GANPAT UNIVERSITY
U. V. PATEL COLLEGE OF ENGINEERING

2CEIT502 SOFTWARE ENGINEERING

UNIT 5

SOFTWARE PROJECT MANAGEMENT (SPM)

Prepared by: Prof. Ravi Raval (Asst. Prof in C.E Dept. , UVPCE)

Unit 5: Software Project Management

Contents:

Unit 5: Software Project Management

5.1 Introduction

- The main goal of software project management is to enable a group of software developers to work efficiently towards successful completion of the project.

Unit 5: Software Project Management

5.2 Responsibilities of a Software Project Manager:

Job Responsibilities:

1) The responsibilities and activities of a software project manager is large and varied.
2) Building up team morale to highly visible customer presentations
3) Project proposal writing
4) Project cost estimation
5) Scheduling
6) Project staffing
7) Software Process tailoring
8) Project monitoring and control
9) Software configuration management
10) Risk management
11) Interfacing with clients
12) Managerial report writing and presentations

Unit 5: Software Project Management

5.2 Responsibilities of a Software Project Manager:

Job Responsibilities:

- Classification of all responsibilities

1. Project Planning:
   - Estimating activities
   - Planning according to estimation
   - Feasibility study
   - Revise plan
2. Project monitoring and controlling activities:
   - Ensure development progress as per plan

Unit 5: Software Project Management

5.2 Responsibilities of a Software Project Manager:

Skills to be possessed by software project manager:

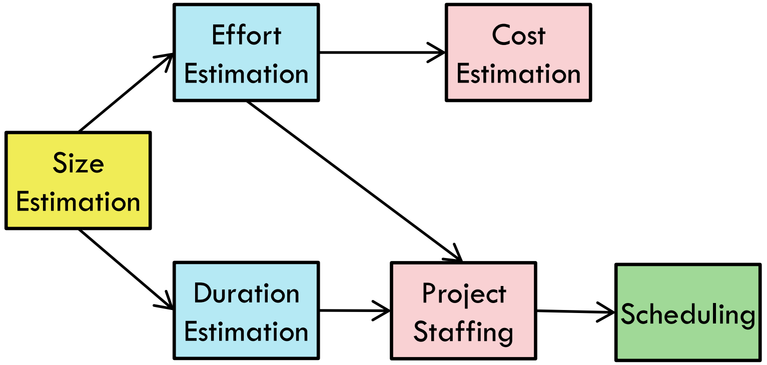
1) Good quality judgment
2) Decision making capabilities
3) Knowledge of latest software project management techniques
    1) Cost estimation
    2) Risk management
    3) Configuration management
4) Good communication skills
5) Ability to get work done
6) Skills to track and control software progress
7) Customer interaction
8) Managerial presentation
9) Team building

Unit 5: Software Project Management

5.3 Project Planning and it's activities

Activities of project planning:

1) Estimation
   1) Cost
   2) Duration
   3) Effort
2) Scheduling manpower and resources
3) Staffing
4) Risk Management
5) Miscellaneous plans: e.g. Quality assurance plan, configuration management plan etc.



Unit 5: Software Project Management

<u>5.5 The Software Project Management Plan (SPMP) Document</u>

- Once the project planning is complete, project managers document their plans in a SPMP document. Here is the template of SPMP document.

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 organisation**
(a) Team Structure
(b) Management Reporting
6. **Risk management plan**
(a) Risk Analysis
(b) Risk Identification
(c) Risk Estimation
(d) Risk Abatement Procedures
7. **Project tracking and control plan**
(a) Metrics to be tracked
(b) Tracking plan
(c) Control plan
8. **Miscellaneous plans**
(a) Process Tailoring
(b) Quality Assurance Plan
(c) Configuration Management Plan
(d) Validation and Verification
(e) System Testing Plan
(f) Delivery, Installation, and Maintenance Plan

Unit 5: Software Project Management

5.6 Metrics for Project size estimation

- Project size? What we mean by size of the project? And Why we need to find it?

The term project size is a measure of the problem complexity in terms of the effort and time required to develop the software.

- Currently there are two metrics popular in size estimation:

1) Lines of Code (LoC) metric
2) Function Point (FP) metric

Unit 5: Software Project Management

5.6 Metrics for Project size estimation

(1) Lines of Code (LoC)

- It measures the size of project by counting the no. of source instruction in the developed program. While counting it ignores comments and header lines of program.
- Estimating LoC is easy at the end of project. Anybody can do it. But how to get an estimation of LoC at the beginning of the project?
- So, what is the solution to get a near about perfect estimation of LoC at the beginning of the project?

