International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

COCOMO Model - Software Engineering

Mr. Sudarshan Ray[1], Dr. Subba Rao Ch.D.V Rao[2], Ms. Vemulamma Polem[3], Mr. Viswanath Kasi[4]

[1,3]Assistant Professor, CSE. Vaagdevi College Of Engineering Bollikunta Warangal Telangana 506005
[2]Professor, CSE, SVUCE Sri Venkateswara University Tirupati
[4]Assistant Professor, CSE, GEC vijayawada 521356

Abstract
Software engineering is core concept as it Manages complexity in large software projects.
Ensures software is reliable and meets specific goals.
Helps control costs and timelines for better performance of the system with more automation of the project activities

The **Constructive Cost Model (COCOMO)** It was proposed by **Barry Boehm** in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models.
It is a **Software Cost Estimation Model** that helps predict the effort, cost, and schedule required for a software development project.

Table of Content

- What is the COCOMO Model?
- Types of Projects in COCOMO Model
- Structure of COCOMO Model
- Importance of the COCOMO Model
- Types of COCOMO Model
- CASE Studies and Examples
- Advantages of the COCOMO Model
- Disadvantages of the COCOMO Model
- Best Practices for Using COCOMO Model

What is the COCOMO Model?

COCOMO Model is a procedural cost estimate model for **Software Projects** and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality.
The key parameters that define the quality of any **Software Product**, which are also an outcome of COCOMO, are primarily effort and schedule.

Types of Projects in COCOMO Model

In the COCOMO model, software projects are categorized into three types based on their complexity,

size, and the development environment. These types are:

1. Organic

A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.

2. Semi-detached

A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environments lie in between organic and embedded.

The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered Semi-Detached types.

3. Embedded

A software project requiring the highest level of complexity, creativity, and experience requirement falls under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

Comparison of Types of Projects in COCOMO Model

Here is the Comparison in detail where the project types of COCOMO Model

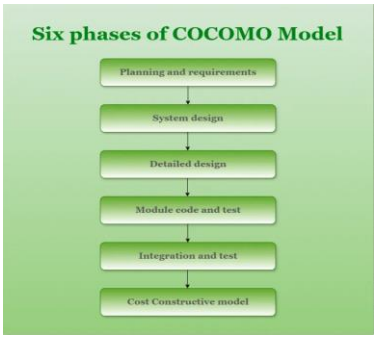
Aspects	Organic	Semidetached	Embedded
Project Size	2 to 50 KLOC	50-300 KLOC	300 and above KLOC
Complexity	Low	Medium	High
Team Experience	Highly experienced	Some experienced as well as inexperienced staff	Mixed experience, includes experts
Environment	Flexible, fewer constraints	Somewhat flexible, moderate constraints	Highly rigorous, strict requirements
Effort Equation	$E = 2.4(400)^{1.05}$	$E = 3.0(400)^{1.12}$	$E = 3.6(400)^{1.20}$
Example	Simple payroll system	New system interfacing with existing systems	Flight control software

Structure of COCOMO Model

Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the **Software Engineering Process.**

In detailed COCOMO, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:



Phases of COCOMO Model

1. **Planning and requirements:** This initial phase involves defining the scope, objectives, and constraints of the project. It includes developing a project plan that outlines the schedule, resources, and milestones
2. **System design:** : In this phase, the high-level architecture of the software system is created. This includes defining the system's overall structure, including major components, their interactions, and the data flow between them**.**
3. **Detailed design:** This phase involves creating detailed specifications for each component of the system. It breaks down the system design into detailed descriptions of each module, including data structures, algorithms, and interfaces.
4. **Module code and test:** This involves writing the actual source code for each module or component as defined in the detailed design. It includes coding the functionalities, implementing algorithms, and developing interfaces.
5. **Integration and test:** This phase involves combining individual modules into a complete system and ensuring that they work together as intended.
6. **Cost Constructive model:** The Constructive Cost Model (COCOMO) is a widely used method for estimating the cost and effort required for software development projects.

Different models of **COCOMO** have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of the constant to be used in subsequent calculations. These characteristics of different system types are mentioned below. Boehm's definition of organic, semidetached, and embedded systems:

Importance of the COCOMO Model

1. **Cost Estimation:** To help with resource planning and project budgeting, COCOMO offers a methodical approach to software development cost estimation.
2. **Resource Management:** By taking team experience, project size, and complexity into account, the model helps with efficient resource allocation.
3. **Project Planning**: COCOMO assists in developing practical project plans that include attainable objectives, due dates, and benchmarks.
4. **Risk management**: Early in the development process, COCOMO assists in identifying and mitigating potential hazards by including risk elements.
5. **Support for Decisions**: During project planning, the model provides a quantitative foundation for choices about scope, priorities, and resource allocation.

