❖ Empirical Estimation Models - COCOMO II Model

COCOMO 2 is another cost-estimating methodology utilized to calculate the cost of software development. It is a modified version of the original COCOMO that was developed at the University of Southern California. It was mainly designed to resolve the shortcomings of the COCOMO 1 model. Its primary goal is to offer methodologies, quantitative analytic structure, and tools. It computes the total development time and effort that is based on the estimates of all individual subsystems.

It is based on the non-linear reuse formula. This approach assists in offering estimates that represent one standard deviation of the most common estimate. Five scale factors define its effort equation exponent, and it has 17 cost drivers attributed to it. It is useful in the software development cycle (SDLC) non-sequential, reuse, rapid development, and reengineering models.

COCOMO 2 estimation models

There are mainly 4 types of COCOMO 2 estimation models. These models are as follows:

1. Application Composition Model

It is intended for usage with reusable components to create prototype development estimates and operates on the basis of object points. It is more suited to prototype system development.

2. Early Design Model

After gathering requirements, the model is utilized during the system design phase. It generates estimates based on function points, which are subsequently converted into numerous lines of source code. The estimates at this level are based on the basic algorithm model formula. You may utilize the following formula:

Effort = A * Size B * M

In this formula,

- A is a constant whose value should be 94.
- B represents the increase in effort ranges from 1 to 1.24 depending on scaling factors such as development flexibility, precedentedness, team cohesion, and many others.
- The M is determined by the cost drivers or the project parameters.

3. Reuse Model

This model computes the effort that is required to join reusable components and/or program code generated by design or program conversion tools. There are mainly two kinds of reused codes: white-box and black-box code. When there is no knowledge of the code, and no alteration is conducted in it, black box code is employed. In contrast, the white box is utilized when the new code is added. The effort needed to integrate this code is estimated as follows:

E = (ALOC * AT/100)/ATPROD

In this formula,

- ALOC represents the number of LOC that must be modified in a component.
- AT represents the automatically produced adapted code percentage.
- ATPROD represents the productivity in code combining. It may be worth up to 2400 LOC every month.

4. Post Architecture Model

A more accurate software estimate may be generated after designing the system architecture, and it is regarded as the most detailed of all models capable of producing an accurate estimate. The post-architecture model effort may be calculated using the following formula:

Effort = A * Size B * M

Comparison between COCOMO 1 and COCOMO 2

COCOMO 1	COCOMO 2
COCOMO 1 is an abbreviation for Constructive Cost Model 1.	COCOMO 2 is an abbreviation for Constructive Cost Model 2.
It is utilized in the waterfall model of SDLC.	It is useful in quick development, non-sequential, and reuses software models.
It is based on the linear reuse formula.	It is based on the non-linear formula.
It offers estimates of the effort and timeline.	It offers estimates th It has extra factors for expressing software size, including lines of code, object points, and function points.t are one standard deviation from the most common estimate.
The program's size is indicated in code lines.	It has extra factors for expressing software size, including lines of code, object points, and function points.
The requirements given to the software come first in the development process.	It utilizes the spiral type of development.
The 3 development methods define the exponent of the effort equation.	The 5 scale methods define the exponent of the effort equation.
It has 3 submodels.	It has 4 submodels.

It utilizes 15 cost drivers.	It utilizes 17 cost drivers.