Divide the main problem (software to be developed) into modules; modules into sub-modules; and so on until the leaf-level modules can be approximately predicted

Unit 5: Software Project Management

5.6 Metrics for Project size estimation

(1) Lines of Code (LoC)

Disadvantages:

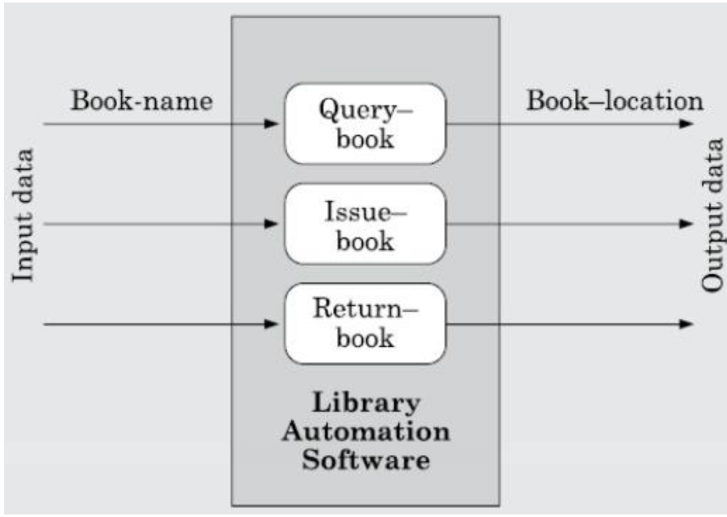
- Numerical value of a problem size can vary programmer to programmer.
- LoC is a measure of coding activity alone. Problem size not only depends on code only but design and test also.
- LoC correlates poorly with the quality and efficiency of the code. "Large code doesn't imply high quality/efficiency.
- LoC metric penalizes use of higher-level programming languages, Code reuse etc. Several library routine call will result lower count.
- LoC matric measures the lexical complexity of a program and does not address the more important but subtle issues of logical or structural complexities. i.e. complex logic program requires more effort than simple logic.
- Biggest shortcoming: It's very difficult to accurately estimate LoC in the final product from just the problem specification.

Unit 5: Software Project Management

5.6 Metrics for Project size estimation

(2) Functional Point (FP) Metric

- It overcomes several shortcomings of LoC.
- The important advantage of using the FP metric is that it can be easily used to estimate the size of project directly from software specification where as in LoC, the size can be accurately determined only after the product has fully developed.
- The conceptual idea behind the function point metric is that the size of a software product is directly dependent on the no. of different functions or features it supports.



Unit 5: Software Project Management

5.6 Metrics for Project size estimation

(2) **Functional Point (FP) Metric**

Steps to calculate function point (FP)

1) **Step 1**: First of all calculate "***unadjusted function point (UFP)***"

UFP = (Number of inputs) * 4 + (Number of outputs) * 5 + (number of inquiries) * 4 + (number of files) * 10 + (number if interfaces) * 10

Where,

- **No. of inputs:** Each data item input by the user is counted. Data inputs should be distinguished from user inquiries.
- **No. of outputs:** reports printed, screen outputs, error messages, audio
- **No. of inquiries:** user commands to perform specific action
- **No. of files:** Each logical file (group of logical data) is counted.
- **No. of interfaces:** interfaces used to exchange info with other system.

Unit 5: Software Project Management

5.6 Metrics for Project size estimation

(2) Functional Point (FP) Metric

Steps to calculate function point (FP)

2) **Step 2**: Refine ***UFP*** in this step. But how to refine?

- The complexity level of each of the parameters as are graded as simple, average or complex. The weights for the different parameters can then be compared based on this table.
- Technical complexity factor (TCF): are such factors which can influence the development effort such as:
  - High transaction rates
  - Response time requirements
  - Scope of reuse etc.

