There are many techniques that can be used to rigorously estimate or measure effort and cost for a software project, such as:

- Function Point (FP)
- Source Lines of Code (SLOC) .
- COnstructive COst MOdel (COCOMO)
- Delphi

4.1. SLOC Technique (Source Line of Code Technique)

- The SLOC technique is an objective method of estimating or calculating the size of the project.
- The project size helps determine the resources, effort, cost, and duration required to complete the project.
- It is also used to directly calculate the effort to be spent on a project.
- We can use it when the programming language and the technology to be used are predefined.
- This technique includes the calculation of lines of codes (LOC), documentation of pages, inputs, outputs, and components of a software program.

4.1.1. Counting SLOC

-The use of SLOC techniques can be used in the case of the technology or language remains unchanged throughout the project. Generally, it can be used when you are using third- generation language, such as FORTRAN or COBOL.

-To count the SLOC the following must be considered:

→ **The count includes:**
- The SLOC delivered to client.
- The SLOC written only by the development team are counted
- The declaration statements are counted as source lines of code

→ **The count excludes:**
- The code created by application generators.
- The comments inserted to improve the readability of program.

- Once, you get the numbers of line of code of SLOC, you can estimate or calculate the total effort and cost to complete the given project.

Example:

- Assume estimated lines of code of a system is: 33,200 LOC
- Average productivity for system of this type is: 620 LOC/person-month
- There are 6 developers
- Labor rate is: $ 800 per person-month

→Calculate the total effort and cost required to complete the above project.

Solution

+Way1

=> Total Effort = Total LOC/Productivity = 33200/620=53.54 ≈ 54 person-months

=> 6 developers →Effort = Total Effort/6 = 54/6 = 9 months

=> Total Cost = Total Effort * Labor Rate = 54 * 800 ≈ $43,200

+Way2

=> Cost per LOC = Labor Rate /Productivity=800/620=$1.29 ≈ $1.3

=> Total Cost = Total LOC * Cost per LOC = 33,200* 1.3=$43160 ≈ $43,200

=> Total Effort = Total Cost / Labor Rate = 43,200/800 = 54 person-months

4.1.2. Disadvantages of Using SLOC Technique

- The SLOC technique is language-dependent (***i.e*** It depends on specific type of language experienced).
- The effort required to calculate source lines of code. This may not be the same for all languages.

4.2. FP Technique (Function Point Technique)

4.2.1- History .

- Function points were developed as an alternative to lines of codes to measure the size of software.
- Allan.J.Albrecht invented function point estimates in 1979 at IBM.
- Thereafter, he introduced General System Characteristics (GSCs) in 1984.
- From 1990 onwards, International Function Point Users Group (IFPUG) made periodic revision to the function point theory.

4.2.2- Features

- The total size of a software project is expressed in total function points.
- It is independent 'of the computer language, development methodology, technology, or capability of the project team developing the software project.
- To calculate FP for a project, some components are required.
- The FP technique is a direct indicator or measurement of a software application functionality from the user's perspective.
- It is the most popular technique used to estimate the size of a software project.
- The total size of a project is estimated in the term of FP.
- After calculating the total size of a project in FP, you divide the total FP into the different phases of the software development life cycle. Or you can calculate separately for each phase.
- You can also use it to determine how much effort per FP is required to complete the particular phase. ***For example***, the testing phase is planned for 20 FP of work. And the project managers, basing on their past project experience, determine the amount of effort in person-months(or man-months), person-days, or person-hours required in the testing phase. For instance, they experienced spend 2 days per FP, then the total effort in this case is (2 * 20)/(number of worker) days to complete this phase.
- Similarly, you can express the cost required to complete FP of work for a particular phase.
- At the end of a project, you can also express the number of defects reported per FP for a phase.
- Basing on the size of the project in FP, you can estimate or calculate the numbers of LOC by multiplying the average number of LOC/ FP for a given language (AVC) by the total number of function points of the project.

