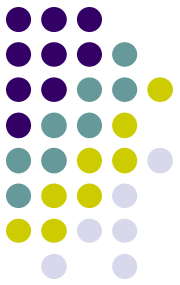


# Chapter 4

## Measuring Effort for Software Project



Referent textbooks:

**1. Software Engineering**

**A Practitioner's Approach**

***Roger S. Pressman***

**Fifth Edition, 2001**

**2. Basic of Software Project Management**

***NIIT***

**Eastern Economic Edition, 2004**

**3. Software Project Management**


**Bob Hughes and Mike Cotterell, 2001**

# Measuring Effort for an SW Project

---

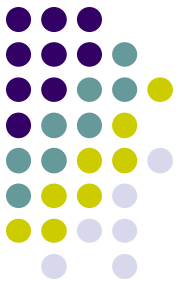


There are many techniques that can use to accurately estimate effort, such as:

- Source Lines of Code (SLOC)
- Function Point (FP)
- Constructive Cost Model (COCOMO)
- Delphi ()

# 4.1. SLOC Technique

---



- The SLOC technique is an objective method of estimating the size of the project.
- The project size helps determine the resources, effort, cost, and duration of 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, documentation of pages, inputs, outputs, and components of a SW program.

# \* Counting SLOC

---

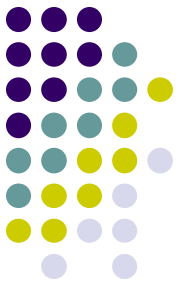
- The use of SLOC techniques requires that the technology or language remain unchanged throughout the project.
- Generally, you can use the SLOC technique when you are using third-generation language, such as FORTRAN or COBOL.

The general consideration include the following:

- Only the delivered lines of code are included in SLOC calculation.
- Only the source lines of code written only by the development team are counted. This excludes the code created by applications generators.
- Only declaration statements are counted as source lines of code. This excludes comments inserted to improve the readability of program.

# Example:

---



- Assume estimated lines of code of a system is:  
33,200 LOC
  - Average productivity for system of this type is:  
620 LOC/pm
  - Labor rate is: \$ 8,000 per month
- ⇒ Cost per line of code is :  $8,000/620 \approx \$13$
- ⇒ Total estimated project is :  $33,200 * 13 \approx \$431,000$
- ⇒ Estimated effort is :  $431,000/8,000 \approx 54$  person-months

# Example:

---



## \* សម្មតិកម្ម

- គេប្រាប់ចំនួន LOC សរុបនៃ system មួយ
- គេប្រាប់ផលិតភាពជាមធ្យមនៃ system មួយ
- គេប្រាប់អត្រាថ្លៃឈ្នួលពលកម្ម
- ចំនួនមនុស្សជាក់ស្តែងចូលរួមបង្កើត system

⇒  $\text{Cost per LOC} = \text{Labor rate} / \text{Productivity}$

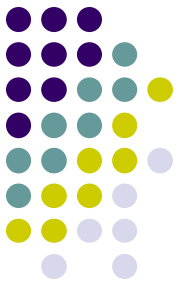
⇒  $\text{Total Cost of Project} = \text{Cost per LOC} * \text{Total number of LOCs}$

⇒  $\text{Estimated Effort} = \text{Total Cost of Project} / \text{Labor rate}$

⇒  $\text{Duration} = \text{Effort} / \text{People}$

# Disadvantages of Using SLOC

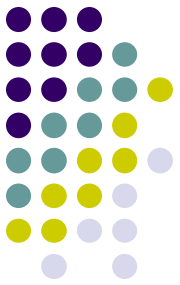
---



1. The number of lines of code delivered is dependent upon the skill level of the programmer. In fact, the higher skill level of the programmer the fewer lines of code they will develop to perform the same function.
2. Higher-level languages such as Delphi, Progress 4GL, Forte, Dynasty, VB, Java Script, or other visual languages require far fewer lines of code than Assembler, COBOL, or C to perform the same functionality. That is, there is an inverse relationship between level of language and work output (when work output is LOC).

# Disadvantages of Using SLOC

---

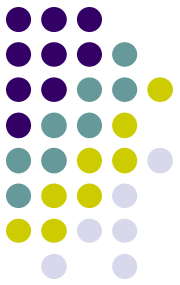


3. The actual number of LOC is not known until the project is almost completed. Therefore, LOC cannot be used to estimate the effort or schedule of a project. Function Points can be derived from requirements and analysis documents that are available early in a project life cycle.
4. There is no agreed upon method to count lines of code. The statement and type of statements used in Visual C++, Assembler, COBOL, SQL are completely different. It is common for applications to have a combination of different languages being utilized.



