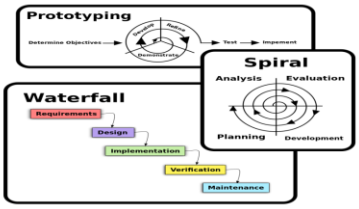
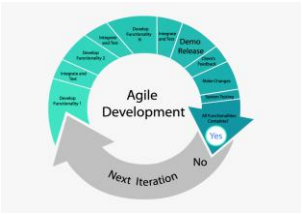
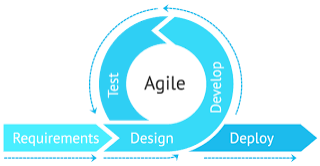


Software Engineering And Testing

Software Project Management





Outline

- Software Project Management Complexities
- Responsibilities of a Software Project Manager,
- Project Planning
  - Sliding Window Planning
  - The SPMP Document of Project Planning.
- Metrics for Project Size Estimation
  - Lines of Code (LOC),
  - Function Point (FP) Metric.
- Project Estimation Techniques
  - Empirical Estimation Techniques [Expert Judgement, Delphi Cost Estimation],
  - Heuristic Techniques [COCOMO],

Outline

- Project Estimation Techniques
  - Heuristic Techniques
  - Analytical Estimation Techniques [Halstead's Software Science].
- Scheduling
  - Work Breakdown Structure
  - Activity Networks, Critical Path Method (CPM)
  - PERT Charts, Gantt Charts.
- Organization and Team Structures
- Staffing and its Estimation,
- Risk Management,
- Software Configuration Management

3

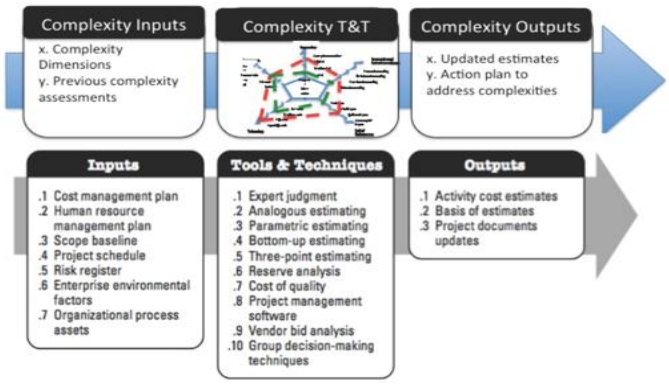
Software Project Management

- The main goal of software project management is to enable a group of developers to work effectively towards the successful completion of a project.
- **Software Project Management Complexities**
- **Invisibility:** Invisibility of software makes it difficult to assess the progress of a project and is a major cause for the complexity of managing a software project.
- **Changeability:** Frequent changes to the requirements and the invisibility of software are possibly the two major factors making software project management a complex task.
- **Complexity:** Even a moderate sized software has millions of parts (functions) that interact with each other in many ways—data coupling, serial and concurrent runs, state transitions, control dependency, file sharing, etc.

4

Software Project Management

- **Complexity:**



- **Uniqueness:** Every software project is usually associated with many unique features or situations.

5

Software Project Management

- **Exactness of the solution:** The requirement not only makes it difficult to get a software product up and working, but also makes reusing parts of one software product in another difficult. This requirement of exact conformity of the parameters of a function introduces additional risks and contributes to the complexity of managing software projects.
- **Team-oriented and intellect-intensive work:** In a software development project, the life cycle activities not only highly intellect-intensive, but each member has to typically interact, review, and interface with several other members, constituting another dimension of complexity of software projects.

6

Responsibilities of Software Project Manager

- **Job Responsibilities for managing software Project**
  - **Project planning** : Project planning involves estimating several characteristics of a project and then planning the project activities based on these estimates made.
  - **Project monitoring and controling** : The focus of project monitoring and control activities is to ensure that the software development proceeds as per plan.
- **Skills Necessary for Managing Software Projects**
  - Knowledge of project management techniques.
  - Decision taking capabilities.
  - Previous experience in managing similar projects.

