Software Engineering
CA210

Unit 3:
Project Estimation
&
Scheduling

➢ Project management

- Software project management includes the tools, techniques, and knowledge essential to deal with the growth of software products.
- In Software Project Management, the end users and developers require to know the cost of the project, duration and length.
- It is a process of managing, allocating and timing resources to develop computer software that meets necessities.

It consists of eight tasks:

1. **Problem Identification**
2. **Problem Definition**
3. **Project Planning**
4. **Project Organization**
5. **Resource Allocation**
6. **Project Scheduling**
7. **Tracking, Reporting and Controlling**
8. **Project Termination**

What is Project?

A project is a sequence of unique, complex, and connected activities having one goal or purpose and that must be completed by a specific time, within budget, and according to specification. This definition tells us quite a bit about a project

Project Management:

- The methods and regulation used to define goals, plan and monitor tasks and resources, identify and resolve issues, and control costs and budgets for a specific project is known as project management.
- Project management is "the application of knowledge, skills, tools and techniques to project activities to meet the project requirements."
- A project should be initiated with a **feasibility study**, where a clear definition of the goals and ultimate benefits need to be determined.

➢ The Management Spectrum:

- The management spectrum describes the management of a software project or how to make a project successful.
- The Management Spectrum provides a holistic approach to managing software projects by addressing key areas: **People, Product, Process, and Project.**
- Effective management across these dimensions helps in reducing risks, improving productivity, and ensuring that the software meets its intended goals.
- Effective software project management focuses on the four P's:
- **people, product, process, and project**. The order is not arbitrary.

❑**The People :**

- This involves defining the roles and responsibilities of every team member.
- Key roles might include **project managers, software engineers, testers, and stakeholders.**
- Ensuring that team members have the necessary skills and knowledge to perform their tasks effectively.
- The people capability maturity model (CMM) defines the following key practice areas for software people: staffing, communication and coordination, work environment, performance management, training, compensation, competency analysis and development, career development, workgroup development, team/culture development, and others.

❑Product

- It defines Project Scope: Clearly defining the scope of the software product, including functional and non-functional requirements.
- Establishing the goals and objectives that the product must meet, often tied to customer needs or business goals.
- Before a project can be planned, product objectives and scope should be established, alternative solutions should be considered and technical and management constraints should be identified
- Without this information, it is impossible to define reasonable(and accurate) estimates of the cost, an effective assessment of risk, a realistic breakdown of project tasks, or a manageable project schedule that provides a meaningful indication of progress.

❑The process

- Software process provides the framework from which a comprehensive plan for software development can be established.
- A small number of framework activities are applicable to all software projects, regardless of their size or complexity.
- A number of different tasks set—tasks, milestones, work products and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.
- Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement—overlay the process model .
- Umbrella activities are independent of any one framework activity and occur throughout the process
- Once the process model is chosen, populate it with the minimum set of work tasks and work products that will result in a high-quality product—avoid process overkill!

❑The Project

- Developing a detailed project plan that outlines tasks, timelines, resources, and budgets.
- The Project We conduct planned and controlled software projects for one primary reason—it is the only known way to manage complexity.
- To avoid project failure, a software project manager and the software engineers who build the product must avoid a set of common warning signs, understand the critical success factors that lead to good project management, and develop a commonsense approach for planning, monitoring, and controlling the project.
- Identifying potential risks that could impact the project and developing strategies to mitigate these risks.

➢ The WWWWWHH (W5HH) Principle

- **What questions need to be answered in order to develop a project plan? For this Bohem suggests W5HH Principle**
- In an excellent paper on software process and projects, Barry Boehm states: "you need an organizing principle that scales down to provide simple [project] plans for simple projects."
- Boehm suggests an approach that addresses project objectives, milestones and schedules, responsibilities, management and technical approaches, and required resources.
- **He calls it the WWWWWHH principle, after a series of questions that lead to a definition of key project characteristics and the resultant project plan:**

Why is the system being developed?
The answer to this question enables all parties to assess the validity of business reasons for the software work. Stated in another way, does the business purpose justify the expenditure of people, time, and money?
What will be done, by when? The answers to these questions help the team to establish a project schedule by identifying key project tasks and the milestones that are required by the customer.
Who is responsible for a function? Earlier in this chapter, we noted that the role and responsibility of each member of the software team must be defined. The answer to this question helps accomplish this.
Where are they organizationally located? Not all roles and responsibilities reside within the software team itself. The customer, users, and other stakeholders also have responsibilities.

