Unit-III
Estimation and Scheduling

Presented by,
Prof. Barkha M Shahaji
Department of Computer Engineering,
Trinity College of Engineering and research, Pune-48.

Syllabus

Unit III	Estimation and Scheduling	(07 Hours)
Estimation for Software Projects: The Project Planning Process, Defining Software Scope and Checking Feasibility, Resources management, Reusable Software Resources, Environmental Resources, Software Project Estimation, Decomposition Techniques, Software Sizing, Problem-Based Estimation, LOC-Based Estimation, FP-Based Estimation, Object Point (OP)-based estimation, Process-Based Estimation, Process-Based Estimation, Estimation with Use Cases, Use-Case–Based Estimation, Reconciling Estimates, Empirical Estimation Models, The Structure of Estimation Models, The COCOMO II Mode, Preparing Requirement Traceability Matrix **Project Scheduling:** Project Scheduling, Defining a Task for the Software Project, Scheduling. **Suggested Free Open Source Tools:** Gantt Project, Agantty, Project Libre.		
#Exemplar/Case Studies	Write SRS in IEEE format for selected Project Statement/ case study, Study SRS of Online Voting system, Library management System (http://dos.iitm.ac.in/OOSD_Material/CaseStudies/CaseStudy2/eVote-srs.pdf),	
*Mapping of Course Outcomes for Unit III	CO1, CO3, CO7	

Software Project

Definition of Software Project:

- A Software Project is the complete procedure of software development from requirement gathering to testing maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.

Purpose of Software Project:

- It is very important for businesses as it helps them distinguish from competitors and become more competitive.
- Software development can improve the client's experiences, bring more feature-rich and innovative products to market, and make setups more safe, productive, and efficient.

Cont'd..

Examples of Software Project:

- Fingerprint-based ATM system
- Fingerprint voting system
- Weather forecasting system
- Android local train ticketing system

Project Planning

- Software project planning is task, which is performed before the production of software actually starts.
- Software Project Planning objective is to provide a framework that enables the managers to make responsible estimates of resources cost and schedule.
- Estimates should attempt to define best case and worst case scenarios so that project outcomes can be bounded

Cont’d..

Definition of Project planning :

- Project planning is an organized and integrated management process, which focuses on activities required for successful completion of the project.

- It prevents obstacles that arise in the project such as changes in projects or organization's objectives, non-availability of resources, and so on.

Cont'd..

Objectives of project planning :

- It defines the roles and responsibilities of the project management team members.
- It ensures that the project management team works according to the business objectives.
- It checks feasibility of the schedule and user requirements.
- It determines project constraints

Principles of Project Planning

Objectives of Effective project planning :

- Planning is necessary.
- Risk analysis.
- Tracking of project plan.
- Meet quality standards and produce quality deliverables.
- Description of flexibility to accommodate changes.