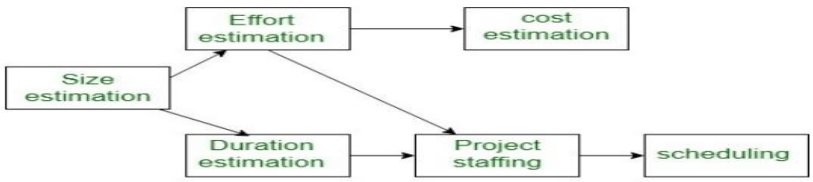
7

Project Planning

- Once a project has been found to be feasible, software project managers undertake project planning.
- Initial project planning is undertaken and completed before any development activity starts.
- Project manager performs the following activities:
  - Estimation: The following project attributes are estimated.
    - Cost: How much is it going to cost to develop the software product?
    - Duration: How long is it going to take to develop the product?
    - Effort: How much effort would be necessary to develop the product?
    - The effectiveness of all later planning activities such as scheduling and staffing are dependent on the accuracy with which these three estimations have been made.

Project Planning

- Scheduling: After all the necessary project parameters have been estimated, the schedules for manpower and other resources are developed.
- Staffing: Staff organisation and staffing plans are made.
- Risk management: This includes risk identification, analysis, and abatement planning.
- Miscellaneous plans: This includes making several other plans such as quality assurance plan, and configuration management plan, etc.



Precedence ordering among planning activities

Sliding Window Planning

- Most project managers take a phased approach to project design.
- By breaking out a project's design into phases, managers can avoid taking on significant responsibilities too soon. This method is called "**window designing**."
- Using the window technique, the project is more carefully planned in successive development phases after a preliminary setup.
- Project managers often start with very little information on the key components of the project.
- The project's database grows gradually as it moves through different phases.
- The project managers will arrange the following sections with a greater degree of precision and certainty as each one is finished.

11

SPMP Document of Project Planning

1. Introduction : (a) Objectives (b) Major Functions (c) Performance Issues (d) Management and Technical Constraints
2. Project Estimates : (a) Historical Data Used (b) Estimation Techniques Used (c) Effort, Resource, Cost, and Project Duration Estimates
3. Schedule : (a) Work Breakdown Structure (b) Task Network Representation (c) Gantt Chart Representation (d) PERT Chart Representation
4. Project Resources : (a) People (b) Hardware and Software (c) Special Resources
5. Staff Organization : (a) Team Structure (b) Management Reporting
6. Risk Management Plan : (a) Risk Analysis (b) Risk Identification (c) Risk Estimation (d) Risk Abatement Procedures

12

SPMP Document of Project Planning

7. Project Tracking and Control Plan
8. Miscellaneous Plans : (a) Process Tailoring (b) Quality Assurance Plan (c) Configuration Management Plan

13

Which Project Parameters to be Estimated?

- What is the **size** of the software to be developed?
- How much **effort** is required?
- How much **calendar** time is needed?
- What is the **total cost** ?

Software Cost Components

- Hardware and software costs.
- Travel and training costs.

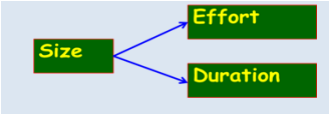
Effort costs: The dominant factor in most projects : Salaries of the project team

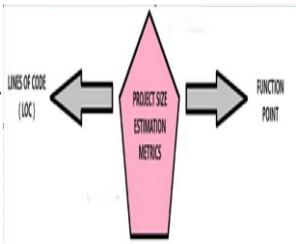
- **Overheads:**
  - Costs of building, heating, lighting.
  - Costs of networking and communications.
  - Costs of shared facilities (e.g library, staff restaurant, etc.).

14

Metrics For Project Size Estimation

- The project size is a measure of the problem complexity in terms of the effort and time required to develop the product.
- The size of a project is obviously not the number of bytes that the source code occupies, neither is it the size of the executable code.
- Two metrics are popularly being used to measure size
  - Source Lines of code (SLOC)
  - Function point (FP)
- **SLOC is conceptually simple**
- But, FP is now-a-days favoured over SLOC
- Because of the many shortcomings of SLOC.