## 4.2. FP Technique

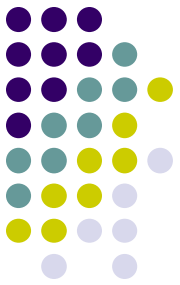
---



- The FP technique is a direct indicator of the functionality of a software application from the user's perspective.
- This is the most popular technique used to estimate the size of a software project.
- You use the FP technique to estimate the size of a project.
- The total size of a project is estimated as a single FP value.
- 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.

## 4.2. FP Technique

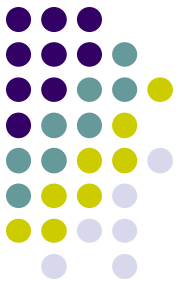
---



- This way, you can determine how much effort per FP is required in that particular phase.
- For example, the testing phase is planned for 20 FP of work.
- The project managers, based on their past project experience, determine the amount of effort in person/man months required in the testing 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 in terms of per FP for a phase.

## 4.2. FP Technique

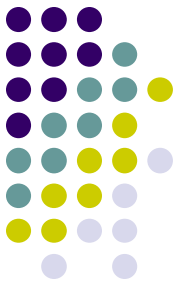
---



- FPs can be used to estimate LOC depending on the average number of LOC per FP for a given language
  - **LOC = AVC \* number of function points**
  - AVC is the average number of LOC/FP for a given language

# \* History

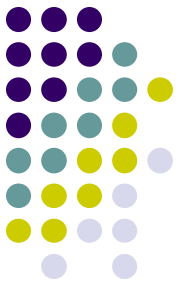
---



- 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 points theory.

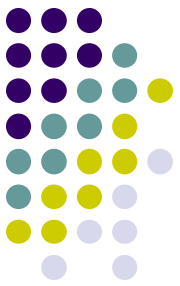
# \* Features of Function Points

---



- 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 major components are required.

# \* Features of Function Points (➡)



$$\text{FP} = \text{Count total} \times [0.65 + 0.01 \times \Sigma (F_i)]$$

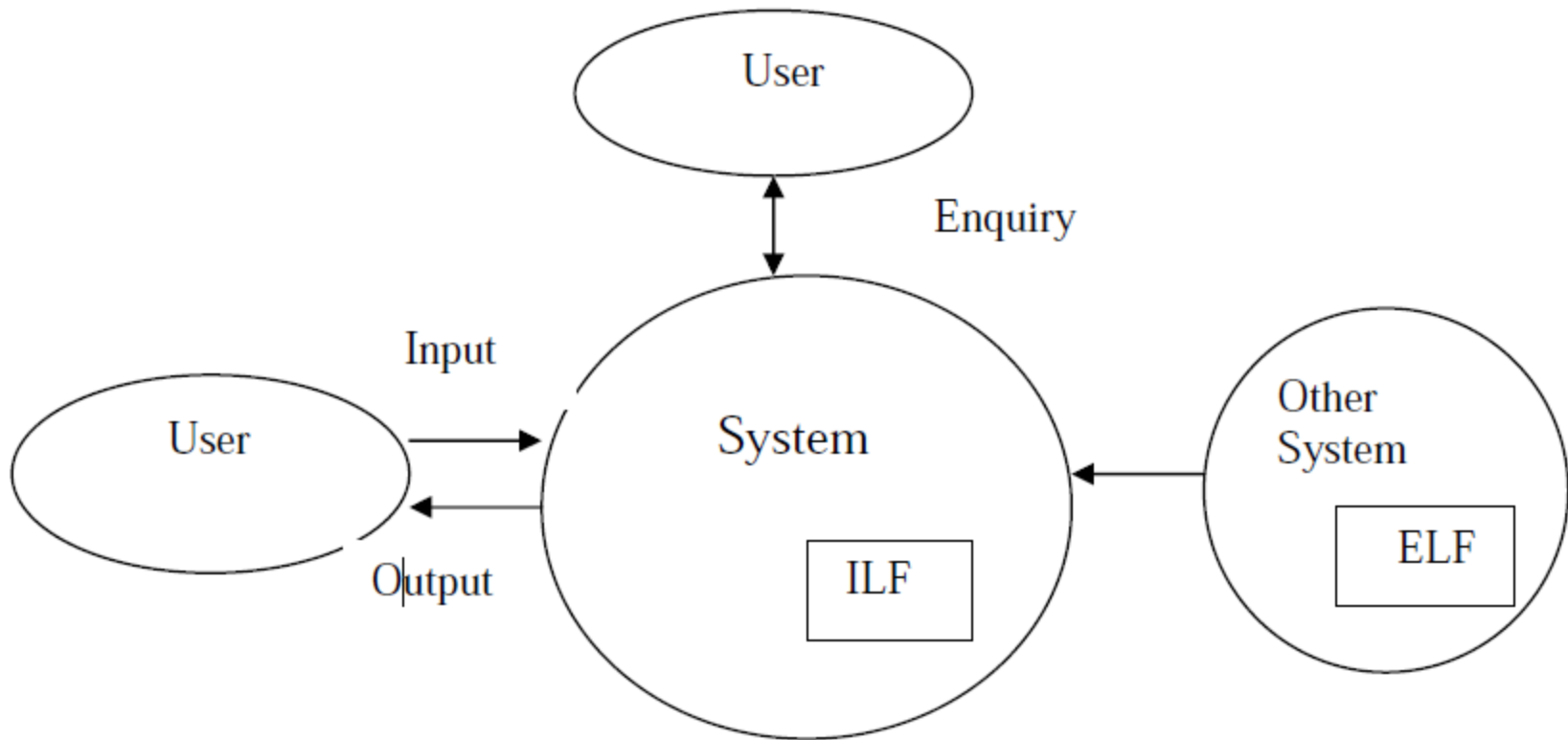
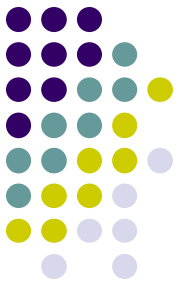
Where