How will the job be done technically and managerially? Once product scope is established, a management and technical strategy for the project must be defined.

How much of each resource is needed? The answer to this question is derived by developing based on estimates. This question involves estimating the costs associated with the project, including development, resources, tools, and other expenses.

Boehm's W5HH principle is applicable regardless of the size or complexity of a software project. The questions noted provide an excellent planning outline for the project manager and the software team.

➢ Introduction of Estimation

- Estimation is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable.
- Software Project Estimation is the process of estimating various resources required for the completion of a project.
- Estimation determines how much money, effort, resources, and time it will take to build a specific system or product.
- Estimation is based on −

    - Past Data/Past Experience
    - Available Documents/Knowledge
    - Assumptions
    - Identified Risk

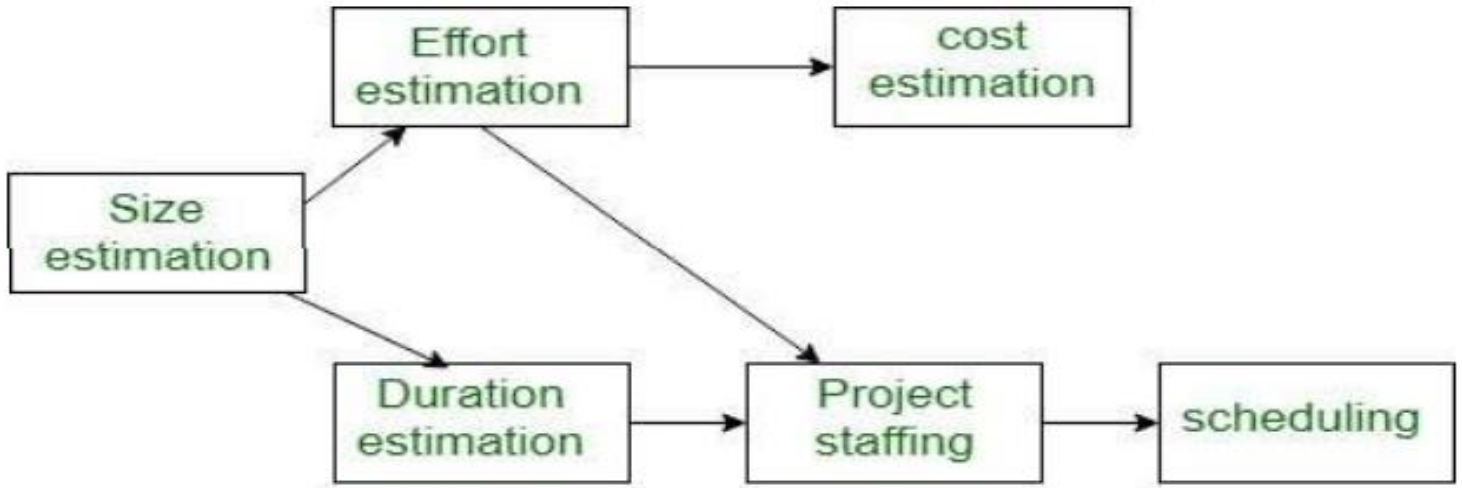
The four basic steps in software project estimation are:

1) **Estimate the size of the development product**. The size can be estimated by using either Lines of Code (LOC) or Function Points (FP).

2) **Estimate the effort** in person-months or person-hours (man-month or man-hour). Man-month is an estimate of personal resources required for the project.

3) **Estimate the schedule** in calendar days /months/ years based on total man-month required and manpower allocated to the project.

4) **Estimate the project cost** in local currency.



