Software Project Management

Project planning

- Once a project is found to be feasible, software project managers undertake project planning.
- Project planning is undertaken and completed even before any development activity starts.

Essential activities in Project Planning:

1) Estimating the following attributes of the project:

- **Project size:** What will be problem complexity in terms of the effort and time required to develop the product?
- **Cost:** How much is it going to cost to develop the project?
- **Duration:** How long is it going to take to complete development?
- **Effort:** How much effort would be required?

➢ The effectiveness of the subsequent planning activities is based on the accuracy of these estimations.

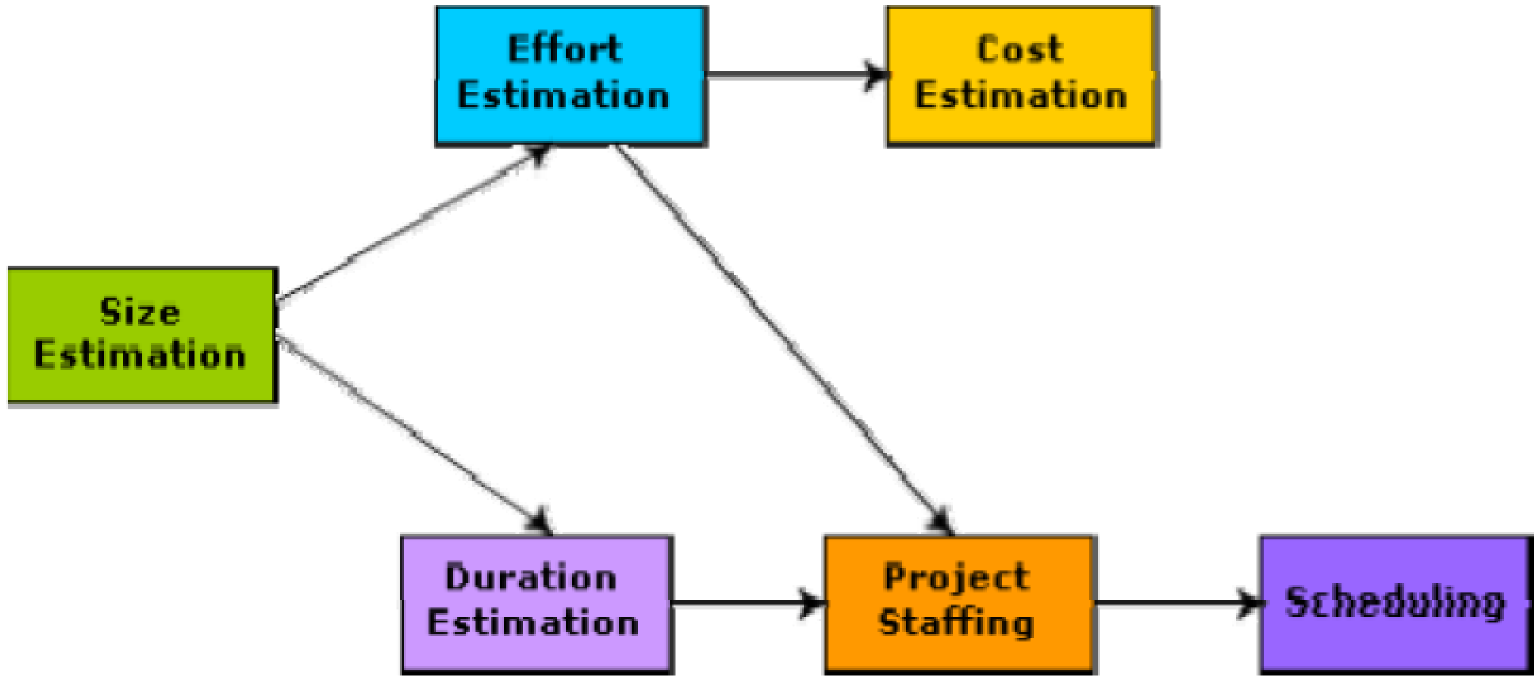
2) Scheduling manpower and other resources

3) Staff organization and staffing plans

4) Risk identification

5) Miscellaneous plans such as quality assurance plan, configuration management plan, etc.

Precedence ordering among project planning activities



Problem in Project Planning

- Commitment to unrealistic time and resource estimates result in schedule slippage.
- Schedule delays can cause customer dissatisfaction and adversely affect team morale.
- However, project planning is a very challenging activity. Especially for large projects, it is very much difficult to make accurate plans.
- A part of this difficulty is due to the fact that the proper parameters, scope of the project, project staff, etc. may change during the span of the project.

Solution: Sliding Window Planning

- Planning a project over a number of stages protects managers from making big commitments too early.
- This technique of staggered planning is known as Sliding Window Planning.
- Starting with an initial plan, the project is planned more accurately in successive development stages.
- After the completion of every phase, the project managers can plan each subsequent phase more accurately and with increasing levels of confidence.

Responsibilities of a software project manager

- Overall responsibility of steering a project to success.
- The job responsibility of a project manager ranges from invisible activities like building up team morale to highly visible customer presentations.
- Activities can be broadly classified into project planning, and project monitoring and control activities.
- It includes:

  project proposal writing, project cost estimation, scheduling, project staffing, software process tailoring, project monitoring and control, software configuration management, risk management, interfacing with clients, managerial report writing and presentations, etc.