+ Count total is the sum of all FP entries obtained from figure below

Measurement Parameter	Count		Simple	Average	Complex		Total
Number of user inputs	<input type="text"/>	X	3	4	6	=	<input type="text"/>
Number of user outputs	<input type="text"/>	X	4	5	7	=	<input type="text"/>
Number of user inquiries	<input type="text"/>	X	3	4	6	=	<input type="text"/>
Number of files	<input type="text"/>	X	7	10	15	=	<input type="text"/>
Number of external interfaces	<input type="text"/>	X	5	7	10	=	<input type="text"/>
Count Total	<input type="text"/>						<input type="text"/>

# \* Features of Function Points

---



## Functional units for Function Count

+  $F_i$  ( $i = 1$  to  $14$ ) are “complexity adjustment values” based on responses to the following questions:

## *General System Characteristics Table*

No.	Characteristics	Brief 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 Configuration	The degree to which the computer resources, where the application runs, are used
6	Transaction Rate	The frequency of transactions that are executed, on a daily, weekly, or monthly basis



7	On-line Data Entry	The percentage of data that is entered by using interactive transaction
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

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 the master file 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?

- Each of these questions is answered using a scale that ranges from 0 to 5 in the table below:

Degree of influence in value	Description of the degree of influence
0	No influence
1	Incidental influence
2	Moderate influence
3	Average influence
4	Significant influence
5	Very critical