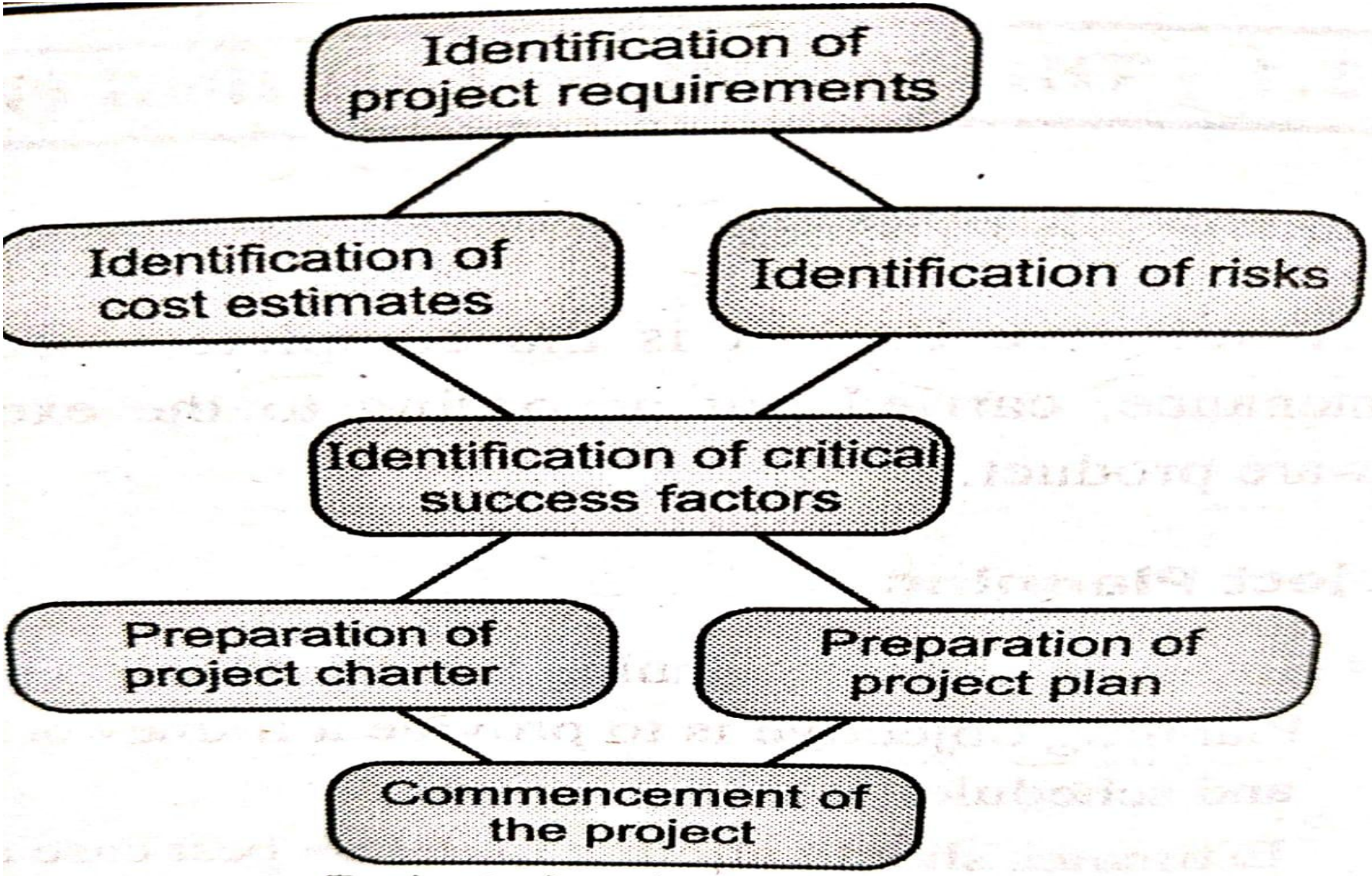
Project Planning Process

The project planning process involves a set of interrelated activities followed in an orderly manner to implement user requirements in software and includes the description of a series of project planning activities and individual(s) responsible for performing these activities.

The project planning process consists of the following things.

1. Objectives and scope of the project .
2. Techniques used to perform project planning
3. Effort (in time) of individuals involved in project
4. Project schedule and milestones
5. Resources required for the project
6. Risks associated with the project.

Project Planning Activities



Project planning activities

Software Scope

Once scope has been identified check whether build software meet this scope is the project feasible.

Software scope described by the following things:

1.The functions and features that are to be delivered to end users
2.Contents in the form of input and output from the system.
3.The contents that is presented to user as a consequences software

Scope is defined using two techniques:
1. The narrative description for the software obtained after Communication with all stakeholders.
2. A set of use cases have developed by end users.

Cont'd..

- After the scope has been identified two questions are asked;

1. Can we build software to meet this scope?
2. Is the project feasible?

- Generally the software scope describe the following:

1. Performance
2. Function
3. The data & control used to define the constraints
4. Reliability
5. Interfaces

Software feasibility

Feasibility study is carried out based on many purposes to analyze whether software product will be right terms of development, implementation, and contribution of project to the organization.