	Simple	Average	Complex
Input (I)	3	4	6
Output (O)	4	5	7
Inquiry (E)	3	4	6
No. of links (F)	7	10	15
No. of interfaces	5	7	10

Unit 5: Software Project Management

5.6 Metrics for Project size estimation

(2) Functional Point (FP) Metric

Steps to calculate function point (FP)

2) **Step 2 (continue):**

- There are 14 such factors which can influence development.
- Assign values all those 14 factors as per following:

  0 – not present / no influence to

  6 – strong influence
- The resulting numbers are summed which calls degree of influence (DI).
- Now TCF can be computed as : (0.65 + 0.01 * DI)

  (DI can vary from 0 to 84 and TCF can vary 0.65 to 1.35)

3) **Step 3:** Finally, FP = UFP * TCF.

Unit 5: Software Project Management

5.6 Metrics for Project size estimation

Shortcomings of Function Point (FP)

- **Major:** it does not take into account the algorithmic complexity of a function.

That means FP assumes that the efforts required to design and develop any two different functionalities of the system is the same. Which is not true.

For Ex. In Library management system, "create member" function would be much simpler to develop then "loan from remote library" function.

- FP only considers the number of functions that the system supports, without distinguishing the difficulty levels of developing the various functionalities.

To overcome this, an extension to FP metric called feature point metric has been proposed which includes algorithmic complexity as an extra parameter.

Unit 5: Software Project Management

5.6 Metrics for Project size estimation

Difference between FP and LOC

Function Point (FP)	Lines of Code (LoC)
1. FP is specification based.	1. LOC is an analogy based.
2. FP is language independent.	2. LOC is language dependent.
3. FP is user-oriented.	3. LOC is design-oriented.
4. It is extendible to LOC.	4. It is convertible to FP (backfiring)

Unit 5: Software Project Management

<u>5.7 Project estimation Techniques</u>

- What to estimate? Size, efforts, duration and cost
- Why to estimate? For resource planning, scheduling and giving quotation to customers.
- Lots of estimation techniques are available but they are classified into three categories:

  5.7.1 Empirical Estimation Techniques
  - Expert Judgment Technique
  - Delphi Cost Estimation.

5.7.2 Heuristics Techniques
- Single Variable Model (Basic COCOMO Model)
- Multi variable model (Intermediate COCOMO Model)

5.7.3 Analytical estimation techniques (Halstead's Software Science)
- Length and Vocabulary
- Program volume
- Potential Minimum Volume
- Effort and Time
- Length Estimation

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.1 Empirical Estimation Techniques

- Based on similar products development experience
- Based on making an educated guess of the project parameters.

1) **Expert Judgment Technique:**
- Experts makes an educated guess of problem size after analysis
- Subjected to human error and individual bias.
- Sometimes experts have not expertise on all aspect of project

2) **Delphi Cost Estimation:** Coordinators and Experts group
- Coordinator will provide a copy of SRS to each expert and a form to record their estimation.
- Expert will estimate and submit estimation individually and anonymously.
- Coordinator will prepare a summary of responses of all estimation.
- Again, share the summary to expert to re-estimate
- Iterate the chain until the final compiled result is produced.

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.2 Heuristic Techniques

- Identify the basic project parameters.
- Create other (dependent) parameters from basic parameters.

1) **Single Variable Model**

Estimated Parameter = $C_1 * e^{d1}$ Where,
e: characteristics of software which has been already estimated

- Estimated parameter is dependent parameter to be estimated (e.g. effort, duration, staff size)
- C1 & D1 are constants values collected from past projects / historical data
- E.g. COCOMO model

2) **Multi Variable Model**

- Estimated Resources = $C_1 * ep_1^{d1} + C_2 * ep_2^{d2} + \dots$ Where,

- Ep_1, ep_2-Basic characteristics of the software already estimated
- $C_1, C_2, D_1, D_2, \dots$ are constants
- It gives more accurate estimation.
- E.g. Intermediate COCOMO model.

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.2 Heuristic Techniques

1) COnstructive COst estimation MOdel (COCOMO)

Any software development project can be classified into any one of the following three categories based on the development complexity:

- Organic
- Semidetached
- Embedded

We are categorizing the software which is to be developed in three categories:

a) Application Programs: e.g. Word processing software, accounting software etc.

b) Utility Programs: e.g. Compiler, linker etc.

c) System Softwares: e.g. real time system, operating system etc.

- The ratio of development complexity is: 1:3:9
- That means if any application software which requires say ‘X’ efforts to develop then utility software will require 3X times efforts and 9X times efforts for system softwares.

