

Software Engineering(SE)

CSC 601



Subject Incharge

Varsha Nagpurkar

Assistant Professor

Room No. 407

email: varshanagpurkar@sfit.ac.in



Module-3

Project Scheduling and Tracking

6/1/2021

Module-3 Syllabus

3.0	Project Scheduling and Tracking	08
3.1	Management Spectrum, 3Ps (people, product and process)	
3.2	Process and Project metrics	
3.3	Software Project Estimation: LOC, FP, Empirical Estimation Models • COCOMO II Model, Specialized Estimation Techniques	
3.4	Project scheduling: Defining a Task Set for the Software Project, Timeline charts, Tracking the Schedule, Earned Value Analysis	

THE MANAGEMENT SPECTRUM

- Effective software project management focuses on the four P's: people, product, process, and project.
- The People
- Every organization needs to continually improve its ability to attract, develop, motivate, organize, and retain the workforce needed to accomplish its strategic business objectives”.
- The people management capability maturity(PM-CMM) model developed by Software Engineering Institute(SEI) defines the following key practice areas for software people:

THE MANAGEMENT SPECTRUM-The People

- Recruiting, selection, performance management, training, compensation, career development, organization and work design, and team/culture development
- Organizations that achieve high levels of maturity in the people management area have a higher likelihood of implementing effective software project management practices.
- The People-CMM is a companion to the Software Capability Maturity Model—Integration that guides organizations in the creation of a mature software process

THE MANAGEMENT SPECTRUM-The People-Stakeholders

- The software process(and every software project)is populated by stakeholders who can be categorized into one of five categories:
 - 1.Senior managers- who define the business issues that often have a significant influence on the project.
 2. Project (technical) managers- who must plan, motivate, organize, and control the practitioners who do software work

THE MANAGEMENT SPECTRUM-The People-Stakeholders

3. Practitioners- who deliver the technical skills that are necessary to engineer a product or application.
4. Customers-who specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
5. End users- who interact with the software once it is released for production use.

THE MANAGEMENT SPECTRUM-The People-Team Leaders

- In an excellent book of technical leadership, Jerry Weinberg suggests a MOI Model of leadership
- 1. Motivation- The ability to encourage (by “push or pull”) technical people to produce to their best ability.
- 2. Organization.- The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product

THE MANAGEMENT SPECTRUM-The People-Team Leaders

- 3. Ideas or innovation.The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

Characteristics that define effective project management

- Problem solving-Project manager can diagnose the technical and organizational issues that are most relevant, systematically structure a solution or properly motivate other practitioners to develop the solution
- Managerial identity- A good project manager must take charge of the project. She must have the confidence to assume control when necessary and the assurance to allow good technical people to follow their instincts.

Characteristics that define effective project management

- Achievement-To optimize the productivity of a project team,a manager must reward initiative and accomplishment
- Influence and team building-An effective project manager must be able to “read” people;she must be able to understand verbal and nonverbal signals and react to the needs of the people sending these signals

The Software Team

- The “best” team structure depends on the management style of the organization, the number of people who will populate the team and their skill levels, and the overall problem difficulty
- Marilyn Mantei from The University of Michigan suggested following 7 factors for considerations before structuring a team

The Software Team

- The difficulty of the problem to be solved
- The “size” of the resultant program(s) in lines of code or function points
- The time that the team will stay together(team lifetime)
- The degree to which the problem can be modularized
- The required quality and reliability of the system to be built

The Software Team

- The rigidity of the delivery date
- The degree of communication required for the project

The Software Team-

To achieve a high-performance team:

- Team members must have trust in one another
- The distribution of skills must be appropriate to the problem
- Independent-minded person may have to be excluded from the team,if team unity is to be maintained
 - Regardless of team organization,the objective for every project manager is to help create a team that exhibits unity among the team members

Agile teams

- In recent years, agile software development has been proposed
- The agile philosophy provides the following
 - It encourages customer satisfaction by early incremental delivery of software
 - It provides small highly motivated project teams
 - It consists of informal methods
 - It provides minimal software engineering work products
 - It results in overall development simplicity

Agile teams

- The small sized and highly motivated project team, also called an agile team, adopts many of the characteristics of successful software project teams.

Coordination and Communication Issues

- There are many reasons that software projects get into trouble
- The scale of many development efforts is large, leading to complexity, confusion, and significant difficulties in coordinating team members
- Uncertainty is common, resulting in a continuing stream of changes
- Interoperability-New software must communicate with existing software and conform to predefined constraints imposed by the system

Coordination and Communication Issues

- Characteristics of modern software-Scale,uncertainty, and interoperability-are facts of life
- To deal with them effectively,a Software Engineering team must establish effective methods for coordinating the people who do the work
- To accomplish this mechanisms for formal and informal communication among team members and between multiple teams must be established

Coordination and Communication Issues

- Formal communication is accomplished through “writing”, structured meetings, and other relatively noninteractive and impersonal communication channels
- Informal communication is more personal-Members of a team share ideas on an adhoc basis, ask for help as problem arise, and interact with one another on a daily basis

The Product

- As requirements may change regularly as the project proceeds,a plan is needed
- So the first activity in software project management is the determination of software scope
- Scope is defined by answering the following questions:
 - Context-How does the software to be built fit into a larger system,product,or business context, and what constraints are imposed as a result of the context?

The Product

- Information objectives-What customer-visible data objects are produced as output from the software? What data objects are required for input?
- Function and performance-What functions does the software perform to transform input data into output?

The Product

- Software project scope must be unambiguous and understandable at the management and technical levels
- Software scope must be bounded-
 - Quantitative data(e.g.,number of simultaneous users,size of mailing list,maximum allowable response time)are stated explicitly;constraints and/or limitations(e.g.,product cost restricts memory size)are noted

The Process

- The framework activities that characterize the software process are applicable to all software projects
- The project manager must decide which process model is most appropriate for
 - The customers who have requested the product and the people who will do the work
 - The characteristics of the product itself, and
 - The project environment in which the software team works

The Process

- When a process model has been selected, the team then defines a preliminary project plan based on the set of process framework activities.
- Once the preliminary plan is established, process decomposition begins
- That is, a complete plan, reflecting the work tasks required to populate the framework activities, must be created

The Project

- To manage a successful software project, we must understand what can go wrong so that problems can be avoided
- In an excellent paper on software projects, John Reel defines 10 signs that indicate that an information systems project is in risk
 - Software people don't understand their customer's needs.
 - The product scope is poorly defined
 - Changes are managed poorly
 - The chosen technology changes
 - Business needs change
 - Deadlines are unrealistic
 - Users are resistant
 - Sponsorship is lost
 - The project team lacks people with appropriate skills
 - Managers and practitioners avoid best practices and lessons learned

The Project

- How does a manager act to avoid the problems just noted?
- Reel suggests a five-part common-sense approach to software projects:
 - Start on the right foot
 - Maintain momentum
 - Track progress
 - Make smart decisions
 - Conduct a postmortem analysis

The Project-A five part common-sense approach to software projects

- Start on the right foot-Accomplished by working hard to understand the problem that is to be solved and then setting realistic objectives and expectations for everyone who will be involved in the project. It is accomplished by building the right team
- Maintain Momentum- Many projects get off to a good start and then slowly disintegrate. To maintain momentum , the project manager must provide incentives to keep turnover of personnel to an absolute minimum
- Track Progress-For a software project,progress is tracked as work products(e.g.,models,source code,sets of test cases)are produced and approved

The Project-A five part common-sense approach to software projects

- Make smart decisions-Whenever possible,decide to use commercial off-the shelf software or existing software components,decide to avoid custom interfaces when standard approaches are available,decide to identify and then avoid risks,and decide to allocate more time than you think is needed to complex and risky tasks
- Conduct a postmortem analysis-Establish a consistent mechanism for extracting lessons learned for each project.Evaluate the planned and actual schedules,collect and analyze software project metrics,get feedback from team members and customers



Metrics in the Process and Project Domains

Measurements and Metrics

- **Measurements and Metrics**
- A **measurement** is an indication of the size, quantity, amount or **dimension** of a particular attribute of a product or process.
- A **Metric** is a **measurement** of the degree that any attribute belongs to a system, product or process. For example the number of errors per person per hour would be a **metric**

What are Metrics

- Software process and project metrics are quantitative measures
- They are a management tool
- They offer insight into the effectiveness of the software process and the projects that are conducted using the process as a framework
- Basic quality and productivity data are collected
- These data are analyzed, compared against past averages, and assessed

What are Metrics

- The goal is to determine whether quality and productivity improvements have occurred
- The data can also be used to pinpoint problem areas
- Remedies can then be developed and the software process can be improved

Uses of Measurement

- Can be applied to the software process with the intent of improving it on a continuous basis
- Can be used throughout a software project to assist in estimation, quality control, productivity assessment, and project control
- Can be used to help assess the quality of software work products and to assist in tactical decision making as a project proceeds

Metrics in the Process Domain

- Process metrics are collected across all projects and over long periods of time
- They are used for making strategic decisions
- The intent is to provide a set of process indicators that lead to long-term software process improvement
- The only way to know how/where to improve any process is to
 - Measure specific attributes of the process
 - Develop a set of meaningful metrics based on these attributes
 - Use the metrics to provide indicators that will lead to a strategy for improvement

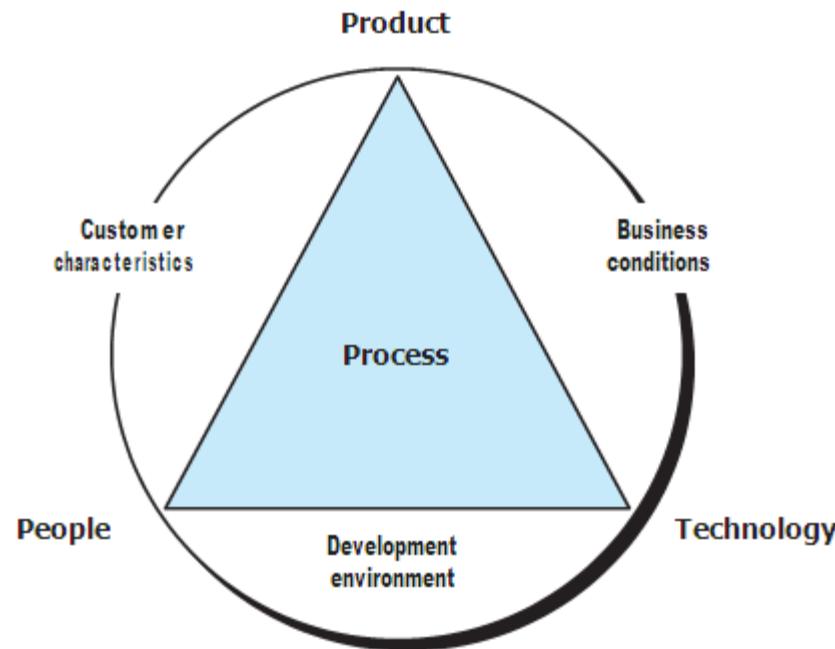
Metrics in the Process Domain

- We measure the effectiveness of a process by deriving a set of metrics based on outcomes of the process such as
 - Errors uncovered before release of the software
 - Defects delivered to and reported by the end users
 - Work products delivered
 - Human effort expended
 - Calendar time expended
 - Conformance to the schedule
 - Time and effort to complete each generic activity

Etiquette of Process Metrics

- Use common sense and organizational sensitivity when interpreting metrics data
- Provide regular feedback to the individuals and teams who collect measures and metrics
- Don't use metrics to evaluate individuals
- Work with practitioners and teams to set clear goals and metrics that will be used to achieve them
- Never use metrics to threaten individuals or teams
- Metrics data that indicate a problem should not be considered “negative”
 - Such data are merely an indicator for process improvement
- Don't obsess on a single metric to the exclusion of other important metrics

Determinants for software quality and organizational effectiveness

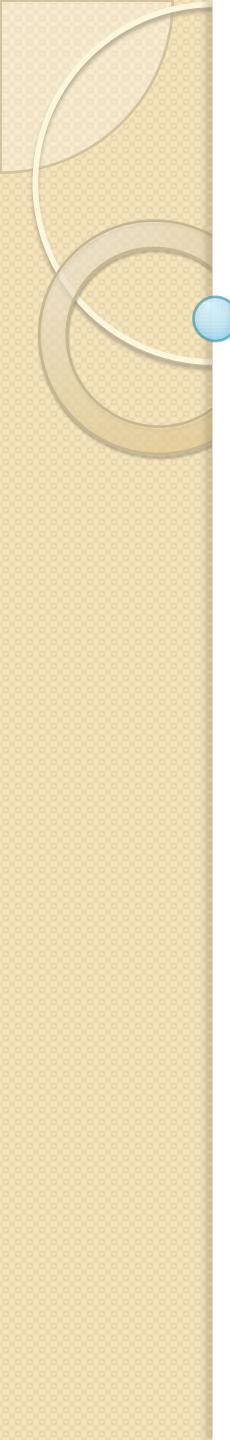


Determinants for software quality and organizational effectiveness

- Process sits at the center of a triangle connecting 3 factors that have a profound influence on software quality and organizational performance.
- The skill and motivation of people have been shown [Boe81] to be the most influential factors in quality and performance.
- The complexity of the product can have a substantial impact on quality and team performance.
- The technology (i.e., the software engineering methods and tools) that populates the process also has an impact.

Determinants for software quality and organizational effectiveness

- In addition, the process triangle exists within a circle of environmental conditions that include the development environment (e.g., integrated software tools), business conditions (e.g., deadlines, business rules), and customer characteristics (e.g., ease of communication and collaboration).



Metrics in the Project Domain

Metrics in the Project Domain

- Project metrics enable a software project manager to
 - Assess the status of an ongoing project
 - Track potential risks
 - Uncover problem areas before their status becomes critical
 - Adjust work flow or tasks
 - Evaluate the project team's ability to control quality of software work products
- Many of the same metrics are used in both the process and project domain
- Project metrics are used for making tactical decisions
 - They are used to adapt project workflow and technical activities

Use of Project Metrics

- The first application of project metrics occurs during estimation
 - Metrics from past projects are used as a basis for estimating time and effort
- As a project proceeds, the amount of time and effort expended are compared to original estimates
- As technical work commences, other project metrics become important
 - Production rates are measured (represented in terms of models created, review hours, function points, and delivered source lines of code)
 - Error uncovered during each generic framework activity (i.e, communication, planning, modeling, construction, deployment) are measured

Use of Project Metrics

- Project metrics are used to
 - Minimize the development schedule by making the adjustments necessary to avoid delays and mitigate potential problems and risks
 - Assess product quality on an ongoing basis and, when necessary, to modify the technical approach to improve quality
- In summary
 - As quality improves, defects are minimized
 - As defects go down, the amount of rework required during the project is also reduced
 - As rework goes down, the overall project cost is reduced

University Questions

- Explain four P's of Software Engineering
- What is process and project metrics

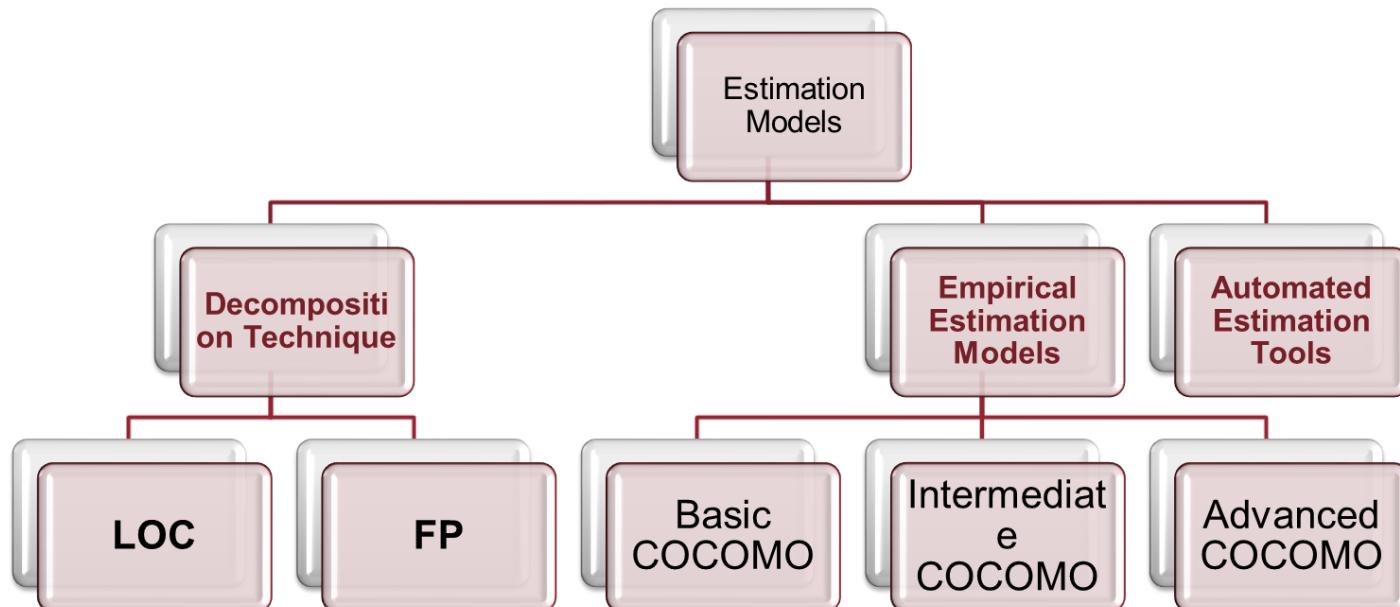
3.3 Software Project Estimation

- LOC
- FP

Software Measurement

- It can be categorized in two ways
 - Direct measures of the software process(e.g.,cost and effort applied) and product(e.g.,lines of code(LOC) produced,execution speed, and defects reported over some set period of time)
 - Indirect measures of the product that include functionality,quality,complexity,efficiency,reliability, maintainability etc.

Software Estimation Methods



Software Estimation:-

- **Decomposition Technique**
 - Lines of Code
 - Function Point
- **Empirical Estimation Models**
 - **Constructive Cost Model (COCOMO)**
 - Model 1 : Basic COCOMO
 - Model 2 : Intermediate COCOMO
 - Model 3 : Advanced COCOMO
- **Automated Estimation Tools**

- The five essential, fundamental metrics:
 - Size (LOC, etc.)
 - Cost (in dollars)
 - Duration (in months)
 - Effort (in person-month)
 - Quality (number of faults detected)

Steps in Estimating

- Estimate cost and effort
 - Decompose the problem
 - Develop two or more estimates using size, function points, process tasks or use-cases
 - Reconcile the estimates

Estimation

- Software cost estimation is the process of predicting the effort required to develop a software system.
- Estimation of resources, cost, and schedule for a software engineering effort requires
 - experience
 - access to good historical information
- Estimation carries inherent risk and this risk leads to uncertainty

Project Estimation



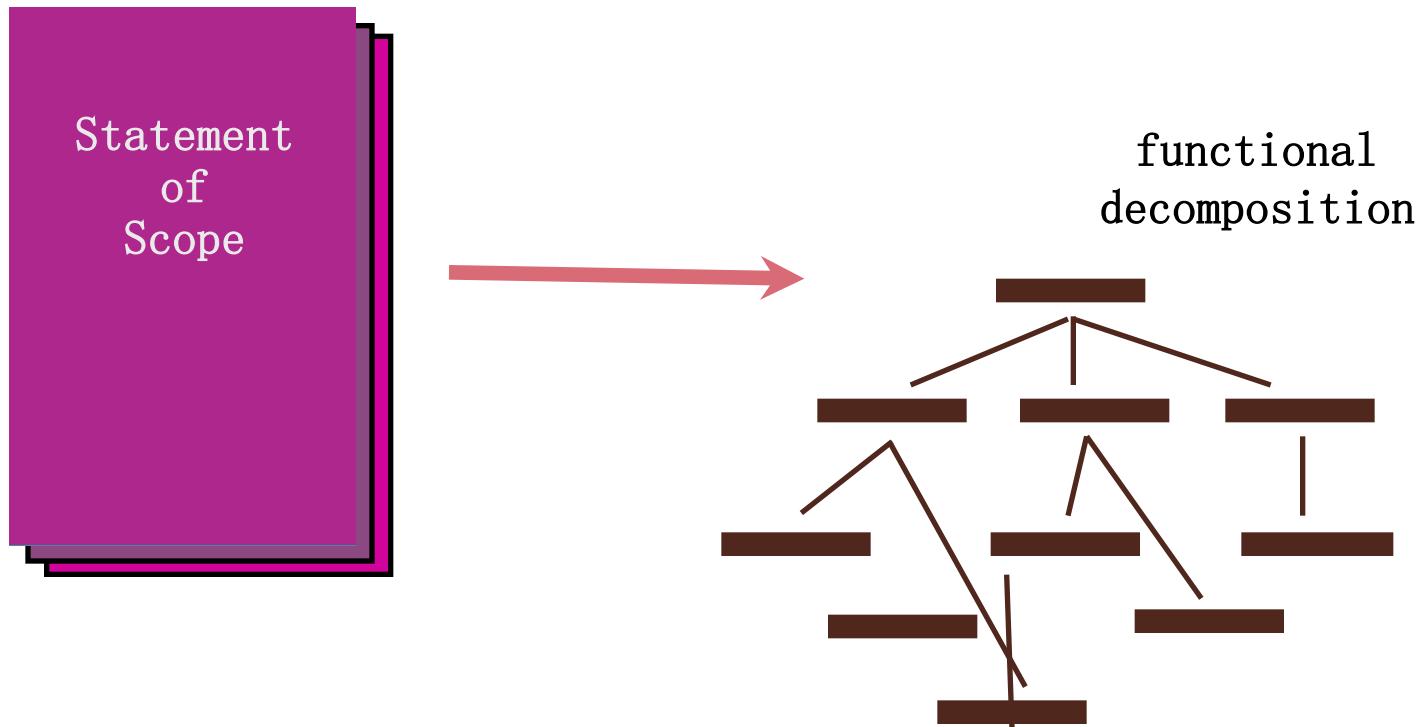
- Project scope must be understood
- Elaboration (decomposition) is necessary
- Historical metrics are very helpful
- At least two different techniques should be used
- Uncertainty is inherent in the process

Estimation Techniques

- Past (similar) project experience
- Conventional estimation techniques
 - task breakdown and effort estimates
 - size (e.g., FP) estimates
- Empirical models
- Automated tools



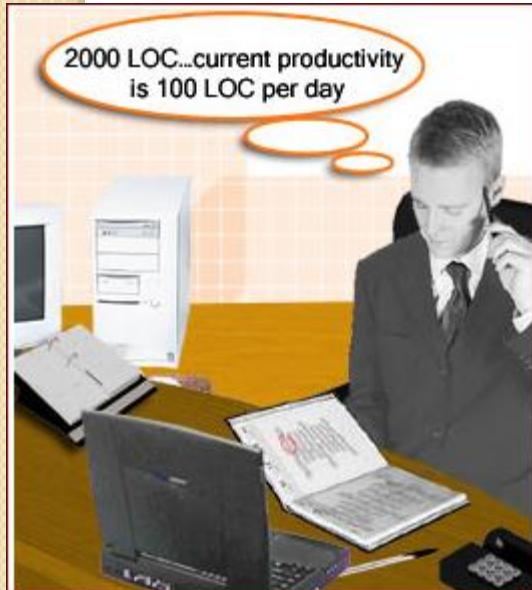
Functional Decomposition



Conventional Methods: LOC/FP Approach

- compute LOC (lines of code)/FP (function points) using estimates of information domain values
- use historical data to build estimates for the project

Example: LOC



Ap

Function	Estimated LOC
user interface and control facilities (UICF)	2,300
two-dimensional geometric analysis (2D GA)	5,300
three-dimensional geometric analysis (3D GA)	6,800
database management (DBM)	3,380
computer graphics display facilities (CGDF)	4,980
peripheral control (PC)	2,100
design analysis modules (DAM)	8,400
<i>estimated lines of code</i>	33,200

Average productivity for systems of this type = 620 LOC/pm.

