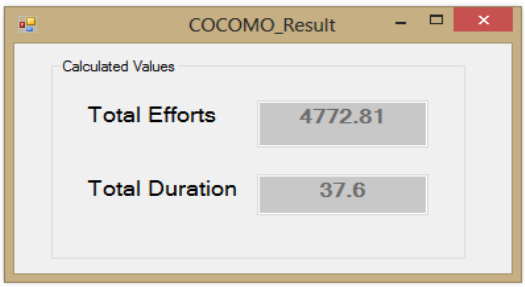


Fig 5: Basic COCOMO Result Panel

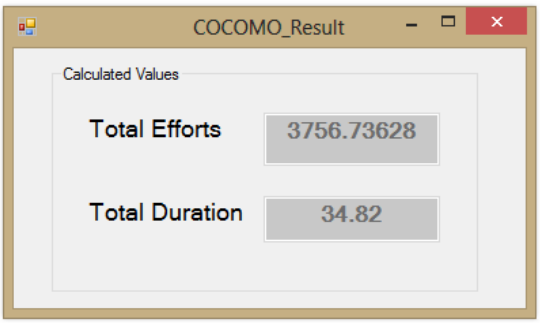


Fig 6: Basic COCOMO Result Panel

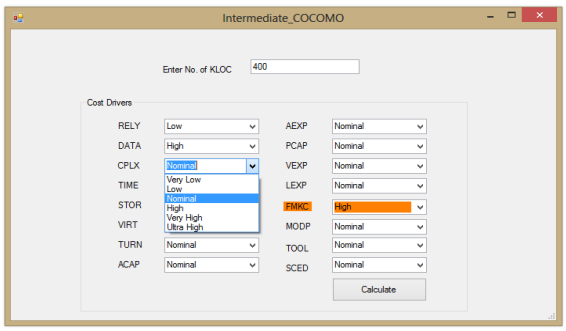


Fig 7: E-COCOMO Result Panel

The Login panel (Figure 2) is used to take the user login credentials. After entering the correct login credentials the user will be able to enter the tool and he/she will be switched to the model selection panel (Figure 3). This panel is used to choose any one model between Basic COCOMO and E-COCOMO. If the user chooses the Basic COCOCMO then he/she will switch to Basic COCOMO input panel (Figure 4) and the panel will asked for number of KLOC. After entering the KLOC and clicking on calculate button the user will get the result shown in Basic COCOMO result panel (Figure 5). In the same way if the user choose E-COCOMO from model selection panel (Figure 6) then he/she will be switched to E-COCOMO input panel (Figure 7). In this panel the user has to enter the value for KLOC and the level for different 16 cost drivers. In panel shown in figure 7, there are six possible levels for the different cost drivers has been given to the user. If the user chooses a level for a cost driver then it will pass the value as per the table 4. Otherwise it passes nominal level for rest of the cost drivers.

After entering the values of the various factors on E-COCOMO Input Panel and clicking on calculate button the user will get the required result on E-COCOMO result panel (Figure 7). The tool has been designed to solve the real-time problem of effort calculation of the software projects.

7. CONCLUSION

In this work, implemented E-COCOMO using C# and developed a new tool (*i.e.*, OpenECOCOMO). This tool can be used to calculate the efforts and time for basic model and E-COCOCMO model both on a single click. In the future work the tool will be evolved with some new parameters for some new approaches like Agile Development, Component based design [9-10].

8. REFERENCES

[1] M. Dyer, 1992. The Cleanroom Approach to Quality Software Development, Wiley.

[2] H. D. Mills, M. Dyer and R. Linger, 1987. Cleanroom Software Engineering, IEEE Software, pp. 19–25.

[3] M. Dyer, 1987. An Approach to Software Reliability Measurement, Information and Software Technology, Volume 29, Issue 8.

[4] G. E. Head, 1994. Six-Sigma Software Using Cleanroom Software Engineering Techniques, Hewlett-Packard Journal, pp. 40–50.

[5] R. Oshana, 1996. Quality Software via a Cleanroom Methodology, Embedded Systems Programming, pp. 36–52.

[6] B. Boehm, 2000. Software Cost Estimation in COCOMO II, Prentice-Hall.

[7] C. Jones, 1996. How Software Estimation Tools Work, American Programmer, Volume 9, Issue 7, pp. 19–27.

[8] J. Matson, B. Barrett, J. Mellichamp, 1994. Software Development Cost Estimation Using Function Points, IEEE Trans. Software Engineering, Volume SE-20, Issue 4, pp. 275–287.

[9] D. Minoli, 1995. Analyzing Outsourcing, Mc Graw-Hill.

[10] D. Phillips, 1998. The Software Project Manager's Handbook, IEEE Computer Society Press.