Skills necessary for software project management

- Theoretical knowledge of different project management techniques like cost estimation, risk management, configuration management
- Good qualitative judgment and decision taking capabilities.
- Good communication skills and the ability get work done.

Software Project Management Plan (SPMP)

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

Software Project Management Plan (SPMP)

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

Project Estimation techniques

- Empirical estimation techniques
- Heuristic techniques
- Analytical estimation techniques

Empirical Estimation Techniques

- Based on making an **educated guess** of the **project parameters.**
- **Prior experience** with development of similar products is helpful.
- Although empirical estimation techniques are based on **common sense**, different activities involved in estimation have been formalized over the years.

Two popular empirical estimation techniques are:

1) Expert judgment technique
2) Delphi cost estimation

Expert Judgment Technique

- An expert makes an educated guess of the **problem size** after analyzing the problem thoroughly.
- Estimates the **cost** of the different components (i.e. modules or subsystems) of the system and then combines them to arrive at the overall estimate.
- However, this technique is subject to **human errors and individual bias.**
- For example, he may be conversant with the database and user interface parts but may not be very knowledgeable about the computer communication part.
- A more refined form of expert judgment is the estimation made by **group of experts**.

Delphi cost estimation

- It overcomes some of the shortcomings of the expert judgment approach.
- Delphi estimation is carried out by a team comprising of a **group of experts and a coordinator.**
- No discussion among the estimators is allowed during the entire estimation process.

Delphi cost estimation

- The coordinator provides each estimator with a copy of the SRS document and a form for recording his cost estimate.
- Estimators complete their **individual estimates** anonymously and submit to the coordinator.
- The coordinator prepares and distributes the summary of the responses of all the estimators, and includes any unusual rationale noted by any of the estimators.
- Based on this summary, the **estimators re-estimate.**
- This process is iterated for several rounds.

Heuristic Techniques

- It assumes that the **relationships among the different project parameters** can be modelled using **suitable mathematical expressions**.
- Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting the value of the basic parameters in the mathematical expression.
- Two classes:

  1) single variable model and
  2) multi variable model

Single variable estimation models

- It provides a means to estimate the desired characteristics of a problem, using some previously estimated basic (independent) characteristic of the software product such as its size.

$$\text{Estimated Parameter} = c_1 * e^{d_1}$$

- e is the characteristic of the software which has already been estimated (independent variable)
- c1 and d1 are constants determined using data collected from past projects (historical data).

Multivariable cost estimation model

$$\text{Estimated Resource} = c_1 * e_1^{d_1} + c_2 * e_2^{d_2} + \ ...$$

- Where e1, e2, … are the basic (independent) characteristics of the software already estimated, and c1, c2, d1, d2, … are constants.
- Multivariable estimation models are expected to give more accurate estimates.

Project Types

Organic:

Routine project

- Well understood domain
- Team works well and efficiently together
- Project expected to run smoothly
- Typically a smaller system and smaller team

Embedded:

- Difficulties expected
- Project that is hard (control software for a nuclear plant, or spacecraft)
- Team has little experience in domain
- New or inexperienced team
- Tend to be large projects with lots of constraints

Semidetached:

- In the middle
- Complex system, but something the company is familiar with
- Teams may be made up of experienced and inexperienced members
- System not huge, but not small either

Person-Month (PM)

1. It is popular unit for effort measurement

2. It is considered to be an appropriate unit of measuring effort because developers are typically assigned to a project for certain number of months

Basic COCOMO Model

- COnstructive COst MOdel
- The basic COCOMO model gives an approximate estimate of the project parameters.

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

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

Estimation of development effort

Organic :	$\textbf{Effort} = 2.4(KLOC)^{1.05}$ PM
Semi-detached :	$\textbf{Effort} = 3.0(KLOC)^{1.12}$ PM
Embedded :	$\textbf{Effort} = 3.6(KLOC)^{1.20}$ PM

Estimation of development time

Organic	:	$\mathbf{Tdev = 2.5(\mathit{Effort})^{0.38}}$	**Months**
Semi-detached	:	$\mathbf{Tdev = 2.5(\mathit{Effort})^{0.35}}$	**Months**
Embedded	:	$\mathbf{Tdev = 2.5(\mathit{Effort})^{0.32}}$	**Months**

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 and the nominal development time and cost to develop the product.

From the basic COCOMO estimation formula for organic software:

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

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

Cost required to develop the product = **14 x 15,000**

= **Rs. 210,000/-**

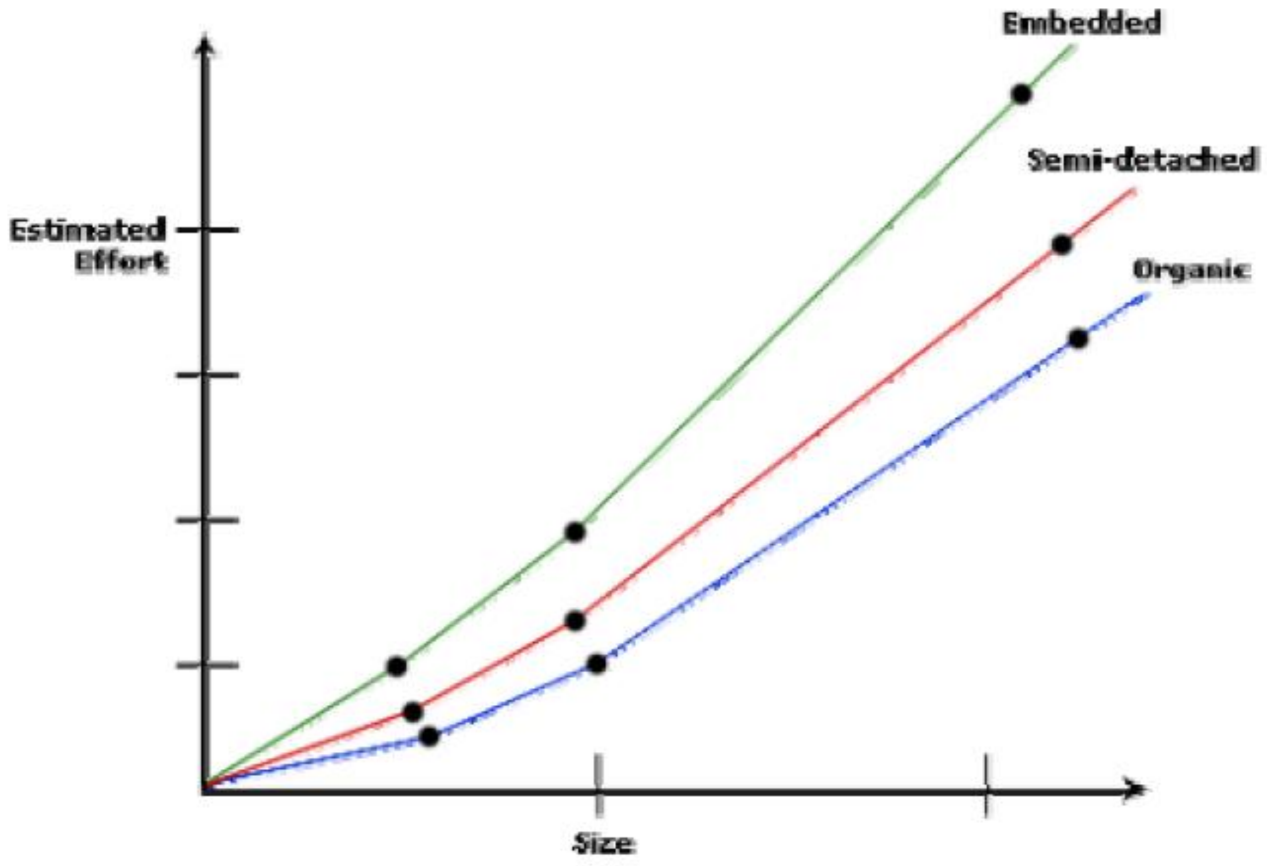


Fig. 11.4: Effort versus product size

Intermediate COCOMO model

- The basic COCOMO model assumes that effort and development time are **functions of the product size** alone.
- However, a host of other project parameters besides the product size affect the effort required to develop the product as well as the development time.
- Therefore, in order to obtain **an accurate estimation** of the effort and project duration, the **effect of all relevant parameters** must be taken into account.

$$E = a(KLOC)^{b} \underbrace{\times C}_{\text{New}}$$

- Where
  - E is the effort
  - a and b are constants (as before)
  - KLOC is thousands of lines of code
  - C is the effort adjustment factor

15 Cost Drivers

- Boehm requires the project manager to **rate these 15 different parameters** for a particular project on a scale of one to three.
- Then, depending on these ratings, he suggests **appropriate cost driver values** which should be multiplied with the initial estimate obtained using the basic COCOMO.

Classification of Cost Drivers

Product: The characteristics of the product that are considered include the inherent complexity of the product, reliability requirements of the product, etc.

Computer: Characteristics of the computer that are considered include the execution speed required, storage space required etc.

Personnel: The attributes of development personnel that are considered include the experience level of personnel, programming capability, analysis capability, etc.

Development Environment: Development environment attributes capture the development facilities available to the developers. An important parameter that is considered is the sophistication of the automation (CASE) tools used for software development.

Project Characteristics Table

Cost adjustments for computing the EAF (Effort Adjustment Factor)

	v.low	low	nominal	high	v.high	ex.high
product attributes						
required software reliability	0.75	0.88	1.00	1.15	1.40	
database size		0.94	1.00	1.08	1.16	
product complexity	0.70	0.85	1.00	1.15	1.30	1.65
computer attributes						
execution time constraints			1.00	1.11	1.30	1.66
main storage constraints			1.00	1.06	1.21	1.56
virtual machine volitility	0.87	1.00	1.15	1.30		
computer turnaround time		0.87	1.00	1.07	1.15	
personnel attributes						
analyst capability	1.46	1.19	1.00	0.86	0.71	
applications experience	1.29	1.13	1.00	0.91	0.82	
programmer capability	1.42	1.17	1.00	0.86	0.70	
virtual machine experience	1.21	1.10	1.00	0.90		
programming language experience	1.14	1.07	1.00	0.95		
project attributes						
use of modern programming practices	1.24	1.10	1.00	0.91	0.82	
use of software tools	1.24	1.10	1.00	0.91	0.83	
required development schedule	1.23	1.08	1.00	1.04	1.10	