Burdened labor rate = \$8000 per month, the cost per line of code is approximately \$13.

Based on the LOC estimate and the historical productivity data, the total estimated project cost is **\$431,600**

LOC Based Estimation

- You are required to establish estimates for Virtual Class room System. For first increment you have to provide automation of course registration, Attendance, lectures.
- LOC estimates for each Functionality are provided:
 - User screens 3000
 - Course registration 1550
 - Attendance 1800
 - Lectures 3000
 - Student query 800

Suppose productivity of your Organization is 500 LOC/pm
Labor rate is \$8000/pm

Calculate Effort and Total cost of this increment

Total LOC

- User screens 3000
- Course registration 1550
- Attendance 1800
- Lectures 3000
- Student query 800
= Total LOC=10150 LOC

- LOC/ Productivity
- = $10150/500$
- = 20.3 pm or 21 pm



$$\begin{aligned}\text{Total cost} &= \text{Rate} * \text{pm} \\ &= 8000 * 21 \\ &= 168000\text{Rs.}\end{aligned}$$

$$\begin{aligned}\text{Cost per LOC} &= 168000 / 10150 \\ &= 16.55 \text{ Rs.}\end{aligned}$$

Total Cost

- Total cost=Rate * pm
 - = 8000 * 21
 - = 168000Rs.
-
- Cost per LOC
=168000/10150
=16.55 Rs.



Function Points

- The function point metric(FP),first proposed by Albrecht,can be used effectively as a means for measuring the functionality delivered by the system
- Using historical data,the FP can then be used to
 - i)estimate the cost or effort required to design,code, and test the software;
 - ii)predict the number of errors that will be encountered during testing,
 - and iii)forecast the number of components and/or the number of projected source lines in the implemented system

Function Points

- Function points are derived using an empirical relationship based on countable(direct)measures of software's information domain and assessment of software complexity.
- Information domain values are defined in the following manner

Function Points

- Number of external inputs(EIs)-Each external input originates from a user or is transmitted from another application and provides distinct application-oriented data or control information.Inputs are often used to update internal logical files
- Number of external outputs(EOs)-Each external output is derived within the application and provides information to the user.External output refers to reports,screens,error messages, and so on

Function Points

- Number of external inquiries(EQs)-An external inquiry is defined as an online input that results in generation of some immediate software response in the form of an on-line output
- Number of internal logical files(ILFs)-Each internal logical file is a logical grouping of data that resides within the application's boundary and is maintained via external inputs

Function Points

- Number of external interface files(EIFs)-
Each external interface file is a logical grouping of data that resides external to the application but provides data that may be of use to the application

Function Points

- A function point is a unit of measurement to express the amount of business functionality an information system (as a product) provides to a user. The cost (in dollars or hours) of a single unit is calculated from past projects

Important Factors to be consider for FP are...



Number of external inputs – from user or anc

Number of external inquiries – request from user for an on-line output (E.G. IRCTC)

Number of external outputs

Number of internal logical files (maintained by system)

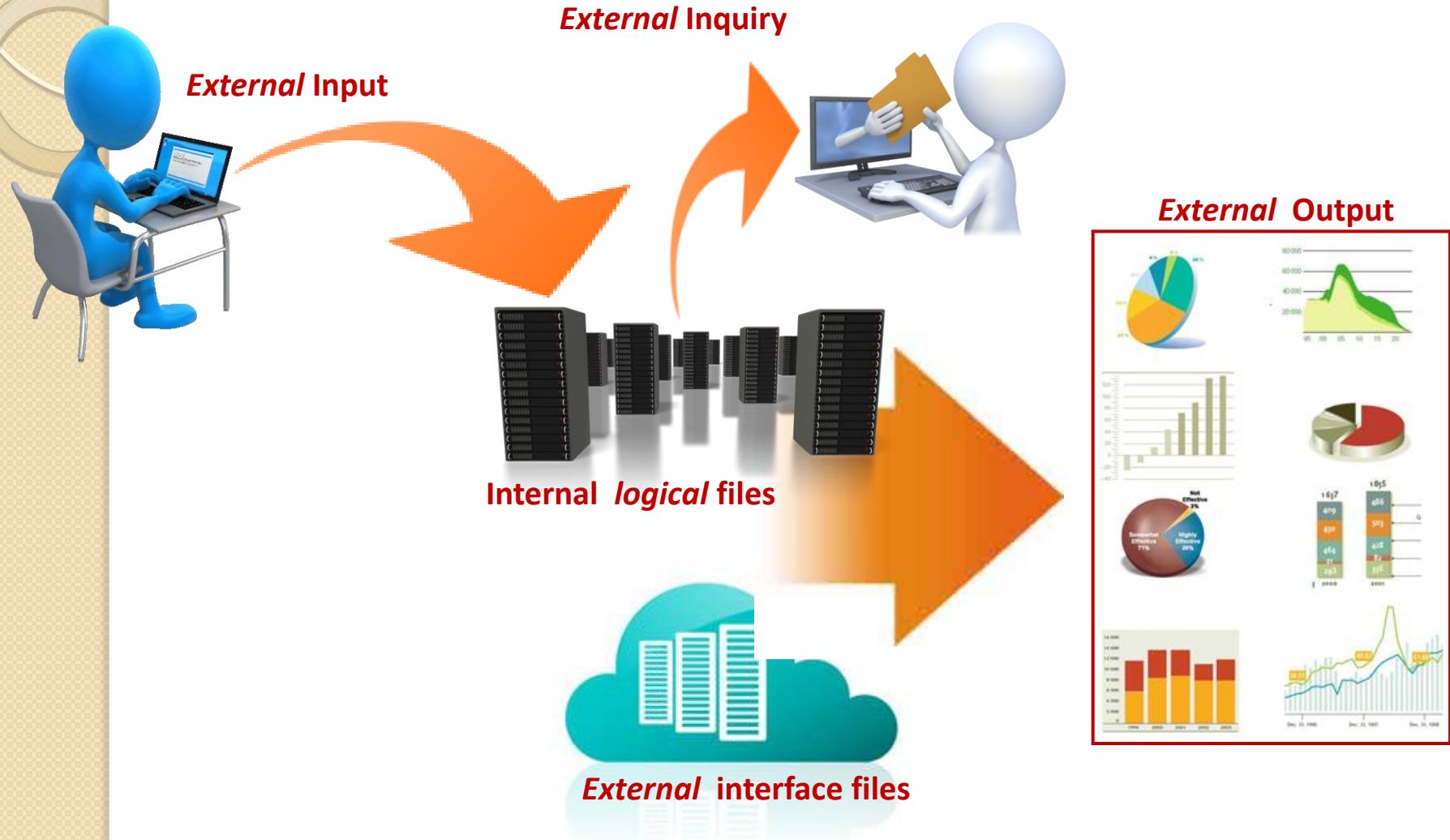
Number of external interface files (provides data but not maintained by system, The **external interface file** is an internal logical file for another application. An application may count a **file** as either a EIF or ILF not both.) (Library system may use students DB for calculating fine)



Size estimation

Functional Size Measurement – Function Point Analysis

67



Count-Total

- Count-Total = sum(number X weight)
- Where weights are:

	Simple	Average	Complex
External Inp	3	4	6
External Out	4	5	7
External Inq	3	4	6
Internal Files	7	10	15
External Files	5	7	10

Value adjustment factors(F_i)

- Total of 1-5 rating of following 14 questions:
 - Does the system require reliable back-up/recovery?
 - Are specialized data communications required?
 - Are there distributed processing functions?
 - Is performance critical?
 - Will run in heavily utilized operating environment?
 - On-line data entry required?
 - For on-line data entry, will it require multiple screens?
 - Are ILF's updated on-line?
 - Are input, output, files, or inquiries complex?
 - Is the internal processing complex?
 - Is the code designed to be reusable?
 - Are conversion and installation included?
 - Is the system designed for installation in different organizations?
 - Is the application designed to facilitate change and ease of use?

General System Characteristic		Brief Description
1.	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2.	Distributed data processing	How are distributed data and processing functions handled?
3.	Performance	Was response time or throughput required by the user?
4.	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5.	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6.	On-Line data entry	What percentage of the information is entered On-Line?
7.	End-user efficiency	Was the application designed for end-user efficiency?
8.	On-Line update	How many ILF's are updated by On-Line transaction?
9.	Complex processing	Does the application have extensive logical or mathematical processing?
10.	Reusability	Was the application developed to meet one or many user's needs?
11.	Installation ease	How difficult is conversion and installation?
12.	Operational ease	How effective and/or automated are start-up, back-up, and recovery procedures?
13.	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14.	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Examlpe



FP based Estimation

- For above mentioned system
 - Student can be registered
 - Courses can be added/updated.
 - Student can view courses
 - Student can inquire his/her registration status

	Count	Simple	Average	Complex
External Inp	3*	3	4	6 = 9
External Out	4*	4	5	7 = 16
External Inq	1*	3	4	6 = 3
Internal Files	4*	7	10	15 = 28
External Files	0*	5	7	10 = 0
Count Total =56				

Calculation of sum(F_i)

- Total of 0-5 rating of following 14 questions:
 - Does the system require reliable back-up/recovery? 3
 - Are specialized data communications required? 2
 - Are there distributed processing functions? 0
 - Is performance critical? 2
 - Will run in heavily utilized operating environment? 1
 - On-line data entry required? 4
 - For on-line data entry, will it require multiple screens? 5
 - Are ILF's updated on-line? 5
 - Are input, output, files, or inquiries complex? 2
 - Is the internal processing complex? 3
 - Is the code designed to be reusable? 0
 - Are conversion and installation included? 1
 - Is the system designed for installation in different organizations? 0
 - Is the application designed to facilitate change and ease of use? 1

- The estimated number of FP is derived:
$$FP_{estimated} = \text{count-total} \times [0.65 + 0.01 \times \text{Sum } (F_i)]$$
$$FP_{estimated} = 56 * (0.65 + 0.01 * 29)$$
$$= 52.64 \text{ or } 53$$
- If organizational average productivity = 5 FP/pm.
- labor rate = \$8000 per month,
- Cost = labor rate * Person Month
- Based on the FP estimate and the historical productivity data,
- **the total estimated project cost is 84800Rs. and the estimated effort is 10.6 person-months.**

Function Point (FP) is an element of software development which helps to approximate the cost of development early in the process. It may measures functionality from user's point of view.

Counting Function Point (FP):

- Step-1:

$$F = 14 * \text{scale}$$

Scale varies from 0 to 5 according to character of Complexity Adjustment Factor (CAF). Below table shows scale:

- 0 - No Influence
- 1 - Incidental
- 2 - Moderate
- 3 - Average
- 4 - Significant
- 5 - Essential

- **Step-2:** Calculate Complexity Adjustment Factor (CAF).

$$\text{CAF} = 0.65 + (0.01 * F)$$

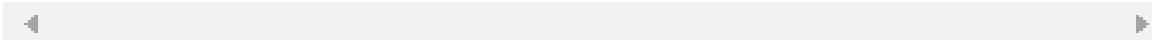
- - -

- - -

Step-3: Calculate Unadjusted Function Point (UFP).

TABLE (Required)

	Function Units	Low	Avg	High
EI		3	4	6
EO		4	5	7
EQ		3	4	6
ILF		7	10	15
EIF		5	7	10



Multiply each individual function point to corresponding values in TABLE.



- **Step-4: Calculate Function Point.**

$$FP = UFP * CAF$$

Example:

Given the following values, compute function point when all complexity adjustment factor (CAF) and weighting factors are average.

User Input = 50

User Output = 40

User Inquiries = 35

User Files = 6

External Interface = 4

- **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.

$$P = (50*4) + (40*5) + (35*4) + (6*10) + (4*7) = 628$$



• Step-4:

$$\text{Function Point} = 628 * 1.07 = 671.96$$

Cost Estimation

- Assume organizational average productivity = 25 FP/pm.
- labor rate = \$8000 per month,
- No of months required=FP/average productivity
 - $=672/25=26.88=27\text{ months}$
- Estimated cost for project=\$8000*27=\$2,16,000
- Based on the FP estimate and the historical productivity data,
- **the total estimated project cost is \$2,16,000 and the estimated effort is 27 months.**

University Questions

- What is cost estimation? Explain LOC method.(5 marks)
- Differentiate between FP based and LOC based cost estimation techniques.(10 marks)

Project Estimation Techniques

- Estimation of various project parameters is an important project planning activity. The different parameters of a project that need to be estimated include-
 - Project Size
 - Effort required to complete the project,
 - Project duration and
 - Cost
- Accurate estimation of these parameters is important for resource planning and scheduling
- Estimation Techniques can be classified as
 - Empirical Estimation Techniques
 - Heuristic Estimation Techniques
 - Analytical Estimation Techniques

Empirical Estimation Techniques

- Empirical Estimation Techniques are based on making an educated guess of the project parameters and common sense
- This technique is based on prior experience of development of similar products and projects
- An educated guess based on past experience
- Two popular empirical estimation techniques are
 - Expert Judgement Technique
 - Delphi Cost Estimation

Expert Judgement Technique

- In this an expert makes an educated guess of the problem size after analyzing the problem thoroughly
- The expert estimates the cost of the different components of the system:e.g GUI,database module,communication module,billing module etc
- Combines them to arrive at the overall estimate

Delphi Cost Estimation

- Is carried out by a team comprising of a group of experts and a coordinator
- 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

Heuristic Techniques

- In Heuristic techniques the relationship that exists among the different project parameters can modeled using suitable mathematical expressions
- Once the independent parameters are known, the dependent parameters can be easily determined by substituting the values of the independent parameters in the corresponding mathematical expressions

Heuristic Techniques

- Estimated parameter is the dependent parameter to be estimated. The dependent parameters to be estimated could be effort, duration, staff size etc.
- Assume that the characteristic to be estimated can be expressed in terms of some mathematical expression
 - Can be classified as Single variable and multivariable models

COCOMO Model-Constructive Cost Model

- Was first proposed by Dr. Barry Boehm in 1981
- Is a heuristic estimation technique-this technique assumes that relationship among different parameters can be modeled using some mathematical expression
- This approach implies that size is primary factor for cost,other facors have lesser effects
- Constructive implies that the complexity
- COCOMO prescribes a three stage process for project estimation

COCOMO Model-Constructive Cost Model

- An initial estimate is obtained, and over next two stages the initial estimate is refined to arrive at a more accurate estimate
- Projects used in this model have following attributes
 - Ranging in size from 2000 to 10000 lines of code
 - Programming languages ranging from assembly to PL/I
 - These projects were based on the waterfall model of software development

COCOMO Model-Constructive Cost Model

- The Constructive Cost Model (COCOMO) is a procedural software cost estimation model developed by Barry W. Boehm.
- The model parameters are derived from fitting a regression formula using data from historical projects

COCOMO Model-Constructive Cost Model

- Boehm stated that any software development project can be classified into three categories
 - Organic
 - Semi-detached
 - Embedded

COCOMO Model-Constructive Cost Model

- **Organic-**
 - If the project deals with developing a well understood application program
 - The size of development is reasonably small and experienced
 - The team members are experienced in developing similar kind of projects
 - Examples of this type of projects are simple business systems, simple inventory management systems, and data processing systems.

COCOMO Model-Constructive Cost Model

- **Semidetached-**
 - If the development team consists of a combination of both experienced and inexperienced staff
 - Team members have limited experience about some aspects but are totally unfamiliar with some aspects of the system being developed
 - Mixed experience
 - Example of Semidetached system includes developing a new operating system (OS), a Database Management System (DBMS), and complex inventory management system.

COCOMO Model-Constructive Cost Model

- **Embedded-**
 - If the software being developed is strongly coupled to complex hardware
 - Software projects that must be developed within a set of tight software,hardware and operational constraints
 - For Example: ATM,Air Traffic control.

<i>Mode</i>	<i>Project size</i>	<i>Nature of Project</i>	<i>Innovation</i>	<i>Deadline of the project</i>	<i>Development Environment</i>
Organic	Typically 2-50 KLOC	Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.	Little	Not tight	Familiar & In house
Semi detached	Typically 50-300 KLOC	Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large project, Real time systems, Complex interfaces, Very little previous experience. For example: ATMs, Air Traffic Control etc.	Significant	Tight	Complex Hardware/ customer interfaces required

PERSON MONTH(PM)

- The effort estimation is expressed in units of person-months (PM).
- An effort of 100 PM does not imply that 100 persons should work for 1 month nor does it imply that 1 person should be employed for 100 months, but it denotes the area under the person-month curve .
- It is the area under the person-month plot.

Person month is a measurement unit for effort in software engineering. 1 person month means effort put by a person in one month. But 100 person does not mean, work effort put by 100 person in one month or 1 person in 100 months. As requirement of staff varies time to time in the development so there is not constant no of people is there to work. It is calculated from the graph(calculate area under time axis) between no of people working and time(in month).

COCOMO Model-Cost Constructive Model

- According to Boehm, software cost estimation should be done through three stages:
- Basic COCOMO Model
- Intermediate COCOMO Model
- Detailed COCOMO Model

1. BASIC COCOMO MODEL

- Basic COCOMO model computes software development effort, time and cost as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC, KLOC).

$$\begin{aligned} \text{EFFORT} &= a_1 \times (\text{KLOC})^{a_2} \text{ PM} \\ T_{\text{dev}} &= b_1 \times (\text{Effort})^{b_2} \text{ Months} \end{aligned}$$

Where

- KLOC is the estimated number of delivered lines (expressed in thousands) of code for project, estimated size of software product.
- The coefficients a_1 , a_2 , b_1 and b_2 are constants for each category of software products.
- T_{dev} is the estimated time to develop the software, in months.
- Effort is the total efforts required to develop the software product, expressed in Person Months (PM)

ESTIMATION OF DEVELOPMENT EFFORT

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

ESTIMATION OF DEVELOPMENT TIME

Organic	$T_{dev} = 2.5(Effort)^{0.38}$	Months
Semi-detached	$T_{dev} = 2.5(Effort)^{0.35}$	Months
Embedded	$T_{dev} = 2.5(Effort)^{0.32}$	Months

CALCULATING EFFORT AND PRODUCTIVITY

When effort and development time are known, the average staff size to complete the project may be calculated as:

$$\text{Average staff size (SS)} = \frac{E}{D} \text{ Persons}$$

When project size is known, the productivity level may be calculated as:

$$\text{Productivity (P)} = \frac{KLOC}{E} \text{ KLOC / PM}$$

Software Projects	a_1	a_2	b_1	b_2
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

- Example: consider a software project using semi-detached mode with 30,000 lines of code . We will obtain estimation for this project as follows:

(1) Effort estimation $E = a_1(\text{KLOC}) \exp(a_2)$ person-months

$$E = 3.0(30) \exp(1.12) \text{ where lines of code} = 30000 = 30 \text{ KLOC}$$

$$E = 135 \text{ person-month}$$

(2) Duration estimation $D = b_1(E) \exp(b_2)$ months $= 2.5(135) \exp(0.35)$

$$D = 14 \text{ months}$$

(3) Person estimation $N = E/D$

$$= 135/14 N = 10$$



CALCULATING EFFORT AND PRODUCTIVITY

When effort and development time are known, the average staff size to complete the project may be calculated as:

$$\text{Average staff size (SS)} = \frac{E}{D} \text{ Persons}$$

When project size is known, the productivity level may be calculated as:

$$\text{Productivity (P)} = \frac{KLOC}{E} \text{ KLOC / PM}$$

$$\frac{30000}{135}$$

Merits-

- Basic COCOMO is good for quick , rough and early estimate of software costs.

Demerits-

- It does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.
- The accuracy of this model is limited because it does not consider certain factors for cost estimation of software.

Example on Basic COCOMO Mode

- **Example I:** 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.
- $\text{Effort} = a_1 * (\text{KLOC})^a_2 \text{ PM}$
- $\text{Time} = b_1 * (\text{effort})^{b_2} \text{ months}$

Software Projects	a_1	a_2	b_1	b_2
Organic	2.4	1.05	2.5	0.38
Semi-Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

- **Solution:** The basic COCOMO equation takes the form:
 $\text{Effort} = a_1 * (\text{KLOC})^{a_2} \text{ PM}$
 $T_{\text{dev}} = b_1 * (\text{efforts})^{b_2} \text{ Months}$
Estimated Size of project= 400 KLOC
- **(i)Organic Mode**
 $E = 2.4 * (400)^{1.05} = 1295.31 \text{ PM}$
 $D = 2.5 * (1295.31)^{0.38} = 38.07 \text{ Months}$

- **(ii) Semidetached Mode**
- $E = 3.0 * (400)1.12 = 2462.79 \text{ PM}$
 $D = 2.5 * (2462.79)0.35 = 38.45 \text{ M}$
- **(iii) Embedded Mode**
- $E = 3.6 * (400)1.20 = 4772.81 \text{ PM}$
 $D = 2.5 * (4772.8)0.32 = 38 \text{ M}$

Software Projects	a ₁	a ₂	b ₁	b ₂
Organic	2.4	1.05	2.5	0.38
Semi - Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

- **Example2:** A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the Effort, development time, average staff size, and productivity of the project.
- Semidetached
- $\text{Effort} = a_1 * (\text{KLOC})^{\exp a_2}$
- $T = b_1 * (\text{effort})^{\exp b_2}$ months
- Avg staff size = E/T
- Productivity = KLOC/Effort

- **Solution:** The semidetached mode is the most appropriate mode, keeping in view the size, schedule and experience of development time.
 - Hence $E = 3.0(200) \times 1.12 = 1133.12 \text{ PM}$
 $D = 2.5(1133.12)0.35 = 29.3 \text{ PM}$

$$\text{Average Staff Size (SS)} = \frac{E}{D} \text{ Persons}$$
$$= \frac{1133.12}{29.3} = 38.67 \text{ Persons}$$

$$\text{Productivity} = \frac{\text{KLOC}}{E} = \frac{200}{1133.12} = 0.1765 \text{ KLOC/PM}$$

$$P = 176 \text{ LOC/PM}$$

Intermediate COCOMO

- Basic COCOMO model assumes
 - effort and development time depend on product size alone.
- However, several parameters affect effort and development time:
 - Reliability requirements
 - Availability of CASE tools and modern facilities to the developers
 - Size of data to be handled

Intermediate COCOMO

- For accurate estimation,
 - the effect of all relevant parameters must be considered:
 - Intermediate COCOMO model recognizes this fact:
 - refines the initial estimate obtained by the basic COCOMO by using a set of 15 cost drivers (multipliers).

Intermediate COCOMO (CONT.)