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.2 Heuristic Techniques

1) COnstructive COst estimation MOdel (COCOMO)

A) Organic: < 50 K LOC (e.g. Pay Load System)

- well-understood application
- Development team size is reasonably small
- Experienced team members

B) Semidetached: 50 K < LOC < 300 K (e.g. Compiler)

- Mixture of experienced and inexperienced development team
- Limited experience on related system development.
- Unfamiliar with some aspects of system being developed.

C) Embedded: LOC > 300 K (ATM, Air traffic control system)

- If the software being developed its strongly coupled to complex hardware, or if stringent regulations on the operation exist

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.2 Heuristic Techniques

1) COnstructive COst estimation MOdel (Basic COCOMO)

- Gives estimate about project parameter

$$\text{Effort} = a_1 \times (\text{KLOC})^{a2} \text{ PM (Person months)}$$

$$T_{dev} = b_1 \times (\text{effort})^{b2} \text{ Months}$$

Where, KLOC - estimated size of software in Kilo Lines of Code

a_1, a_2, b_1, b_2 – constants for each software product category

T_{dev} – estimated time to develop (in Months)

Efforts – efforts required to develop (in Person Months)

	Development effort estimation	Development time estimation
Organic	$2.4\ (\text{KLOC})^{1.05}$ PM	$2.5\ (\text{effort})^{0.38}$ Months
Semidetached	$3.0\ (\text{KLOC})^{1.12}$ PM	$2.5\ (\text{effort})^{0.35}$ Months
Embedded	$3.6\ (\text{KLOC})^{1.20}$ PM	$2.5\ (\text{effort})^{0.32}$ Months

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.2 Heuristic Techniques

1) COnstructive COst estimation MOdel (Basic COCOMO)

Ex: Assume that the size of an organic type of software product has been estimated to be 32,000 lines of source code. Assume that the average salary of software developers in Rs. 15,000 per month. Determine the effort required to develop the software product, the nominal development time, and the cost of develop the product.

Solution:

Efforts = $2.4 \times (32)^{1.05}$ = 91 Person Months (PM)

Nominal development time = $2.5 \times (91)^{0.38}$ = 14 Months

Cost = $91 \times 15{,}000$ = Rs. 14,65,000

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.2 Heuristic Techniques

2) Intermediate COnstructive COst estimation MOdel (Intermediate COCOMO)

- Basic COCOMO assumes that effort and development time are functions of the product size alone.
- The intermediate COCOMO model recognize all other project parameters which may affect effort and development time; It refines the initial estimate obtained using the basic COCOMO expression by using a set of 15 cost drivers (multipliers) based on various attributes of software development
- Different cost drivers:

1) Product related: complexity of the product, reliability
2) Computer related: Execution speed, storage space etc.
3) Personnel related: experience level, programming / analysis capability
4) Development environment related: automation tools (e.g. CASE tool)

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.3 Analytical Estimation Techniques : Halstead's software science

- To measure size, development effort, and development cost of software products

For a given program, Let

n_1- no. of **unique operators** used in the program

n_2- no. of **unique operands** used in the program

N_1- total **no. of operators** used in the program

N_2- total **no. of operands** used in the program

- **How to identify operators?**

Arithmetic, Logical, assignment operators

() and {}

Label of GOTO statement

If, if…then…else…endif, while, for, do ;

- **How to identify operands?**

Function name, variable name, function name in function call, arguments in function call

But remember…! Parameter list in function declaration is not considered as operands

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.3 Analytical Estimation Techniques : Halstead's software science

a) Length and vocabulary:

Length of a program (N) = $N_1 + N_2$

Program's vocabulary (n) = $n_1 + n_2$

b) Program's Volume:

- Length of a program depends on choice of operators and operands
- For the same programming problem, length of program may differ because of programming style and programming language used

$V = N \operatorname{Log}_2 n$ where

V - min. no. of bits need to encode the program

n - unique different identifiers / program's vocabulary

So, we need $N \operatorname{Log}_2 n$ bits to store a program having length N

c) Potential Minimum Volume:

- The potential minimum volume V* is defined as the volume of the most succinct program in which a problem can be coded. The minimum volume is obtained when the program can be expressed using a single source code instruction, say a function call like foo ();
- Thus if an algorithm operates on input and output data $(d_1, d_2, \dots, d_n)$ the most succinct program would be $f(d_1,d_2,\dots,d_n)$; for which $n_1=2$, $n_2=n$ therefore