Precedence ordering among planning activities

Project Estimation Process

➢Software Project Estimation

- Estimation begins with a description of the scope of the problem.
- The problem is then decomposed into a set of smaller problems, and each of these is estimated using historical data and experience as guides.
- Problem complexity and risk are considered before a final estimate is made.
- Estimation of resources, cost, and schedule for a software engineering effort requires experience, access to good historical information (metrics), and the courage to commit to quantitative predictions when qualitative information is all that exists.
- Estimation carries inherent risk, and this risk leads to uncertainty

To achieve reliable cost and effort estimates, a number of options arise:

1. Delay estimation until late in the project (obviously, we can achieve 100 percent accurate estimates after the project is complete!).
2. Base estimates on similar projects that have already been completed.
3. Use relatively simple decomposition techniques to generate project cost and effort estimates.
4. Use one or more empirical models for software cost and effort estimation

DECOMPOSITION TECHNIQUES

- The decomposition approach from two different points of view: **decomposition of the problem** and **decomposition of the process.**
- Estimation uses one or both forms of partitioning.
- But before an estimate can be made, the project planner must understand the scope of the software to be built and generate an estimate of its "size."
- **It has following concepts:**
    1 Software Sizing
    2 Problem-Based Estimation
        An Example of LOC-Based Estimation
        An Example of FP-Based Estimation
    3 Process-Based Estimation

1 Software Sizing

- The “size” of software to be built can be estimated using a direct measure, LOC, or an indirect measure, FP.
- The accuracy of a software project estimate is predicated on a number of things:
  (1) the degree to which you have properly estimated the size of the product to be built;
  (2) the ability to translate the size estimate into human effort, calendar time, a function of the availability of reliable software metrics from past projects
  (3) the degree to which the project plan reflects the abilities of the software team
  (4) the stability of product requirements and the environment that supports the software engineering effort

2. Problem-Based Estimation

Lines of code and function points were described as measures from which productivity metrics can be computed. LOC and FP data are used in two ways during software project estimation:

(1) as estimation variables to “size” each element of the software and

(2) as baseline metrics collected from past projects and used in conjunction with estimation variables to develop cost and effort projections. LOC and FP estimation are distinct estimation techniques.

An Example of LOC-Based Estimation

As an example of LOC and FP problem-based estimation techniques, I consider a software package to be developed for a computer-aided design application for mechanical components.

The software is to execute on an engineering workstation
and must interface with various computer graphics peripherals including a mouse, digitizer, high-resolution color display, and laser printer

Decomposition for FP-based estimation focuses on information domain values rather than software functions.

Process-Based Estimation

- The most common technique for estimating a project is to base the estimate on the process that will be used.
- That is, the process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated.

The COCOMO Model

- Barry Boehm introduced a hierarchy of software estimation models bearing the name COCOMO, for **COnstructive COst MOdel**.
- The original COCOMO model became one of the most widely used and discussed software cost estimation models in the industry.
- Like all estimation models for software, the COCOMO models require sizing information.
- Three different sizing options are available as part of the model hierarchy: **object points, function points, and lines of source code.**

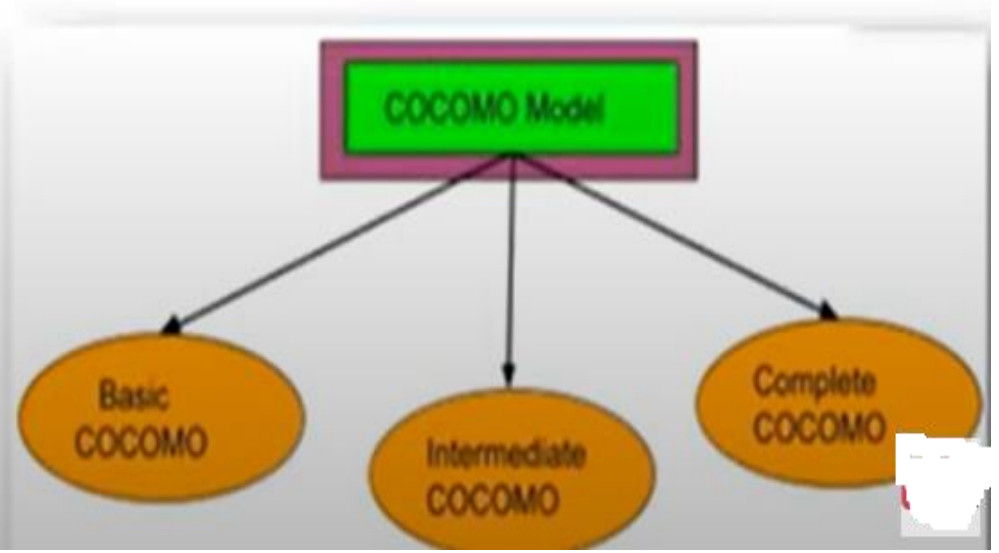
It is actually a hierarchy of estimation models that address the following areas:

- **Application composition model**. Used during the early stages of software engineering, when prototyping of user interfaces, consideration of software and system interaction, assessment of performance, and evaluation of technology maturity are paramount.
- **Early design stage model**. Used once requirements have been stabilized and basic software architecture has been established.
- **Post-architecture-stage model**. Used during the construction of the software.

Function points measure the size of an application system based on the functional view of the system.

the object point is an indirect software measure that is computed using counts of the number of (1) screens (at the user interface), (2) reports, and (3) components likely to be required to build the application.

- It was developed by a scientist Barry Boehm in 1981.
- The COCOMO **(Constructive Cost Model)** is one of the most popularly used software cost estimation models.
- This model depends on the size means number of lines of code for software product development.
- It estimates Effort required for project, Total project cost & Scheduled time of project.



In COCOMO, projects are categorized into three types:

1. Organic Type:

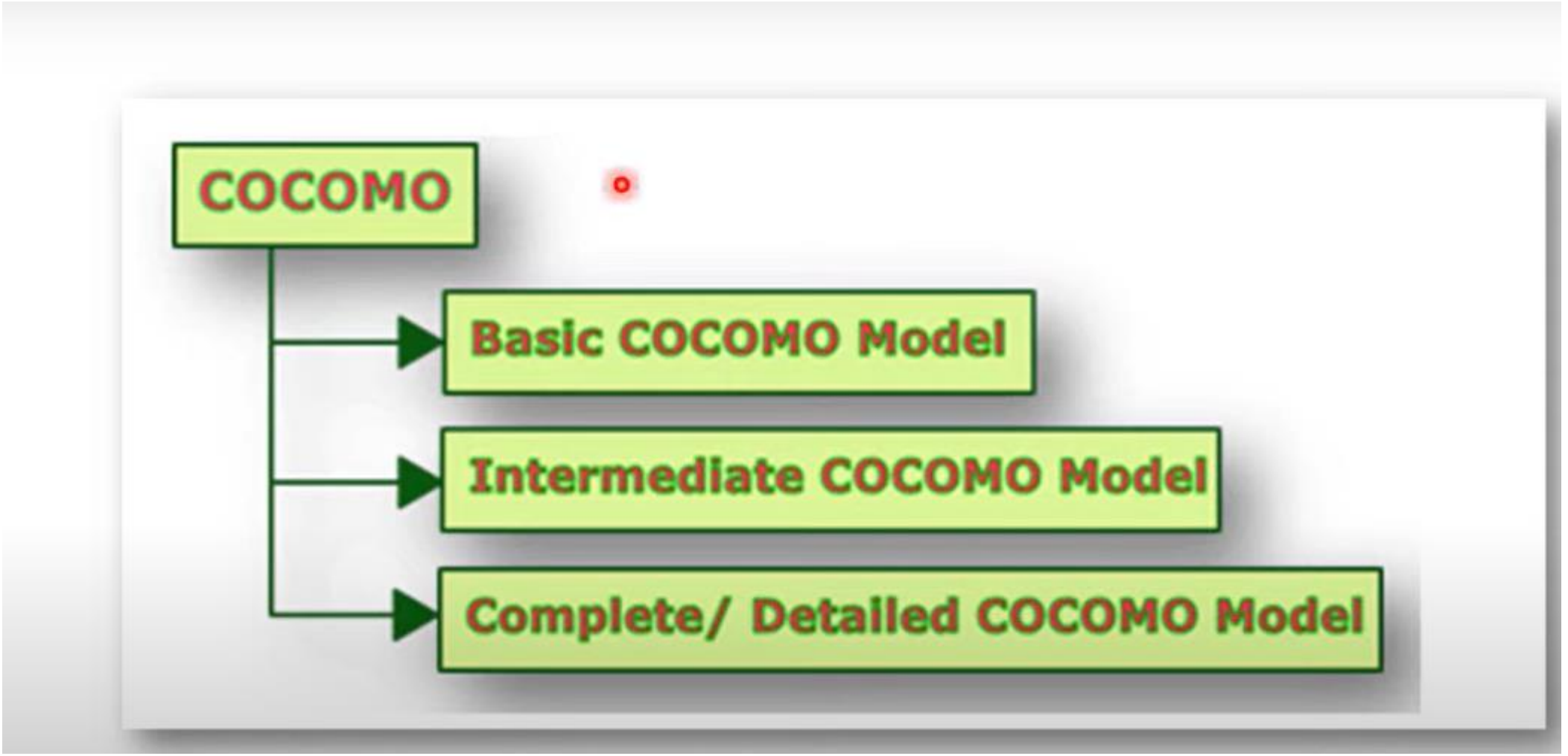
- Project is small and simple. (2-50 KLOC)
- Few Requirements of projects.
- Project team is small with prior experience.
- The problem is well understood and has been solved in the past.
- **Examples:** Simple Inventory Management Systems & Data Processing Systems.