- Rate different parameters on a scale of one to three:
 - multiply cost driver values with the estimate obtained using the basic COCOMO.

INTERMEDIATE COCOMO MODEL

- Intermediate COCOMO computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes.
- The Intermediate COCOMO model refines the initial estimate obtained from the basic COCOMO by scaling the estimate up or down based on attributes of software development.
- This model uses a set of 15 cost drivers; these cost drivers are multiplied with the initial cost and effort estimates to scale the estimates up and down.
- This extension considers a set of "cost drivers", each with a number of subsidiary attributes:
- Product attributes
 - ❖ Required software reliability
 - ❖ Size of application database
 - ❖ Complexity of the product

- Hardware attributes
 - ❖ Run-time performance constraints
 - ❖ Memory constraints
 - ❖ Volatility of the virtual machine environment
 - ❖ Required turnabout time
- Personnel attributes
 - ❖ Analyst capability
 - ❖ Software engineering capability
 - ❖ Applications experience
 - ❖ Virtual machine experience
 - ❖ Programming language experience
- Project attributes
 - ❖ Use of software tools
 - ❖ Application of software engineering methods
 - ❖ Requirements development schedule



Cost Drivers	Very Low	Low	Nominal	High	Very High
Product Attributes					
Required Software Reliability	0.75	0.88	1.00	1.15	1.40
Size of Application Database		0.94	1.00	1.08	1.16
Complexity of The Product	0.70	0.85	1.00	1.15	1.30
Hardware Attributes					
Runtime Performance Constraints			1.00	1.11	1.30
Memory Constraints			1.00	1.06	1.21
Volatility of the virtual machine environment	0.87	1.00	1.15	1.30	
Required turnabout time	0.94	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	
Software engineer 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						
Application of software engineering methods	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	

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
Required Reliability	.75	.88	1.00	1.15	1.40	1.40
Database Size	.94	.94	1.00	1.08	1.16	1.16
Product Complexity	.70	.85	1.00	1.15	1.30	1.65
Execution Time Constraint	1.00	1.00	1.00	1.11	1.30	1.66
Main Storage Constraint	1.00	1.00	1.00	1.06	1.21	1.56
Virtual Machine Volatility	.87	.87	1.00	1.15	1.30	1.30
Comp Turn Around Time	.87	.87	1.00	1.07	1.15	1.15
Analyst Capability	1.46	1.19	1.00	.86	.71	.71
Application Experience	1.29	1.13	1.00	.91	.82	.82
Programmers Capability	1.42	1.17	1.00	.86	.70	.70
Virtual machine Experience	1.21	1.10	1.00	.90	.90	.90
Language Experience	1.14	1.07	1.00	.95	.95	.95
Modern Prog Practices	1.24	1.10	1.00	.91	.82	.82
SW Tools	1.24	1.10	1.00	.91	.83	.83
Required Dev Schedule	1.23	1.08	1.00	1.04	1.10	1,10

Intermediate COCOMO equation:

$$E = a_i (KLOC) b_i * EAF$$

$$D = c_i (E) d_i$$

Intermediate COCOMO Model

Software Projects	a	b
Organic	3.2	1.05
Semi Detached	3.0	1.12
Embedded	2.8	1.20

Example:

Consider a project having 30,000 lines of code which in an embedded software with critical area hence reliability is high. The estimation can be

$$E = a_1 (KLOC)^{a_2} \times (EAF)$$

As reliability is high EAF=1.15(product attribute)

$$2.8(30) \exp 1.20 * 1.15$$

$$\text{productivity} = \text{KLOC/Effort} = 30/191$$

Software Projects	a ₁	a ₂	b ₁	b ₂
Organic	2.4	1.05	2.5	0.38
Semi - Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Example:

Consider a project having 30,000 lines of code which in an embedded software with critical area hence reliability is high. The estimation can be

$$E = a_1 (KLOC)^{a_2} \times (EAF)$$

As reliability is high EAF=1.15(product attribute)

$$a_1 = 2.8$$

$$b_1 = 1.20 \text{ for embedded software}$$

$$E = 2.8(30)^{1.20} \times 1.15$$

$$= 191 \text{ person month}$$

$$D = b_1 (E)^{b_2}$$

$$= 2.5(191)^{0.32}$$

$$= 13 \text{ months approximately}$$

$$N = E/D = 191/13$$

$$N = 15 \text{ persons approx.}$$

Merits:

- This model can be applied to almost to entire software product for easy and rough cost estimation during early stage.
- It can be applied at the software product component level for obtaining more accurate cost estimation.

Demerits:

- The effort multipliers are not dependent on phases.
- A product with many components is difficult to estimate.

SHORTCOMING OF BASIC AND INTERMEDIATE COCOMO MODELS

- Both models:
 - ❖ consider a software product as a single homogeneous entity;
 - ❖ However, most large systems are made up of several smaller sub-systems.
 - ❖ Some sub-systems may be considered as organic type, some may be considered embedded, etc.
 - ❖ For some the reliability requirements may be high, and so on.
 - ❖ So, complete COCOMO was proposed to overcome these limitations of basic and intermediate COCOMO.

Complete COCOMO

- Cost of each sub-system is estimated separately.
- Costs of the sub-systems are added to obtain total cost.
- Reduces the margin of error in the final estimate.

Complete COCOMO Example

- A Management Information System (MIS) for an organization having offices at several places across the country:
 - Database part (**semi-detached**)
 - Graphical User Interface (GUI) part (**organic**)
 - Communication part (**embedded**)
- Costs of the components are estimated separately:
 - summed up to give the overall cost of the system.

Relation between LOC and FP

Relation between LOC and FPs

Language	LOC/FP
assembly	320
C	128
Cobol	105
Fortan	105
Pascal	90
Ada	70
OO languages	30
4GL languages	20

- Relationship:

- $LOC = Language\ Factor * FP$
- where
 - LOC (Lines of Code)
 - FP (Function Points)

Language	QSM SLOC/FP Data			
	Avg	Median	Low	High
ABAP (SAP) *	28	18	16	60
ASP*	51	54	15	69
Assembler *	119	98	25	320
Brio +	14	14	13	16
C *	97	99	39	333
C++ *	50	53	25	80
C# *	54	59	29	70
COBOL *	61	55	23	297
Cognos Impromptu Scripts +	47	42	30	100
Cross System Products (CSP) +	20	18	10	38
Cool:Gen/IEF *	32	24	10	82
Datastage	71	65	31	157
Excel *	209	191	131	315
Focus *	43	45	45	45
FoxPro	36	35	34	38
HTML *	34	40	14	48

Java *	53	53	14	134
JavaScript *	47	53	31	63
JCL *	62	48	25	221
LINC II	29	30	22	38
Lotus Notes *	23	21	19	40
Natural *	40	34	34	53
.NET *	57	60	53	60
Oracle *	37	40	17	60
PACBASE *	35	32	22	60
Perl *	24	15	15	60
PL/1 *	64	80	16	80
PL/SQL *	37	35	13	60
Powerbuilder *	26	28	7	40
REXX *	77	80	50	80
Sabretalk *	70	66	45	109
SAS *	38	37	22	55
Siebel *	59	60	51	60
SLOGAN *	75	75	74	75
SQL *	21	21	13	37
VB.NET *	52	60	26	60
Visual Basic *	42	44	20	60

Elaborate COCOMO method of cost estimation

CHAPTER 3

Project Planning and Scheduling

“Failing to plan is planning to fail”

- **Planning:**

- “what” is going to be done, “how”, “where”, by “whom”, and “when”
- for effective monitoring and control of complex projects

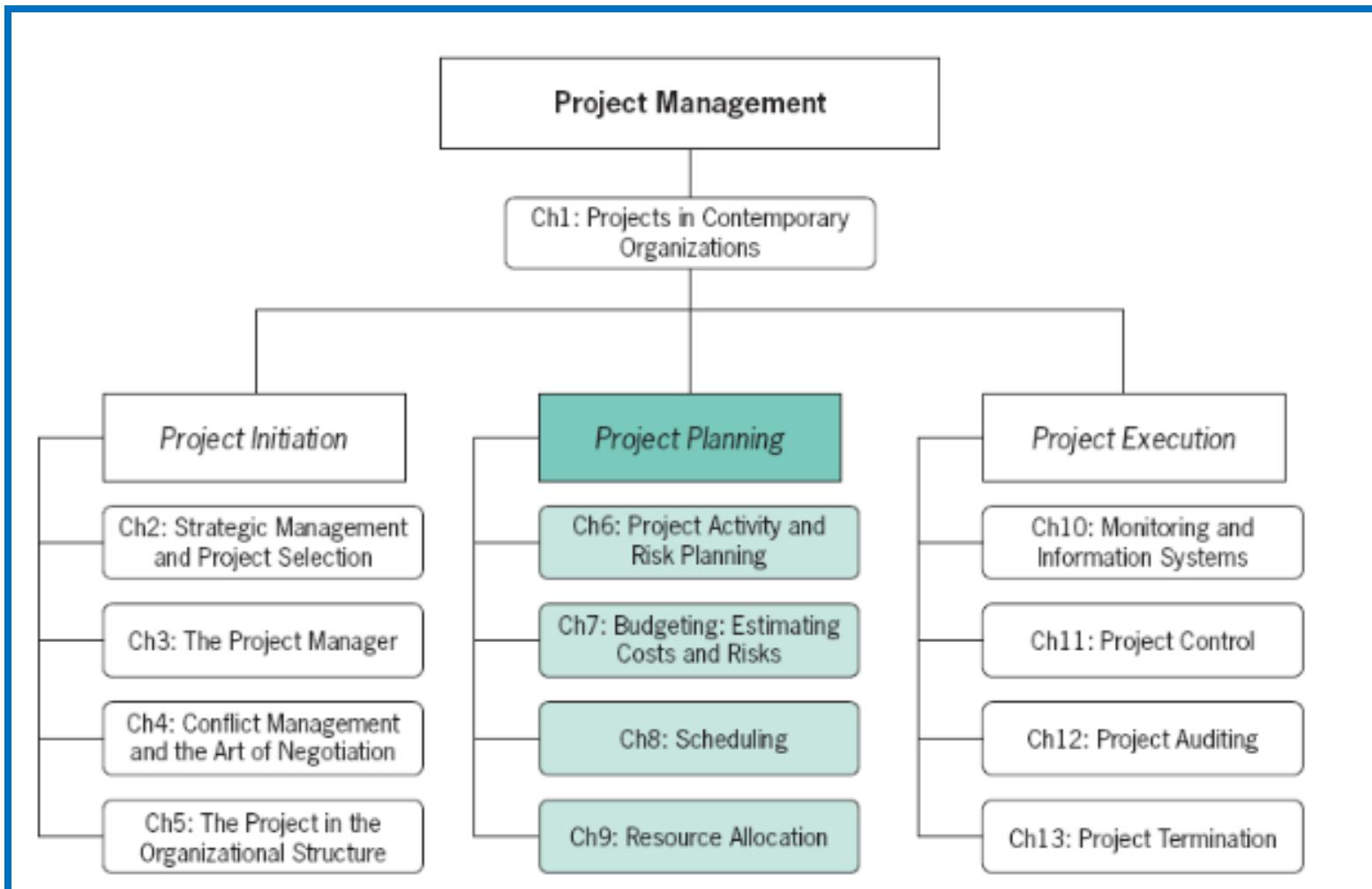
“It's about time”

- **Scheduling:**
 - “what” will be done, and “who” will be working
 - relative timing of tasks & time frames
 - a concise description of the plan

“Once you plan your work, you must work your plan”

- Planning and Scheduling occurs:
 - **AFTER** you have decided how to do the work
 - “The first idea is not always the best idea.”
- Requires discipline to “work the plan”
 - The act of development useful,
 - But need to monitor and track
 - only then, is a schedule an effective management tool
 - as-built schedules

Project Management



Contents

- Project Planning and Scheduling: Work Breakdown structure (WBS) and linear responsibility chart,
- Project cost estimation and budgeting,
- Top down and bottoms up budgeting,
- Networking and Scheduling techniques.
- PERT, CPM, GANTT chart,
- Introduction to Project Management Information System (PMIS).



- As the Beagle 2 Mars probe designed jointly by the European Space Agency and British National Space Center headed to Mars in December of 2003, contact was lost and it was never heard from again.
- In retrospect, it appears that inadequate project planning (and preplanning) was to blame.
- Excessive pressure on time, cost, and weight compromised the mission right from the start.
- With insufficient public funding, the design team had to spend much of their time raising private funds instead of addressing difficult technical issues.
- In addition, late changes forced the team to reduce the Beagle's weight from 238 pounds to 132 pounds! And when the three airbags failed to work properly in testing, a parachute design was substituted but inadequately tested due to lack of time.

- Requisite financing be available at the outset of a project
- Formal project reviews be conducted on a regular basis
- Milestones should be established where all stakeholders reconsider the project
 - Expectations of potential failure should be included in the funding consideration
 - Robust safety margins should be included and funded for uncertainties

PLANNING

- The purpose of planning is to facilitate the later accomplishment.
- Smooth the path from idea to outcome.
- Planning can be interpreted as “setting objective”, “defining the scope”, “Identifying requirements” .
- Budget and schedule are main important parts of proj plan.

Initial Project Coordination and the Project Charter

- Project launch meetings are used to decide on participating in the project with clear goal by senior mgmt people.
- Discussion should be according to complexity of project.
- Change management (“change without communication”)
- Outcomes include:
 - Technical scope
 - Areas of responsibility
 - Delivery dates or budgets
 - Risk management group
 - Clear deliverables

Project Management in Practice

Child Support Software a Victim of Scope Creep



In March 2003, the United Kingdom's Child Support Agency (CSA) started using their new £456 million (\$860 million) software system for receiving and disbursing child support payments. However, by the end of 2004 only about 12 percent of all applications had received payments, and even those took about three times longer than normal to process. CSA thus threatened to scrap the entire system and withhold £1 million (\$2 million) per month in service payments to the software vendor. The problem was thought to be due to both scope creep and the lack of a risk management strategy. The vendor claimed that the proj-

ect was disrupted constantly by CSA's 2500 change requests, while CSA maintained there were only 50, but the contract did not include a scope management plan to help define what constituted a scope change request. And the lack of a risk management strategy resulted in no contingency or fallback plans in case of trouble, so when project delays surfaced and inadequate training became apparent, there was no way to recover.

Source: Project Management Institute. "Lack of Support," *PM Network*, January 2005, p. 1.

Outside Clients

- When it is for outside clients, specifications cannot be changed without the client's permission
- Client may place budget constraints on the project
- May be competing against other firms
- Sales vs.. Design/Functional

Project Charter Elements

- Purpose
- Objectives
- Overview
- Schedules
- Resources
- Personnel
- Risk management plans
- Evaluation methods

Systems Integration

- Performance
- Effectiveness
- Cost

Starting the Project Plan: The WBS

- What is to be done
- When it is to be started and finished
- Who is going to do it

Starting the Project Plan: The WBS

Continued

- Some activities must be done sequentially
- Some activities may be done simultaneously
- Many things must happen when and how they are supposed to happen
- Each detail is uncertain and subjected to risk

Hierarchical Planning

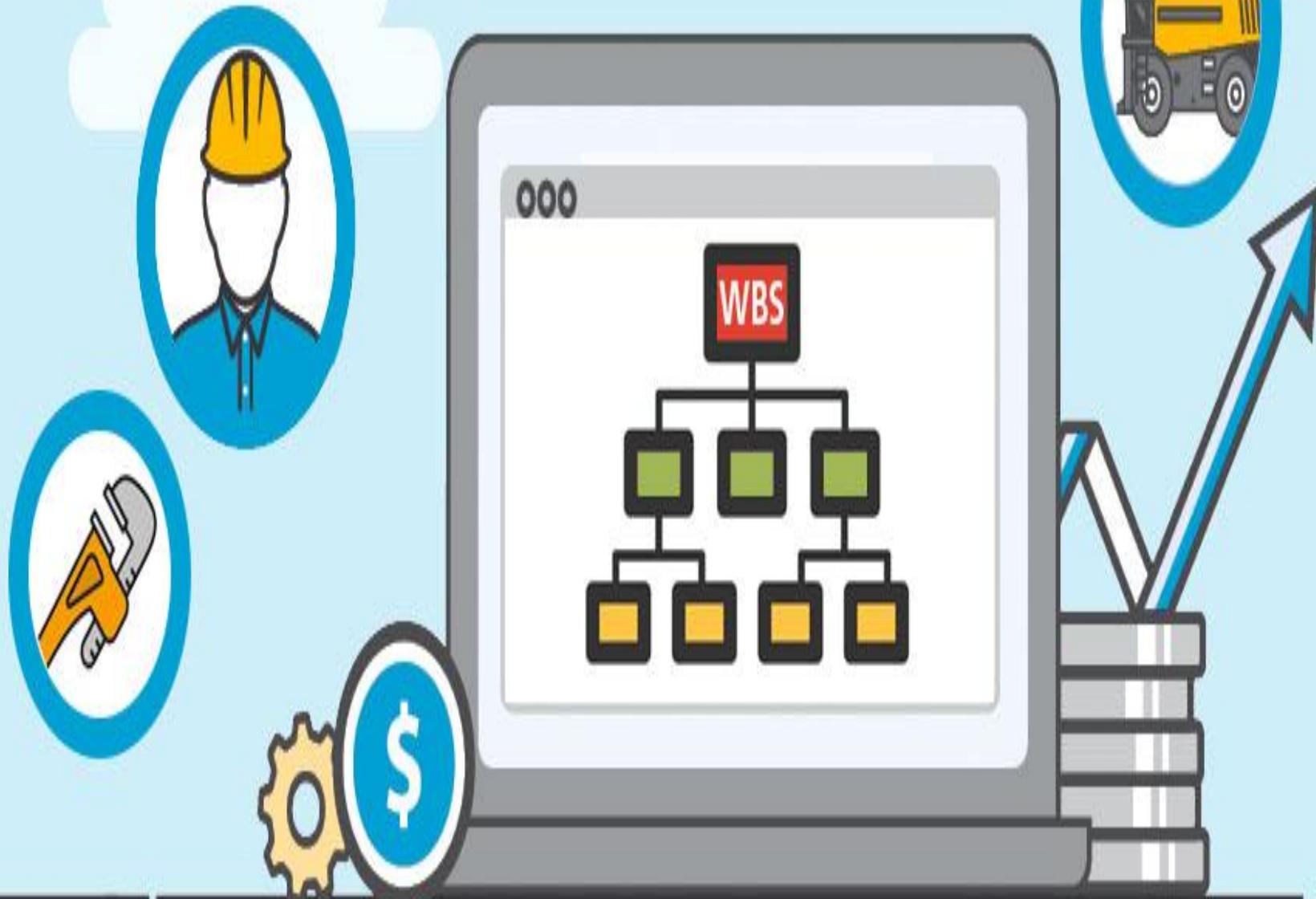
- Major tasks are listed
- Each major task is broken down into detail
- This continues until all the activities to be completed are listed
- Need to know which activities “depend on” other activities

A Form to Assist Hierarchical Planning

ACTION PLAN				
Deliverables _____ _____				
Measure(s) of accomplishment _____ _____				
Key constraints and assumptions _____ _____				
TASKS	ESTIMATED RESOURCES	IMMEDIATE PREDECESSOR TASKS	ESTIMATED TIME DURATION(S)	ASSIGNED TO

Figure 6-2

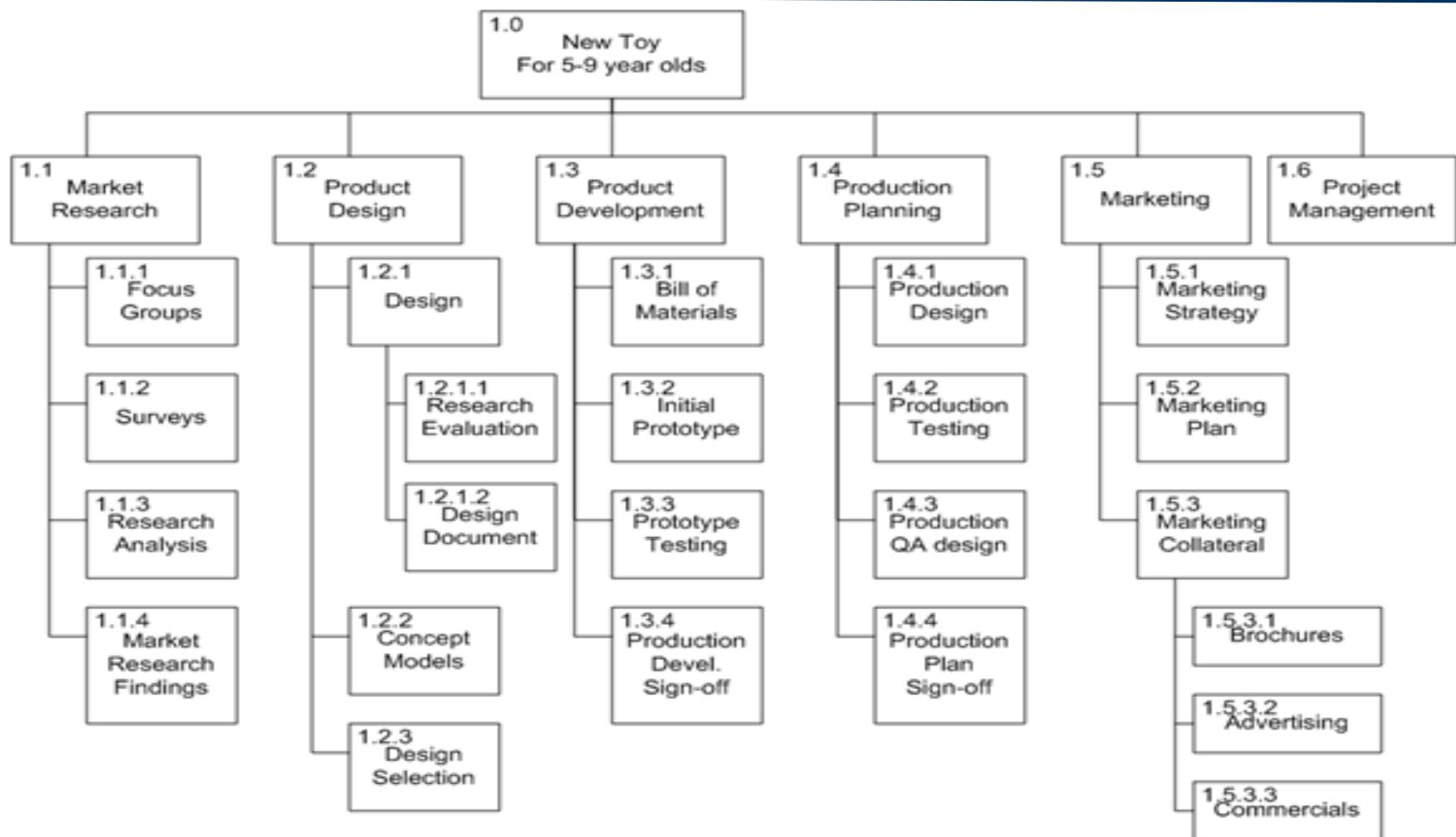
Work Breakdown Structure



The Work Breakdown Structure (WBS)