Lines of Code

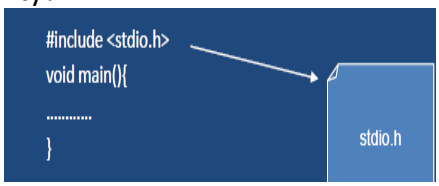
- The simplest among all metrics available to estimate project size.
- Project size estimated by counting the number of source instructions
- Lines used for commenting, header lines ignored
- To find LOC at the beginning of a project divide module into sub modules and so on until size of each module can be predicted.
- Accurate estimation of LOC count at the beginning of a project is a very difficult task.

Shortcomings of Lines of Code

- Gives a numerical value of problem size that vary widely with individual coding style

```
If( x>y )                    x > y ? x++ : y++;
   then x++;
else
   y++;
```

- Good problem size ⇒ Overall complexity of Problem + Effort needed.
- Effort needed for analysis, design , coding, testing etc (not just coding)
- Larger Code size ⇒ Better Quality?
- Impact of Code Reuse on LOC ?



Shortcomings of Lines of Code

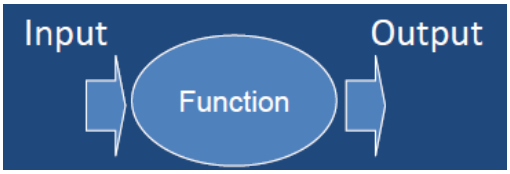
- Logical Complexity?
  - Complex Logic ⇒ More Effort
  - Simple Logic ⇒ Less Effort

```
while(i<4){                  printf("testing");
   printf("testing");        printf("testing");
}                            printf("testing");
                             printf("testing");
```

- Accurate computation of LOC only after project completion!!

Function Point (FP) Metric

- **Function points are independent of the language, tools, or methodologies used for implementation**
- **Function points can be estimated early in analysis and design**
- **Since function points are based on the user's external view of the system:**
  - Non-technical users of the software have a better understanding of what function points are measuring
- Size of software product computed directly from problem specification
- Size of software = number of different functions/ features it supports



19

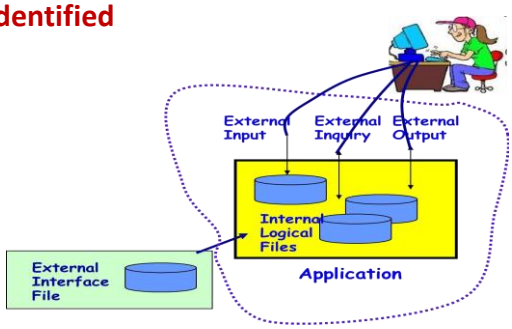
Function Point (FP) Metric

- Apart from that size depends on
  - number of files
  - number of interfaces
  - number of enquiries
  - external inputs and outputs;
- Size of Function Point (FP)= Weighted sum of these five problem characteristics
- UFC = $\sum$(number of elements of given type) × (weight)

20

Function Point (FP) Metric

- **Five key components are identified**
  - **External Inputs**
  - **External Outputs**
  - **External Inquiries**
  - **Internal Logical Files**
  - **External Interface Files**
- **Five function types:**



1. **Logical interface file (LIF) types :** Equates roughly to a data store in the system that is created and accessed by the system
2. **External interface file types (EIF) :** Represents data retrieved from a data store which is actually maintained by a different application.
3. **External input (EI) types:** Transactions which update internal computer files

21

Function Point (FP) Metric

4. **External output (EO) types:** Transactions which extract and display data from internal computer files. Generally involves creating reports.
5. **External inquiry (EQ) types:** User initiated transactions which provide information but do not update computer files.
   - Normally the user inputs some data that guides the system to the information the user needs.

Albrecht Complexity Multipliers

External user types	Low complexity	Medium complexity	High complexity
EI	3	4	6
EO	4	5	7
EQ	3	4	6
LIF	7	10	15
EIF	5	7	10

Function Point (FP) Metric

- **FP is computed in 2 steps:**

1. Compute Unadjusted Function Point (UFP)
2. Technical Complexity Factor (TCF)

- **Step 1:** Compute Unadjusted Function Point UFP
  - UFP = (Number of inputs) * 4 + (Number of outputs) * 5 + (Number of inquiries) * 4 + (Number of files)* 10 + (Number of interfaces) * 10
- **Step 2:** Compute Technical Complexity Factor (TCF)
  - Considers 14 factors e.g. response time requirements, throughput etc
  - Each assigned a value from 0 (absent) to 5 (strong influence) Sum resulting numbers = Total Degree of Influence (DI)
  - TCF = (0.65 + 0.01 * DI) TCF varies from 0.65 to 1.35
  - **FP = UFP * TCF**