# Ex1:

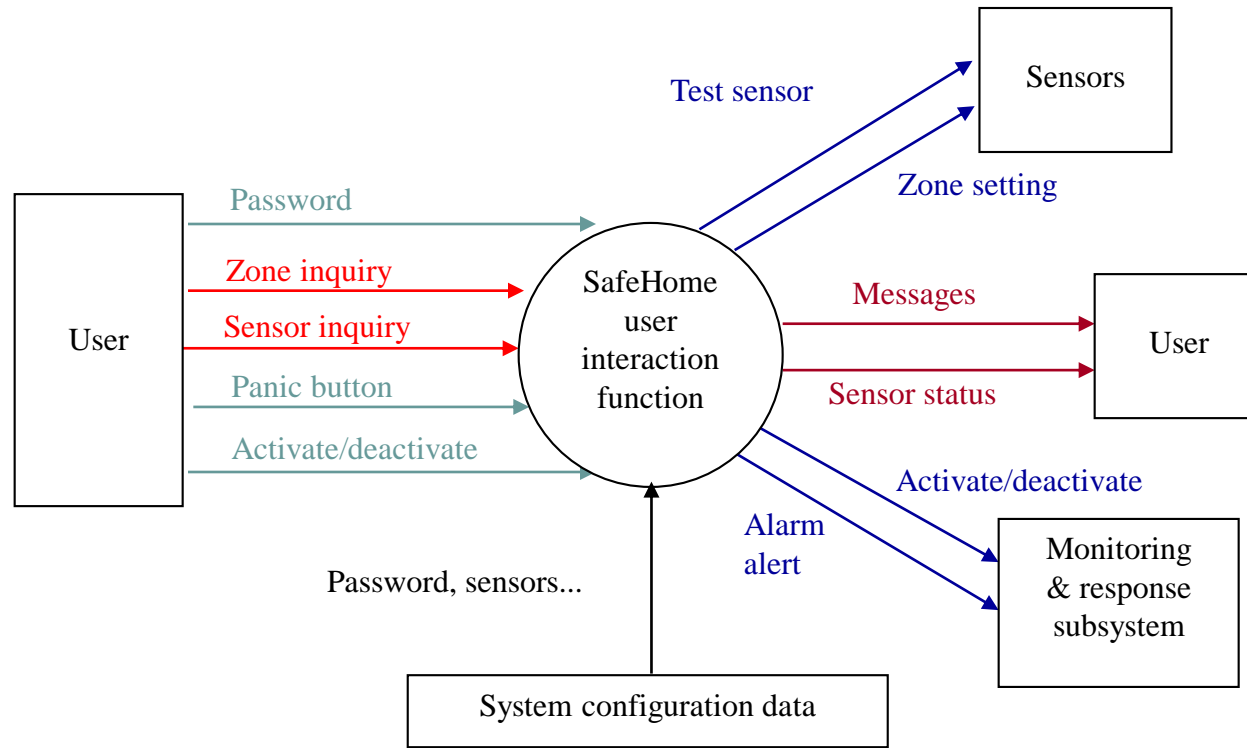
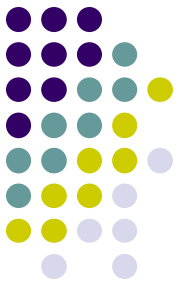
---

Measurement parameter	Count	Simple (Low)	Average (Medium)	Complex (High)	Total
User inputs	19	$3 * 5 = 15$	$4 * 8 = 32$	$6 * 6 = 36$	83
User outputs	15	$4 * 5 = 20$	$5 * 6 = 30$	$7 * 4 = 28$	78
User inquiries	5	$3 * 2 = 6$	$4 * 2 = 8$	$6 * 1 = 6$	20
Internal files	6	$7 * 1 = 7$	$10 * 2 = 20$	$15 * 3 = 45$	72
External interfaces	3	$5 * 0 = 0$	$7 * 2 = 14$	$10 * 1 = 10$	24
Count total					277

	<i>General System Characteristics</i>	<i>DI Value</i>
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 value	48

$$\Rightarrow FP = 277 * (0.65 + [0.01 * 48]) = 313$$

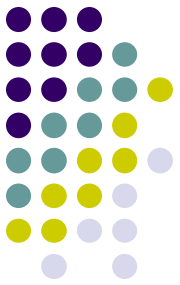
## Ex2: Analysis Model of SafeHome SW



- Number of user inputs = 3 (password, panic button, and activate/deactivate)
  - Number of user inquiries = 2 (zone inquiry and sensor inquiry)
  - Number of file = 1 (system configuration file)
  - Number of user outputs = 2 (messages and sensor status)
  - Number of external interfaces = 4 (test sensor, zone setting, activate/deactivate, and alarm alert)
- \* Assume Weighting Factor is simple:  $\Rightarrow \text{Count total} = (3 \times 3 + 4 \times 2 + 3 \times 2 + 7 \times 1 + 5 \times 4) = 50$
- \* And assume  $\sum F_i = 46$
- $\Rightarrow \text{FP} = 50 \times [0.65 + (0.01 \times 46)] = 55.50$

# \* Using FP for Initial Estimation

---

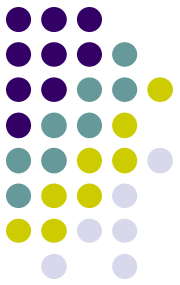


- To complete one FP of work, the project requires 10 hours
- Assume in a project, the total FP estimated is 200 FP
- Therefore, to complete a project of 200 FP, you require  $200 \text{ FP} * 10\text{hrs} = 2000\text{hrs}$
- In relation to 8 hours on a working day, the value of working hours is 250 person days
- Assuming 20 working days in a month, to complete a project of 200 FP, you require 12.5 person months
- If you deploy three persons on the project, they would require four months to complete the work