- A hierarchical planning process
- Breaks tasks down into successively finer levels of detail
- Continues until all meaningful tasks or work packages have been identified
- These make tracking the work easier
- Need separate budget/schedule for each task or work package

WBS EXAMPLE



A Visual WBS

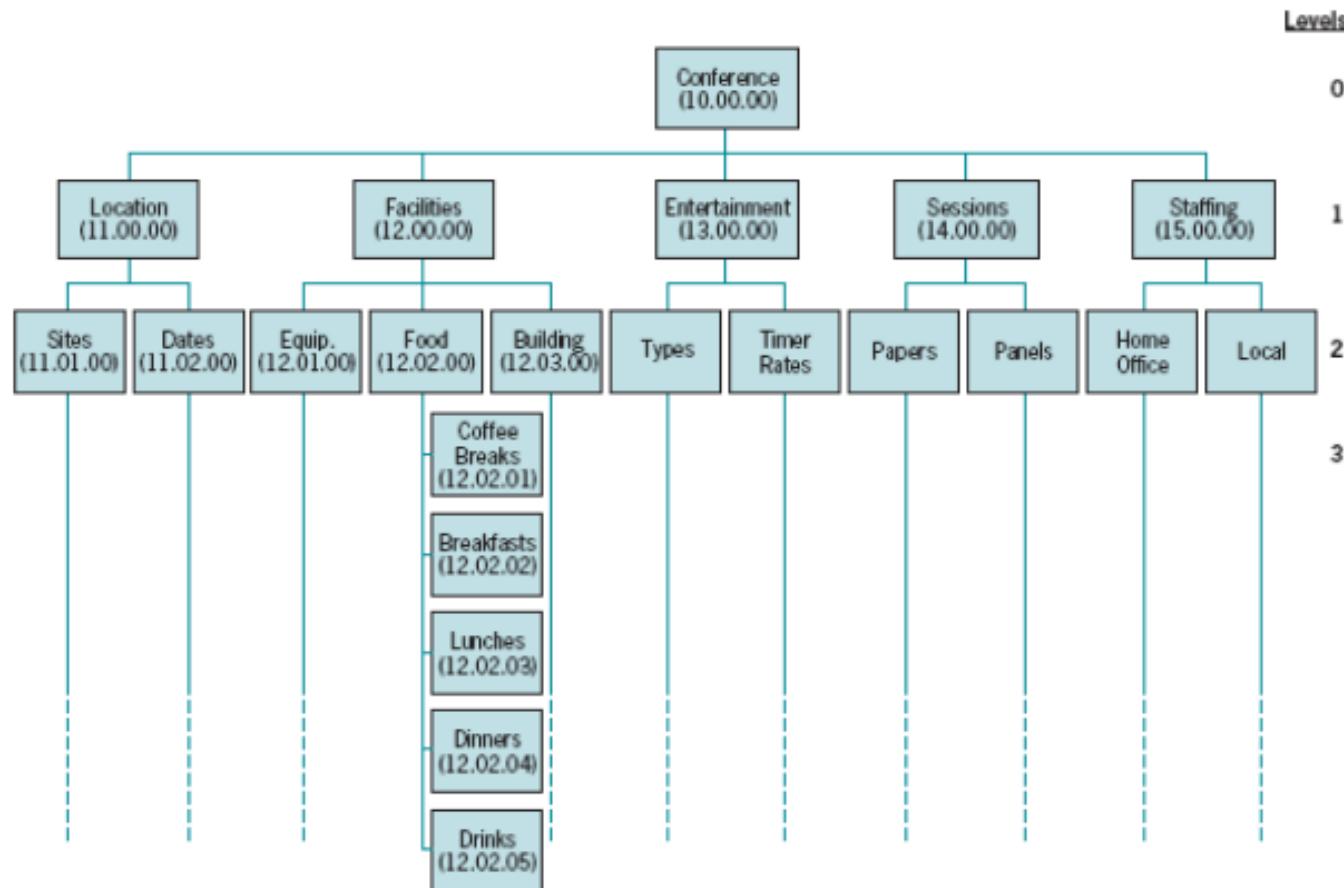
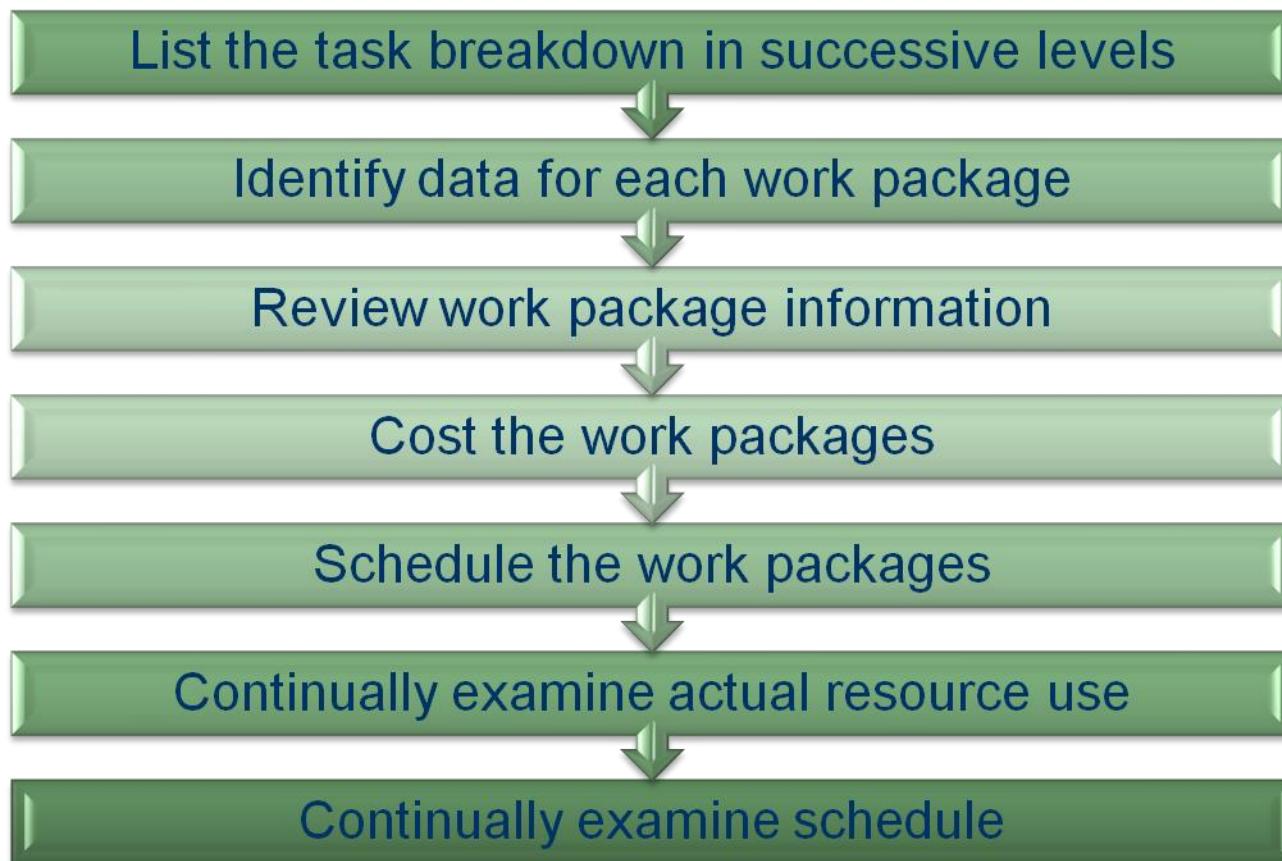


Figure 6-3

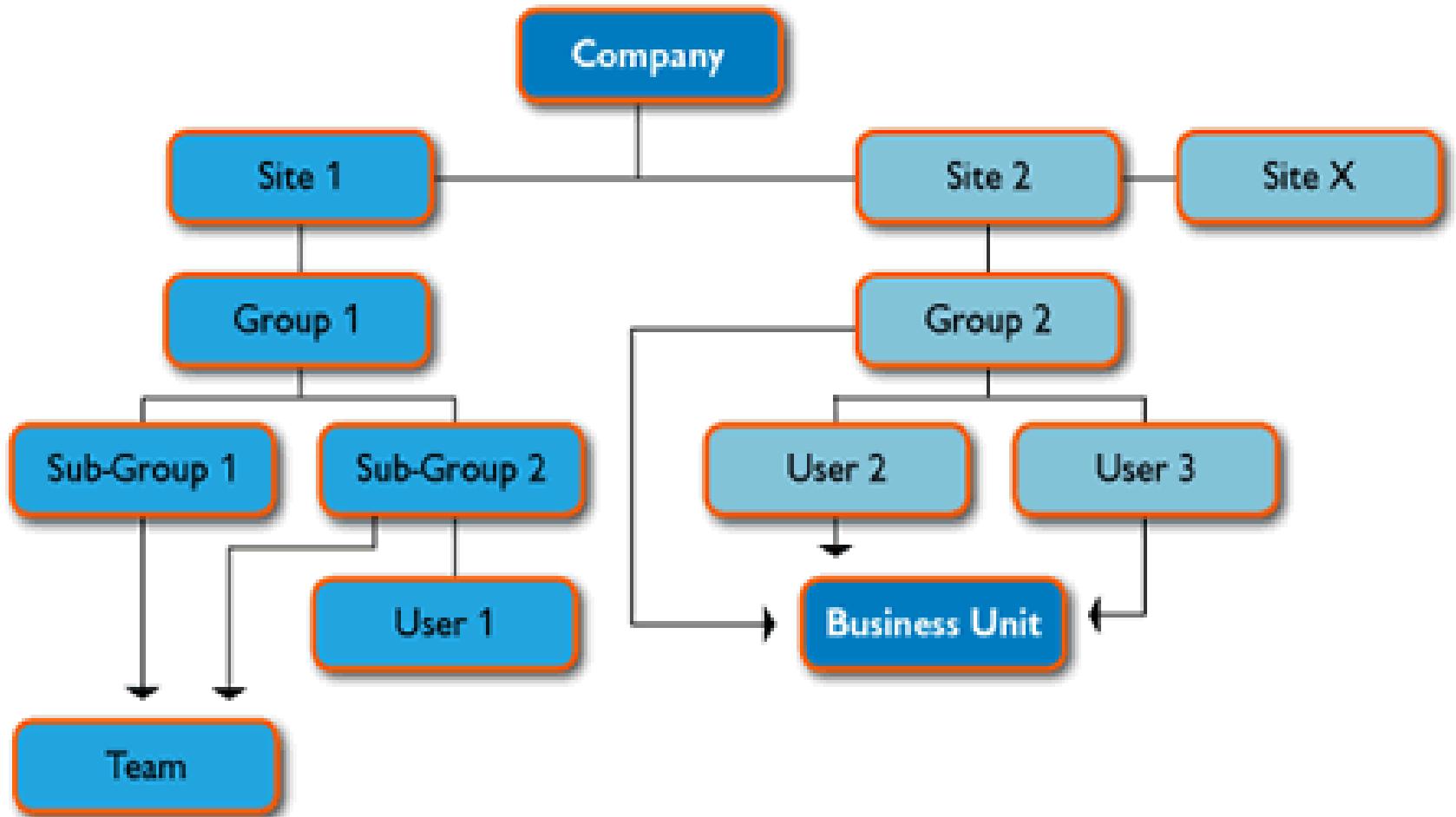
Steps to Create a WBS

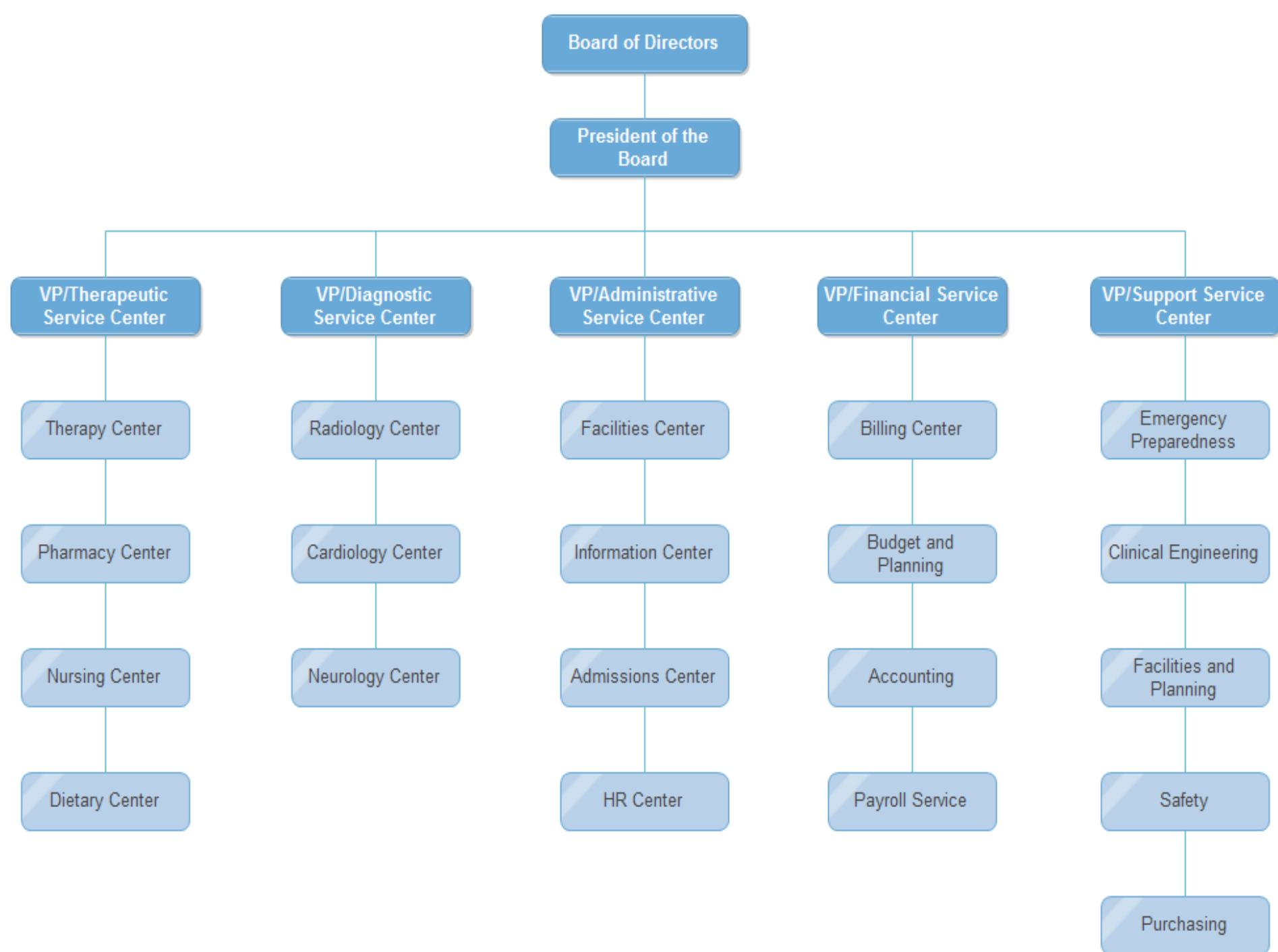


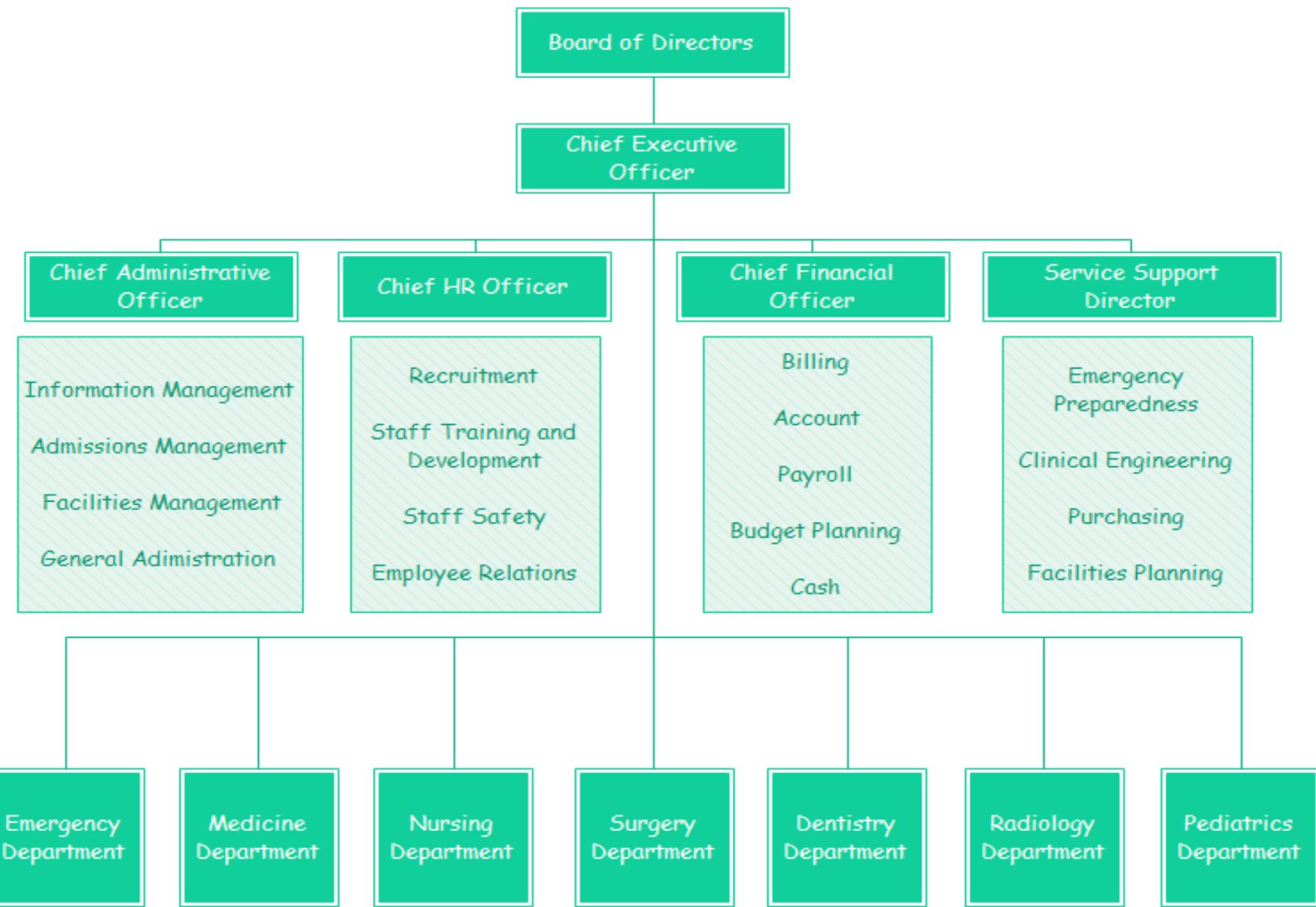
Human Resources

- Useful to create a table that shows staff needed to execute **WBS** tasks
- One approach is a **organizational breakdown structure(OBS)**
 - Organizational units responsible for each WBS element
 - Who must approve changes of scope
 - Who must be notified of progress
- WBS and OBS may not be identical

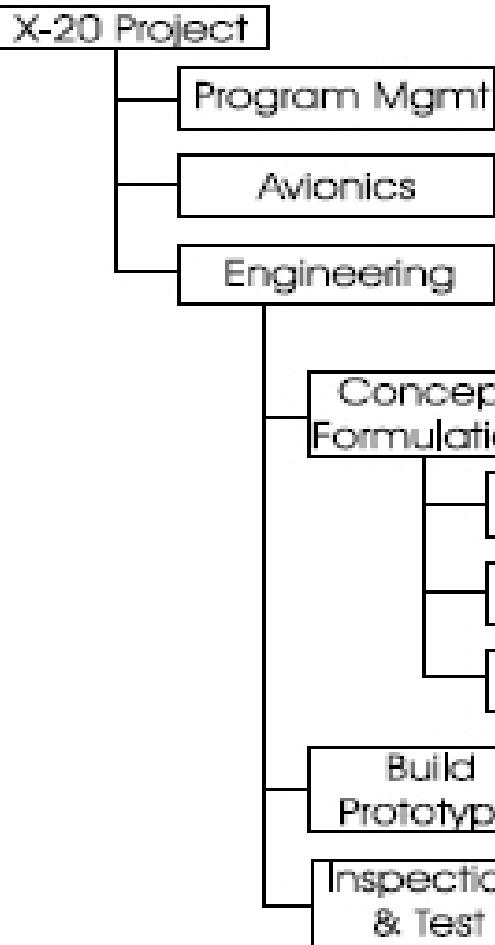
OBS



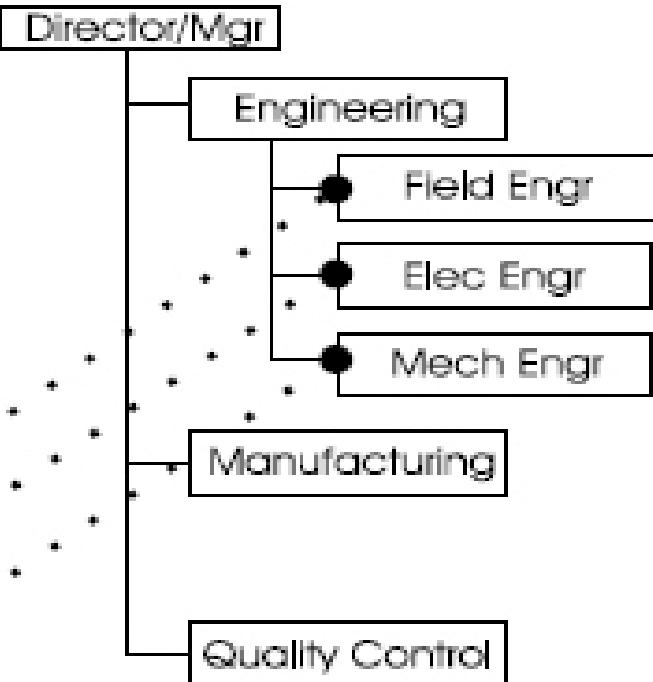




WBS Tree



OBS Tree



The Responsibility (RACI) Matrix

- Another approach is the **Responsible, Accountable, Consult, Inform (RACI)** matrix
 - Also known as a responsibility matrix, a **linear responsibility chart**, an assignment matrix, a responsibility assignment matrix
- Shows critical interfaces
- Keeps track of who must approve what and who must be notified



Dave Gappeler

"MISS WILCOX, SEND IN SOMEONE TO BLAME."