Feasibility is defined as the practical extent to which a project can be performed successfully.

Objectives of feasibility study are listed below:

- To analyze whether the software will meet organizational requirements.

- To determine whether the software can be implemented using the current technology and within the specified budget and schedule.

Cont'd..

- Types of Feasibility:

1. Technical Feasibility
2. Operational Feasibility
3. Economic Feasibility

RESOURCE MANAGEMENT

- All elements used to develop a software product may be assumed as resource for that project.

- Consider the use the following resources in software development:
  1. Time
  2. Human Resources
  3. Computer Resources
  4. Money

Cont'd..

Resource Management Components:

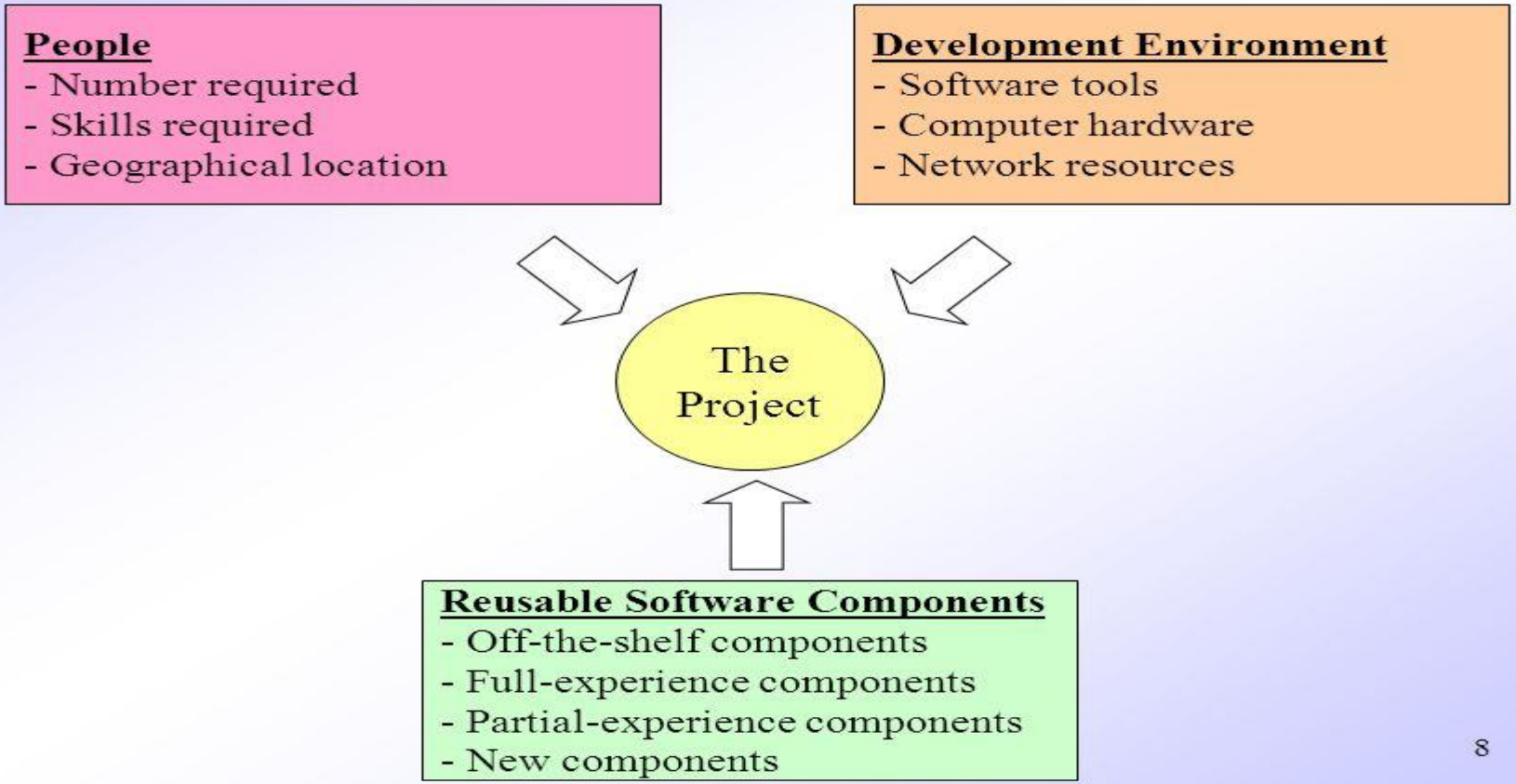
- Resource Monitoring
- Resource Allocation
- Resource Discovery