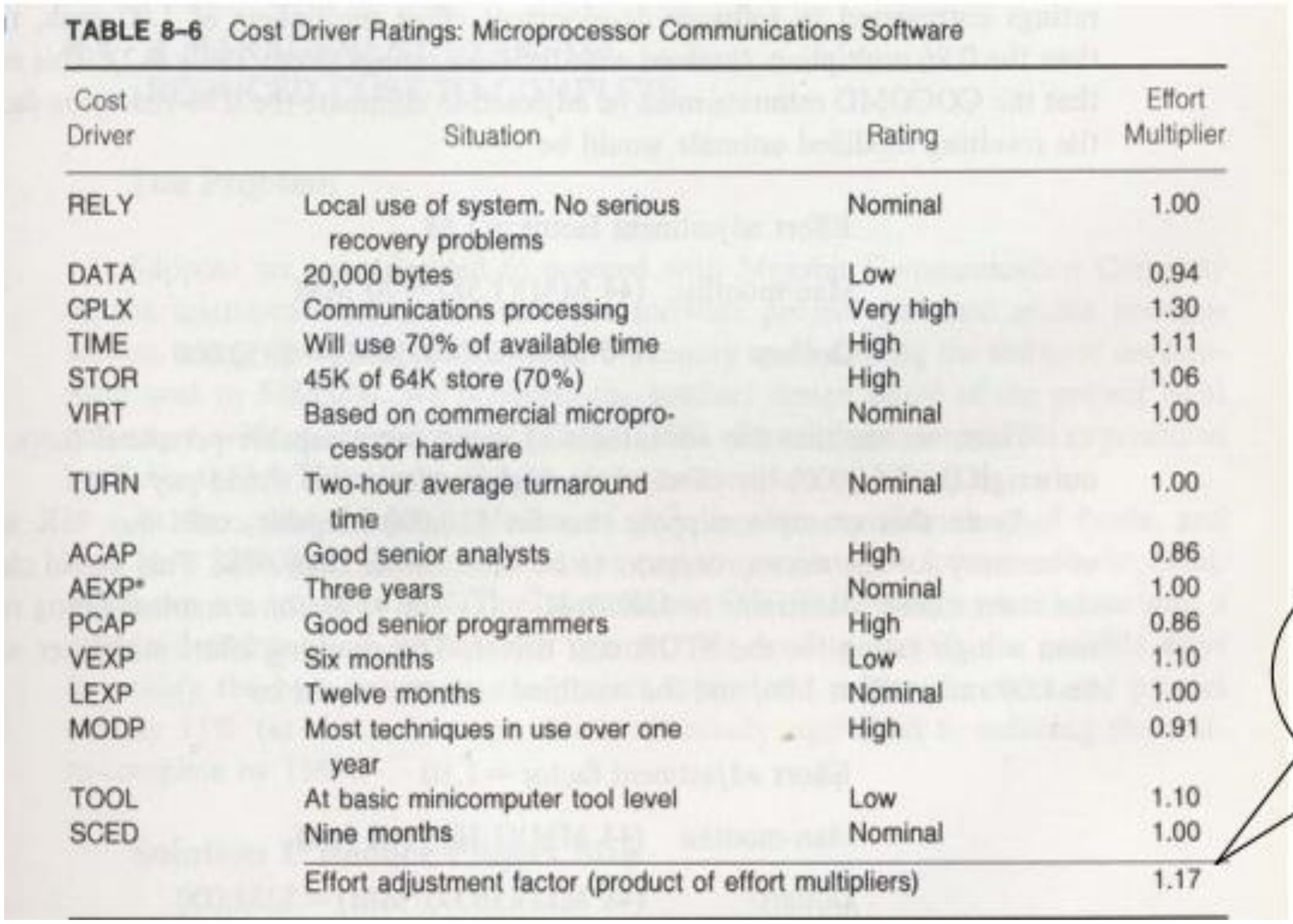


TABLE 8–6 Cost Driver Ratings: Microprocessor Communications Software

Cost Driver	Situation	Rating	Effort Multiplier
RELY	Local use of system. No serious recovery problems	Nominal	1.00
DATA	20,000 bytes	Low	0.94
CPLX	Communications processing	Very high	1.30
TIME	Will use 70% of available time	High	1.11
STOR	45K of 64K store (70%)	High	1.06
VIRT	Based on commercial microprocessor hardware	Nominal	1.00
TURN	Two-hour average turnaround time	Nominal	1.00
ACAP	Good senior analysts	High	0.86
AEXP*	Three years	Nominal	1.00
PCAP	Good senior programmers	High	0.86
VEXP	Six months	Low	1.10
LEXP	Twelve months	Nominal	1.00
MODP	Most techniques in use over one year	High	0.91
TOOL	At basic minicomputer tool level	Low	1.10
SCED	Nine months	Nominal	1.00
	Effort adjustment factor (product of effort multipliers)		1.17

1.17

Complete COCOMO model

- A major shortcoming of both the basic and intermediate COCOMO models is that they consider a software product as a **single homogeneous entity.**
- However, most large systems are made up several smaller sub-systems.
- These subsystems may have widely different characteristics.
- For example, some subsystems may be considered as organic type, some semidetached, and some embedded.

Example

- 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
- Of these, the communication part can be considered as embedded software.
- The database part could be semi-detached software, and the GUI part organic software.