# \* FP-Based Estimation

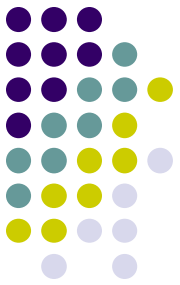
---



- Assume function point of a system is: 375
  - Average productivity for system of this type is: 6.5FP/pm
  - Labor rate is: \$ 8,000 per month
- ⇒ Cost per FP is :  $8,000/6.5 \approx \$1,230$
- ⇒ Total estimated project is :  $1,230 \times 375 \approx \$461,000$
- ⇒ Estimated effort is :  $461,000/8,000 \approx 58$  person-months

# \* FP-Based Estimation

---



## \* សម្មតិកម្ម

- គេប្រាប់ចំនួន FP សរុបនៃ system មួយ
- គេប្រាប់ផលិតភាពជាមធ្យមនៃ system មួយ
- គេប្រាប់អត្រាថ្លៃឈ្នួលពលកម្ម
- ចំនួនមនុស្សជាក់ស្តែងចូលរួមបង្កើត system
- គេប្រាប់ AVC (the average number of LOC/FP for a given language)

⇒  $\text{Cost per FP} = \text{Labor rate} / \text{Productivity}$

⇒  $\text{Total Cost of Project} = \text{Cost per FP} * \text{Total number of FPs}$

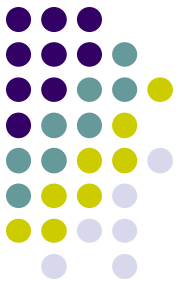
⇒  $\text{Estimated Effort} = \text{Total Cost of Project} / \text{Labor rate}$

⇒  $\text{Duration} = \text{Effort} / \text{People}$

⇒  $\text{LOC} = \text{AVC} * \text{Total number of FPs}$

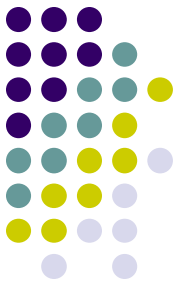
## \* Using FP for Post-Project analysis

---



- FP can use to express factors such as productivity, effort, defects, and cost used in a 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

# Ex: (←)



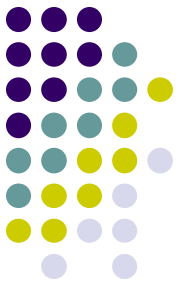
	<i>Total number of defects reported</i>	<i>Total project size in FP</i>
<b>Project 1</b>		
	10	150
Defect density	$10/150=0.067$ defects/FP	
<b>Project 2</b>		
	20	200
Defect density	$20/200=0.10$ defects/FP	
<b>Project 3</b>		
	40	1000
Defect density	$40/1000=0.04$ defects/FP	

\* 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. COCOMO Techniques

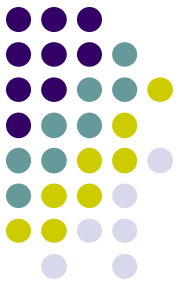
---



- COCOMO is one of the most widely used software estimation models in the world.
- It was developed by Barry Boehm in 1981
- COCOMO predicts the effort and schedule (duration) for a software product development based on inputs relating to the **size** of the software and a number of **cost drivers** that affect productivity.

## 4.3. COCOMO Techniques

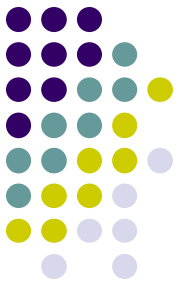
---



- COCOMO technique has three levels of implementation:
  - Basic
  - Intermediate
  - Advanced

# a) Basic COCOMO

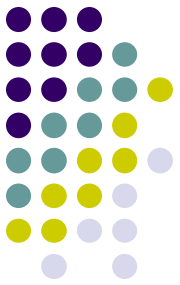
---



- The basic COCOMO technique 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 maintenance project.
- Estimating the effort in basic COCOMO technique involves three steps:
  1. Estimating the total delivered lines of code
  2. Determine the effort constants based on the type of the project
  3. Substituting values for lines of code and effort constants in a formula

# a) Basic COCOMO

---

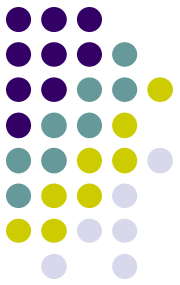


- The next step is to determine the type of the project being developed
- There are three types of projects to calculate effort(➡)
  - 1) Organic
    - Organic projects have sufficient and defined objectives
    - The organizations that undertake organic projects have ample experience in development and use small development teams
    - These are simple business and applications, such as a banking system and inventory system



# a) Basic COCOMO

---



## 1) Organic

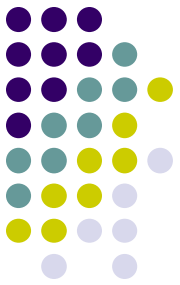
- Project Size: 2-50 KLOC
- Team Size: Small
- Nature of Project: Familiar
- Deadline: Not tight