There are three major software resources categories

1. Human Resources
2. Reusable Software Components
3. Environment Resources

Project Resources

Categories of Resources



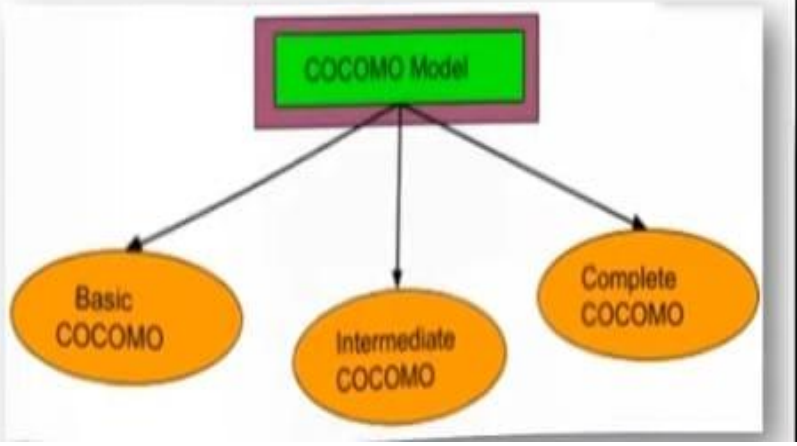
Cont'd..