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.3 Analytical Estimation Techniques : Halstead's software science

$$V^* = (2 + n_2) \log_2 (2 + n_2)$$

- Program level $L = V^*/V$
- Higher the program level → less effort to develop

d) Efforts and Time:

- Efforts need to develop a program E = V (volume of the program) / L (Level of programming lang. used)
- $E = V^2/V$ (where E is no. of mental discrimination required to implement the program and also the effort required to read and understand the program.)
- $T = E/S$ (S-speed of mental discriminations S=18 empirically psychologically decided)

e) Length Estimation:

- Halstead suggests a way to determine the length of a program using the no. of unique operators and operands used in the program.
- By this method, length, cost, volume, efforts etc. can be determined before even starting the programming.
- Any program of length N consist of N/n unique strings of n (length). Now it is standard combinational result that for any given alphabet of size K, these are exactly K^r different strings of length r. Thus

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.3 Analytical Estimation Techniques : Halstead's software science

$$N/n \leq n^n \text{ or } N \leq n^{n+1}$$

- Upper bound can be (as operators and operands may duplicates)

$$\text{So } N \leq n n_1^{n1} n_2^{n2}$$

- N must include all possible subsets of that ordered set.

$$2N = n n_1^{n1} n_2^{n2}$$

- Take log both side: $N = \log_2 n + \log_2(n_1^{n1} n_2^{n2})$
- So we get, $N = \log_2(n_1^{n1} n_2^{n2})$ (approximately, by ignoring $\log_2 N$)
- Or $N = \log_2 n_1^{n1} + \log_2 n_2^{n2} = n_1 \log_2 n_1 + n_2 \log_2 n_2$

Unit 5: Software Project Management

5.7 Project estimation Techniques

5.7.3 Analytical Estimation Techniques : Halstead's software science

Example: Find out estimated length and volume of following C program

```
main()
{
int a, b, c, avg;
scanf("%d %d %d", &a, &b, &c);
avg=(a+b+c)/3;
printf("avg=%d",avg);
}
```

Solution: Step-1: First, Find out the total number of unique operators used: 12

main, (), {}, int, scanf, &, ",", ";", = , +, /, printf

Step-2: Find out the total number of unique operands used:11

"%d %d %d",a+b+c, a, b, c, &a, &b, &c, avg=%d, avg, 3

Step-3: Therefore, here $n_1=12$, $n_2=11$

Now, estimated Length

$$(N) = n_1\log_2 n_1 + n_2\log_2 n_2$$
$$= (12*\log_2 12) + (11*\log_2 11)$$
$$= 43+38$$
$$= 81$$

$$\text{Volume}(V) = N * \text{Log}_2\, n$$
$$= 81 * \log_2(23)$$
$$= 81 * 4.52$$
$$= 366$$

Unit 5: Software Project Management

5.8 Scheduling

- Scheduling defines that which task should be taken up when
- How to schedule project activities:

1) Identify all tasks needed to complete the project.
2) Breakdown large task into small activities.
3) Determine the dependency among different activities
4) Establish the most likely estimates for the time duration necessary to complete the activities
5) Allocate resources to activities
6) Plan the starting and ending dates for various activities
7) Determine the critical path: A **critical path** is a chain of activities that determines the duration of project.

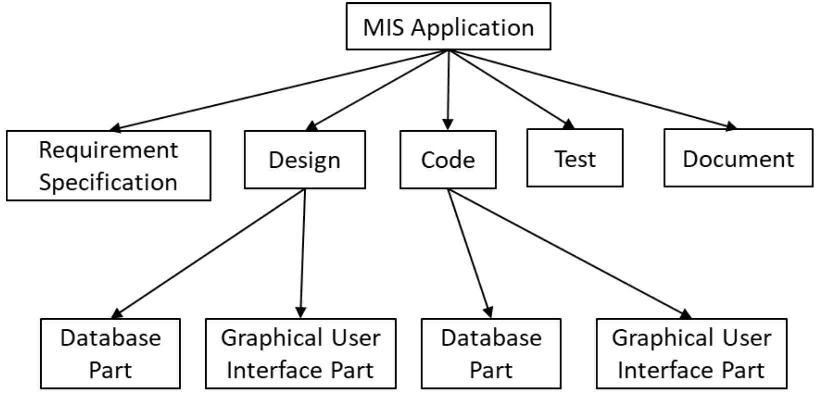
The end of each activity is called **Milestone**

Unit 5: Software Project Management

5.8 Scheduling

5.8.1 Work Breakdown Structure

- Work breakdown structure (WBS) is used to recursively decompose a given set of activities into smaller activities.
- First, let us understand why it Is necessary to break down project activities into tasks.
- Once project Activities have been decomposed into a set of Tasks using WBS, the time frame when each activity is to be performed is to be determined. The end of each important activity is called a milestone. The project manager tracks the progress of a project by monitoring the timely completion of the milestones. If he observes that some milestones start getting delayed, he carefully monitors and controls the progress of the tasks, so that the overall deadline can still be met.