## 2) Semidetached

- Semidetached projects are combination of the preceding two types of SW projects
- Examples: a new operating system, a database management system

# a) Basic COCOMO

---



## 2) Semidetached

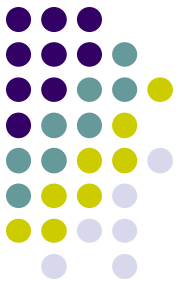
- Project Size: 50-300 KLOC
- Team Size: Medium
- Nature of Project: Medium
- Deadline: Medium

## 3) Embedded

- Embedded projects 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

# a) Basic COCOMO

---



## 3) Embedded

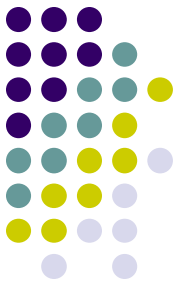
- Project Size: Over 300 KLOC
- Team Size: Large
- Nature of Project: Complex
- Deadline: Tight
- The last step is to substitute the value of lines of code and effort constants, development time constants in the following formula:

$$E_i = a1 * (KLOC)^{a2}$$

$$D = a3 * (E_i)^{a4} \quad P = E_i/D$$

Where  $E_i$  is the effort for a project,  $D$  is development time,  $P$  is the number of people required,  $a1$  and  $a2$  are the effort constants,  $a3$  and  $a4$  are development time constants depend on the type of project being developed

## a) Basic COCOMO (←) (→)



Project Type	a1	a2	a3	a4
Organic	2.4 (3.2)	1.05	2.5	0.38
Embedded	3.6 (2.8)	1.20	2.5	0.32
Semidetached	3.0	1.12	2.5	0.35

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

$$E_i = 2.4 * 4^{1.05} = 2.4 * 4.28 \approx 10 \text{ person months}$$

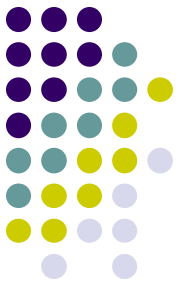
$$D = 2.5 * (10)^{0.38} = 2.5 * 2.4 = 6 \text{ months}$$

$$\Rightarrow P = E_i / D = 10 / 6 \approx 1.67 \text{ Person}$$

$$\Rightarrow \text{Productivity of Software} = \text{KLOC} / E_i = 4 / 10 = 0.4 \\ \text{KLOC/PM or 400 LOC/PM}$$

## b) Intermediate COCOMO

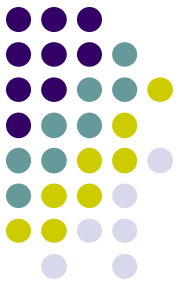
---



- 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:

## b) Intermediate COCOMO

---



1) Estimate the initial development effort by using SLOC in the following formula:

$$E_i = a1 * (KLOC)^{a2}$$

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

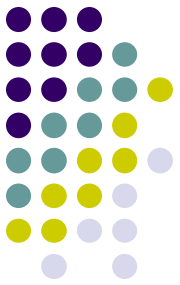
3) Finally, you calculate the actual effort by multiplying the weighted cost driver attributes with the initial effort estimate

- There are 15 commonly used project, personnel, and product-related cost driver attributes

	Cost Drivers	Rating					
		Negligible	Low	Average	High	Very high	Extremely critical
Product attributes	Required Software Reliability (RELY)	0.75	0.88	1.00	1.15	1.40	
	Database Size (DATA)		0.94	1.00	1.08	1.16	
	Software Product Complexity (CPLX)	0.70	0.85	1.00	1.15	1.30	1.65
Computer attributes	Execution Time Constraint (TIME)			1.00	1.11	1.30	1.66
	Main Storage Constraint (STOR)			1.00	1.06	1.21	1.56
	Virtual Machine Volatility (VIRT)		0.87	1.00	1.15	1.30	
	Computer Turnaround Time (TURN)		0.87	1.00	1.07	1.15	
Personnel attributes	Analyst Capability (ACAP)	1.46	1.19	1.00	0.86	0.71	
	Applications Experience (APEX)	1.29	1.13	1.00	0.91	0.82	
	Programmer Capability (PCAP)	1.42	1.17	1.00	0.86	0.70	
	Virtual Machine Experience (VEXP)	1.21	1.10	1.00	0.90		
	Programming Language Experience (LEXP)	1.14	1.07	1.00	0.95		
Project attributes	Modern Programming Practices (MODP)	1.24	1.10	1.00	0.91	0.83	
	Use of Software Tools (TOOL)	1.24	1.10	1.00	0.91	0.82	
	Required Development Schedule (SCED)	1.23	1.08	1.00	1.04	1.10	