There are four characteristics of resources

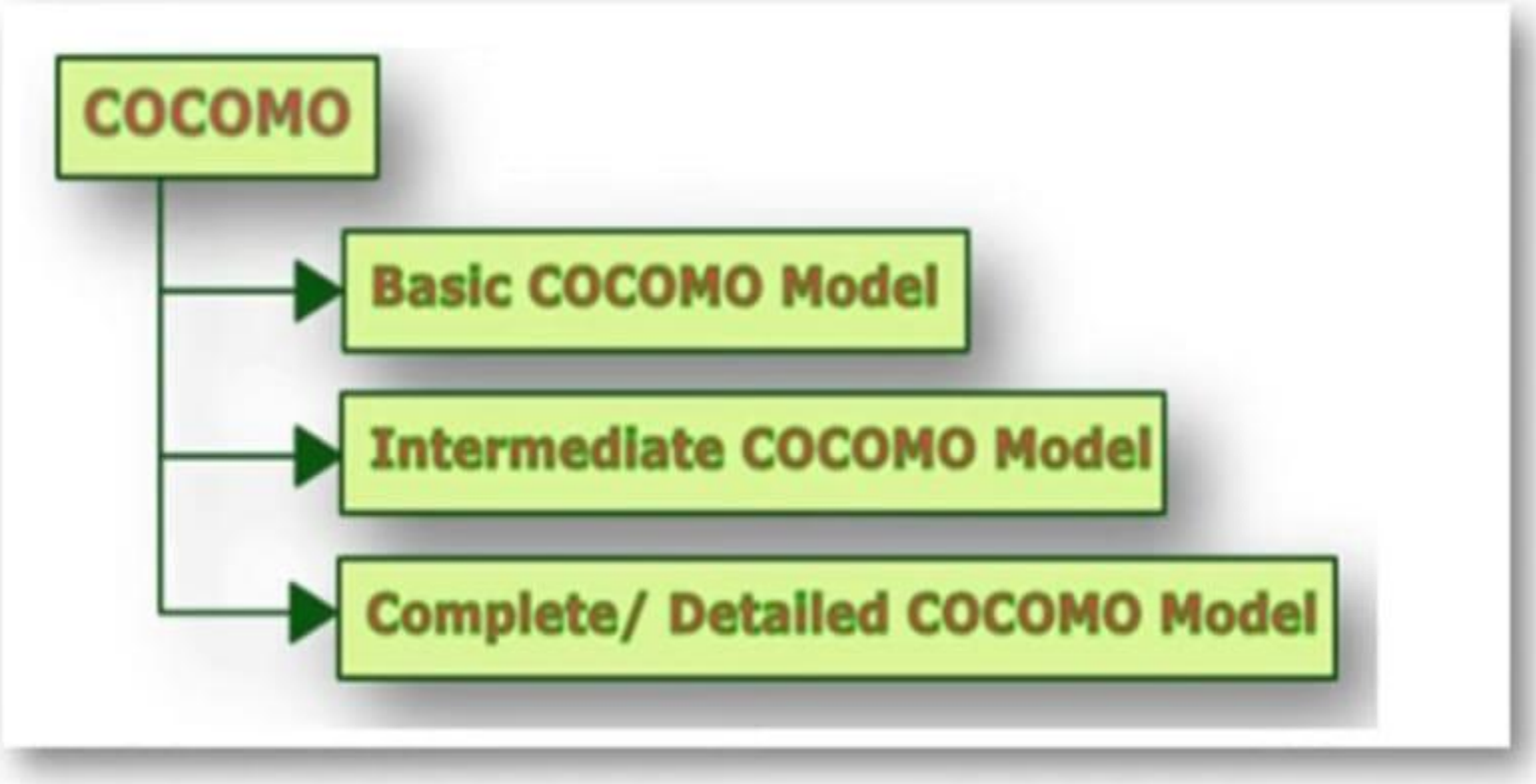
1. Description of resource
2. Resource availability
3. Time of resource when it will be available
4. Duration of resource availability

About COCOMO Model

- 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.



Types of COCOMO Model



Type 1: Basic COCOMO Model

- 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.

1. Basic COCOMO Model:

The basic COCOMO model provide an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model:

Effort=a_1*(KLOC) a_2 PM

Tdev=b_1*(efforts)b_2 Months

Where

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

a_1,a_2,b_1,b_2 are constants for each group of software products,

Tdev is the estimated time to develop the software, expressed in months,

Effort is the total effort required to develop the software product, expressed in **person months (PMs)**.

Estimation of development effort

For the three classes of software products, the formulas for estimating the effort based on the code size are shown below:

Organic: Effort = 2.4(KLOC) 1.05 PM
Semi-detached: Effort = 3.0(KLOC) 1.12 PM
Embedded: Effort = 3.6(KLOC) 1.20 PM

Average staff size(SS)=E/D (persons)
Productivity= KLOC/E (LOC/PM)

Type 2: Intermediate COCOMO Model

- 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.

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

Example1: Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three model i.e., organic, semi-detached & embedded.

Solution: The basic COCOMO equation takes the form:

Effort=a_1*(KLOC) a_2 PM

Tdev=b_1*(efforts)b_2 Months

Estimated Size of project= 400 KLOC

(i)Organic Mode

E = 2.4 * (400)1.05 = 1295.31 PM

D = 2.5 * (1295.31)0.38=38.07 PM

(ii)Semidetached Mode

E = 3.0 * (400)1.12=2462.79 PM

D = 2.5 * (2462.79)0.35=38.45 PM

(iii) Embedded Mode

E = 3.6 * (400)1.20 = 4772.81 PM

D = 2.5 * (4772.8)0.32 = 38 PM

Software Project Types

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.

Software Project Types

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.

Problem Based Estimation

- The project planner begins with a bounded statement of software scope.
- Decompose the software into problem functions that can each be estimated individually.
- Problem based estimation is done by Line of Code (LOC) and Function Points (FP) estimation, Process Based and Use Case Estimation.