6. **Benchmarking**: To compare and assess various software development projects to industry standards, COCOMO offers a benchmark.
7. **Resource Optimization:** The model helps to maximize the use of resources, which raises productivity and lowers costs.

Types of COCOMO Model

There are three types of COCOMO Model:

Cocomo Model types

1. Basic COCOMO Model

The Basic COCOMO model is a straightforward way to estimate the effort needed for a software development project. It uses a simple mathematical formula to predict how many person-months of work are required based on the size of the project, measured in thousands of lines of code (KLOC).

It estimates effort and time required for development using the following expression:

$E = a*(KLOC)^b\ PM$

$Tdev = c*(E)^d$

Person required = Effort/ Time

Where,

E is effort applied in Person-Months

KLOC is the estimated size of the software product indicate in Kilo Lines of Code

Tdev is the development time in months

a, b, c are constants determined by the category of software project given in below table.

The above formula is used for the cost estimation of the basic COCOMO model and also is used in the subsequent models. The constant values a, b, c, and d for the Basic Model for the different categories of the software projects are:

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32
Advanced	4.01	1.75	2.75	0.41

1. The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. The development time is measured in months.
2. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, and expertise is taken into account, henceforth the estimate is rough.

Example of Basic COCOMO Model:

Suppose that a Basic project was estimated to be 400 KLOC (kilo lines of code). Calculate effort and time for each of the three modes of development. All the constants value provided in the following tabl-

e:

Solution: From the above table we take the value of constant a,b,c and d.

1. For organic mode,
- effort = $2.4 \times (400)^{1.05} \approx 1295$ person-month.
- dev. time = $2.5 \times (1295)^{0.38} \approx 38$ months.

2. For semi-detach mode,
- effort = $3 \times (400)^{1.12} \approx 2462$ person-month.
- dev. time = $2.5 \times (2462)^{0.35} \approx 38$ months.

3. For Embedded mode,
- effort = $3.6 \times (400)^{1.20} \approx 4772$ person-month.
- dev. time = $2.5 \times (4772)^{0.32} \approx 38$ months.

Below are the programs for Basic COCOMO Model:

```
// Javascript program to implement basic COCOMO

// Function to calculate parameters of Basic COCOMO
function calculate(table,n,mode,size)
{
var effort,time,staff,model;

// Check the mode according to size

if (size >= 2 && size <= 50)
model = 0; // organic

else if (size > 50 && size <= 300)
model = 1; // semi-detached

else if (size > 300)
model = 2; // embedded


console.log("The mode is ",mode[model]);

// Calculate Effort
effort = table[model][0] * (size ** table[model][1]);

// Calculate Time
time = table[model][2] * (effort ** table[model][3]);
```

```
// Calculate Persons Required
staff = effort / time;

console.log("Effort = ",effort," Person-Month");
console.log("Development Time = ",time," Months");
console.log("Average Staff Required = ",Math.round(staff)," Persons");
}


var table = [[2.4,1.05,2.5,0.38],[3.0,1.12,2.5,0.35],[3.6,1.20,2.5,0.32]]
var mode = ["Organic","Semi-Detached","Embedded"]
var size = 4;
calculate(table, 3, mode, size);

// This code is contributed by satwiksuman.
```

Output

The mode is Organic
Effort = 10.289 Person-Month
Development Time = 6.06237 Months
Average Staff Required = 2 Persons

2. Intermediate COCOMO Model

The basic COCOMO model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system's effort and schedule can be solely calculated based on Lines of Code. For that, various other factors such as reliability, experience, and Capability. These factors are known as **Cost Drivers (multipliers)** and the Intermediate Model utilizes 15 such drivers for cost estimation.

Classification of Cost Drivers and their Attributes:

The cost drivers are divided into four categories

Product attributes:

- Required **Software Reliability** extent
- Size of the application database
- The complexity of the product

Hardware attributes:

- Run-time performance constraints
- Memory constraints
- The volatility of the virtual machine environment
- Required turnabout time

Personal attributes:

- Analyst capability
- Software engineering capability
- Application experience

- Virtual machine experience
- Programming language experience

Project attributes:

- Use of **Software Tools**.
- Application of **Software Engineering Methods**.
- Required development schedule.