2. Semidetached Type:

- Medium size and has mixed rigid requirements. (50-300 KLOC)
- Project has Complexity not too high & low.
- Both experienced and inexperienced members in Project Team.
- Project are few known and few unknown modules.
- **Examples:** Database Management System, Difficult inventory management system.

3. Embedded Type:

- Large project with fixed requirements of resources. (More then 300 KLOC)
- Larger team size with little previous experience.
- Highest level of complexity, creativity and experience requirement.
- **Examples:** ATM, Air Traffic control, Banking software.


COCOMO
Basic COCOMO Model
Intermediate COCOMO Model
Complete/ Detailed COCOMO Model

Basic COCOMO

- It is type of static model to estimates software development effort quickly and roughly.
- It mainly deals with the number of lines of code in project.

Formula:

Effort (E) = a*(KLOC)b MM

Scheduled Time (D) = c*(E)d Months(M)

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Where,

- **E** = Total effort required for the project in Man-Months (MM).
- **D** = Total time required for project development in Months (M).
- **KLOC** = The size of the code for the project in Kilo lines of code.
- **a, b, c, d** = The constant parameters for a software project.

Example of Basic COCOMO

❑Problem Statement:

➢Consider a software project using semi-detached mode with 300 Kloc.

Find out Effort estimation, Development time and Person estimation.

❑Solution:

Effort (E) = **a*(KLOC)b** = 3.0*(300)1.12 = 1784.42 PM

Development Time (D) = **c(E)d** = 2.5(1784.42)0.35 = 34.35 Months(M)

Person Required (P) = **E/D** = 1784.42/34.35 = 51.9481 Persons ~52 Persons

$$E = a \times (\mathrm{KLOC})^b$$

Where:

- $a = 3.0$
- $b = 1.12$
- KLOC is the number of thousands of lines of code (300 KLOC

2. **Development Time (Tdev):**

$$Tdev = c \times (E)^d$$

Where:

- $c = 2.5$
- $d = 0.35$
- E is the effort calculated above.

3. **Number of Persons (P):**

$$P = \frac{E}{Tdev}$$

Where:

- E is the effort in person-months.
- $Tdev$ is the development time in months.