Cont'd

LOC and FP data are used in two ways during project estimation:

- These are helpful to estimate size of each variable of the software
- The baseline metric collected from past projects and used in conjunction with data variables LOC and FP to develop cost and efforts.

LOC (Line of Code)

- LOC is line of code in which we count the lines of code. If there are more lines of code then it is said to be big project if it has less line of code then it is said to be small project

- Which lines of code to consider in LOC
  1.Declarations
  2.Actual code including logic and computation.
- What is not LOC
  1. Blank lines
  2. Comments
- Blank lines:

It is included to improve readability of code
Easy to Understand

LOC (Line of Code)

- Comments

It is included to help in code understanding as well as during maintance.

- They do not contribute to any kind of Functionality.

Misused by the developer to give a false notions about productivity.

Advantages of LOC for size estimation

1. Very easy to count and calculate from the developed code

Disadvantages of LOC for size estimation

1. LOC is language and technology dependent.
2. It varies from organisation to organisation.

LOC (Line of Code)

P1:

```
For(int i=1; i<10;i++)
Cout << i ;
```

Loc=2 for P1

P2:

```
For(int i=1; i<10;i++)
{
Cout << i ;
}
```

Loc=4 for P2

LOC (Line of Code)

- Int a ; //declaration

declaration and comment both are on same line now.
Ambiguity is present.

- //declaration

Int a ;

declaration and comment both are on different line.

LOC based Estimation

Using the historical data the project planner expected value by considering following variables-
1.Optimistics 2. most likely 3.pessimistic

Eg. Following formula

$$S = [S_{opt} + 4 * S_m + S_{pess}]/6$$

Considers for “most likely “ estimate where S is the estimation size variable, Sopt represents the optimistic estimates, Sm represents most likely estimate and S pess represents pessimistic estimate values.

Examples of LOC based Estimation

Consider an ABC project with some important modules such as -

1. User interface and control facility
2. 2D graphics analysis
3. 3D graphics analysis
4. Database management
5. Computer graphics display facility
6. Peripheral control function
7. Design analysis model

Estimate the project based on LOC

For estimating the given application we consider each module as separate function and corresponding lines of code can be estimated in the following table as-

Function	Estimated LOC
User interface and control facility	2500
2D graphics analysis (2DGA)	5600
3D graphics analysis (3DGA)	6450
Database management (DBM)	3100
Computer graphics display facility(CGDF)	4740
Peripheral control function(PCF)	2250
Design analysis model (DAM)	7980
Total Estimation in LOC	32620

Expected LOC for 3D graphics analysis function based on 3 point estimation is-

✔Optimistic estimation 4700

✔ most likely estimation 6000

✔Pessimistic estimation 10000

$S = [S_{opt} + 4 * S_m + S_{press}]/6$

Expected value= [4700+(4*6000)+10000]/6 = 6450

A review of historic data indicates -

1.Average productivity is 500 LOC per month

2.Average labor cost is $6000 per month

Then cost for line of code can be estimated as

Cost/LOC=(6000/500)=$12

By considering total estimated LOC as 32620

Total estimated project cost=(32620*12)=$391440

Total estimated project efforts =(32620/500)=65 person/month

FP (Function Points)

- It is not dependent on technology or language.
- It measures functionality from user's point of view.
- User point of view means what the user received from software and what user is asking to design in software.
- It only focuses on what functionality is being delivered.

Five functional units of FP

- ILF(Internal logical files) : the control data or other logically related data that is stored or present with in system.
- EIL(External interface files): the control data or other logical data that is referenced by the system but present in other system.
- EI(External Input): the data or controlled information that comes from outside our system.
- EO(External output) : data that goes out of the system after generation.
- EQ(External Enquires): : It is basically a combination of input and output resulting in data retrieval.

Example of FP based estimation

FP focuses on information domain values rather than software functions. thus we create a function point calculation table for ABC project discussed in previous topic.