23

Function Point (FP) Metric

14 General Systems Characteristics are evaluated and used to compute a Value Adjustment Factor (VAF)

General System Characteristics

Data Communication	On-Line Update
Distributed Data Processing	Complex Processing
Performance Objectives	Reusability
Heavily Used Configuration	Conversion & Install Ease
Transaction Rate	Operational Ease
On-Line Data Entry	Multiple-Site Use
End-User Efficiency	Facilitate Change

Function Point (FP) Metric

- **0 Not present, or no influence**
- **1 Incidental influence**
- **2 Moderate influence**
- **3 Average influence**
- **4 Significant influence**
- **5 Strong influence throughout**

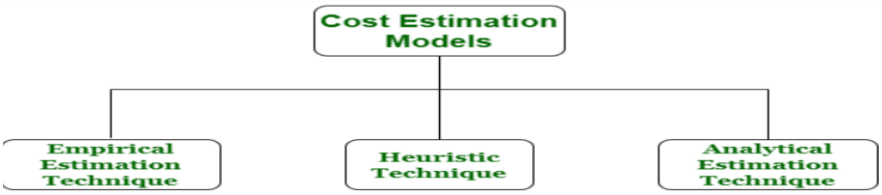
25

Function Point (FP) Metric Example

- **Given the following values, compute function points when all complexity adjustment factors (CAF) and weighting factors are average.**
  - User Input = 50, User Output = 40, User Inquiries = 35, User Files = 6, External Interface = 4
- **Solution:**
  - **Step-1:** As complexity adjustment factor is average (given in question), hence, scale = 3. **F = 14 * 3 = 42**
  - **Step-2:** CAF = 0.65 + (0.01 * 42) = **1.07**
  - **Step-3:** As weighting factors are also average (given in question) hence we will multiply each individual function point to corresponding values in TABLE. **UFP = (50*4) + (40*5) + (35*4) + (6*10) + (4*7) = 628**
  - **Step-4:** Function Point = 628 * 1.07 = **671.96**

26

Project Estimating Techniques



- Empirical techniques: an educated guess based on past experience.
  - Expert Judgement: An educated for guess made by an expert. Suffers from individual bias.
  - Delphi Estimation: overcomes some of the problems of expert judgement.
- Heuristic techniques: assume that the characteristics to be estimated can be expressed in terms of some mathematical expression.
- Analytical techniques: derive the required results starting from certain simple assumptions.

27

COCOMO Model

- **COCOMO (CONSTRUCTIVE COST MODEL) :** First published by Dr. Barry Boehm, 1981
- **Basic model:** $effort = c \times size^k$
  - C and k depend on the type of system: organic, semi-detached, embedded.
  - Size is measured in 'kloc' i.e. Thousands of lines of code
- **Three development environments (modes)**
  - Organic Mode
  - Semidetached Mode
  - Embedded Mode
- **Three increasingly complex models**
  - Basic Model
  - Intermediate Model
  - Detailed Model

28

COCOMO Model

- **Organic :** the project deals with developing a well-understood application program, the size of the development team is reasonably small, and the team members are experienced in developing similar types of projects.
  - **Examples** of this type of projects are simple business systems, simple inventory management systems, and data processing systems.
- **Semidetached :** if the development team consists of a mixture of experienced and inexperienced staff. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed.
  - **Example** of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.

29

COCOMO Model

- **Embedded :** if the software being developed is strongly coupled to hardware, or if stringent regulations on the operational procedures exist. Team members may have limited experience on related systems but may be unfamiliar with some aspects of the system being developed. New product requiring a great deal of innovation.
  - **Example:** ATM, Air Traffic control.