Responsible



Accountable



Consulted



Informed

Step	Project Initiation	Project Executive	Project Manager	Business Analyst	Technical Architect	Application Developers
1	Task 1	C	A/R	C	I	I
2	Task 2	A	I	R	C	I
3	Task 3	A	I	R	C	I
4	Task 4	C	A	I	R	I

Sample RACI Matrix

		Responsibility					
		Project Office				Field Oper.	
WBS	Task	Project Manager	Contract Admin.	Project Eng.	Industrial Eng.	Field Manager	
Subproject	Task	Project Manager	Contract Admin.	Project Eng.	Industrial Eng.	Field Manager	
Determine need	A1	○		●	▲		
	A2	■	○	▲	●		
Solicit quotations	B1	○	■	▲		●	
Write approp. request.	C1	■	▲	○	●		
	C2		●	○	▲		
	C3	●	■	▲		■	
"	"						
"	"						
"	"						

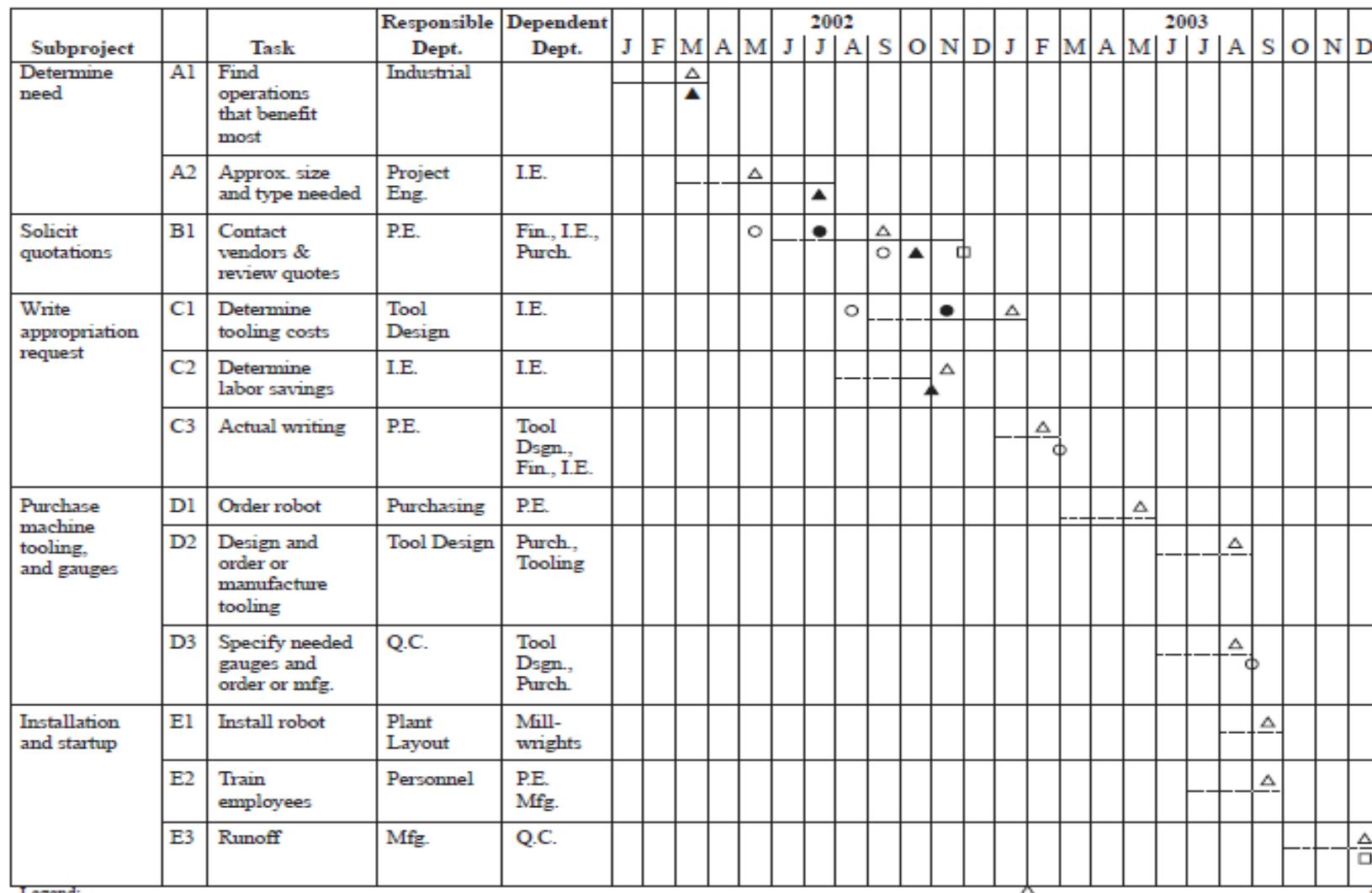
Legend:

- ▲ Responsible
- Support
- Notification
- Approval

	Vice-president	General manager	Project manager	Manager engineering	Manager software	Manager manufacturing	Manager marketing	Subprogram manager manufacturing	Subprogram manager software	Subprogram manager hardware	Subprogram manager services
Establish project plan	6	2	1	3	3	3	3	4	4	4	4
Define WBS		5	1	3	3	3	3	3	3	3	3
Establish hardware specs		2	3	1	4	4	4				
Establish software specs		2	3	4	1		4				
Establish interface specs		2	3	1	4	4	4				
Establish manufacturing specs		2	3	4	4	1	4				
Define documentation		2	1	4	4	4	4				
Establish market plan	5	3	5	4	4	4	1				
Prepare labor estimate			3	1	1	1		4	4	4	4
Prepare equipment cost estimate		3	1	1	1			4	4	4	4
Prepare material costs			3	1	1	1		4	4	4	4
Make program assignments			3	1	1	1		4	4	4	4
Establish time schedules		5	3	1	1	1	3	4	4	4	4

1 Actual responsibility
 2 General supervision
 3 Must be consulted
 4 May be consulted
 5 Must be notified
 6 Final approval

Figure 6-8 Simplified linear responsibility chart.



Legend:

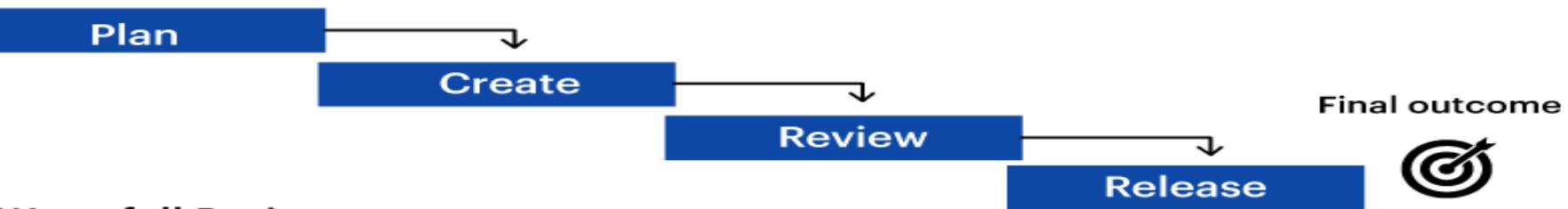
- Project completion
- Contractual commitment
- △ Planned completion
- ▲ Actual completion
- ^ Status date
- Milestone planned
- Milestone achieved
- Planned progress
- Actual progress

Note: As of Jan. 31, 2003, the project is one month behind schedule. This is due mainly to the delay in task C1, which was caused by the late completion of A2.

Figure 6-10 Projected baseline schedule.

Agile Project Planning and Management

- When scope cannot be determined in advance, traditional planning does not work
- Agile project management was developed to deal with this problem in IT
- Small teams are located at a single site
- Entire team collaborates
- Team deals with one requirement at-a-time with the scope frozen
- It is a charting system that illustrates the task's goal and the required action for each person



Agile Project



Daily Scrum

- Done since last meeting
- Plan for today
- Obstacles?

24 hours

Sprint Planning Meeting

- Review Product Backlog
- Sprint Goal
- Estimate Sprint Backlog
- Commit to the Sprint

Backlog tasks

expanded by team

Sprint 1-4 weeks

Sprint Review

- Demo features to all
- Discuss what's done and what wasn't done

Sprint Retrospective

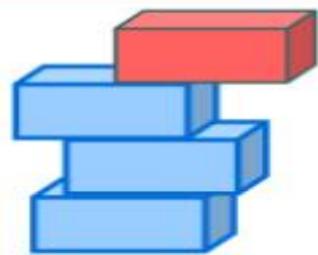
- Inspect and Adapt



Potentially Shippable Product Increment



Product Backlog
Prioritized Features desired by Customer



Sprint Backlog

Features assigned to Sprint
Estimated by team

Agile planning – iteration

...

#1 iteration: weeks 1-2

	Owner	Status	Priority	Due date	
Review dev environment		Done	High	Jan 6	
Plan release		Done	Low	Jan 9	
Analyze deployment's data		Done	Mid	Jan 13	

#1 iteration: weeks 3-4

	Owner	Status	Priority	Due date	
Release plan review		For review	High	Jan 19	
Build release		Working on it	Low	Jan 23	
Test plan approval		For review	Mid	Jan 16	
Test user acceptance		Ongoing	High	Jan 26	

#1 Iteration - Week 1...er	Status	Priority	Type	Deadline	Time Est.	Time Spent
Decide Business ...	Done	Must Have	MILESTONE!	✓ Feb...	5 Hours	5 Hours
Feasibility Study	Working on it	High	Improvements	! Feb ...	10 Hours	2 Hours
plan	Not Started	MILESTONE	MILESTONE!	Feb ...	7 Hours	6.75 Hours
Design	Waiting for Review	High	Increment			
+ Add						
					22 Hours sum	13.75 Hours sum
#2 Iteration - Week 3...er	Status	Priority	Type	Deadline	Time Est.	Time Spent
Release Plan Revie...	Working on it	High	Increment	! Jan ...	2 Hours	
Build Release	Needs Design	Low	Increment	! Jan ...	6 Hours	
Test Plan Approval	Needs Product	Medium	Increment		12 Hours	
Test User Accept...	Not Started	Must Have	Increment		14 Hours	
+ Add					34 Hours sum	0 Hours sum

Project Management-How to use projectlibre.com for time line chart

- <https://www.youtube.com/watch?v=oiVnWX-J5Mo>

Project Scheduling-Timeline Charts

- When creating a software project schedule, the planner begins with a set of tasks (the work breakdown structure)
- If automated tools are used, the work breakdown is input as a task network or task outline
- Effort, duration, and start date are then input for each task
- In addition, tasks may be assigned to specific individuals
- As a consequence of this input, a timeline chart, also called a Gantt chart, is generated

Agile Project Plan MS Project

Task Name	Duration	Start	Finish	Resource	January 2017	February 2017	March 2017												
				Name	18	23	28	2	7	12	17	22	27	1	6	11	16	21	26
▪ User Stories	15 days	Mon 12/19/16	Fri 1/6/17																
Identify Key stakeholders	5 days	Mon 12/19/16	Fri 12/23/16	Business Analyst															
Form project team	15 days	Mon 12/19/16	Fri 1/6/17	Project Manager															
User Story workshops - 1	10 days	Mon 12/19/16	Fri 12/30/16	Business Analyst															
User Story workshops - 2	10 days	Mon 12/19/16	Fri 12/30/16	Business Analyst															
User Stories Walk-through and sign off	10 days	Mon 12/19/16	Fri 12/30/16	Business Analyst															
▪ Product Backlog	5 days	Mon 1/9/17	Fri 1/13/17																
Create Product Backlog	5 days	Mon 1/9/17	Fri 1/13/17	Project Manager															
Story Estimation	5 days	Mon 1/9/17	Fri 1/13/17	Project Team															
Prioritize	5 days	Mon 1/9/17	Fri 1/13/17	Project Manager															
▪ High Level Sprint Planning	5 days	Mon 1/16/17	Fri 1/20/17																
Create project timeline	5 days	Mon 1/16/17	Fri 1/20/17	Project Manager															
Draft resource plan	5 days	Mon 1/16/17	Fri 1/20/17	Project Manager															
Plan project budget	5 days	Mon 1/16/17	Fri 1/20/17	Project Manager															
▪ Sprint - 1	10 days	Mon 1/23/17	Fri 2/3/17																
Sprint 1 - Planning	10 days	Mon 1/23/17	Fri 2/3/17	Project Team															
Sprint 1 - Execution	10 days	Mon 1/23/17	Fri 2/3/17	Project Team															
Sprint 1 - Demo	10 days	Mon 1/23/17	Fri 2/3/17	Project Team															
Sprint 1 - Implementation	10 days	Mon 1/23/17	Fri 2/3/17	Project Team															
Sprint 1 - Retrospective	10 days	Mon 1/23/17	Fri 2/3/17	Project Team															
▪ Sprint - 2	10 days	Mon 2/6/17	Fri 2/17/17																
Sprint 2 - Planning	10 days	Mon 2/6/17	Fri 2/17/17	Project Team															
Sprint 2 - Execution	10 days	Mon 2/6/17	Fri 2/17/17	Project Team															

Techno-PM

Project Management Templates

-
- 1. Analysis 2 days 7th May(Friday) 8 am to 5pm 10th May(Monday)(18 hrs)**
 - 2. Design 3 days 11th May to 14th May**

Interface Coordination Through Integration Management

- Managing a project requires a great deal of coordination
- Projects typically draw from many parts of the organization as well as outsiders
- All of these must be coordinated
- The RACI matrix helps the project manager accomplish this

Integration Management

- If the project is large multidisciplinary team(MT) will work together
- Integration will be Coordinating the work and timing of different groups
- RACI Matrix will be useful here as conflict and uncertainty can be arise in MT(e,g, Bank merging)
- Divide the project in to phases and then accomplish the task by teams

Managing Projects by Phases and Phase-Gates

- Break objectives into shorter term sub-objectives
- Project life cycle is used for breaking a project up into component **phases**
- Focus on specific, short-term output
- Lots of feedback between disciplines
- **Phase-Gate** : create different phases and keep “Review” as a gate to proceed in next phase

Risk Management

- Projects are risky, uncertainty is high
- Project manager must manage this risk
- This is called “risk management”
- Risk varies widely between projects
- Risk also varies widely between organizations
- Risk management should be built on the results of prior projects

Parts to Risk Management

- Risk management planning
- Risk identification
- Qualitative risk analysis
- Quantitative risk analysis
- Risk response planning
- Risk monitoring and control
- The risk management register

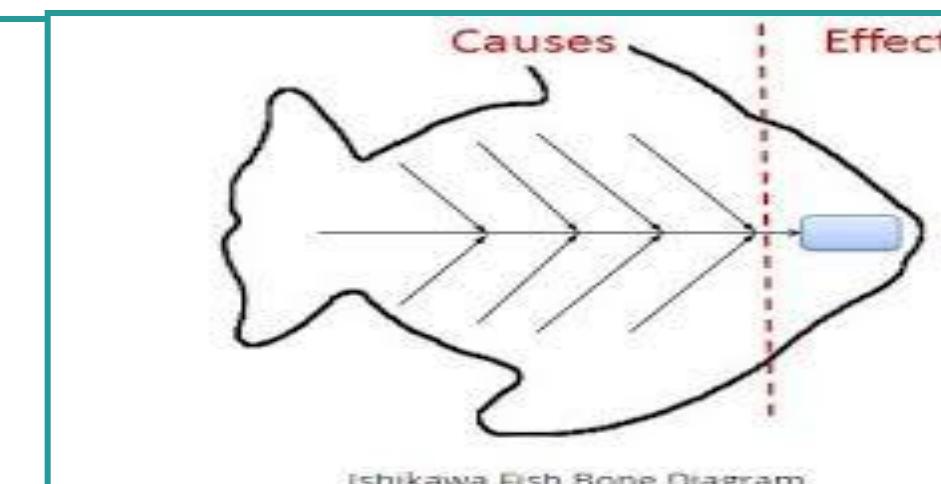
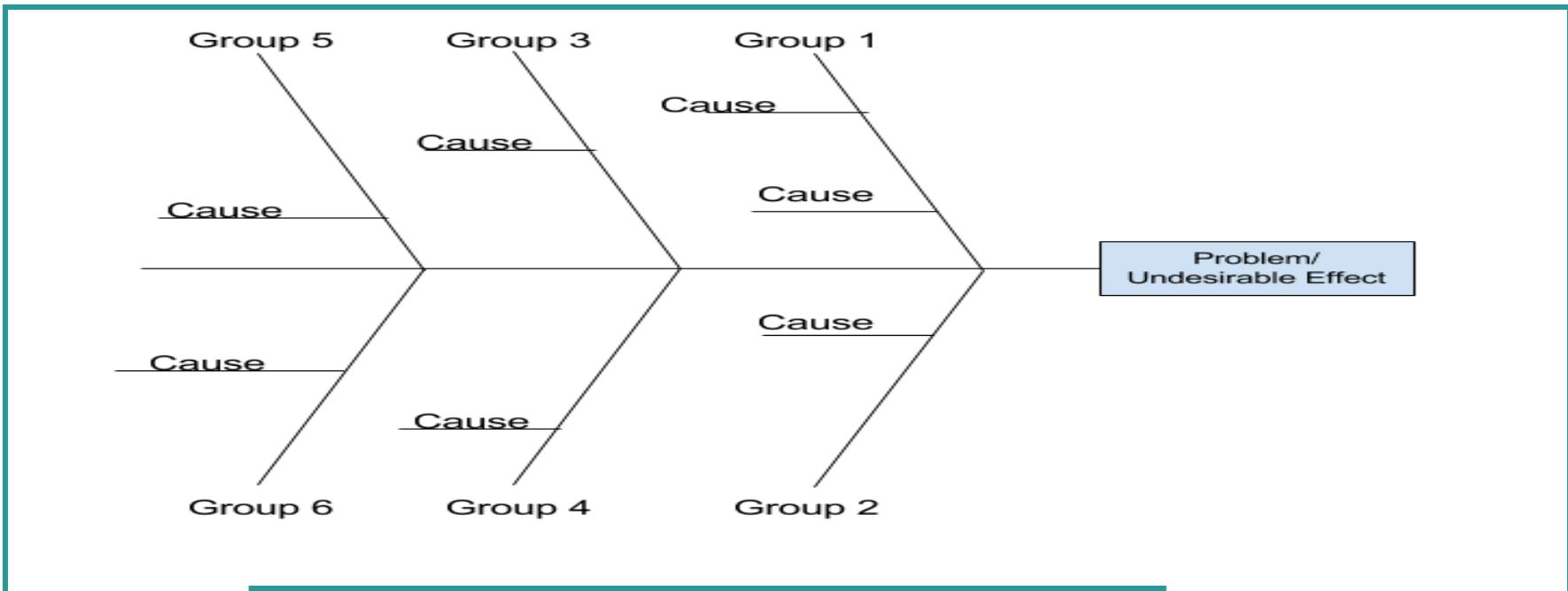
Risk Management Planning

- Need to know the risk involved before selecting a project
- Risk management plan must be carried out before the project can be formally selected
- At first, focus is on externalities
 - Track and estimate project survival
- Project risks take shape during planning
- Often handled by project office
- May include analytical techniques.

Risk Identification

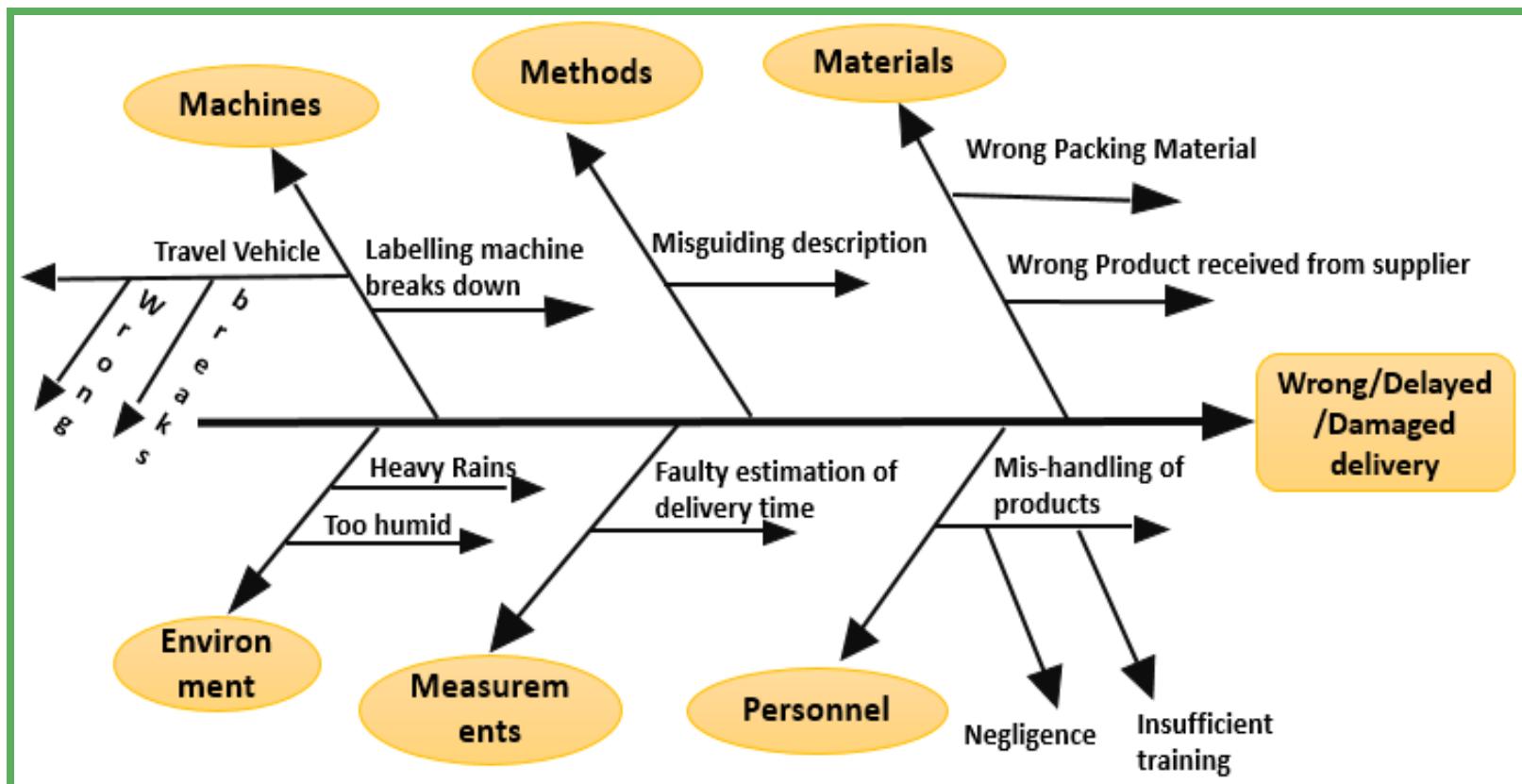
- Risk is dependent on technology and environmental factors e.g. economic, cultural factors
- **Delphi method** is useful for identifying project risks
- Other methods include brainstorming, nominal group techniques, checklists, and attribute listing
- May also use **cause-effect diagrams(fish bone diagram)**, flow charts, influence charts, **SWOT analysis**