Each of the 15 such attributes can be rated on a six-point scale ranging from "very low" to "extra high" in their relative order of importance. Each attribute has an effort multiplier fixed as per the rating. Table give below represents Cost Drivers and their respective rating:

*The **Effort Adjustment Factor (EAF)** is determined by multiplying the effort multipliers associated with each of the 15 attributes.*

The Effort Adjustment Factor (EAF) is employed to enhance the estimates generated by the basic COCOMO model in the following expression:

Intermediate COCOMO Model equation:

$E = a*(KLOC)^b * EAF\ PM$

$Tdev = c*(E)^d$

Where,

- *E is effort applied in Person-Months*
- *KLOC is the estimated size of the software product indicate in Kilo Lines of Code*
- *EAF is the Effort Adjustment Factor (EAF) is a multiplier used to refine the effort estimate obtained from the basic COCOMO model.*
- *Tdev is the development time in months*
- *a, b, c are constants determined by the category of software project given in below table.*

The constant values a, b, c, and d for the Basic Model for the different categories of the software projects are:

Software Projects	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

3. Detailed COCOMO Model

Detailed COCOMO goes beyond Basic and Intermediate COCOMO by diving deeper into project-specific factors. It considers a wider range of parameters, like team experience, development practices, and software complexity. By analyzing these factors in more detail, Detailed COCOMO provides a highly accurate estimation of effort, time, and cost for software projects. It's like zooming in on a project's unique characteristics to get a clearer picture of what it will take to complete it successfully.

CASE Studies and Examples

1. **NASA Space Shuttle Software Development:** NASA estimated the time and money needed to bu-

ild the software for the Space Shuttle program using the COCOMO model. NASA was able to make well-informed decisions on resource allocation and project scheduling by taking into account variables including project size, complexity, and team experience.

2. **Big Business Software Development:** The COCOMO model has been widely used by big businesses to project the time and money needed to construct intricate business software systems. These organizations were able to better plan and allocate resources for their software projects by using COCOMO's estimation methodology.
3. **Commercial Software goods:** The COCOMO methodology has proven advantageous for software firms that create commercial goods as well. These businesses were able to decide on pricing, time-to-market, and resource allocation by precisely calculating the time and expense of building new software products or features.
4. **Academic Research Initiatives:** To estimate the time and expense required to create software prototypes or carry out experimental studies, academic research initiatives have employed COCOMO. Researchers were able to better plan their projects and allocate resources by using COCOMO's estimate approaches.

Advantages of the COCOMO Model

1. **Systematic cost estimation:** Provides a systematic way to estimate the cost and effort of a software project.
2. **Helps to estimate cost and effort:** This can be used to estimate the cost and effort of a software project at different stages of the development process.
3. **Helps in high-impact factors:** Helps in identifying the factors that have the greatest impact on the cost and effort of a software project.
4. **Helps to evaluate the feasibility of a project:** This can be used to evaluate the feasibility of a software project by estimating the cost and effort required to complete it.

Disadvantages of the COCOMO Model

1. **Assumes project size as the main factor:** Assumes that the size of the software is the main factor that determines the cost and effort of a software project, which may not always be the case.
2. **Does not count development team-specific characteristics:** Does not take into account the specific characteristics of the development team, which can have a significant impact on the cost and effort of a software project.
3. **Not enough precise cost and effort estimate:** This does not provide a precise estimate of the cost and effort of a software project, as it is based on assumptions and averages.

Best Practices for Using COCOMO Model

1. **Recognize the Assumptions Underpinning the Model:** Become acquainted with the COCOMO model's underlying assumptions, which include its emphasis on team experience, size, and complexity. Understand that although COCOMO offers useful approximations, project results cannot be predicted with accuracy.
2. **Customize the Model**: Adapt COCOMO's inputs and parameters to your project's unique requirements, including organizational capacity, development processes, and industry standards. By doing this, you can be confident that the estimations produced by COCOMO are more precise and appropriate for your situation.
3. **Utilize Historical Data:** To verify COCOMO inputs and improve estimating parameters, collect and examine historical data from previous projects. Because real-world data takes project-specific

aspects and lessons learned into account, COCOMO projections become more accurate and reliable.

4. **Verify and validate:** Compare COCOMO estimates with actual project results, and make necessary adjustments to estimation procedures in light of feedback and lessons discovered. Review completed projects to find errors and enhance future project estimation accuracy.
5. **Combine with Other Techniques:** To reduce biases or inaccuracies in any one method and to triangulate results, add COCOMO estimates to other estimation techniques including expert judgment, similar estimation, and bottom-up estimation.

Conclusion

Here we discussed the **COCOMO Model** in detail you learned the basic to advance things related to the **COCOMO Model**. The **Software Development Models** are really helpful for the completing the process of development of software easily.