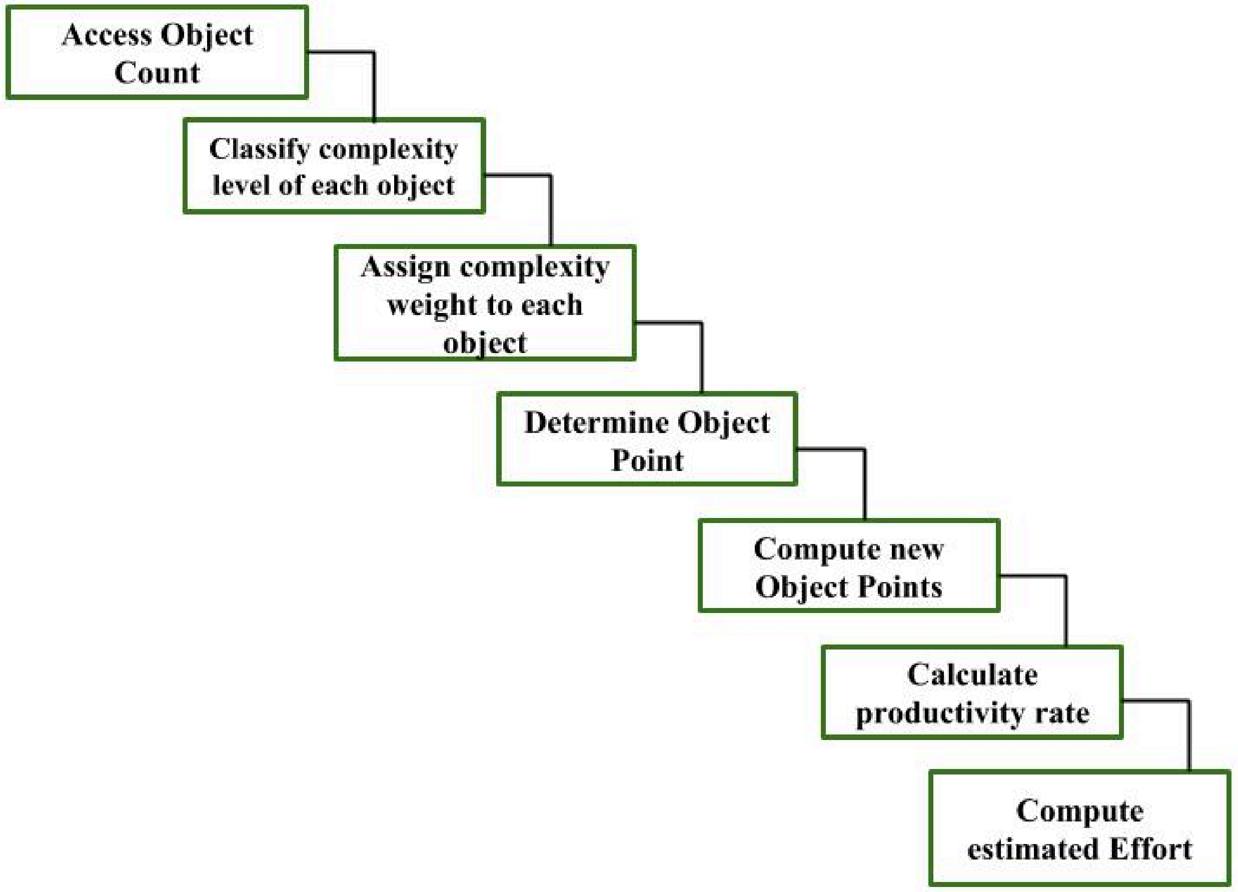
COCOMO II can be used for the following major decision situations:

- Making investment or other financial decisions involving a software development effort.
- Setting project budgets and schedules as a basis for planning and control.
- Deciding on or negotiating tradeoffs among software cost, schedule, functionality, performance or quality factors.
- Making software cost and schedule risk management decisions.
- Deciding which parts of a software system to develop, reuse, lease, or purchase.
- Making legacy software inventory decisions: what parts to modify, phase out, outsource, etc
- Setting mixed investment strategies to improve organization's software capability, via reuse, tools, process maturity, outsourcing, etc.

Application Composition Estimation Model allows one to estimate the cost, and effort at stage 1 of the COCOMO II Model. In this model, size is first estimated using Object Points. Object Points are easy to identify and count. Object Points define screen, reports, and third-generation (3GL) modules as objects. Object Point estimation is a new size estimation technique but it is well suited in the Application Composition Sector.