Info domain value	Opt	Most likely	pessimistic	Estimated value	Weight factor	FP
No of input	25	28	32	28.1	4	112
No of output	14	17	20	17	5	85
No of enquries	17	23	30	23.1	5	116
No of files	5	5	7	5.33	10	53
No of external i/f	2	2	3	2	7	15
Count total						381

Each of the compl… the complexity adjustment factor i…r table given below .(based on questic…

Sr. No	Factor	Value (Fi)
1	Back-up and recovery?	4
2	Data communication?	2
3	Distributed processing?	0
4	Performance critical?	4
5	Existing operational environment?	3
6	Online data entry?	4
7	Input transaction over multiple screens?	5
8	Online updates?	3
9	Information domain value complex?	5
10	Internal processing complex?	5
11	Code designed for reuse?	4
12	Conversion/installation in design?	3
13	Multiple installation?	5

$\sum (F_i) \longrightarrow 52$

The estimated number of adjusted FP is derived using the following formula:

FP estimated=(FP count total * [complexity adjustment factor]

FP estimated=count total*[0.65+(0.01*$\sum (F_i)$)]

Complexity adjustment factor = [0.65+(0.01*52)]=1.17

FP estimated = (381*1.17) = 446 (function point count adjustment with complexity adjustment factor)

A review of the historical data indicates-

1.Average productivity is 6.5 FP/ person month

2.Average labor cost is 6000 per month.

Calculation for cost per function point, total estimated project cost and total effort

1.The cost per function point=(6000/6.5)=923

2.Total estimated project cost = (446*923)=411658

3.Total estimated effort = (446/6.5)=69 person month

Object based Estimation

1. Using requirement model, develop use case and determine the count. As project proceed, the number of the use cases may get changed.
2. Determine the number of key classes using requirement model
3. Categorize the type of interface and develop the multiplier for example

Multiply the key classes by this multiplier to obtain the no. of support classes
Now, multiply the key and support classes by average no. Of work units per class. Approximately there are 15-20 person-days per class

Interface Type	Multiplier
No GUI	2.0
Text based interface	2.25
Graphical User interface	2.5
Complex GUI	3.0

Object Points Computation

1. Similar to FP
2. Compute the no. of screens and classify them as simple,medium,complex.
3. Compute the no. of reports and classify them as simple,medium,complex.
4. Count the no. of modules that have to be developed.
5. Use weight metrices to sum the values above, taking into account reused code.
6. A formula translates Ops into productivity measures.

Process Oriented Estimation

- It is most commonly used method for estimating project based on the processes used.
- This technique decomposes process into relatively small set of tasks and the efforts required to accomplish each task is estimated.
- The estimation starts by identifying software functions obtained from project scope then series of software process activities should perform for each function.
- Create table which contain function and software process activities.
- Finally planner estimates the efforts of each software function.

Consider the project XYZ. We can estimate it using Process based Estimation technique by creating a following table.
process based estimation.docx

- In this technique for each business function software engineering task like analysis, design coding and testing is carried out.
- Estimation function for each these are identified and then sum is obtained.
- CC-Customer communication
- CE-Customer evaluation
- The percentage effort is calculated based on the total 44.
- Total for task design as 22.05
- 22.05*100/44=50 %
- Based on the average labour cost of $6000 per month

1. Total estimated project effort=44 person-month
2. Total estimated project cost=(6000*44)=$264000

Estimation with use cases

- Use cases provides s/w team with insight into s/w **scope and requirements.**
- But there are some difficulties in developing estimation using use cases due to the following reasons:

- Use cases can be described by varieties of ways. There is no unique format for them.
- Use case represents an external view (Users view) of the software and therefore be written in many of abstractions.
- Use case do not address the complexity functions or features that are describing
- Use case also not describes the complex behavior involved in many functions & features.

Cont'd

Following formula is useful to compute this estimation

LOC estimation =N * LOC $_{avg}$ + [(S_a/S_h-1)+(P_a/P_h-1)] * LOC $_{adjust}$

Where N= actual number of use cases.