-To calculate the numbers of line of code, the following formula is used:

$\boxed{\text{LOC = LOC per FP * FP}}$ **OR** $\boxed{\text{LOC = AVC * FP}}$ where:

- AVC: is the average number of LOC/FP for a given language
- LOC: is the numbers of line of code
- FP : is the Total numbers of Function Point

-To define AVC, the following table is used to choose the type of programming languages that will be used when developing a sw project:

Programming Language	LOC/FP (average)	Select
Assembly Language	320	○
C	128	○
COBOL/Fortran	105	○
Pascal	90	○
Ada	70	○
C++	64	○
Visual Basic	32	○
Object-Oriented Languages	30	○
Smalltalk	22	○
Code Generators (PowerBuilder)	15	○
SQL/Oracle	12	○
Spreadsheets	6	○
Graphical Languages (icons)	4	○

Note: You may use the similar one from the table for other languages.

-To calculate the total FP, the following formula is used:

$$\boxed{FP = CT * (0.65 + 0.01 * \Sigma F_i)}$$

Where:

-FP: Total Function Points

-CT: Count Total. To calculate this, you have to fill the correct number for the following count of each of 5 parameters and select the one of sw project complexity levels

Measurement Parameter	Count		Simple ○	Average ○	Complex ○		Total
Number of user inputs		×	3	4	6	=	
Number of user outputs		×	4	5	7	=	
Number of user inquiries		×	3	4	6	=	
Number of files		×	7	10	15	=	
Number of external interfaces		×	5	7	10	=	
Count Total	→						

+Each total=each Count * the corresponding number in the complexity level selected
+CT= The sum of each total or CT= total1 + total2 + total3 + total4 + total5

-**ΣF_i** (i=1 to 14): The sum of all 14 Complexity Adjustment Values(CAV) or Complexity Weighting Factors(CWF) which each value is ranged from 0 to 5 depending on the complexity level of each type. To do this, you have to complete each of the following values in the table below:

Table of Complexity Weighting Factors (CWF)
Each factor on a scale of 0 to 5 is the degree of influence which answers to each question
(0 = No influence, 1 = Incidental, 2 = Moderate, 3 = Average, 4 = Significant, 5 = Critical)
This is called to calculate the Total Weighting Factors (TWF)

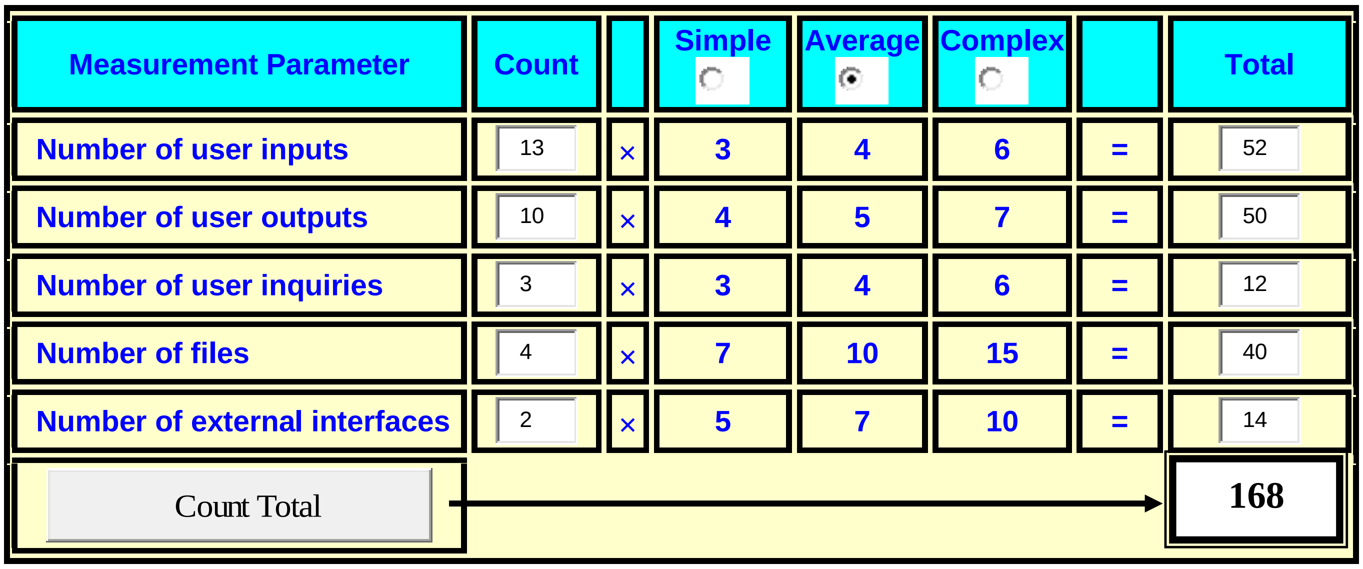
Question	0	1	2	3	4	5
1. Does the system require reliable backup and recovery?	●	○	○	○	○	○
2. Are data communications required?	●	○	○	○	○	○
3. Are there distributed processing functions?	●	○	○	○	○	○
4. Is performance critical?	●	○	○	○	○	○
5. Will the system run in an existing, heavily utilized operational environment?	●	○	○	○	○	○
6. Does the system require on-line data entry?	●	○	○	○	○	○
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?	●	○	○	○	○	○
8. Are there master files updated on-line?	●	○	○	○	○	○
9. Are the inputs, outputs, files, or inquiries complex?	●	○	○	○	○	○
10. Is the internal processing complex?	●	○	○	○	○	○
11. Is the code designed to be reusable?	●	○	○	○	○	○
12. Are conversion and installation included in the design?	●	○	○	○	○	○
13. Is the system designed for multiple installations in different organizations?	●	○	○	○	○	○
14. Is the application designed to facilitate change and ease of use by the user?	●	○	○	○	○	○
Total Weighting Factor(ΣFi)						