Estimation of Efforts

The following steps are taken to estimate the effort to develop a project:

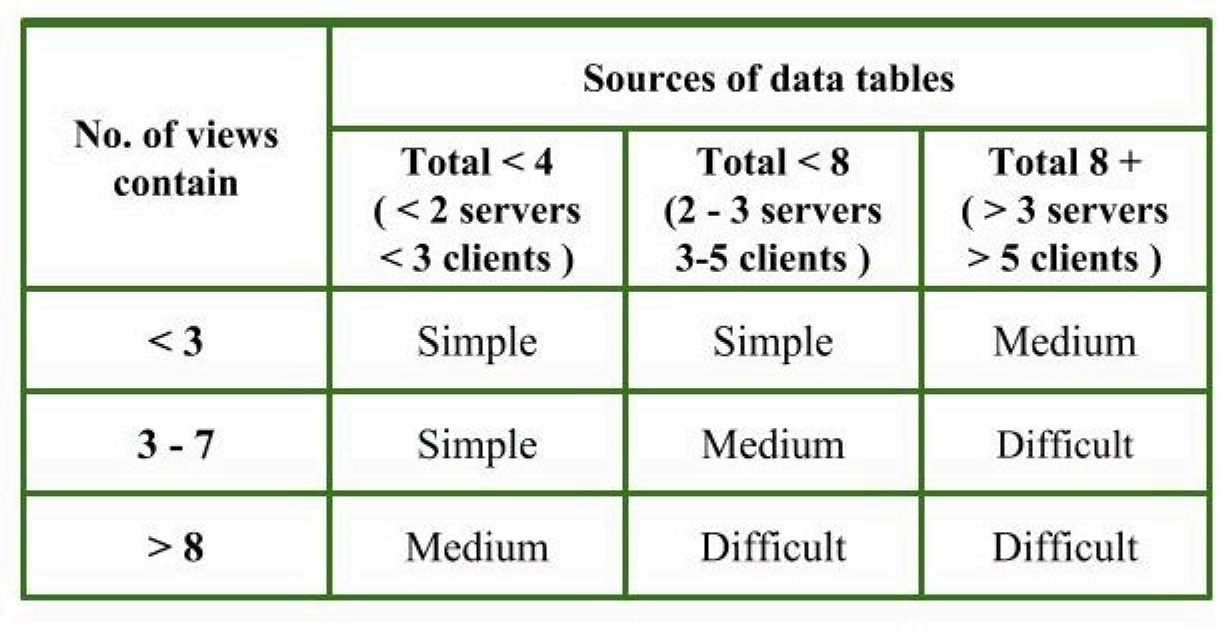

Access Object Count
Classify complexity level of each object
Assign complexity weight to each object
Determine Object Point
Compute new Object Points
Calculate productivity rate
Compute estimated Effort

1. Access Object Counts

Estimate the number of screens, reports and 3GL components that will comprise this application.

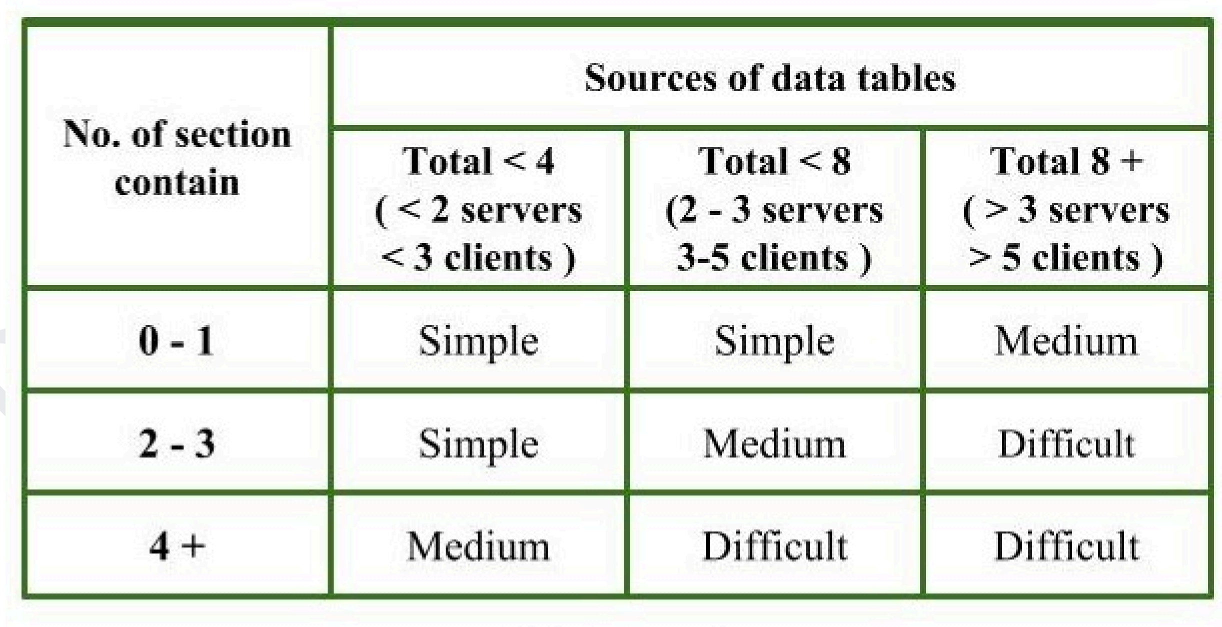
2. Classify Complexity Levels of Each Object