Project scheduling

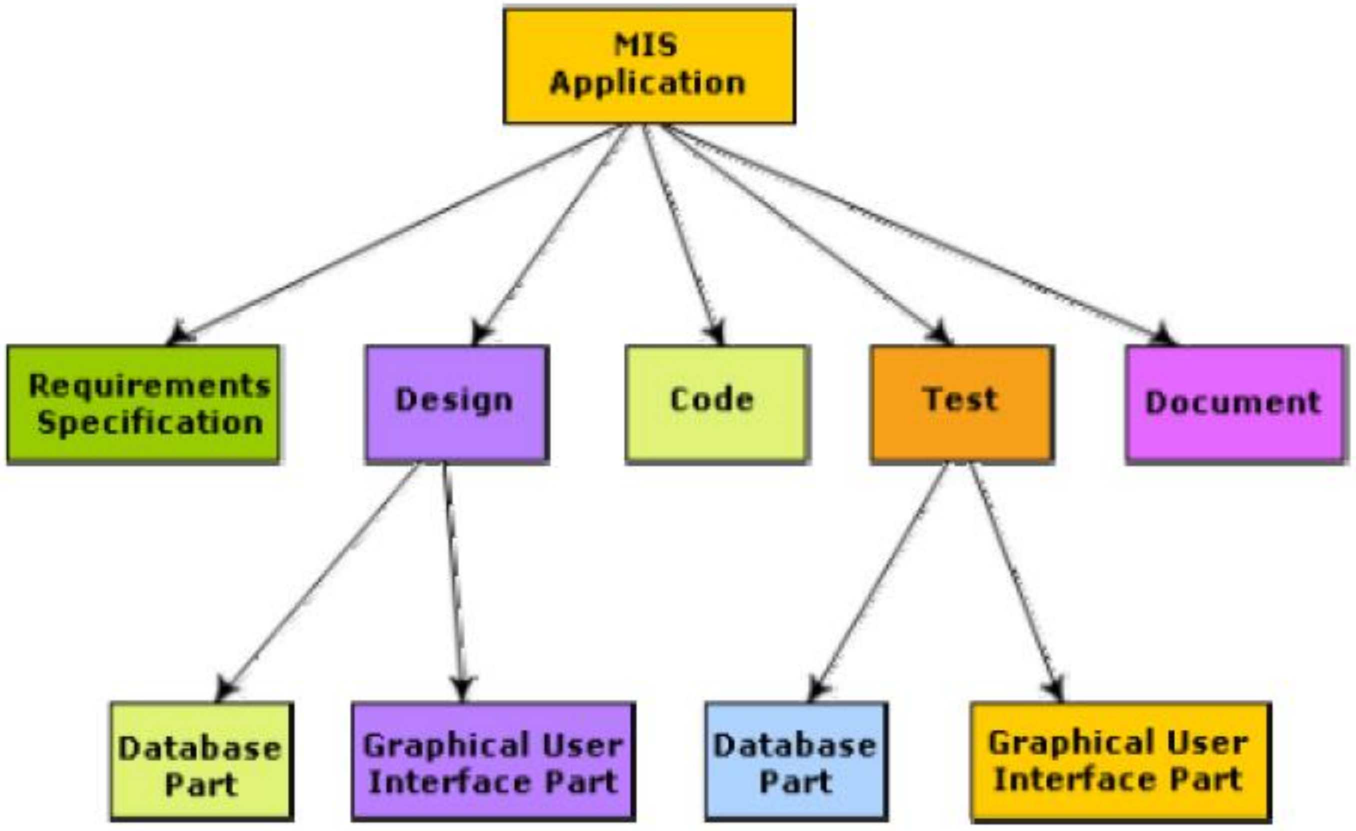
It involves deciding which tasks would be taken up when. Activities undertaken:

1. Identify all the tasks needed to complete the project.
2. Break down large tasks into small activities.
3. Determine the dependency among different activities.
4. Establish the most likely estimates for the time durations 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 the chain of activities that determines the duration of the project.

Work breakdown structure

- Work Breakdown Structure (WBS) is used to decompose a given task set recursively into small activities.
- The root of the tree is labeled by the **problem name.**
- Each node of the tree is broken down into smaller activities that are made the children of the node.
- Each activity is recursively decomposed into smaller sub-activities until at the **leaf level**, the activities requires approximately **two weeks to develop.**