## b) Intermediate COCOMO

---



- Typically, the values that rate each cost driver attribute range from 0.7 through 1.66
- Usually, in organizations, the average rating is assigned a static value of 1.0
- The intermediate COCOMO technique formula is:

$$E = EAF * E_i$$

$$D = a3 * (E)^{a4}$$



# Ex:

- In a customized insurance project, there are four modules which the total effort estimate of the modules is 3.0 KLOC
- There are four cost driver attributes with the respective multiplying factors that might affect the project most
- a1 and a2 are 3.2 and 1.05 (organic project)

$$\Rightarrow E_i = a1 * (KLOC)^{a2} = 3.2 * 3^{1.05} = 3.2 * 3.16 = 10.11 \text{ PM} (\leftarrow)$$

Applicable cost driver attributes	Rating	Multiplying factors
CPLX	High	1.15 <del>(1.2)</del>
TIME	Very high	1.3 <del>(1.35)</del>
ACAP	Low	1.19 <del>(0.95)</del>
MODP	Average	1.0

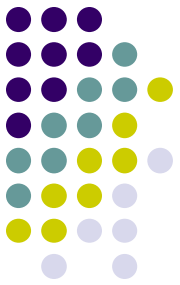
$$EAF = 1.15 * 1.3 * 1.19 * 1.0 = 1.78 \text{ ~~(1.53)~~}$$

$$\Rightarrow E = EAF * E_i = 1.78 * 10.11 \approx 18 \text{ person months}$$

$$\Rightarrow D = a3 * (E)^{a4} = 2.5 * (18)^{0.38} = 2.5 * 3 \approx 7.5 \text{ months}$$

## c) Advanced COCOMO

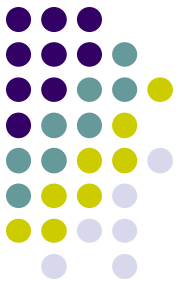
---



- 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

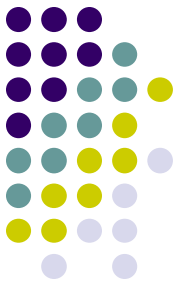
---



- 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 weightage to the 14 GSCs (General System Characteristics) or complexity adjustment values and the various information domain value elements

## 4.4. Delphi Technique

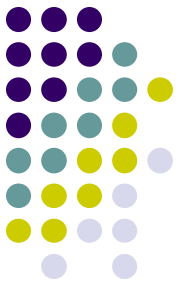
---



- 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:

# Step 1

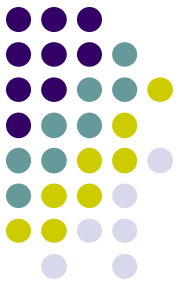
---



- Identify the teams that need to perform the estimation activity.
  - Estimation experts: They usually consist of groups of five or six experienced project managers.
    - The estimation values provided by the project managers 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

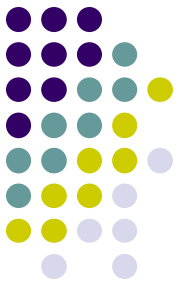
---



- 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

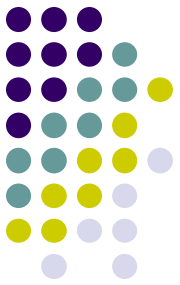
---



- 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.

# Step 4

---

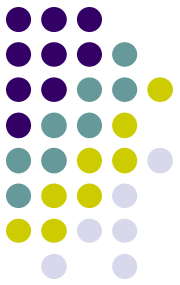


- The coordinator prepares a list of tasks jointly decided by the team and distributes the list to all experts.
- These tasks comprise a project plan.



# Step 5

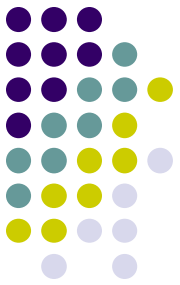
---



- 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.

# Step 6

---



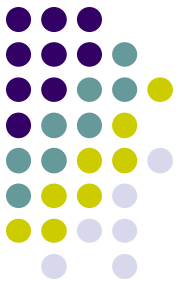
- The coordinator prepares a summary of estimates for each task.
- After calculating the percentage of variance, the coordinator marks each task as accepted or not accepted based on the agreed accepted value.