We have to classify each object instance into simple, medium and difficult complexity levels depending on values of its characteristics. Complexity levels are assigned according to the given table.



No. of views contain	Sources of data tables		
	Total < 4 (< 2 servers < 3 clients)	Total < 8 (2 - 3 servers 3-5 clients)	Total 8 + (> 3 servers > 5 clients)
< 3	Simple	Simple	Medium
3 - 7	Simple	Medium	Difficult
> 8	Medium	Difficult	Difficult

For Screens

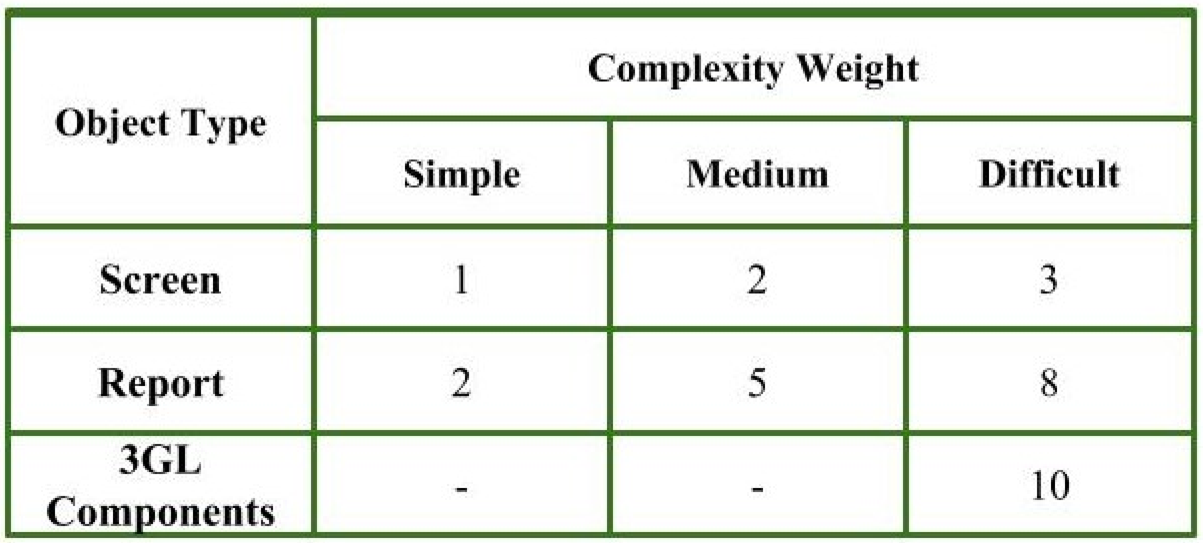


No. of section contain	Sources of data tables		
	Total < 4 (< 2 servers < 3 clients)	Total < 8 (2 - 3 servers 3-5 clients)	Total 8 + (> 3 servers > 5 clients)
0 - 1	Simple	Simple	Medium
2 - 3	Simple	Medium	Difficult
4 +	Medium	Difficult	Difficult

For Reports

3. Assign Complexity Weights to Each Object

The weights are used for three object types i.e, screens, reports and 3GL components. Complexity weights are assigned according to the object's complexity level using the following table.



Object Type	Complexity Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Components	-	-	10

Complexity Weight

4. Determine Object Points

Add all the weighted object instances to get one number and this is known as object point count.