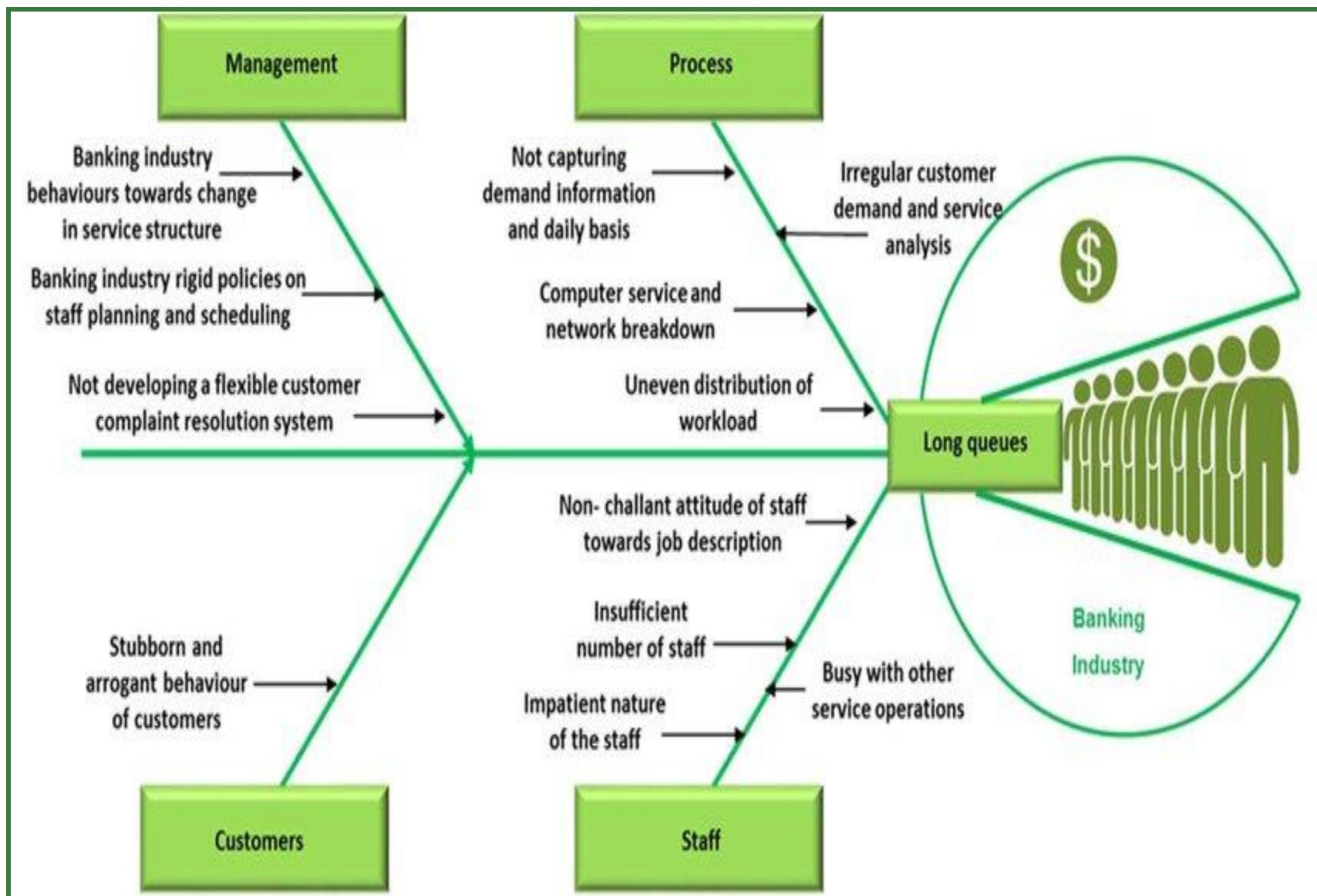
Risk Identification(Cause-Effect Analysis-Fish Bone Diagram)



Ishikawa Fish Bone Diagram

Fish bone diagram for “Delayed/Damage Delivery”

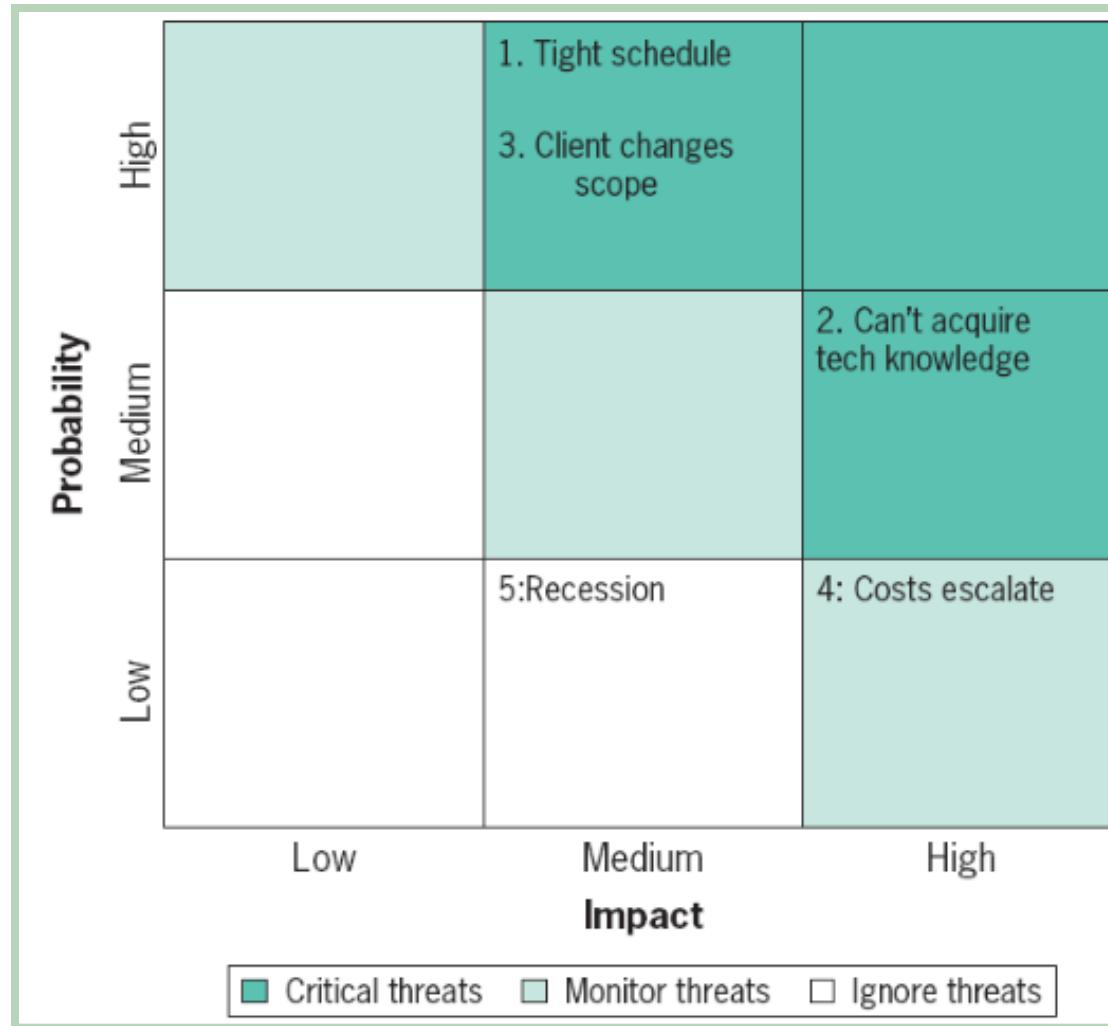




Qualitative Risk Analysis

- Purpose is to prioritize risks
- Identify the **probability of occurrence** of risk and a sense of the **impact** is also needed(e.g. low, medium, high)
- Each objective should be scaled and weighted
- Construct a risk matrix
- Same approach can be used for opportunities

Risk Matrix



		CONSEQUENCES				
		Negligible	Minor	Moderate	Significant	Severe
PROBABILITY	Almost Certain (81%-100%)	Low Risk	Moderate Risk	High Risk	Extreme Risk	Extreme Risk
	Likely (61%-80%)	Minimum Risk	Low Risk	Moderate Risk	High Risk	Extreme Risk
	Moderate (41%-60%)	Minimum Risk	Low Risk	Moderate Risk	High Risk	High Risk
	Unlikely (21%-40%)	Minimum Risk	Low Risk	Low Risk	Moderate Risk	High Risk
	Rare (1%-20%)	Minimum Risk	Minimum Risk	Low Risk	Moderate Risk	High Risk

Quantitative Risk Analysis

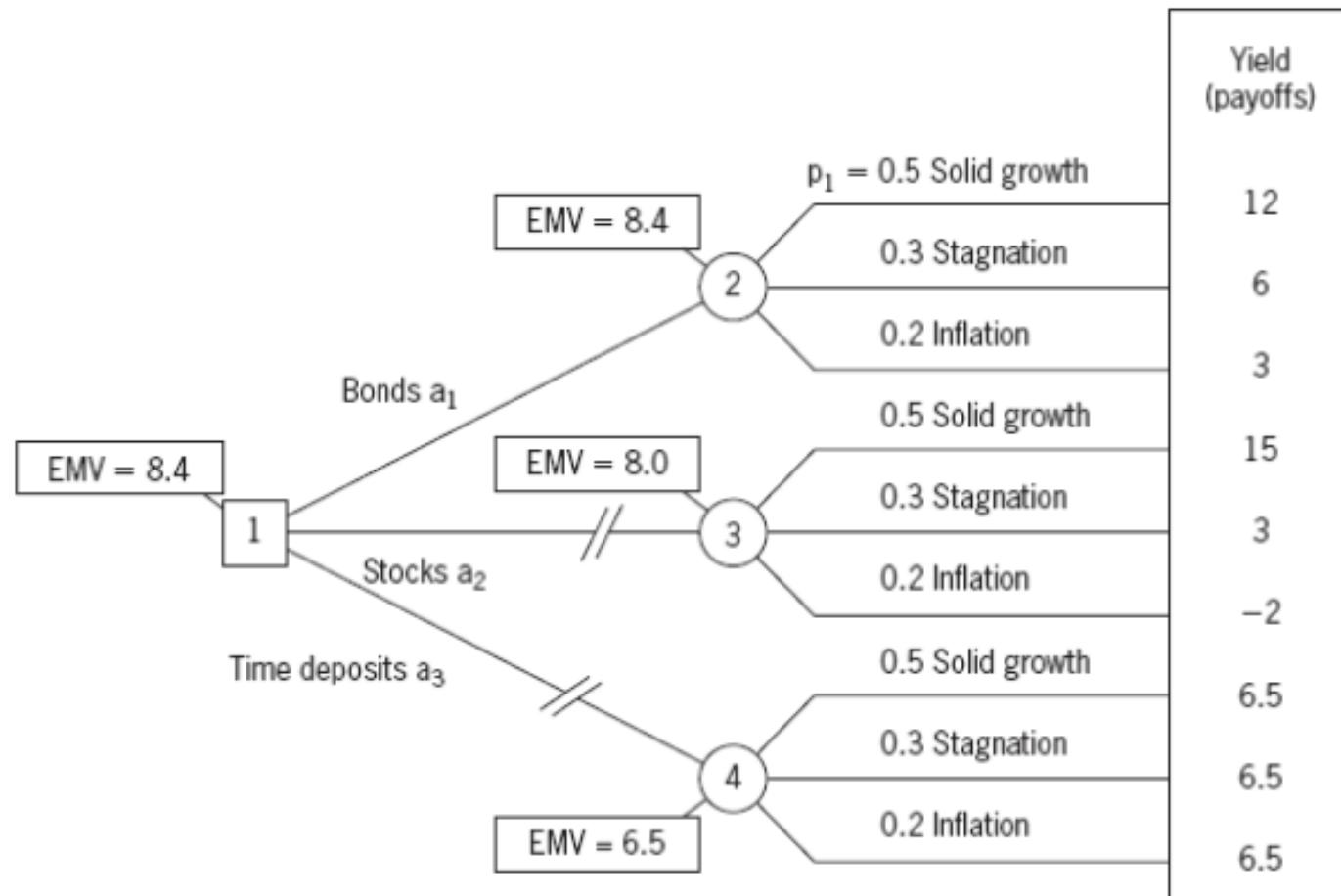
1. List ways a project can fail
2. Evaluate severity (**S**) by “*Failure mode and effect analysis*”(**FMEA**) **FORM 1-10 POINTS**
3. Estimate likelihood(**L**) from 1 -10 points
4. Estimate the inability to detect (**D**)
5. Find the *risk priority number* (RPN)
$$(\text{RPN} = S \times L \times D)$$
6. Consider ways to reduce the S, L, and D for each cause of failure

A FMEA Example

<i>Threat</i>	<i>Severity, S</i>	<i>Likelihood, L</i>	<i>Ability to Detect, D</i>	<i>RPN</i>
1. Tight schedule	6	7.5	2	90
2. Can't acquire tech knowledge	8.5	5	4	170
3. Client changes scope	4	8	5	160
4. Costs escalate	3	2	6	36
5. Recession	4	2.5	7	70

Highest is 170 i.e. “cant acquire tech knowledge”,
Now try to reduce this risk by taking appropriate action

Decision Tree Analysis



The Risk Management Register

- Environments that may impact projects
- Assumptions made
- Risks identified
- List of *categories* and *key words*
- Estimates on risk, states of project's environment, or on project assumptions
- Minutes
- Actual outcomes

Budgeting: Estimating Costs and Risks

Budgeting

- A plan for the costs of project resources
- A budget implies constraints
- Thus, it implies that managers will not get everything they want or need

Budgeting

Continued

- The budget for an activity also implies management support for that activity
- Higher the budget, relative to cost, higher the managerial support
- The budget is also a control mechanism
 - Many organizations have controls in place that prohibit exceeding the budget
 - Comparisons are against the budget

Estimating Project Budgets

- On most projects
 - Material + Labor + Equipment + Capital + Overhead + Profits = Bid
- In other words
 - Resources + Profits = Bid
- So we are left with the task of forecasting resources

Estimating Project Budgets Continued

- Like any forecast, this includes some uncertainty
- There is uncertainty regarding usage and price
 - Especially true for material and labor
- The more standardized the project and components, the lower the uncertainty
- The more experienced the cost estimator, the lower the uncertainty

Rules of Thumb

- Some estimates are prepared by rules of thumb
 - Construction cost by square feet
 - Printing cost by number of pages
 - Lawn care cost by square feet of lawn
- These rules of thumb may be adjusted for special conditions
- However, this is still easier than starting the estimate from scratch

Estimating Budgets is Difficult

- There may not be as much historical data or none at all
- Even with similar projects, there may be significant differences
- Many people have input to the budget

Estimating Budgets is Difficult Continued

- Multiple people have some control over the budget
- There is more “flexibility” regarding the estimates of inputs (material and labor)
- The accounting system may not be set up to track project data
- Usage of labor and material is very lumpy over time

Types of Budgeting

- Top-down
- Bottom-up
- Negotiated

Top-Down Budgeting

- Top managers estimate/decide on the overall budget for the project
- These trickle down through the organization where the estimates are broken down into greater detail at each lower level
- The process continues to the bottom level

Advantages

- Overall project budgets can be set/controlled very accurately
 - A few elements may have significant error
- Management has more control over budgets
- Small tasks need not be identified individually

Disadvantages

- More difficult to get buy in
- Leads to low level competition for larger shares of budget

Bottom-Up Budgeting

- Project is broken down into work packages
- Low level managers price out each work package
- Overhead and profits are added to develop the budget

Advantages

- Greater buy in by low level managers
- More likely to catch unusual expenses

Disadvantages

- People tend to overstate their budget requirements
- Management tends to cut the budget

Work Element Costing

- Labor rates include overhead and personal time
- Direct costs usually do not include overhead
- General and administrative (G&A) charge

An Iterative Budgeting Process— Negotiation-in-Action

- Most projects use some combination of top-down and bottom-up budgeting
- Both are prepared and compared
- Any differences are negotiated

Category Budgeting Versus Program/Activity Budgeting

- Organizations are used to budgeting (and collecting data) by activity
- These activities correspond to “line items” in the budget
 - Examples include phone, utilities, direct labor,...
- Projects need to accumulate data and control expenses differently
- This resulted in program budgeting

Typical Monthly Budget

	<i>Current</i>			
	<i>Actual</i>	<i>Budget</i>	<i>Variance</i>	<i>Pct.</i>
<i>Corporate—Income Statement</i>				
Revenue				
8430 Management fees				
8491 Prtnsp reimb—property mgmt	7,410.00	6,222.00	1,188.00	119.0
8492 Prtnsp reimb—owner acquisition	.00	3,750.00	3,750.00—	.0
8493 Prtnsp reimb—rehab	.00	.00	.00	.0
8494 Other income	.00	.00	.00	.0
8495 Reimbursements—others	.00	.00	.00	.0
Total revenue	7,410.00	9,972.00	2,562.00—	74.3
<i>Operating expenses</i>				
Payroll & P/R benefits				
8511 Salaries	29,425.75	34,583.00	5,157.25	85.0
8512 Payroll taxes	1,789.88	3,458.00	1,668.12	51.7
8513 Group ins & med reimb	1,407.45	1,040.00	387.45—	135.3
8515 Workmen's compensation	43.04	43.00	.04—	100.0
8516 Staff apartments	.00	.00	.00	.0
8517 Bonus	.00	.00	.00	.0
Total payroll & P/R benefits	32,668.12	39,124.00	6,457.88	83.5
Travel & entertainment expenses				
8512 Travel	456.65	300.00	156.65—	152.2
8522 Promotion, entertainment & gift	69.52	500.00	430.48	13.9
8523 Auto	1,295.90	1,729.00	433.10	75.0
Total travel & entertainment exp	1,822.07	2,529.00	706.93	72.1
Professional fees				
8531 Legal fees	419.00	50.00	369.00—	838.0
8532 Accounting fees	289.00	.00	289.00—	.0
8534 Temporary help	234.58	200.00	34.58—	117.2

Table 7-1

Project Budget by Task & Month

Task	Estimate	Monthly Budget (£)							
		1	2	3	4	5	6	7	8
A	7000	5600	1400						
B	9000		3857	5143					
C	10000		3750	5000	1250				
D	6000		3600	2400					
E	12000				4800	4800	2400		
F	3000				3000				
G	9000			2571	5143	1286			
H	5000					3750	1250		
I	8000						2667	5333	
J	6000								6000
	75000	5600	12607	15114	14193	9836	6317	5333	6000

Source: Harrison, 1983.

Improving The Process of Cost Estimation

- Inputs from a lot of areas are required to estimate a project
- May have a professional cost estimator to do the job
- Project manager will work closely with cost estimator when planning a project
- We are primarily interested in estimating direct costs
- Indirect costs are not a major concern

Problems

- Even with careful planning, estimates are wrong
- Most firms add 5-10 percent for contingencies

Learning Curves

- Human performance usually improves when a task is repeated
- This happens by a fixed percent each time the production doubles
- Percentage is called the learning rate

Learning Curve Calculations

$$T_n = T_1 n^r$$

$$r = \frac{\log(\text{rate})}{\log 2}$$

$$\text{Total time} = T_1 \sum_{n=1}^N n^r$$

T_n = Time for n th unit
 T_1 = Time for first unit
 n = Number of units
 r = log decimal
rate/log 2

Other Factors

- Escalation
- Waste
- Bad Luck

Making Better Estimates

- Projects are known for being over budget
 - It is unlikely that this is due to deliberate underestimating
- There are two types of errors
 - Random
 - Bias
- There is nothing we can do about random errors
 - Tend to cancel each other
- Eliminate systematic errors

Risk Estimation

- Duration of project activities varies
- Amounts of various resources needed varies
- Value of accomplishing a project varies
- Can reduce but not eliminate ambiguity
- Want to describe uncertainties in a way that provides useful insight to their nature

Applying Risk Analysis

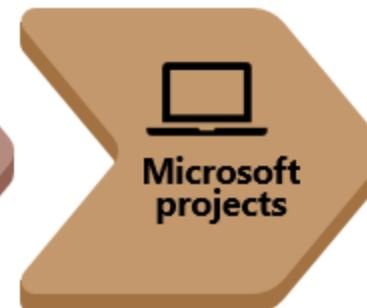
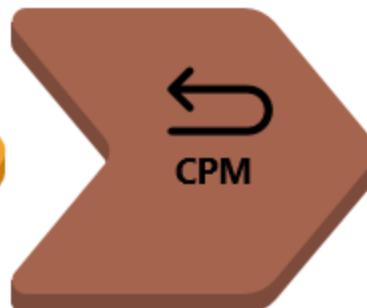
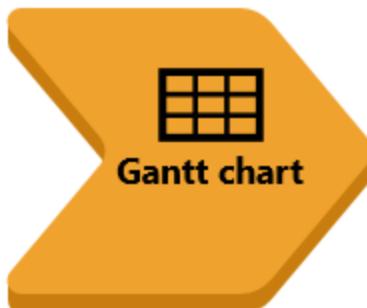
- Must make assumptions about probability distributions
 - Key parameters
 - Variables
- Estimate the risk profiles of the outcomes of the decision
 - Also known as *probability distributions*
- Simulation is often used

General Simulation Analysis

- Simulation combined with sensitivity analysis is useful for evaluating projects
- Would support project if NPV is positive and is the best use of funds
- Should avoid full-cost philosophy
 - Some overheads are not affected by changes
- Analysis gives a picture in terms of costs and times that will be affected
- Project is then reviewed using simulation

Scheduling

Project Scheduling Techniques



Useful Abbreviations

- **CPM** - Critical Path Method
- **PERT** - Program Evaluation and Review Technique

Background

- Schedule is the conversion of a project action plan into an operating timetable
- Basis for monitoring a project
- One of the major project management tools
- Work changes daily, so a detailed plan is essential
- Not all project activities need to be scheduled at the same level of detail

Background Continued

- Most of the scheduling is at the WBS level, not the work package level
- Only the most critical work packages may be shown on the schedule
- Most of the scheduling is based on network drawings

Network Scheduling Advantage

- Consistent framework
- Shows interdependences
- Shows when resources are needed
- Ensures proper communication
- Determines expected completion date
- Identifies critical activities

Network Scheduling Advantage

Continued

- Shows which of the activities can be delayed
- Determines start dates
- Shows which task must be coordinated
- Shows which task can be run parallel
- Relieves some conflict
- Allows probabilistic estimates

Network Scheduling Techniques: PERT (ADM) and CPM (PDM)

- PERT was developed for the Polaris missile/submarine project in 1958
- CPM developed by DuPont during the same time
- Initially, CPM and PERT were two different approaches
 - CPM used deterministic time estimates and allowed project crunching
 - PERT used probabilistic time estimates
- Microsoft Project (and others) have blended CPM and PERT into one approach

Terminology

- **Activity** - A specific task or set of tasks that are required by the project, use up resources, and take time to complete
- **Event** - The result of completing one or more activities
- **Network** - The combination of all activities and events that define a project
 - Drawn left-to-right
 - Connections represent predecessors

Terminology

Continued

- **Path** - A series of connected activities
- **Critical** - An activity, event, or path which, if delayed, will delay the completion of the project
- **Critical Path** - The path through the project where, if any activity is delayed, the project is delayed
 - There is always a critical path
 - There can be more than one critical path

Terminology

Continued

- **Sequential Activities** - One activity must be completed before the next one can begin
- **Parallel Activities** - The activities can take place at the same time
- **Immediate Predecessor** - That activity that must be completed just before a particular activity can begin

Terminology

Continued

- **Activity on Arrow** - Arrows represent activities while nodes stand for events
- **Activity on Node** - Nodes stand for events and arrows show precedence