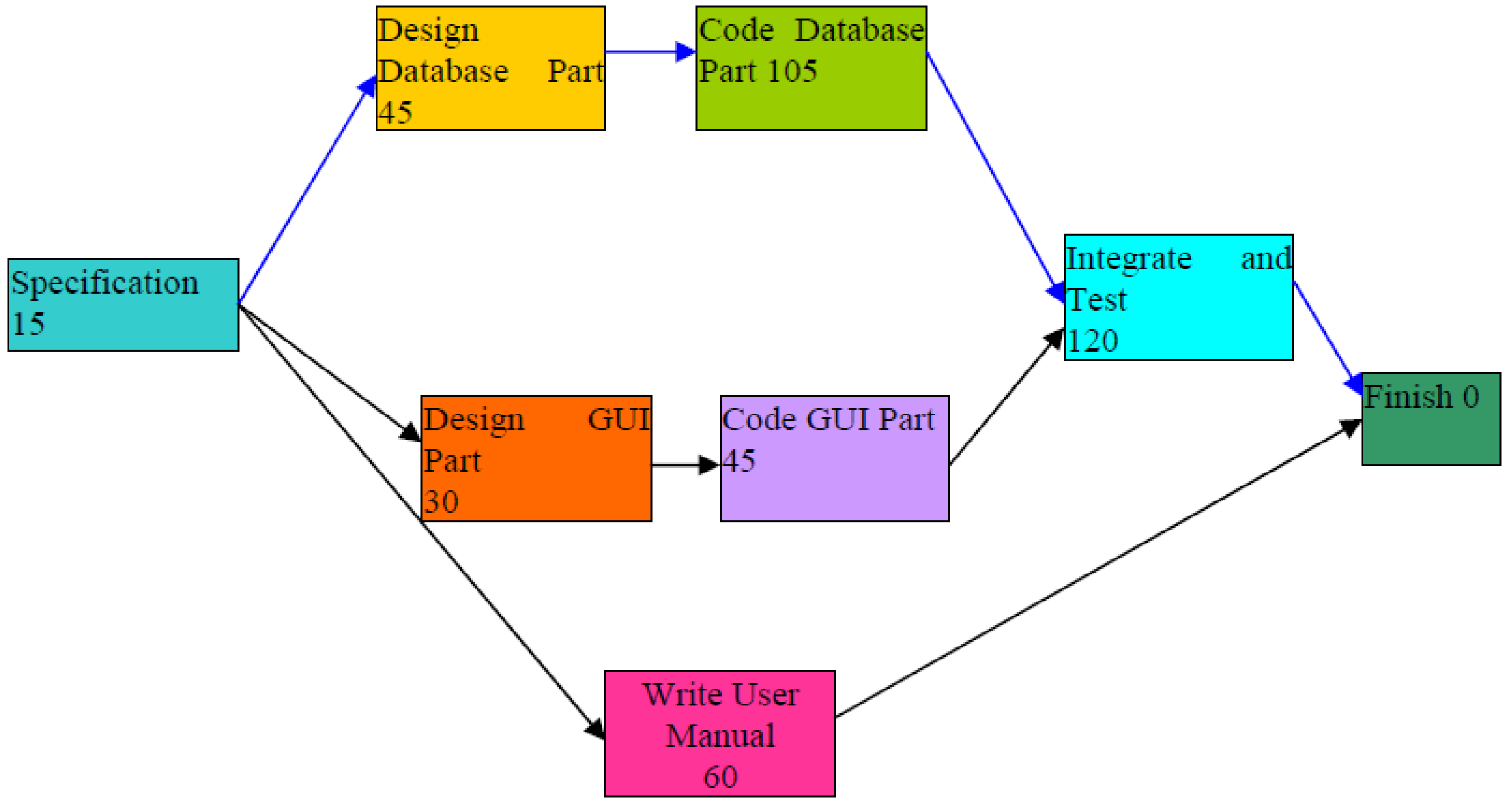
Work breakdown structure of MIS problem


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

Activity networks and critical path method

- WBS representation of a project is transformed into an activity network by representing activities identified in WBS along with their interdependencies.
- An activity network shows the different **activities** making up a project, their estimated **durations**, and **interdependencies.**

Activity network representation of the MIS problem



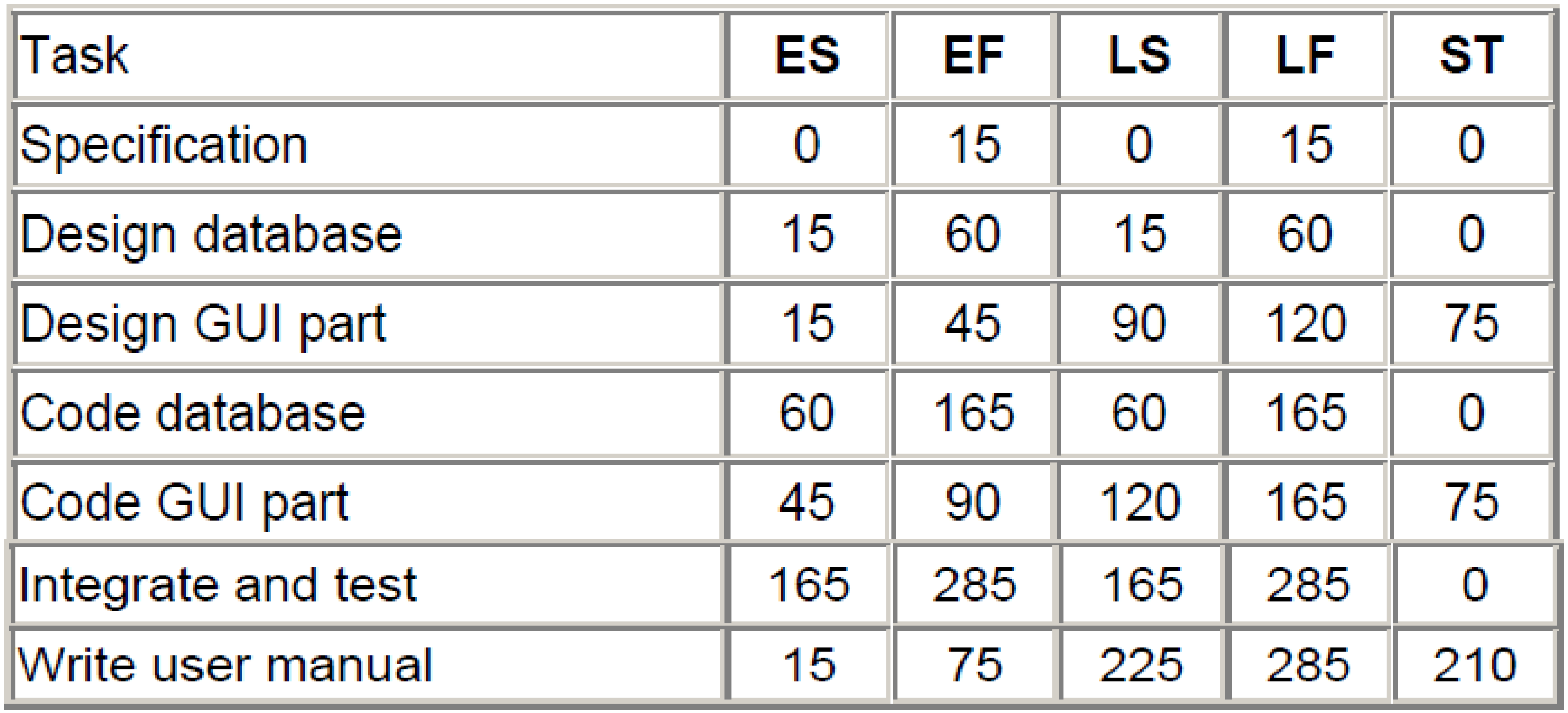
Critical Path Method (CPM)