LOC $_{avg}$ = historical average LOC/use case for this type of subsystem

LOC $_{adjust}$ = adjustment based on 'n%' of LOC avg

n=defined locally and represents the difference between this project and average project.

S_a = actual scenario per use case , S_h = average scenario per use case for this type subsystem

P_a = actual pages per use case , P_h = average pages per use case for this type of subsystem

above expression is used to develop a rough estimation of the number of LOC based on the actual number of use cases by the number of scenarios and page length of the use cases

Empirical Estimation model

- Values for LOC and FP are used into the estimation model.
- The empirical data are derived from a limited sample of projects.
- Hence this estimation model is appropriate for all classes of software project.

Structure of estimation model:

- It is typical estimation model.
- It is derived using regression analysis on data collected from past s/w projects.
- The estimated efforts can be denoted by following formula- $E= A+ B*(e_v)^C$
- Where A,B and C are the constants derived empirically.

E is the effort in persons-months

e_v is the estimation variable derived from either LOC and FP

COCOMO II model

- **COCOMO-II** is the revised version of the original cocomo **model(constructive cost model)** and is developed at University of Southern California.

- It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.

- COCOMO II is actually a hierarchy of three estimation models

- As with all estimation models, it requires sizing information and accepts it in three forms: object points, function points, and lines of source code

Modes of development

- Boehm proposed that there can be three modes of software development project based on development complexity. He considered software size innovation deadline / constraint deadline and development environment and gave three modes which were

Cont'd....

Development Mode	Project Characteristics			
	Size	Innovation	Deadline/ constraints	Dev. Environment
Organic	Small	Little	Not Tight	Stable
Semi-Detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex hardware/ Customer Interface

Development Modes

Cont'd….

- COCOMO model depends on two main equations:

1. Development Efforts:

$$MM = a_1 \times (KLOC)^{a_2} \; PM$$

Based on MM- man month (MM)/ person month(PM)/ staff month is one month of efforts by one person. COCOMO consider 152 hours per person month. It may vary according to organization by 10% to 20%.

2. Effort and Development Time(TDEV)

$$TDEV = b_1 \times (Effort)^{b_2} \; Months$$

Tdev is estimated time to develop the software, expressed in months.

- a1,a2,b1,b2 are constant for each category of software

Project Scheduling

Project-task scheduling is a significant project planning activity. It comprises deciding which functions would be taken up when. To schedule the project plan, a software project manager wants to do the following:

1. Identify all the functions required to complete the project.
2. Break down large functions into small activities.
3. Determine the dependency among various activities.
4. Establish the most likely size for the time duration required to complete the activities.
5. Allocate resources to activities.
6. Plan the beginning and ending dates for different activities.
7. Determine the critical path. A critical way is the group of activities that decide the duration of the project.

Project Scheduling Principles

Scheduling Principles:

Following are the seven scheduling principles those are applied as the project schedule evolves

1. Compartmentalization: Project must be compartmentalized into number of manageable activities, actions and tasks To achieve this both the project and the process are decomposed

2. Interdependency: The Interdependency of each compartmentalized activity or task must be identified. Some task may occur in sequence or parallel.

Cont'd….

Therefore tasks which occur in sequence have to be performed in a sequential order since the output of one task will be the input of the next task. Other tasks can occur independently.

3. Time allocation: Each and every task has to be assigned a specific time period i.e. a start date and a completion date based on whether the work will be performed in a full time or part time basis.

Cont'd….

4. Effort validation: Every project is assigned to a software team. The project manager has to make sure that the effort allocated should not be more than the number of people available to do the work.

5. Defined responsibilities: Each of the scheduled task is assigned to a specific member of the software team.

6. Defined outcomes: Each task has a defined outcome. Work product is the outcome of a software project.

7. Defined milestones: Every task is associated with a milestone. A milestone is an action or event marking a significant change in development process.