MIS Application
Requirement Specification
Design
Code
Test
Document
Database Part
Graphical User Interface Part
Database Part
Graphical User Interface Part

[Figure 5.1 MIS Problem]

Unit 5: Software Project Management

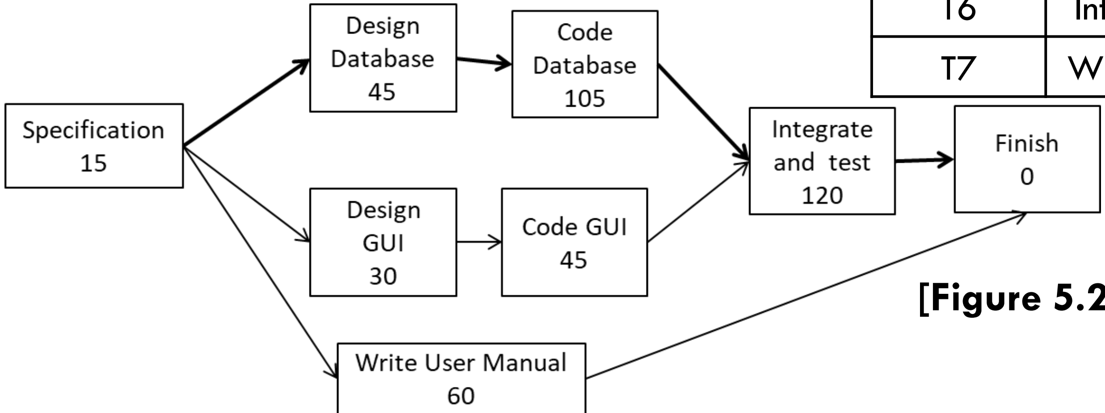
5.8 Scheduling

5.8.2 Activity Networks:

Minimum time: the maximum of all paths from start to finish

So, Here its (15 + 45 + 105 + 120) = **285**

Task No.	Task	Duration	Dependency
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 & T5
T7	Write User Manual	60	T1



[Figure 5.2 Activity Network]