AON and AOA Format

Figure 8-2

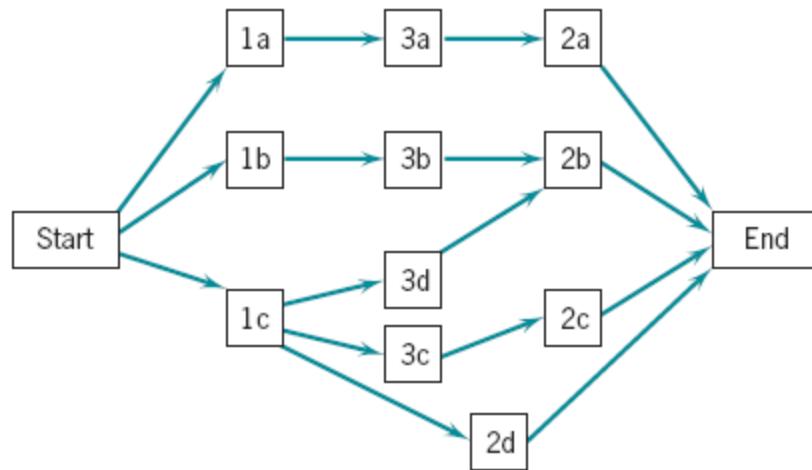
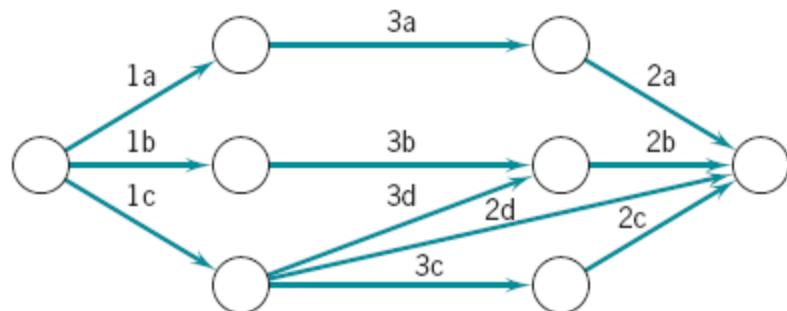


Figure 8-3



Constructing the Network

- Begin with START activity
- Add activities without precedences as nodes
 - There will always be one
 - May be more
- Add activities that have those activities as precedences
- Continue

Gantt (Bar) Charts

- Developed by Henry L. Gantt
- Shows planned and actual progress
- Easy-to-read method to know the current status

Advantages and Disadvantage

- Advantages
 - Easily understood
 - Provide a picture of the current state of a project
- Disadvantage
 - Difficult to follow complex projects

Microsoft Project Gantt Chart

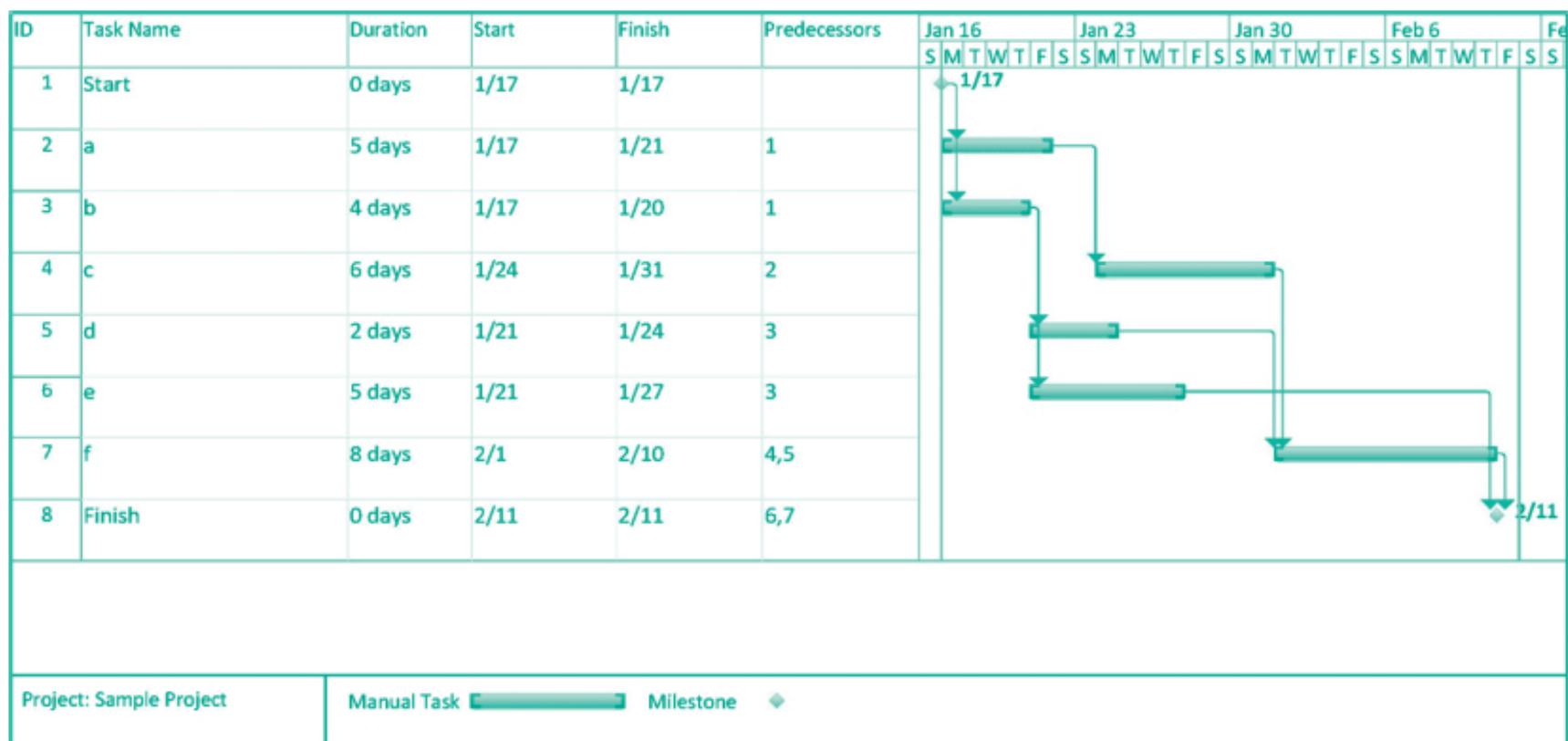


Figure 8-11

Microsoft Project AON Network

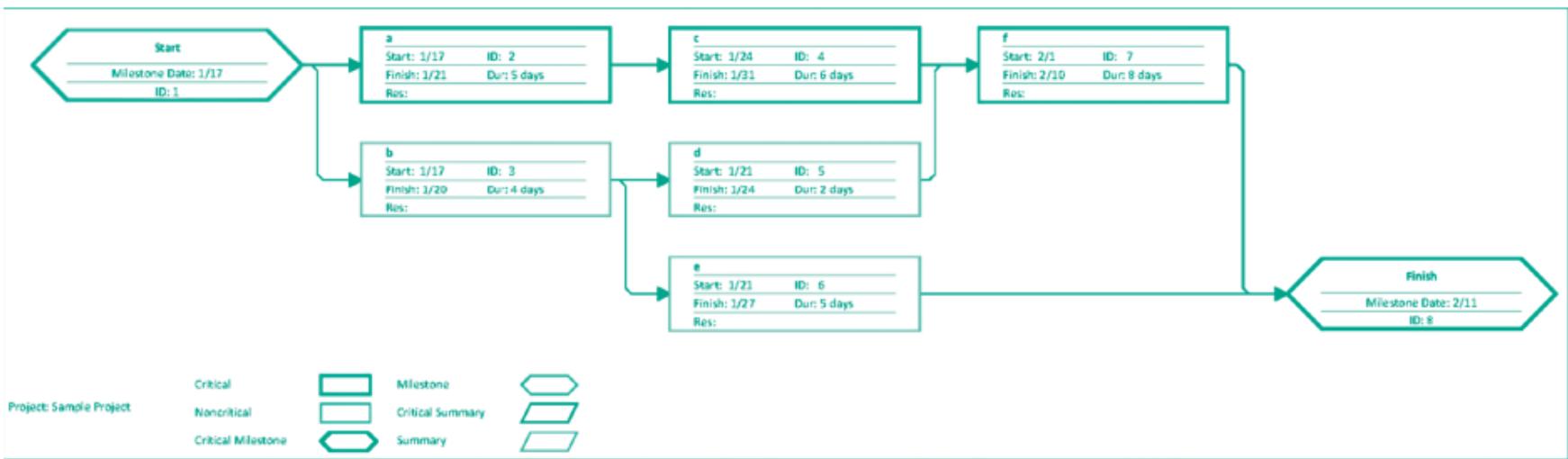


Figure 8-12

Solving the Network

<i>Activity</i>	<i>Optimistic Time</i>	<i>Most Likely Time</i>	<i>Pessimistic Time</i>	<i>Immediate Predecessor Activities</i>
a	10	22	22	—
b	20	20	20	—
c	4	10	16	—
d	2	14	32	a
e	8	8	20	b, c
f	8	14	20	b, c
g	4	4	4	b, c
h	2	12	16	c
i	6	16	38	g, h
j	2	8	14	d, e

The AON Network from the previous table

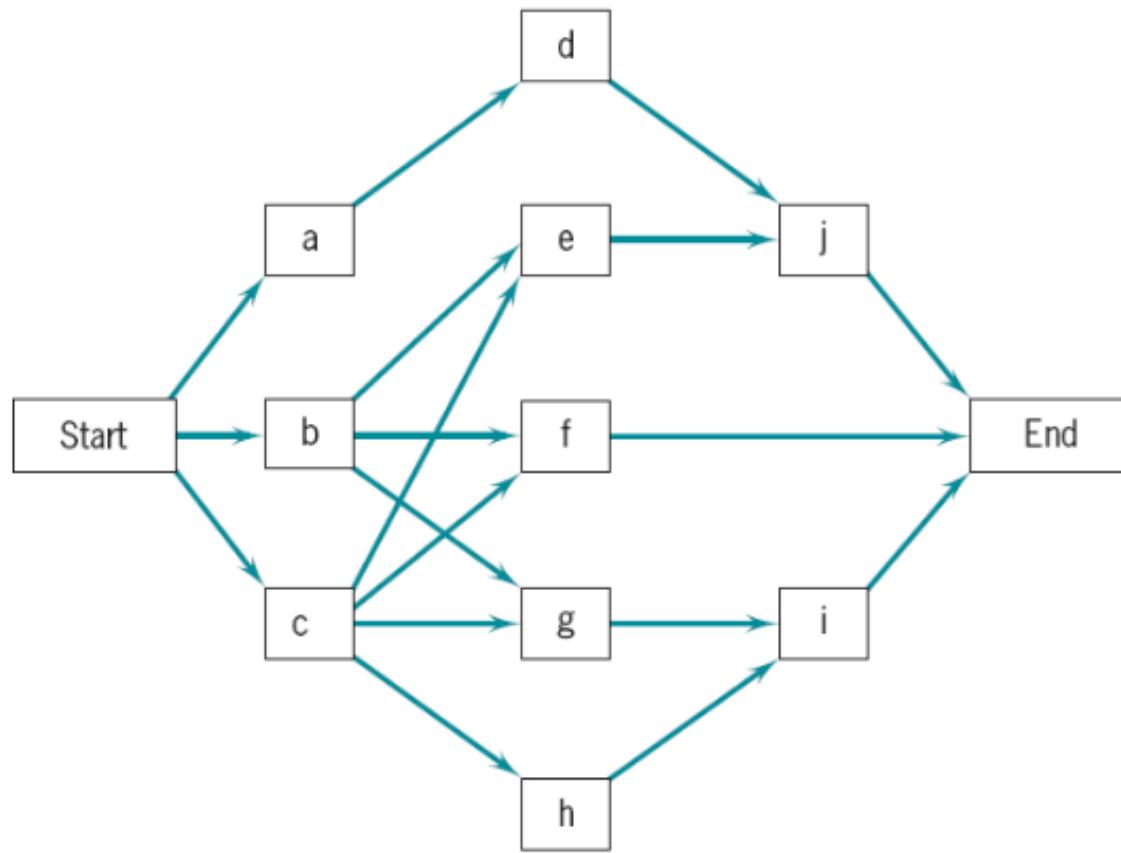


Figure 8-13

Calculating Activity Times

$$TE = \frac{(a + 4m + b)}{6}$$

$$\sigma^2 = \left(\frac{(b - a)}{6} \right)^2$$

$$\sigma = \sqrt{\sigma^2}$$

The Results

<i>Activity</i>	<i>Expected Time, TE</i>	<i>Variance, σ^2</i>	<i>Standard Deviation, σ</i>
a	20	4	2
b	20	0	0
c	10	4	2
d	15	25	5
e	10	4	2
f	14	4	2
g	4	0	0
h	11	5.4	2.32
i	18	28.4	5.33
j	8	4	2