→You can then calculate FP by substituting the values of CT and TWF(ΣF_i) into the FP formula above. And then you substitute the value of obtained FP and LOC/FP into the LOC formula above, you will obtain the total numbers of line of code.

→The general characteristics and brief description of the 14 complexity adjustment values above are shown below:

Nº	Characteristics	Description
1	Operational Ease	The degree to which the application attends to operational aspects, such as backup, start-up, and recovery processes
2	Data Communication	The degree to which an application communicates with the other applications
3	Distributed Functions	The degree to which an application transfers or shares data among the component of applications
4	Performance	The degree of the response time and throughput performance of an application
5	Heavily Used on figuration	The degree to which the computer resources, where the application runs, are used
6	On-line Data Entry	The percentage of data that is entered by using interactive transaction
7	Transaction Rate	The frequency of transactions that are executed, on a daily, weekly, or monthly basis
8	On-line Update	The degree to which the number of internal logical files is updated on-line
9	End-user Efficiency	The degree to which human factors and user friendliness are to be considered
10	Complex Processing	The degree to which the complexity of logic influences the processing logic, in turn, influences the development of the application
11	Reusability	The degree to which the application and code in the application are specifically designed, developed, and supported to be reused in other applications
12	Installation Ease	The degree to which conversion from a previous environment influences the development of the application
13	Multiple Sites	The degree to which the application is developed for multiple locations and User organizations
14	Facilitates Change	The degree to which the application is developed for easy modification of processing logic or data structure

Example1: Assuming we have collected data for a sw project as shown in the two tables below. The development team uses SQL Server for developing this sw project. ***Calculate FP and LOC.***