Unit 5: Software Project Management

5.8 Scheduling

5.8.2 Critical Path Method (CPM):

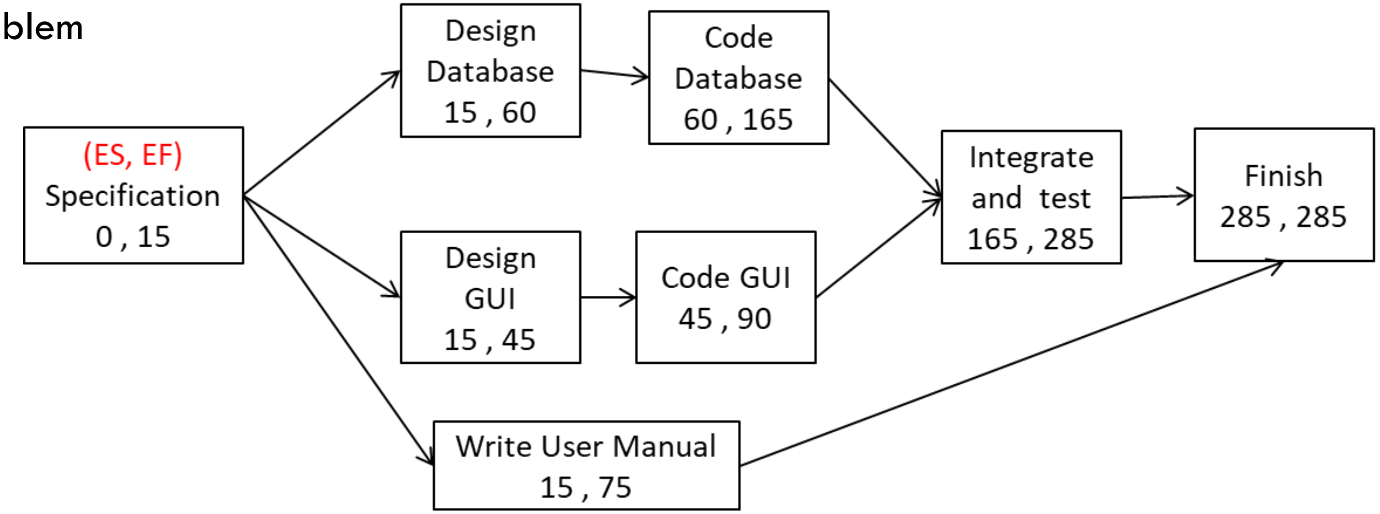
- A ***path*** in the activity network graph is any set of consecutive nodes and edges in this graph from the starting node to the last node.
- A ***critical path*** consists of a set of dependent tasks that need to be performed in a sequence and which together take the longest time to complete.
- A ***Critical task*** is one with a zero slack time. A path from the start node to the finish node containing only critical tasks is called a critical path.
- **Minimum time (MT):** It is the minimum time required to complete the project. It is computed by determining the maximum of all paths from start to finish.
- **Earliest start (ES):** It is the time of a task is the maximum of all paths from the start to this task. The ES for a task is the ES of the previous task plus the duration of the preceding task.
- **Latest start time (LST):** It is the difference between MT and the maximum of all paths from this task to the finish. The LST can be computed by subtracting the duration of the subsequent task from the LST of the subsequent task.
- **Earliest finish time (EF):** The EF for a task is the sum of the earliest start time of the task and the duration of the task.
- **Latest finish (LF):** LF indicates the latest time by which a task can finish without affecting the final completion time of the project. A task completing beyond its LF would cause project delay. LF of a task can be obtained by subtracting maximum of all paths from this task to finish from MT.
- **Slack time (ST):** The slack time (or float time) is the total time that a task may be delayed before it will affect the end time of the project. The slack time indicates the ”flexibility” in starting and completion of tasks. ST for a task is LS-ES and can equivalently be written as LF-EF.

Unit 5: Software Project Management

5.8 Scheduling

5.8.2 Critical Path Method (CPM):

- **Example:** Use the Activity network of Figure 5.2 to determine the ES and EF for every task for the MIS problem
- **Solution:**



[Figure 5.3 Finding Early Start (ES) and Early Finish (ES)]

Unit 5: Software Project Management

5.8 Scheduling

5.8.2 Critical Path Method (CPM):

Example: Use the same Activity network of Figure 5.3 to determine the LS, LF and ST for every task for the MIS problem (Remember that our **Minimum Time (MT)** was **285**)

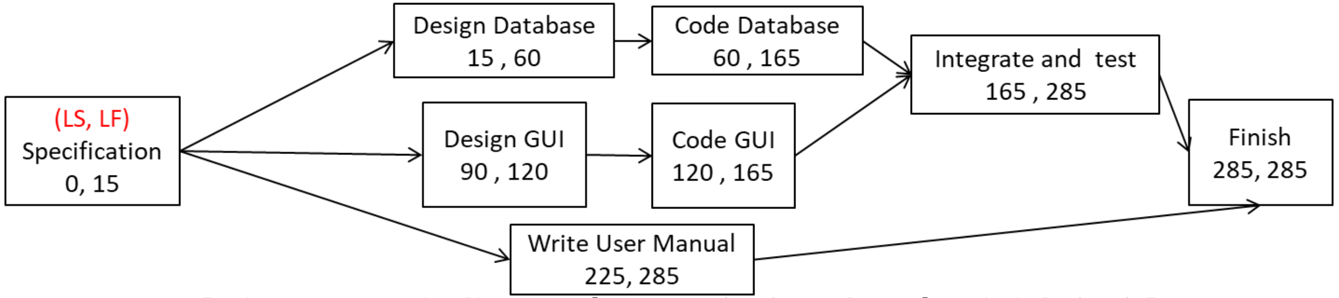
What is Latest Start (LS)? **Difference between MT and max of all paths from this task to finish**

What is Latest Finish (LF)? **MT – (max of all paths from this task to finish)**

Task	ES	EF	LS	LF	ST
Specification	0	15			
Design Database	15	60			
Design GUI	15	45			
Code Database	60	165			
Code GUI Part	45	90			
Integrate and Test	165	285			
Write User Manual	15	75			