30

COCOMO Model

	Organic	Semi-detached	Embedded
Project size (lines of source code)	2,000 to 50,000	50,000 to 300,000	300,000 and above
Team Size	Small	Medium	Large
Developer Experience	Experienced developers needed	Mix of Newbie and experienced developers	Good experience developers
Environment	- Familiar Environment	- Less familiar environment	- Unfamiliar environment (new) - Coupled with complex hardware
Innovation	Minor	Medium	Major
Deadline	Not tight	Medium	Very tight
Example(s)	Simple Inventory Management system	New Operating system	Air traffic control system

Person Month

- **Suppose a project is estimated to take 300 person- months to develop:**
  - Is one person working for 30 days same as 30 persons working for 1 day?
  - Yes/No? Why?
- **How many hours is a man month?**
  - Default Value: 152 hours per month
  - 19 days at 8 hours per day.
- **Why Person-Month and not Person-days or Person Years?**
  - Modern Projects typically take a few months to complete...
  - Person-years is clearly unsuitable.
  - Person-days would make monitoring and estimation overhead large and tedious.

32

COCOMO Model

- **Basic Model :** Used for early rough, estimates of project cost, performance, and schedule
  - Accuracy: within a factor of 2 of actual 60% of time
- **Intermediate Model :** Uses Effort Adjustment Factor (EAF) fm 15 cost drivers
  - Doesn't account for 10 – 20% of cost
  - Accuracy: within 20% of actual 68% of time
- **Detailed Model :** Uses different Effort Multipliers for each phase of project
- (Most project managers use intermediate model)

33

Basic Effort Equation (COCOMO Model)

$$\text{Effort} = A(\text{size})^k$$

- A is a constant based on the developmental mode
  - organic = 2.4
  - semi = 3.0
  - embedded = 3.6
- Size = 1000s Source Lines of Code (KSLOC)
- k is constant for a given mode
  - organic = 1.05
  - semi = 1.12
  - embedded = 1.20

34

Basic Effort Equation (COCOMO Model)

$$\text{Effort} = a_1 \times (\text{KLOC})^{a_2}\ \text{PM}$$

$$\text{Tdev} = b_1 \times (\text{Effort})^{b_2}\ \text{Months}$$

- where, **KLOC** is the estimated size of the software product expressed in Kilo Lines of Code,
- **a1**, **a2**, **b1**, **b2** are constants for each category 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).
- The values of a1, a2, b1, b2 for different categories of products (i.e., organic, semi-detached, and embedded) as given by Boehm are summarized below.

35

Basic Effort Equation (COCOMO Model)

- **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: $\text{Effort} = 2.4 \times (\text{KLOC})^{1.05}\ \text{PM}$

Semi-detached: $\text{Effort} = 3.0 \times (\text{KLOC})^{1.12}\ \text{PM}$

Embedded: $\text{Effort} = 3.6 \times (\text{KLOC})^{1.20}\ \text{PM}$

- **Estimation of Development Time**: For the three classes of software products, the formulas for estimating the development time based on the effort size are given below:

Organic: $\text{Tdev} = 2.5 \times (\text{Effort})^{0.38}\ \text{Months}$

Semi-detached: $\text{Tdev} = 2.5 \times (\text{Effort})^{0.35}\ \text{Months}$

Embedded: $\text{Tdev} = 2.5 \times (\text{Effort})^{0.32}\ \text{Months}$

36

COCOMO Model Example

- Assume that the size of an organic type software product has been estimated to be 32,000 lines of source code. Assume that the average salary of software engineers be Rs. 15,000/- per month. Determine the effort required to develop the software product, the nominal development time, and cost required to develop the product. Also find number of persons required.

37

COCOMO Model Example Solution

- From the basic COCOMO estimation formula for organic software:

$$\text{Effort} = 2.4 \times (32)^{1.05} = 91 \text{ PM}$$

$$\text{Tdev} = 2.5 \times (91)^{0.38} = 14 \text{ Months}$$

$$\text{Cost} = 14 \times 15{,}000 = \text{Rs. } 2{,}10{,}000/-$$

- Person required (P) = Effort / Tdev = 91/14=6.5 Person ~7Person

38

Project Scheduling Activities

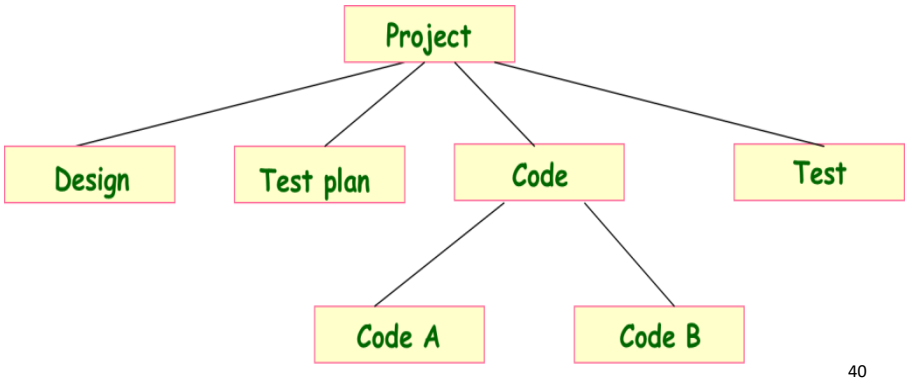
- The project manager should do the following:

1. Identify all the major activities that need to be carried out to complete the project.
2. Break down each activity into tasks.
3. Determine the dependency among different tasks.
4. Establish the estimates for the time durations necessary to complete the tasks.
5. Represent the information in the form of an activity network.
6. Determine task starting and ending dates from the information represented in the activity network.
7. Determine the critical path. **A critical path is a chain of tasks that determines the duration of the project.**

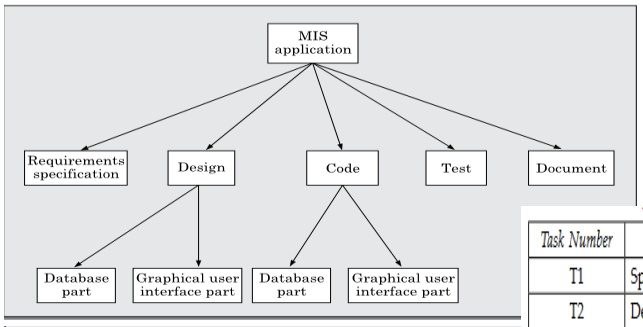
39

Work Breakdown Structure

- **Hierarchical decomposition of a project into subtasks**
  - Shows how tasks are decomposed into subtasks
  - Does not show duration
  - Does not show precedence relations (e.g. task A must be finished before task B can start)

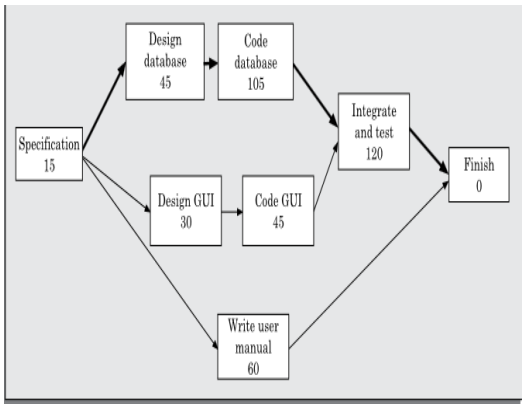


40



WBS

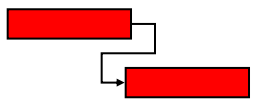
Task Number	Task	Duration	Dependent on Tasks
T1	Specification	15	-
T2	Design database	45	T1
T3	Design GUI	30	T1
T4	Code database	105	T2
T5	Code GUI part	45	T3
T6	Integrate and test	120	T4 and T5
T7	Write user manual	60	T1