- The **minimum time (MT)** to complete the project is the maximum of all paths from start to finish.
- The **earliest start (ES)** time of a task is the maximum of all paths from the start to the task.
- The **latest start time** is the difference between MT and the maximum of all paths from this task to the finish.
- The **earliest finish time (EF)** of a task is the sum of the earliest start time of the task and the duration of the task.
- The **latest finish (LF) time** of a task can be obtained by subtracting maximum of all paths from this task to finish from MT.
- The **slack time (ST)** is LS – ES and equivalently can be written as LF – EF.

Slack Time

- 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.
- 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.**

Example: 3.9 and 3.10 Parameters for different tasks of MIS Problem



Task	ES	EF	LS	LF	ST
Specification	0	15	0	15	0
Design database	15	60	15	60	0
Design GUI part	15	45	90	120	75
Code database	60	165	60	165	0
Code GUI part	45	90	120	165	75
Integrate and test	165	285	165	285	0
Write user manual	15	75	225	285	210

Gantt chart

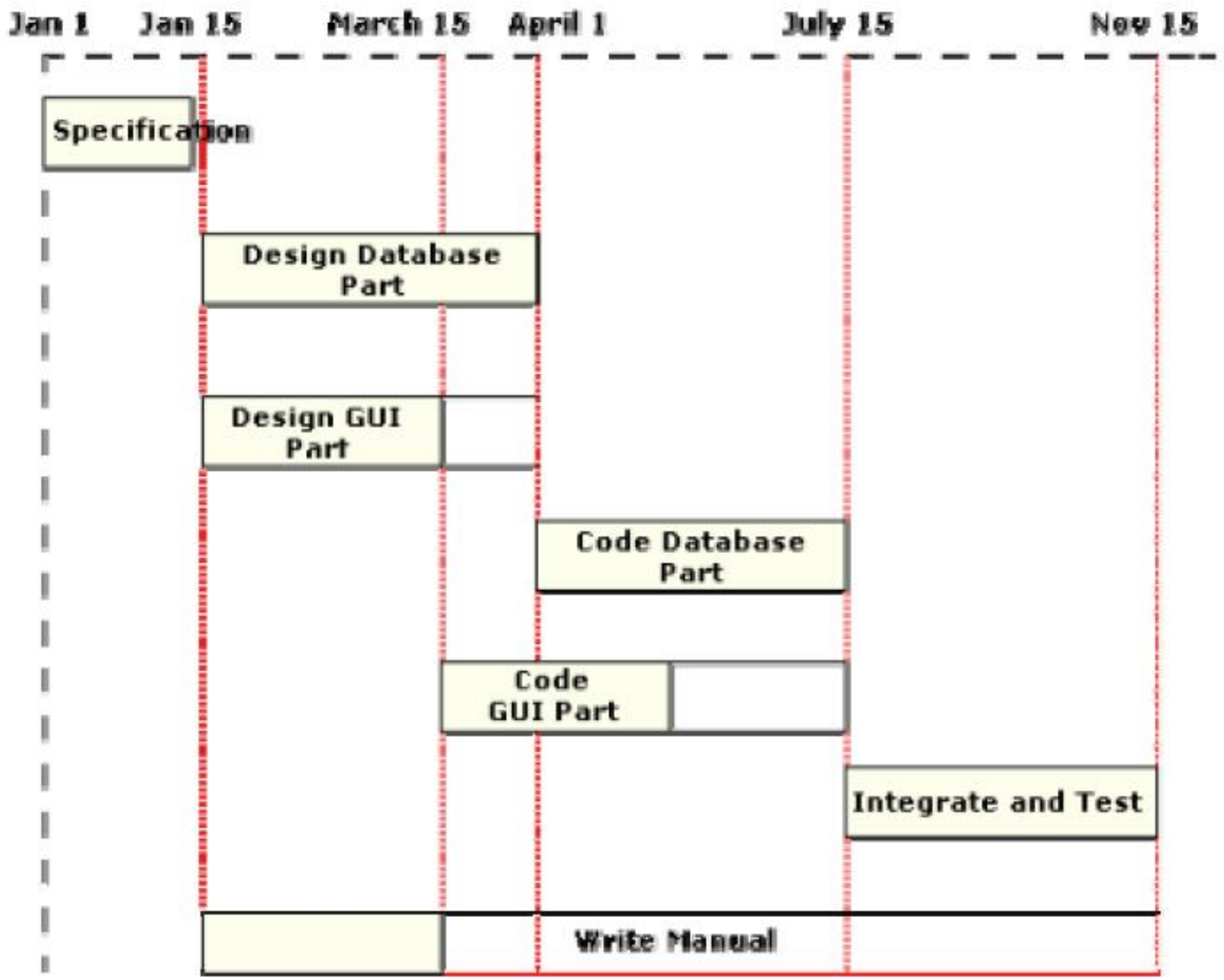
- Gantt Chart represents a project schedule which is helpful in allocating resources to activities (resource planning).