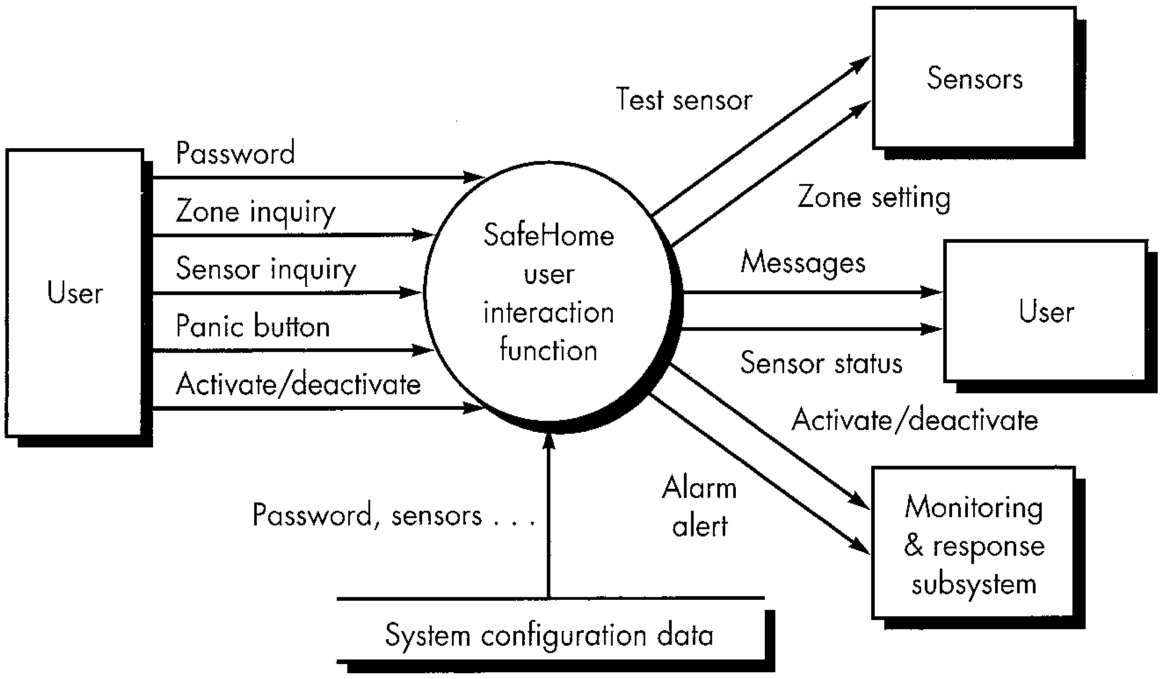
Measurement Parameter	Count		Simple ○	Average [✓ ●]	Complex ○		Total
Number of user inputs	13	×	3	4	6	=	52
Number of user outputs	10	×	4	5	7	=	50
Number of user inquiries	3	×	3	4	6	=	12
Number of files	4	×	7	10	15	=	40
Number of external interfaces	2	×	5	7	10	=	14
Count Total							168

№	*General System Characteristics*	*Degree of Influence (Value)*
1	Operational Ease	2
2	Data Communication	5
3	Distributed Functions	4
4	Performance	5
5	Heavily Used Configuration	2
6	Transaction Rate	3
7	On-line Data Entry	4
8	On-line Update	2
9	End-user Efficiency	3
10	Complex Processing	4
11	Reusability	5
12	Installation Ease	2
13	Multiple Sites	3
14	Facilitates Change	4
Total complexity adjustment or Total Weighting Factor value		48

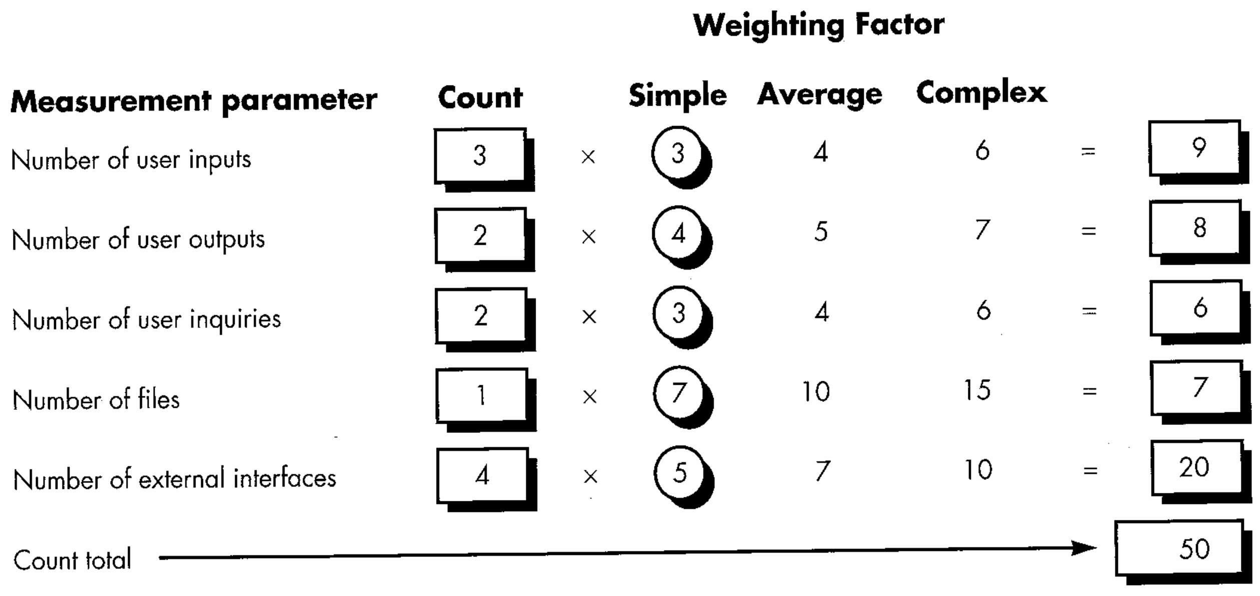
➔ **FP = 168 ∗ (0.65 + 0.01 ∗ 48) = 189.84 ≈ <u>190 FPs</u>**