Object Point = Sigma (number of object instances) * (Complexity weight of each object instance)

5. Compute New Object Points (NOP)

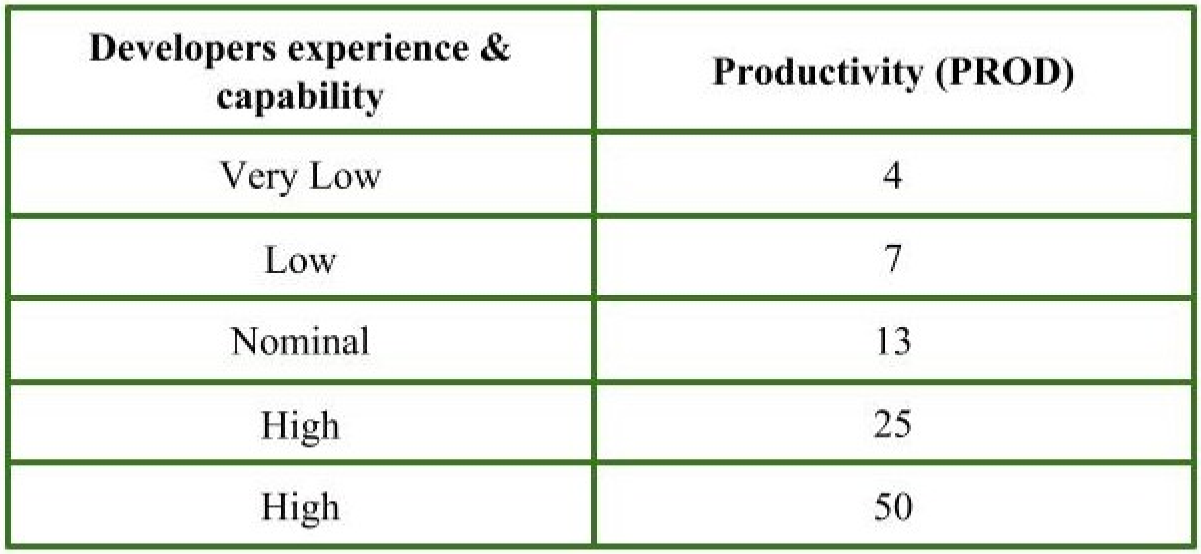
We have to estimate the %reuse to be achieved in a project. Depending on %reuse

NOP = [(object points) * (100 – % reuse)] / 100

NOP are the object point that will need to be developed and differ from the object point count because there may be reuse of some object instance in the project.

6. Calculate Productivity rate (PROD)

Productivity rate is calculated on the basis of information given about developer's experience and capability. For calculating it, we use the following table.



Developers experience & capability	**Productivity (PROD)**
Very Low	4
Low	7
Nominal	13
High	25
High	50

Productivity Rate

7. Compute the estimated Effort

Effort to develop a project can be calculated as:

Effort = NOP / PROD

Effort is measured in person-month.

Example

Consider a database application project with

- The application has four screens with four views each and seven data tables for three servers and four clients.
- Application may generate two reports of six section each from seven data tables for two servers and three clients.
- 10% reuse of object points.
- Developer's experience and capability in a similar environment is low.

Calculate the object point count, New object point and effort to develop such project.

Step 1: Number of screens = 4 and Number of records = 2

Step 2: For screens,

Number of views = 4

Number of data tables = 7

Number of servers = 3

Number of clients = 4

By using above given information and table (For Screens), Complexity level for each screen = medium

For reports,

Number of sections = 6

Number of data tables = 7

Number of servers = 2

Number of clients = 3

By using above given information and table (For Reports), Complexity level for each report = difficult.

Step 3: By using a complexity weight table we can assign complexity weight to each object instance depending upon their complexity level.

Complexity weight for each screen = 2

Complexity weight for each report = 8

Step 4:

Object point count = sigma (Number of object instances) * (its Complexity weight)

= 4 * 2 + 2 * 8 = 24

Step 5:

%reuse of object points = 10% (given)

NOP = [object points * (100 – %reuse)] / 100

= [24 * (100 -10)]/100 = 21.6

Step 6:

Developer's experience and capability is low (given) Using information given about developer and productivity rate table

Productivity rate (PROD) of given project = 7

Step 7:

Effort = NOP / PROD

= 21.6/7

= 3.086 person-month

Therefore, effort to develop the given project = 3.086 person-month.