## Summary of Estimates Table

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

# Step 7

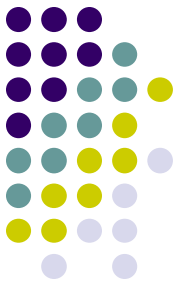
---



- 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 not disclosed or discussed.
- To resolve the high percentage of the variance value, some tasks may be broken down further or combined.

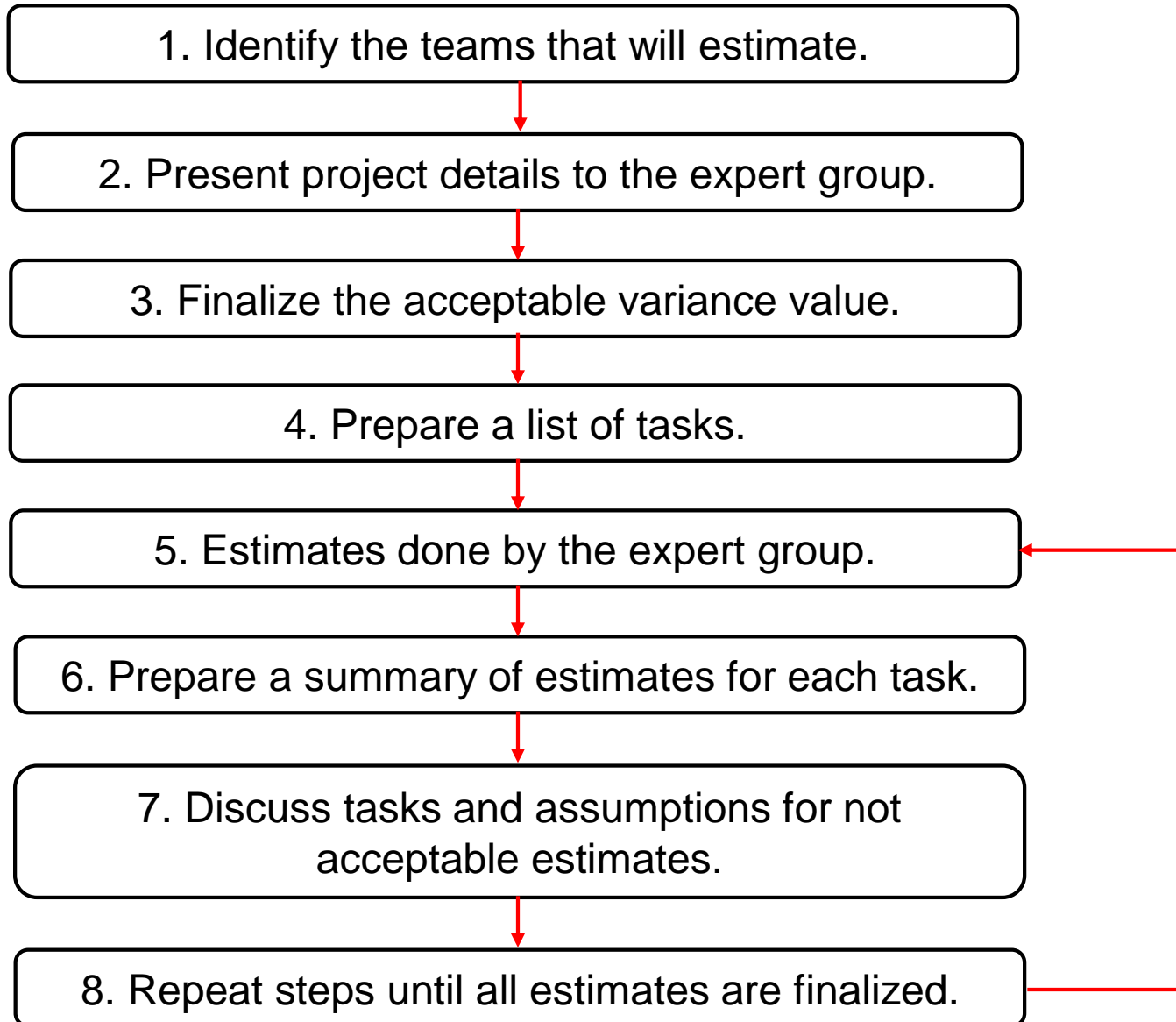
# Step 8

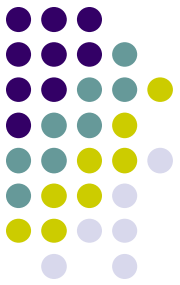
---



- 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.

## Delphi Technique has eight basic steps:





- 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.