Unit 5: Software Project Management

5.8 Scheduling



[Figure 5.3 Finding Early Start (ES) and Early Finish (ES)]

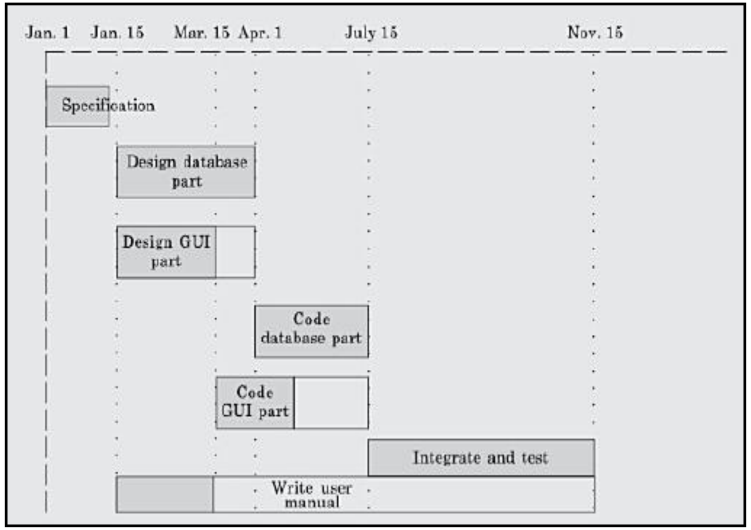
Task	ES	EF	LS	LF	ST (LF-EF)
Specification	0	15	Difference (285,285) = 0	285-(45+105+120)=15	15-15=0
Design Database	15	60	Difference (285,(45+105+120)) = 15	285-(105+120)=60	60-60=0
Design GUI	15	45	Difference (285, (30+45+120)) = 90	285-(45+120)=120	120-45=75
Code Database	60	165	Difference (285,(105+120)) =60	285-(120)=165	165-165=0
Code GUI Part	45	90	Difference (285, (45+120))=120	285-(120)=165	165-90=75
Integrate and Test	165	285	Difference (285, 120) = 165	285-(0)=285	285-285=0
Write User Manual	15	75	Difference (285, 60) = 225	285-(0)=285	285-75=210

Unit 5: Software Project Management

5.8 Scheduling

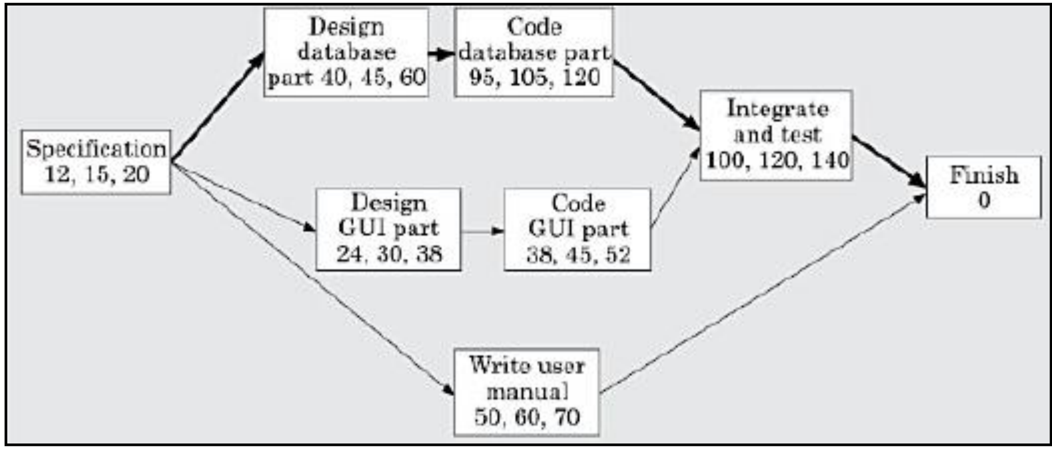
Gantt Chart

- Used to allocate resources to activities
- Shaded part shows length of time where as white bar shows slack time



Pert Chart

- Project Evaluation and Review Technique chart
- It represents statistical variation in the project estimates assuming a normal distribution



Specification 12, 15, 20
Design database part 40, 45, 60
Code database part 95, 105, 120
Design GUI part 24, 30, 38
Code GUI part 38, 45, 52
Write user manual 50, 60, 70
Integrate and test 100, 120, 140
Finish 0