  The resources allocated to activities include staff, hardware, and software.
- A Gantt chart is a special type of bar chart where each bar represents an activity.

➢ The bars are drawn along a time line.

➢ The length of each bar is proportional to the duration of time planned for the corresponding activity.

Gantt chart for MIS problem

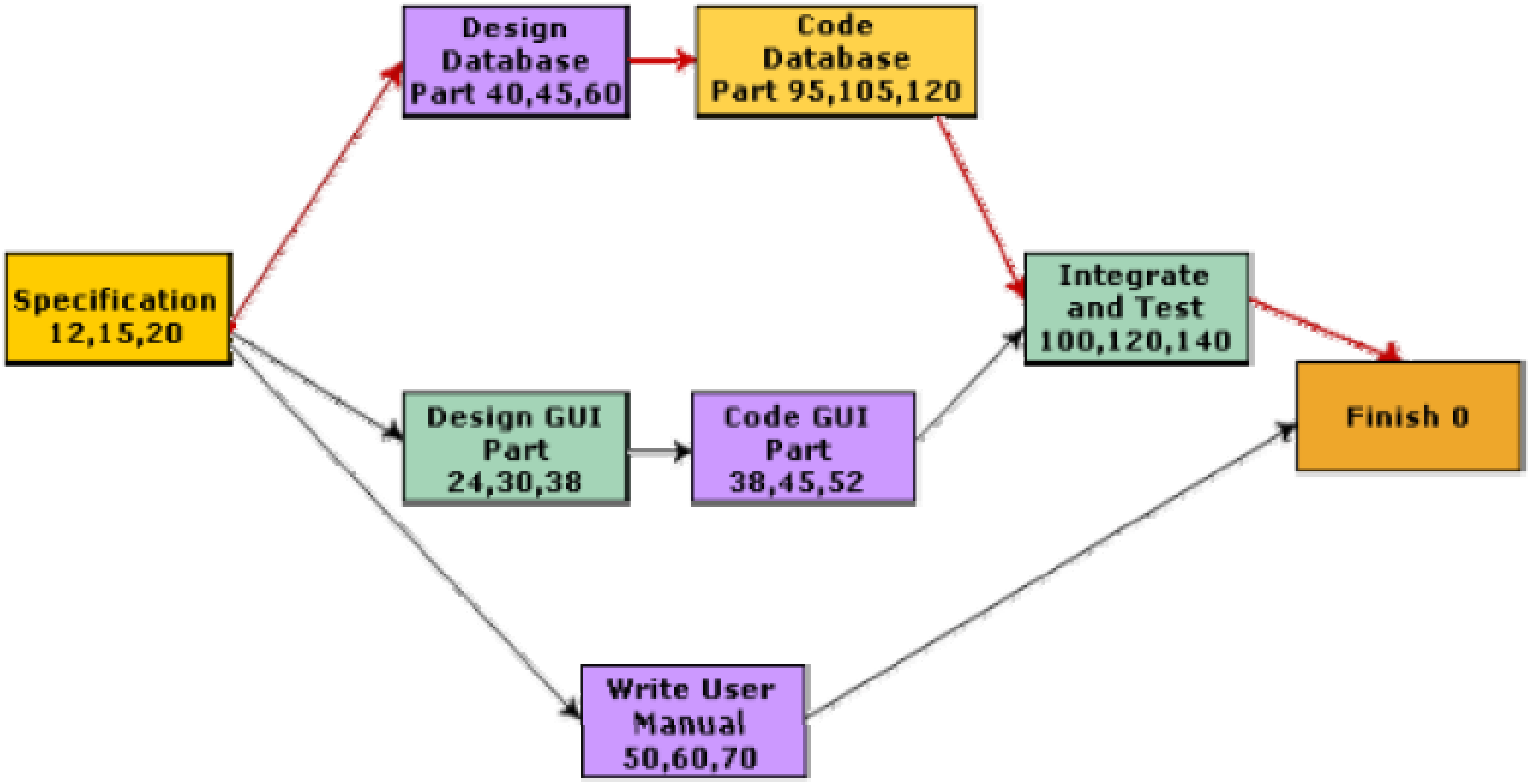


PERT chart

- PERT (Project Evaluation and Review Technique) charts consist of a network of boxes and arrows.
- The boxes represent activities and the arrows represent task dependencies.
- The boxes of PERT charts are usually annotated with the worst , likely, and optimistic estimates for every task.

- Gantt chart representation of a project schedule is helpful in planning the utilization of resources, while PERT chart is useful for monitoring the timely progress of activities.
- Also, it is easier to identify parallel activities in a project using a PERT chart.

PERT chart representation of the MIS problem


Design Database Part 40,45,60
Code Database Part 95,105,120
Specification 12,15,20
Integrate and Test 100,120,140
Design GUI Part 24,30,38
Code GUI Part 38,45,52
Finish 0
Write User Manual 50,60,70

PERT chart

- **Optimistic (O) :** best possible case task completion time
- **Most likely estimate (M):** most likely task completion time
- **Worst case (W):** worst possible case task completion time