➔ **LOC = FP ∗ 12 = 190 ∗ 12 = <u>2280 LOCs</u>**

<u>Example2:</u> Part of analysis model for SafeHome software



- Number of **user inputs**	= 3 (password, panic button, and activate/deactivate)
- Number of **user outputs**	= 2 (massages and sensor status)
- Number of **user inquiries**	= 2 (zone inquiry and sensor inquiry)
- Number of **file**	= 1 (system configuration file)
- Number of **external interfaces**	= 4 (test sensor, zone setting, activate/deactivate, and alarm alert)



			Weighting Factor				
Measurement parameter	Count		Simple	Average	Complex		
Number of user inputs	3	×	[✓ 3]	4	6	=	9
Number of user outputs	2	×	[✓ 4]	5	7	=	8
Number of user inquiries	2	×	[✓ 3]	4	6	=	6
Number of files	1	×	[✓ 7]	10	15	=	7
Number of external interfaces	4	×	[✓ 5]	7	10	=	20
Count total						→	50

+Assume Weighting Factor is simple:

➔ CT = (3*3)+(2*4)+(2*3)+(1*7)+(4*5)= **50**

+And assume ∑Fi = 46

➔ FP = 50 x [0.65 + (0.01 x 46)] = **55.50 ≈ 56FP**

4.2.3- Using FP for Initial Estimation

Example:

+ Assuming that:

- To complete one FP of work, the project requires 10 hours (Productivity = 10hours/FP)
- The total FP estimated is 200 FP for the project
- The project team works 8 person-hours per working day
- There are 20 working days in a month

+ Calculate the effort required to complete the sw project

Solution

To complete a project of 200 FP, you require:

+Effort in person-hours:

Effort = 200 FP * 10hrs/FP = **2000 person-hours**

+Effort in person-days:

Effort = 2000 hours/8 hours = **250 person-days**

+Effort in person-months:

Effort = 250 person-days/20 days = **12.5 person-months**

+Note:

- If there are two developers on the project, then you would require **=**12. 5 months/2 developers **= 6.25 months** to complete the work.

4.2.4- FP-Based Estimation

Example:

- Assume FP of a system is: 375

- Average productivity for system of this type is: 6.5FP/pm
- Labor rate is: $ 800 per month

➔Cost per FP is: 800/6.5 ≈ $123

➔Total Estimated Cost is: $123 * 375 ≈ $46,100

➔Total Estimated Effort is: 46,100/800 = 57.62 ≈ 58 person-months

4.2.5- <u>Using FP for Post-Project analysis</u>

- FP can be used to express factor such as **productivity**, **effort**, **defects**, and **cost** used in an already completed project.
- Expressing these factors in FP helps analyze a project effectively.
- Analysis of a project enables to apply the learning derived from a particular project to future projects

Example:

	Total number of defects reported	Total project size in FP
Project 1	10	150
Defect density	10/150 = **<u>0.067</u>** defects /FP	'''14'
Project 2	20	200
Defect density	20/200 = **<u>0.10</u>** defects/FP	
Project 3	40	1000
Defect density	40/1000 = **<u>0.04</u>** defects /FP	,

$$\boxed{\textbf{Defect density = Total number of defects/total project size in FP}}$$

➔Project 3 is of superior quality because it reported least number of defects . per FP of development effort

4.3. <u>COCOMO Techniques</u> (**CO**nstructive **CO**st **MO**del Techniques)

- COCOMO uses cost driver attributes to calculate the effort and duration of a project.
- COCOMO technique has three levels of implementation:
  - **+ Basic**
  - **+ Intermediate**
  - **+ Advanced**

4.3.1- <u>Basic COCOMO</u>

-The basic COCOMO technique is used to estimates the effort and cost of a SW project by using only the lines of code.