1. KLOC (thousands of lines of code):

$$KLOC = 300$$

2. Effort (E):

$$E = 3.0 \times (300)^{1.12}$$

3. Development Time (Tdev):

$$Tdev = 2.5 \times (E)^{0.35}$$

4. Number of Persons (P):

$$P = \frac{E}{Tdev}$$

For a software project with 300 KLOC in the semi-detached mode,

- **Effort (E)**: 1784.42 person-months
- **Development Time** : 34.35 months
- **Number of Persons (P)**: 51.94 persons = 52

Parameters for Basic COCOMO

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Intermediate COCOMO

- Extension of Basic COCOMO model which enhance more accuracy to cost estimation model result.
- It include cost drivers (Product, Hardware, Resource & project Parameter) of project.

Formula:

Effort (E) = $a*(KLOC)^b * EAF$ MM

Scheduled Time (D) = $c*(E)^d$ Months(M)

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Where,

- **E** = Total effort required for the project in Man-Months (MM).
- **D** = Total time required for project development in Months (M).
- **KLOC** = The size of the code for the project in Kilo lines of code.
- **a, b, c, d** = The constant parameters for the software project.

- **EAF:** Effort Adjustment Factor, which is calculated by multiplying the parameter values of different cost driver parameters. For ideal, the value is 1.

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Product Parameter					
Required Software	0.75	0.88	1	1.15	1.4
Size of Project Database	NA	0.94		1.08	1.16
Complexity of The Project	0.7	0.85		1.15	1.3

Personnel Parameter					
Analysis Capability	146	1.19	1	0.86	0.71
Application Experience	129	1.13		0.91	0.82
Software Engineer Capability	142	1.17		0.86	0.7
Virtual Machine Experience	121	1.1		0.9	NA
Programming Experience	114	1.07		0.95	NA

Hardware Parameter					
Performance Restriction	NA	NA	1	1.11	1.3
Memory Restriction	NA	NA		1.06	1.21
virtual Machine Environment	NA	0.87		1.15	1.3
Required Turnabout	NA	0.94		1.07	1.15

Project Parameter					
Software Engineering Methods	1.24	1.1	1	0.91	0.82
Use of Software Tools	1.24	1.1		0.91	0.83
Development Time	1.23	1.08		1.04	1.1

❑Problem Statement:

- For a given Semidetached project was estimated with a size of 300 KLOC. Calculate the Effort, Scheduled time for development by considering developer having high application experience and very low experience in programming.

❑Solution:

EAF = 0.82 *1.14 = 0.9348

Effort (E) = $a*(KLOC)^b * EAF = 3.0*(300)^{1.12} *0.9348 = 1668.07$ MM

Scheduled Time (D) = $c*(E)^d = 2.5*(1668.07)^{0.35} = 33.55$ Months(M)

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Personnel Parameter					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17		0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming	1.14	1.07		0.95	

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Personnel Parameter					
Analysis Capability	1.46	1.19	1	0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17		0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming Experience	1.14	1.07		0.95	

Type 3: Detailed / Complete COCOMO Model

- The model incorporates all qualities of both Basic COCOMO and Intermediate COCOMO strategies on each software engineering process.
- The whole software is divided into different modules and then apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test
5. Integration and test
6. Cost Constructive model

❑Problem Statement:

A distributed Management Information System (MIS) product for an organization having offices at several places across the country can have the following sub-components:

- Database part
- Graphical User Interface (GUI) part
- Communication part

❑Solution:

✓The communication part can be considered as Embedded software.

✓The database part could be Semi-detached software,

✓The GUI part Organic software.

The costs for these three components can be estimated separately and summed up to give the overall cost of the system.

Advantages

1. Provides a systematic way to estimate the cost and effort of a software project.
2. Estimate cost and effort of software project at different stages of the development process.
3. Helps in identifying the factors that have the greatest impact on the cost and effort of a software project.
4. Provide ideas about historical projects.
5. Easy to implement with various factors.

Disadvantages

1. It ignores requirements, customer skills and hardware issues.
2. It limits the accuracy of the software costs.
3. It is based on assumptions and averages.
4. It mostly depends on time factors.
5. 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.

1. Explain COCOCMO Model for project estimation with suitable example. 8M
2. Explain COCOCMO Model with levels with examples. 8M