Activity Network

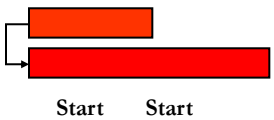
Logical Relationship

- *"Finish to Start" (most common)*

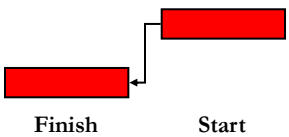


- *"Finish to Finish"*

- *"Start to Start"*

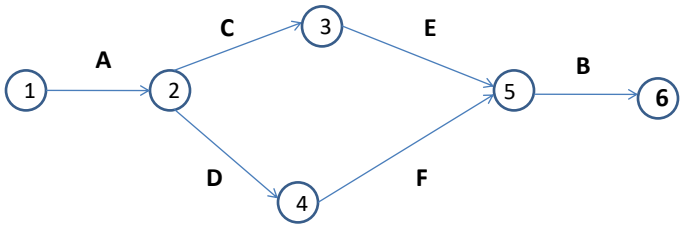


- *"Start to Finish" (very rare, not available in project management software)*



Example

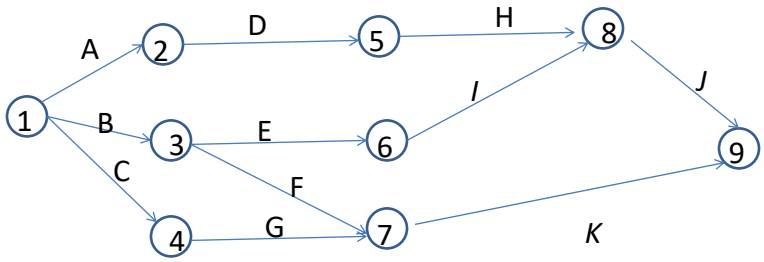
- Draw the logic network for the following: Activities C and D both follow A , activity E follows C , activity F follows D , activity E and F precedes B.



Construct a network for a project whose activities and their predecessor relationship are given in table

Activity	A	B	C	D	E	F	G	H	I	J	K
Predecessor	-	-	-	A	B	B	C	D	E	H,I	F,G

Activity	A	B	C	D	E	F	G	H	I	J	K
PAc	-	-	-	A	B	B	C	D	E	H,I	F,G

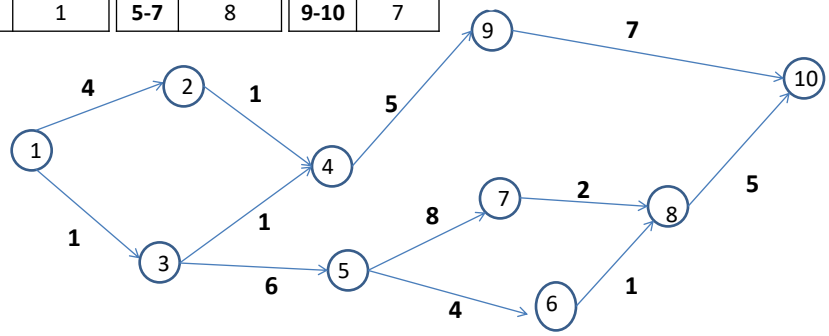


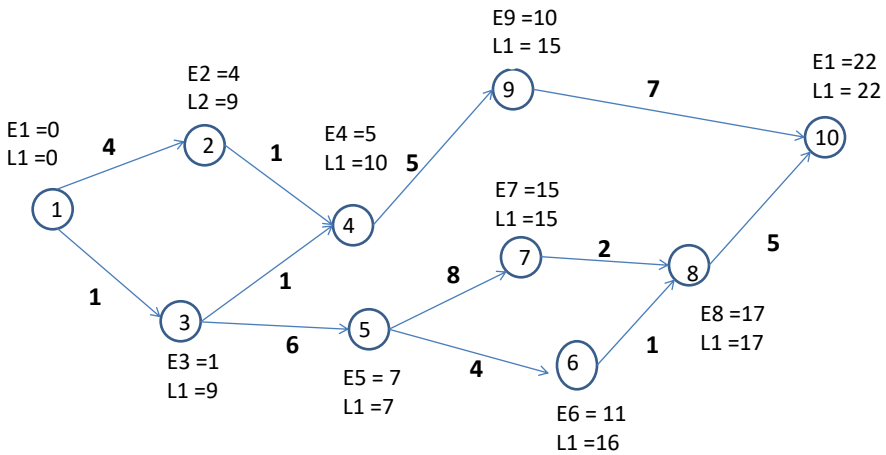
Example

- A project has the following characteristics:

Activity	Days
1-2	4
1-3	1
2-4	1
3-4	1
3-5	6
4-9	5
5-6	4
5-7	8
6-8	1
7-8	2
8-10	5
9-10	7

Activity	Days	Activity	Days	Activity	Days
1-2	4	**3-5**	6	**6-8**	1
1-3	1	**4-9**	5	**7-8**	2
2-4	1	**5-6**	4	**8-10**	5
3-4	1	**5-7**	8	**9-10**	7






THANKS