-We use basic COCOMO when we need a rough estimate of effort, such as during maintaining a project.

-There are three steps involved in estimating the effort using basic COCOMO technique:

S1- Estimating the total delivered lines of code.

S2- Determine the effort constants based on the type of the project

S3- Substituting values for lines of code and effort constants in a formula

-There are three types of projects to be calculated effort. To determine the type of the project being developed, the end user has to select one of the three (3) types of modes, which are **organic**, **semi-detached**, and **embedded**.

1-Organic Mode: Relatively small, simple software projects in which a small team with good application experience work to a set of less than rigid requirement.

-The organic projects must have sufficient and defined objectives

-These are simple businesses and applications, such as a banking system and inventory system

-The equation for the Effort (E) and Development time (D) for this model are:

2-Embedded Mode: A software project that must be developed within a set of tight hardware, software and operational constraints.

-The embedded projects must have stringent and specialized HW, SW, and human resources requirements.

-Organizations usually have less experience in developing such projects.

-Examples of such projects includes real-time operating systems, industrial automation systems, and sophisticated space and aviation systems

-The equation for the Effort (E) and Development time (D) for this model are:

3-Semi-Detached Mode: An intermediate (in size and complexity) software project in which teams with mixed experience levels must meet a mix of rigid and less than rigid requirements.

- Semidetached projects are combination of the preceding two types of SW projects

- Examples: a new operating system, a database management system

-The equation for the Effort (E) and Development time (D) for this model are:

→Once the end user selects his/her model, he/she calculates the (E)ffort and the (D)evelopment time (Duration).
→In summary, you can select the software project type and substitute the values of **C** and **K** from the following table into the formulas below the table:

Software Project Type	C	K	Select
Organic	3.2	1.05	◉
Embedded	2.8	1.20	○
Semi-detached	3.0	1.12	○

$$\boxed{E_i = C * (KLOC)^K}$$

-Where **E**$_i$ is the effort for a project, **C** and **K** are the effort or COCOMO constants depending on the type of project being developed.
-Table of COCOMO Constants

-Example: Estimate the effort of an application (organic type) with 4 KLOC:

→$E_i = 3.2 * 4^{1.05} = 3.2 * 4.28 \approx$ **14 person-months**

4.3.2- Intermediate COCOMO

-Calculating of effort by using the intermediate COCOMO technique involves an additional step of calculating the effort adjustment factor (EAF).

-The effort adjustment factor is calculated by assigning ratings to 15 cost driver attributes. .

-These cost driver attributes relate to the various aspects of a sw project, such as project, product, personnel, and computer attributes.

-Using the intermediate COCOMO technique we can accurately estimate effort and cost required for a project.

-There are **three steps** in calculating the effort:

-**Step1:** Estimate the initial development effort by using SLOC in the following formula: $\mathbf{E_i = C * (KLOC)^K}$

-**Step2:** The second step is to determine the relevant cost driver attributes that affect your project intensively (This provides you with the value for EAF)

-**Step3:** Finally, you calculate the actual effort by multiplying the weighted cost driver attributes with the initial effort estimation.

-There are 15 intermediate cost driver attributes, 3-product attributes, 4-computer attributes, 5-personnel attributes, and 3-project attributes:

+**3 Product attributes:**
-Required Software Reliability
-Database Size
-Software Product Complexity

+**4 Computer attributes:**
-Execution Time Constraint
-Main Storage Constraint
-Virtual Machine Volatility
-Computer Turnaround Time

+**5 Personnel attributes:**
-Analyst Capability
-Application Experience
-Programmer capability
-Virtual Machine Experience
-Programming Language Experience

+**3 Project attributes:**
-Modern Programming Practices
-Use of Software Tools
-Required Development Schedule

Cost Drivers	Rating					
	Negligible	Low	Average	High	Very High	Extremely critical
Required Software Reliability (RSR)						
Database Size (DBS)						
Software Product Complexity (SPC)						
Execution Time Constraint (ETC)						
Main Storage Constraint (MSC)						
Virtual Machine Volatility (VMV)						
Computer Turnaround Time (CTT)						
Analyst Capability (AC)						
Applications Experience (AE)						
Programmer Capability (PC)						
Virtual Machine Experience (VME)						
Programming Language Experience(PLE)						
Modem Programming Practices (MPP)						
Use of Software Tools (TOOL)						
Required Development Schedule (RDS)						

- Typically, the values that rate each cost driver attribute ranges from 0.9 through 1.4 .
- Usually, in organizations, the average rating is assigned a static value of 1.0
- The intermediate COCOMO technique formula is: **E=EAF*E$_i$**
- **For example**, in a customized insurance project, there are four modules which the total effort estimate of the modules is 3 KLOC. Assuming there are four cost driver attributes with the respective multiplying factors that might affect the project most. C and K are 3.2 and 1.05 (organic project)

=> $E_i = \mathbf{C} * (KLOC)^{\mathbf{K}} = 3.2 * 3^{1.05} = 3.2 * 3.16 = 10.11$

Applicable cost driver attributes	Rating	Multiplying factors
SPC	High	1.2
ETC	Very high	1.35
AC	Low	0.95
MPP	Average	1.0

EAF = 1.2 * 1.35 * 0.95 * 1.0 = 1.53

=> $E = EAF * E_i = 1.53 * 10.11 = 15.5$ person months

4.3.3- <u>Advanced COCOMO</u>

-The advanced COCOMO technique uses the steps of the intermediate COCOMO technique

- In addition, it uses cost driver attributes assigned to each phase of the sw development life cycle such as analysis and design

→Applicability of COCOMO

- COCOMO is flexible and capable of using SLOC and FP
- You can use COCOMO when the size of a project is extensive and the requirements of the project are vague
- In contrast, SLOC and FP can be used for projects where either the requirements are more or less known or developers possess the relevant experience in developing projects
- Generally, you can use COCOMO when the sw development environment is new to an organization
- You can use COCOMO when you do not have baseline data about past projects
- You can use FP or SLOC techniques when you have enough past project data to assign accurate weight age to the 14 GSCs (General System Characteristics) or complexity adjustment values and the various information domain value elements

4.4. Delphi Techniques

- The Delphi technique is a human-based estimation technique
- Human-based estimation techniques use human experience and analytical skills to estimate the size, productivity, and effort required for a project.
- The rationale of using the Delphi technique is that when many experts independently arrive at the same estimate on the basis of similar assumptions, the estimate is likely to be correct.
- The Delphi technique has eight basic steps:

+Step1: Identify the teams that need to perform the estimation activity.

→**Estimation experts:** usually consist of groups of five or six experienced project managers.

- The estimation values provided by the project mangers are based on past project history and their knowledge .
- However, only those project managers should be invited for estimation whose experience of a past project matches that of the current project.

→**Estimation coordinator:** An estimation coordinator is very similar to a moderator in a usual meeting. The coordinator facilitates the meeting and ensures that the goals of the meeting are fully achieved.

→**Author:** An author is similar to a recorder of minutes in a meeting.

+Step 2:

- The author present the project details including client's needs and system requirements to the group of experts
- The author also describes the expectation from the group.
- The author and experts jointly identify the tasks that need to be estimated.
- They also identify the valid assumptions that they need to consider while estimating.

+Step 3:

- The author and experts arrive at a consensus that any estimation with a specific variance value will not be accepted.
- For example, they may decide that any variance above 25% will not be accepted as an estimation value for computing the project effort or the productivity.

+<u>Step 4</u>**:**

-The coordinator prepares a list of tasks jointly decided by the team and distributes the list to all experts.

-These tasks comprise a project plan.

+<u>Step 5</u>:

-The experts independently make their estimates for each task.

-After recording their estimates, they hand over their estimates to the coordinator.

-This is a critical step

-While making estimates, no discussions or consultations are permitted because a mutual discussion may influence the estimation logic of the fellow experts.

-The coordinator and the author jointly ensure this.

+<u>Step 6</u>:

-The coordinator prepares a summary of estimates for each task.

-After calculating the percentage of variance, the coordinator marks each task as accepted or not accepted basing on the agreed accepted value.

<u>Summary of Estimates Table</u>

Task	Maximum Estimation (Hours)	Minimum Estimation (Hours)	Percentage of Variance	Accepted or not accepted *(A/NA)*
Cost And benefit analysis	20	15	25	A
High level design	50	30	40	NA

+<u>Step 7</u>:

-The coordinator hands over the summary to the group of experts and the author.

-The group of experts and the author discuss tasks and assumptions where the percentage of variance is more than the acceptable level.

-The maximum and minimum estimates of tasks are disclosed.

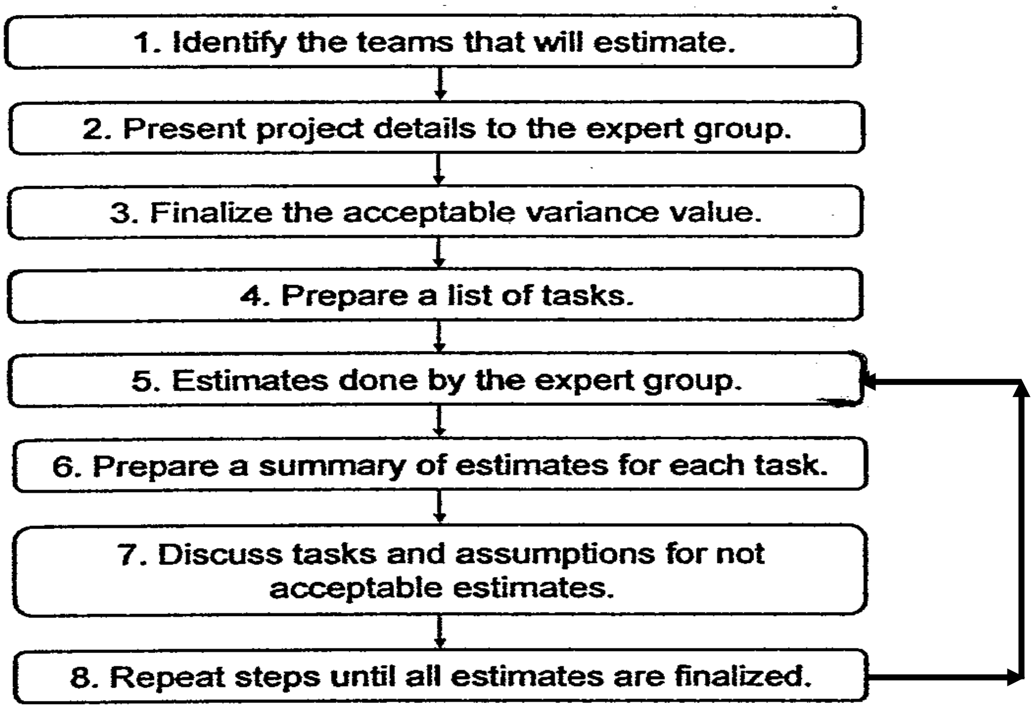
-To resolve the high percentage of the variance value, some tasks may be broken down further or combined.

+<u>Step 8</u>:

-Revert to step 5 and repeat the steps.

-You do this until all tasks are assigned estimates that have an acceptable percentage of variance value.

The Eight basic steps of Delphi Techniques:


1. Identify the teams that will estimate.
2. Present project details to the expert group.
3. Finalize the acceptable variance value.
4. Prepare a list of tasks.
5. Estimates done by the expert group.
6. Prepare a summary of estimates for each task.
7. Discuss tasks and assumptions for not acceptable estimates.
8. Repeat steps until all estimates are finalized.

-The Delphi technique is a simple and subjective method of estimation.
-However, it is a very effective method because most of the estimates are tried and tested.
-You can use this method if the project is small or if you have the data and expertise that can enable unambiguous estimates.