Table 8-2

Critical Path and Time

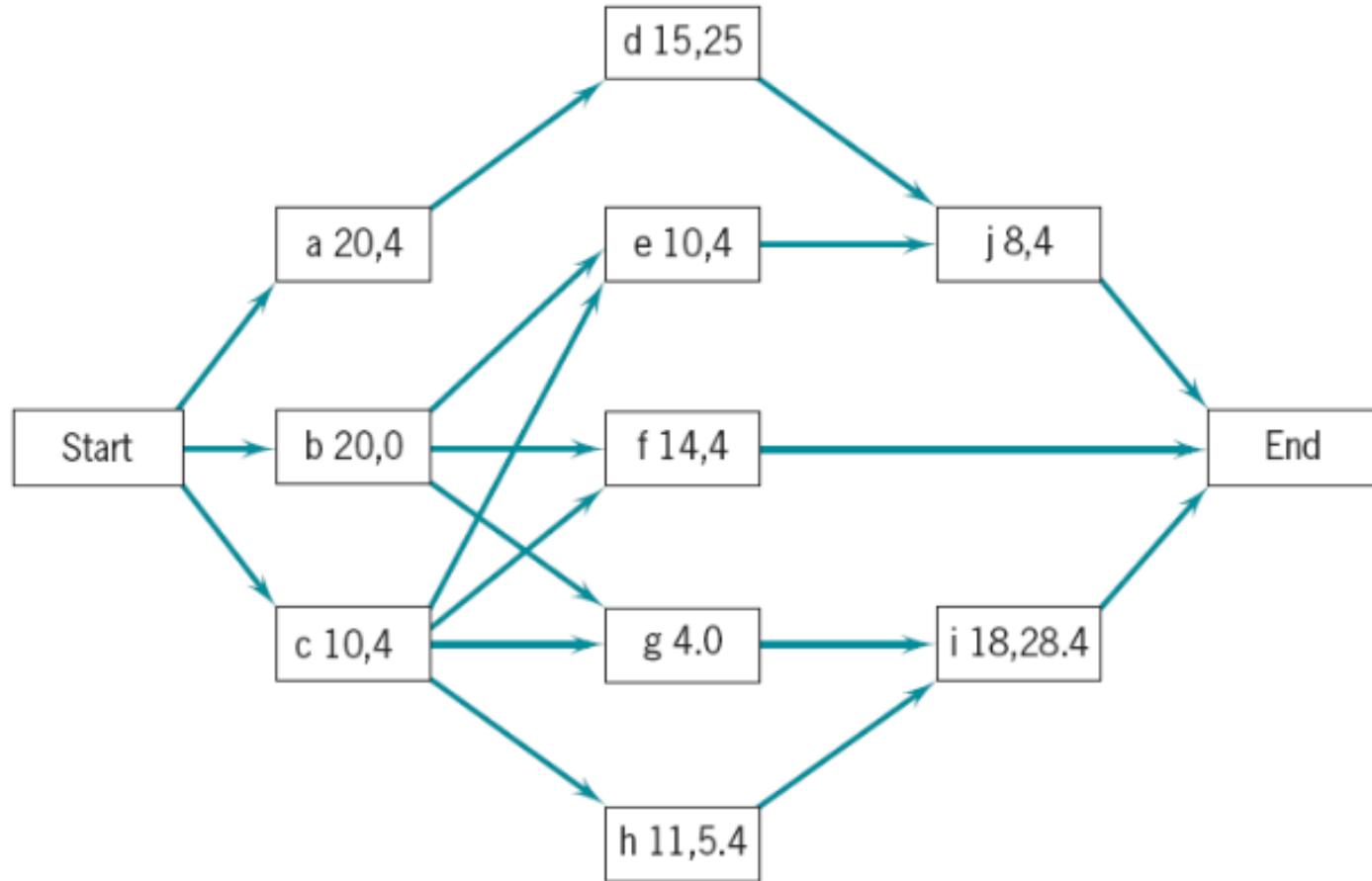


Figure 8-15

Critical Path and Time Continued

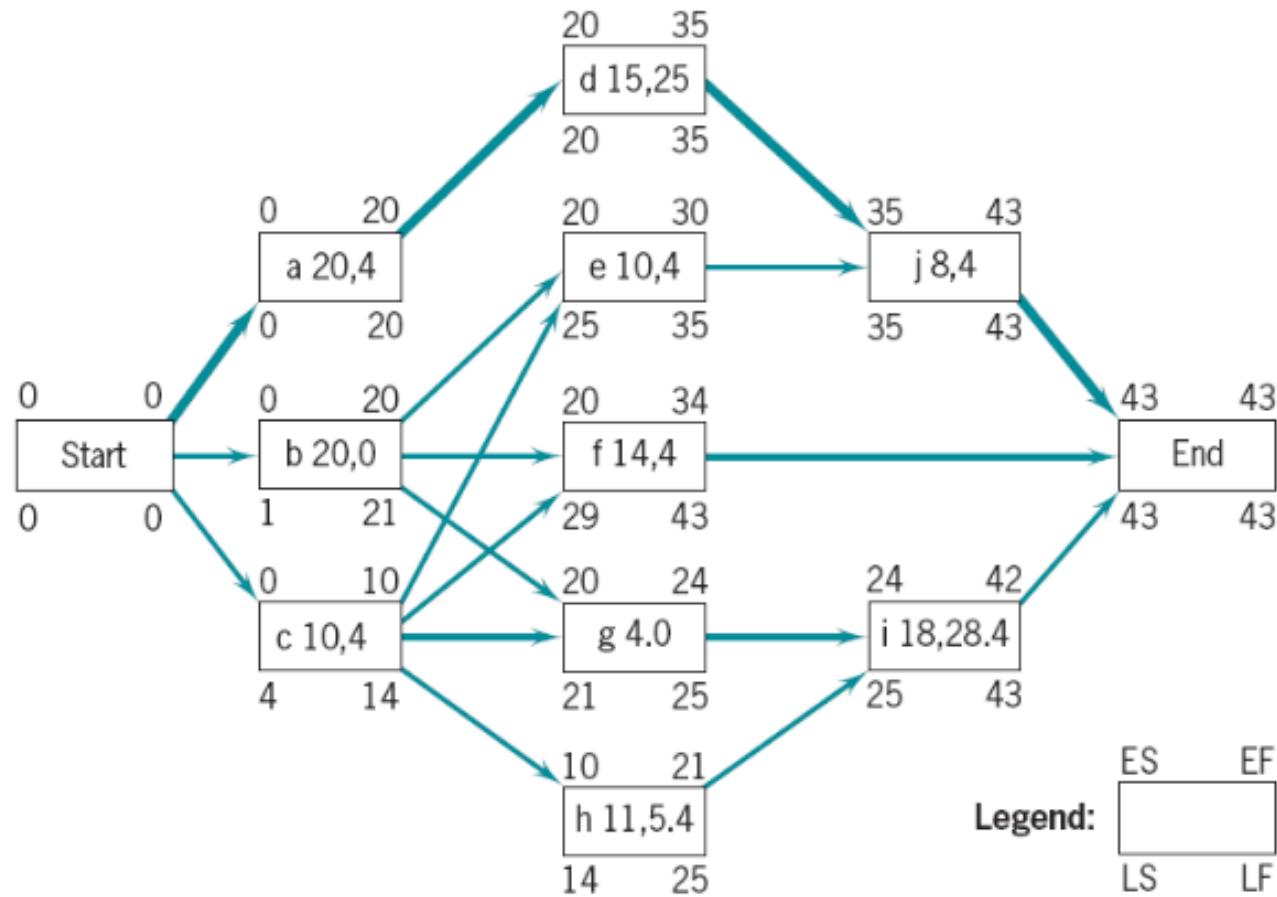


Figure 8-16

Slack

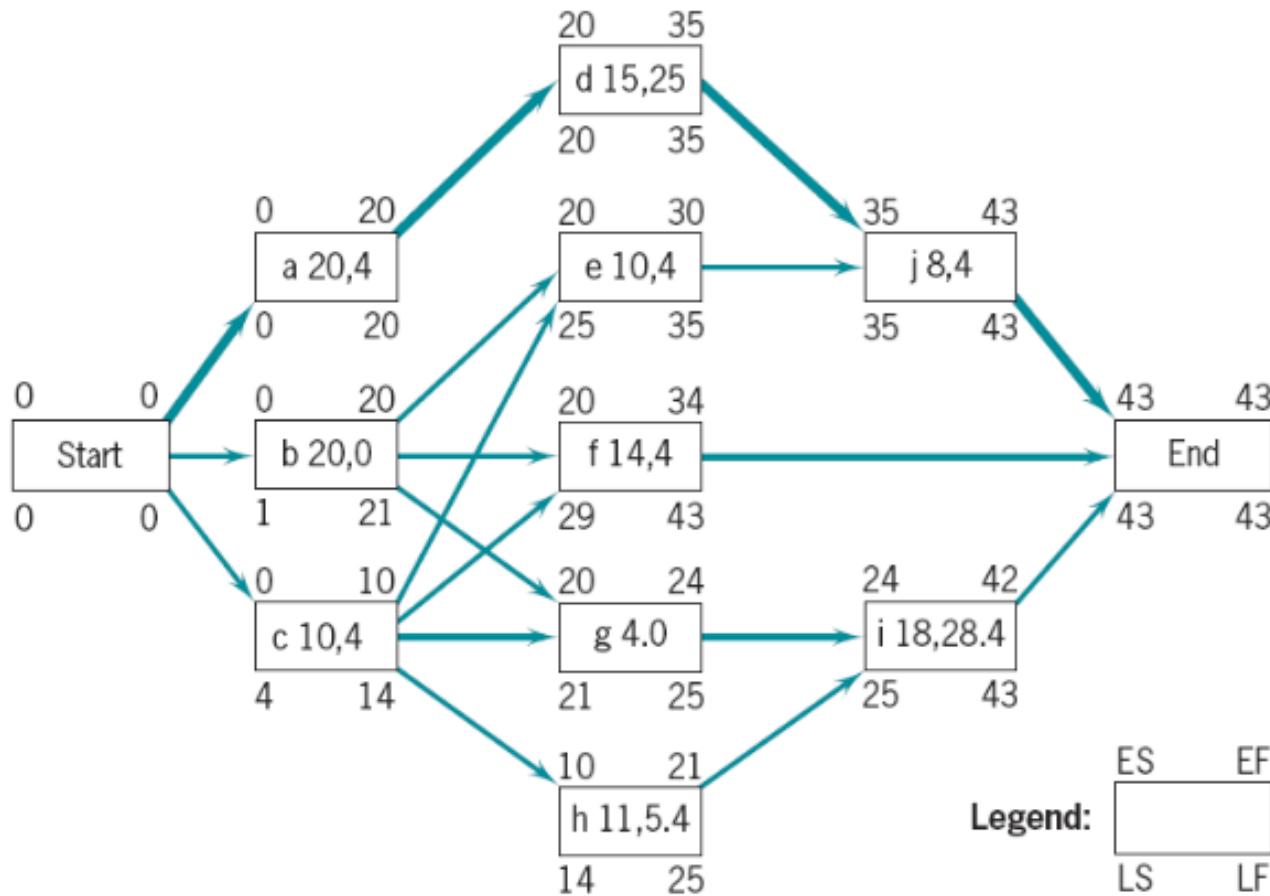


Figure 8-16

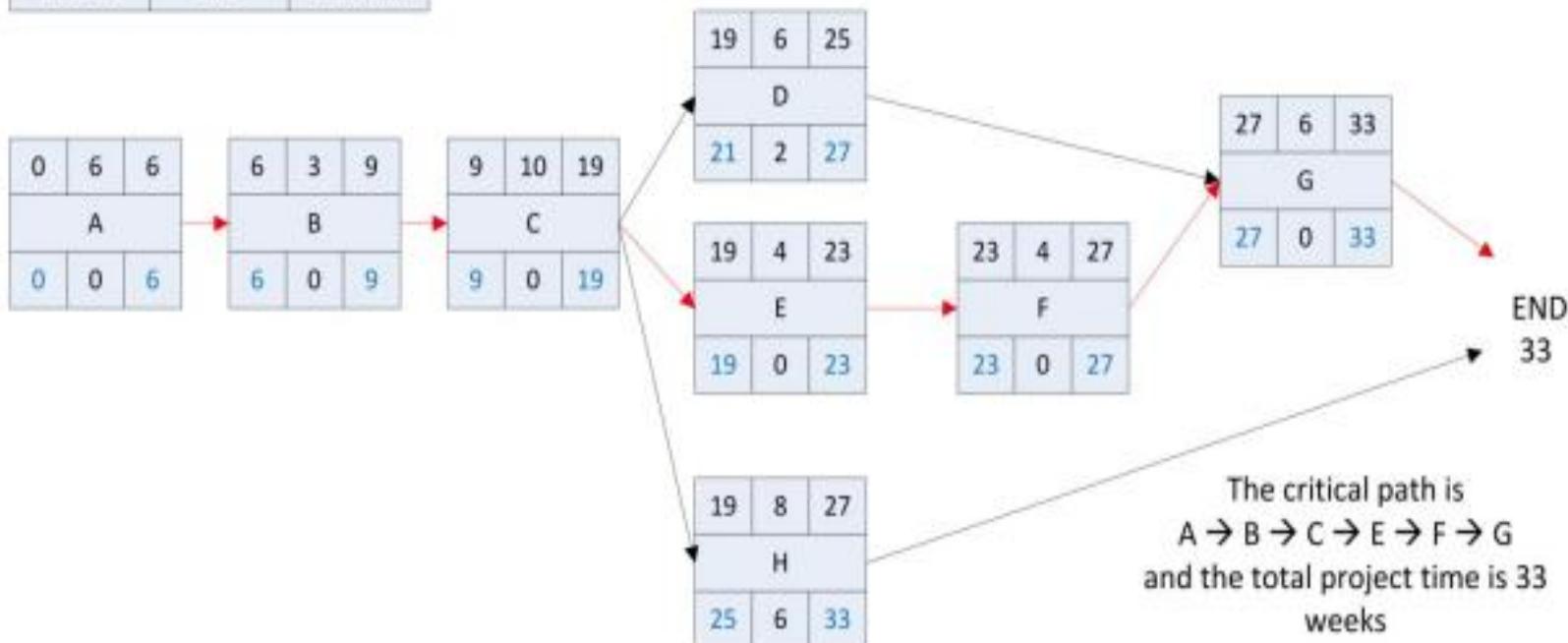
Slack Values

<i>Activity</i>	<i>LS</i>	<i>ES</i>	<i>Slack</i>
a	0	0	0
b	1	0	1
c	4	0	4
d	20	20	0
e	25	20	5
f	29	20	9
g	21	20	1
h	14	10	4
i	25	24	1
j	35	35	0

Precedence Diagramming

- Finish to start
- Start to start
- Finish to finish
- Start to finish

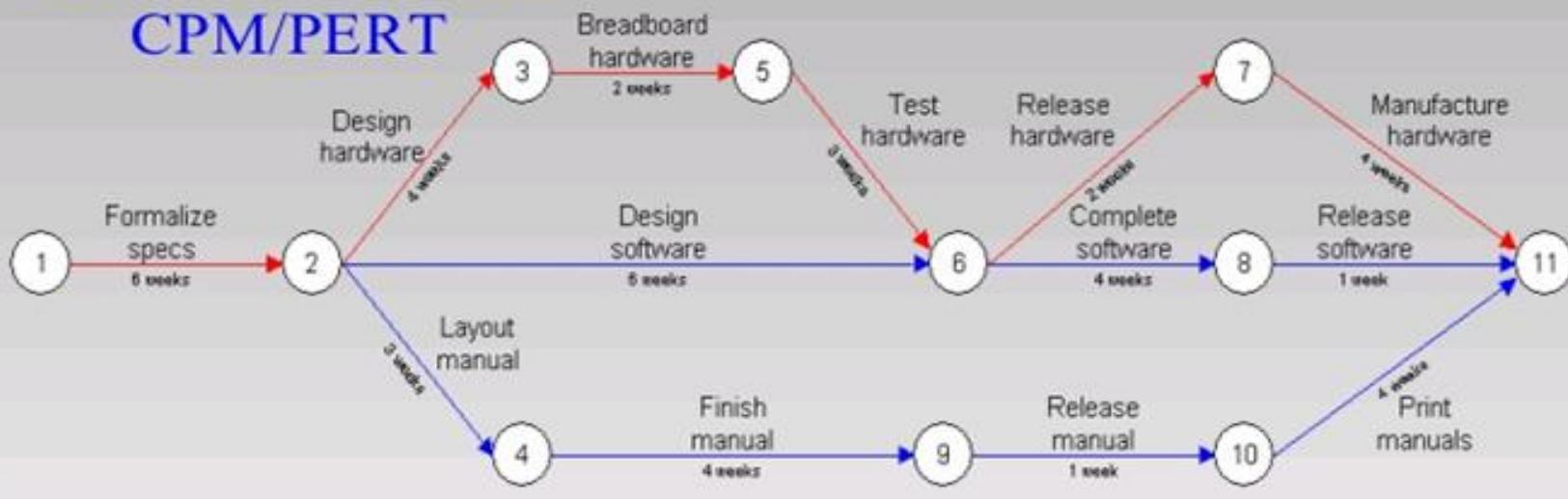
Early Start	Duration	Early Finish
Task Name		
Late Start	Slack	Late Finish



The critical path is
 A → B → C → E → F → G
 and the total project time is 33 weeks

PERT/CPM Chart - PC Card

Project Management CPM/PERT

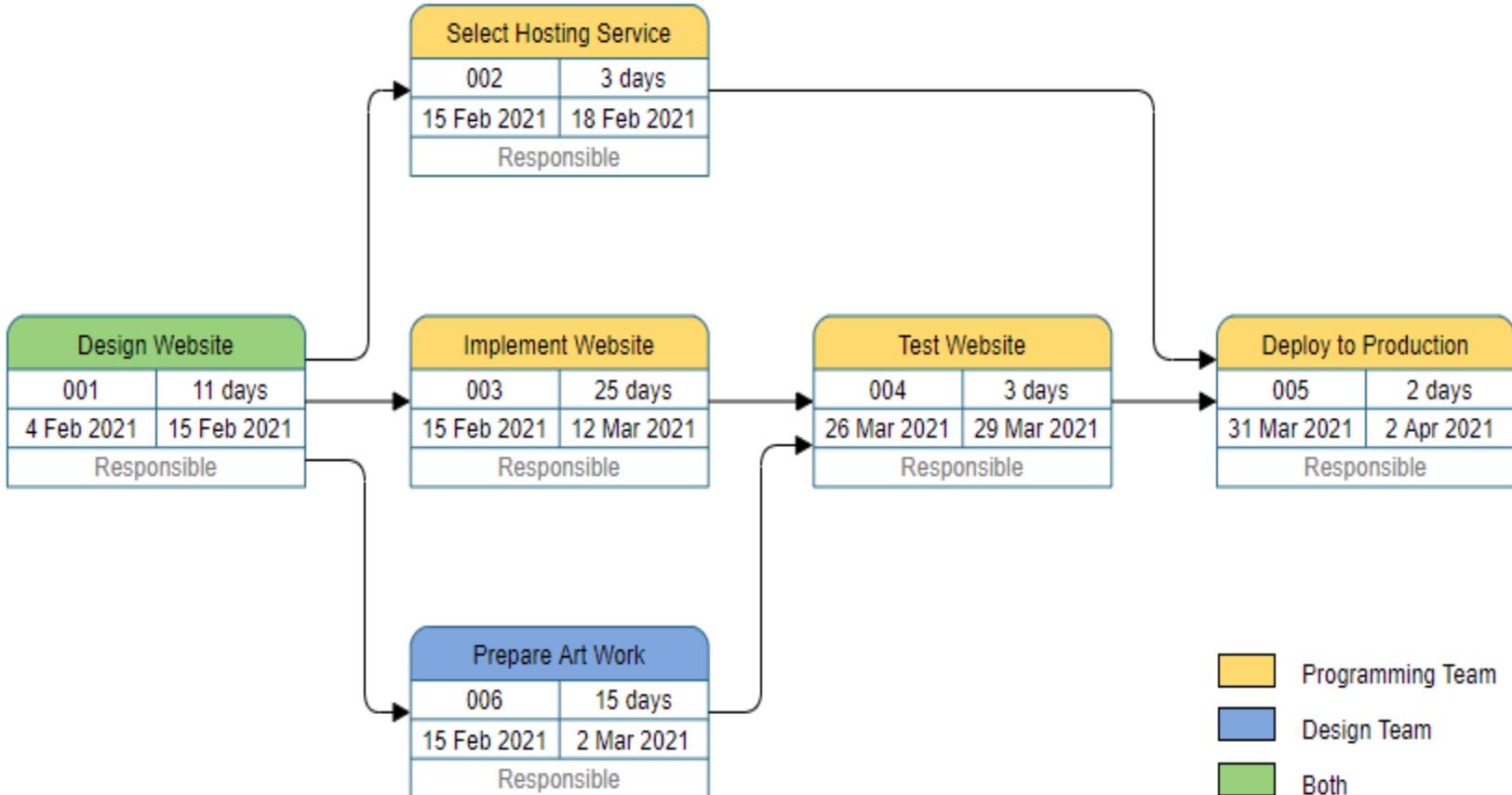


- 1 Start project
- 2 All specs finalized
- 3 Hardware design completed
- 4 Manual layout completed

- 5 Breadboard running
- 6 Hardware fully functional
- 7 PC Board released
- 8 Software finished

- 9 Manual finalized
- 10 Manual ready for printer
- 11 Project complete

→ Critical path



Precedence Diagramming Conventions



Figure 8-17

Microsoft Projects

<i>ID</i>	<i>Task Name</i>	<i>Predecessors</i>	<i>Duration</i>	<i>Optimistic Duration</i>	<i>Expected Duration</i>	<i>Pessimistic Duration</i>
1	Start		0 days	0 days	0 days	0 days
2	a	1	20 days	10 days	22 days	22 days
3	b	1	20 days	20 days	20 days	20 days
4	c	1	10 days	4 days	10 days	16 days
5	d	2	15 days	2 days	14 days	32 days
6	e	3, 4	10 days	8 days	8 days	20 days
7	f	4, 3	14 days	8 days	14 days	20 days
8	g	3, 4	4 days	4 days	4 days	4 days
9	h	4	11 days	2 days	12 days	16 days
10	i	9, 8	18 days	6 days	16 days	38 days
11	j	5, 6	8 days	2 days	8 days	14 days
12	Finish	10, 11, 7	0 days	0 days	0 days	0 days

Table 8-4

Gantt Chart

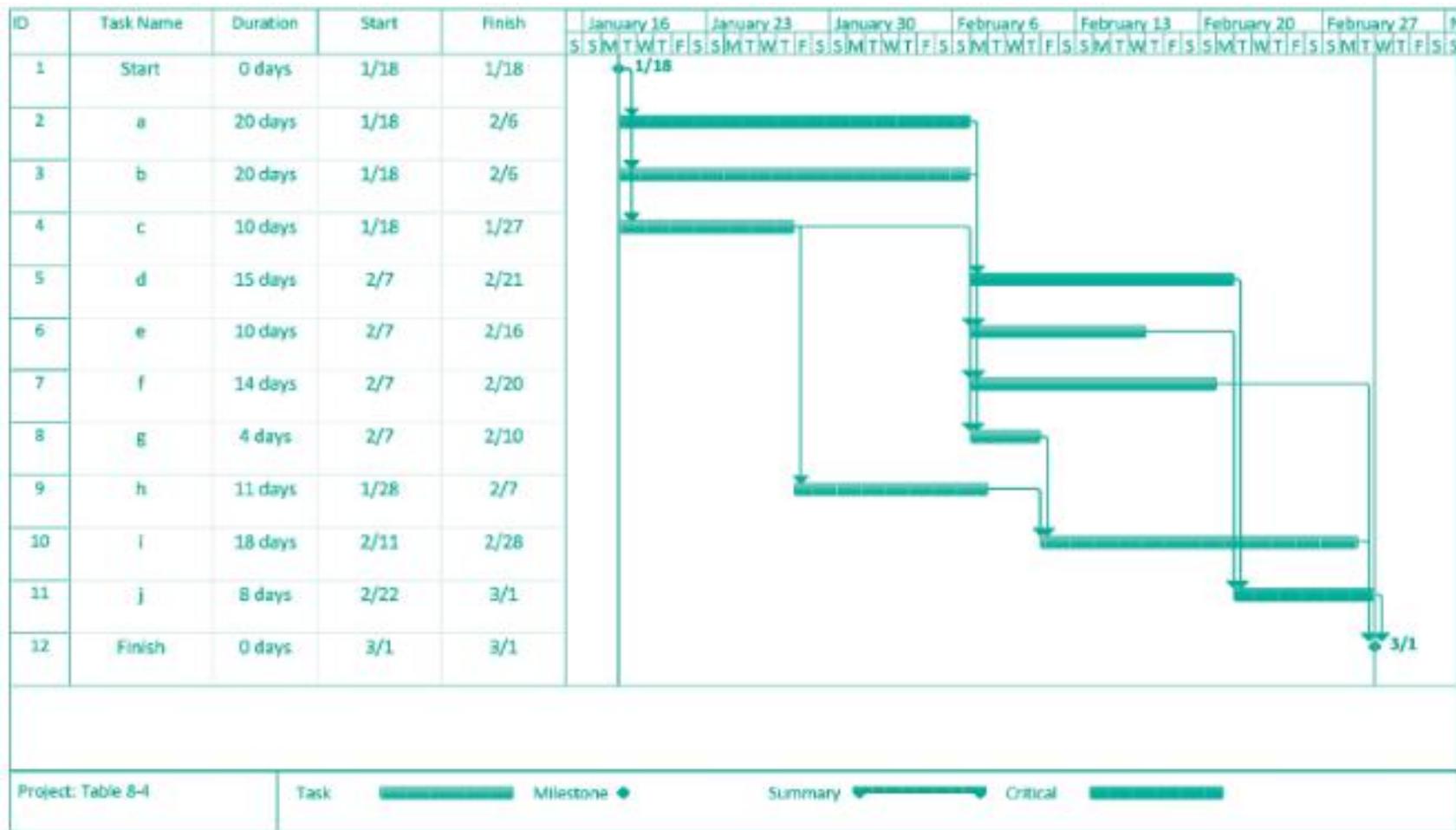


Figure 8-18

AON Network

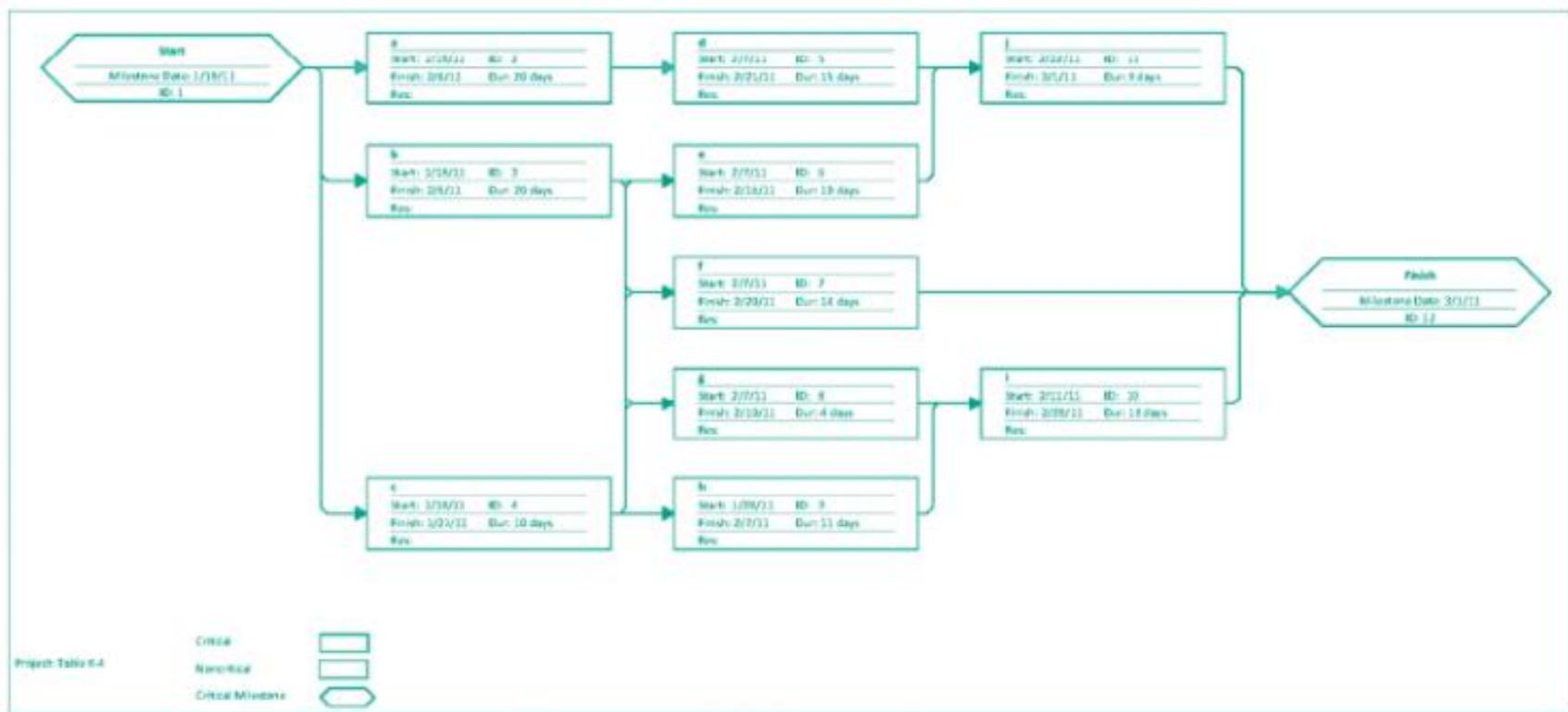


Figure 8-19

Microsoft Project Calendar

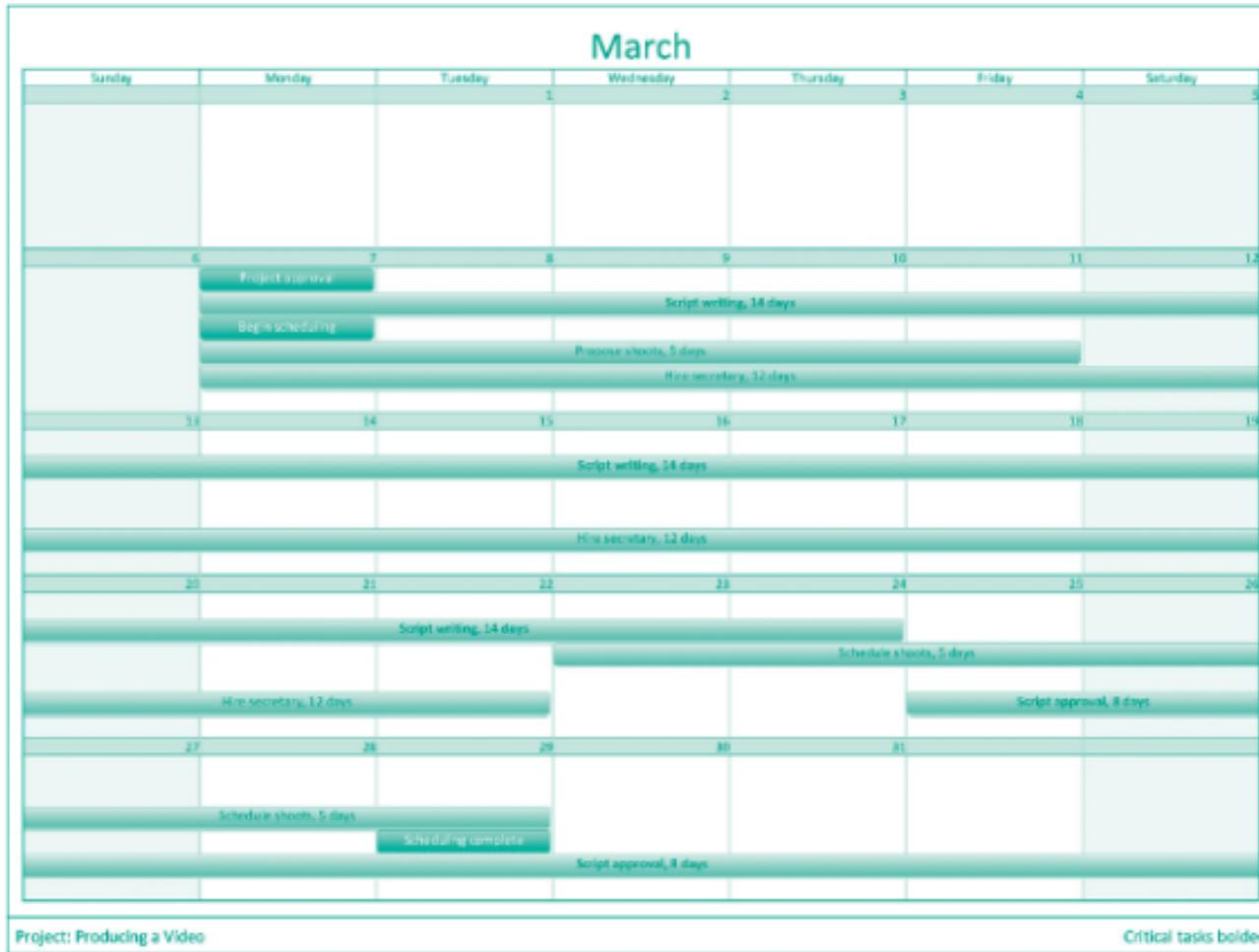


Figure 8-23

Uncertainty of Project Completion Time

- Assume activities are statistically independent
- Variance of a set of activities is the sum of the individual variances
- Interested in variances along the critical path

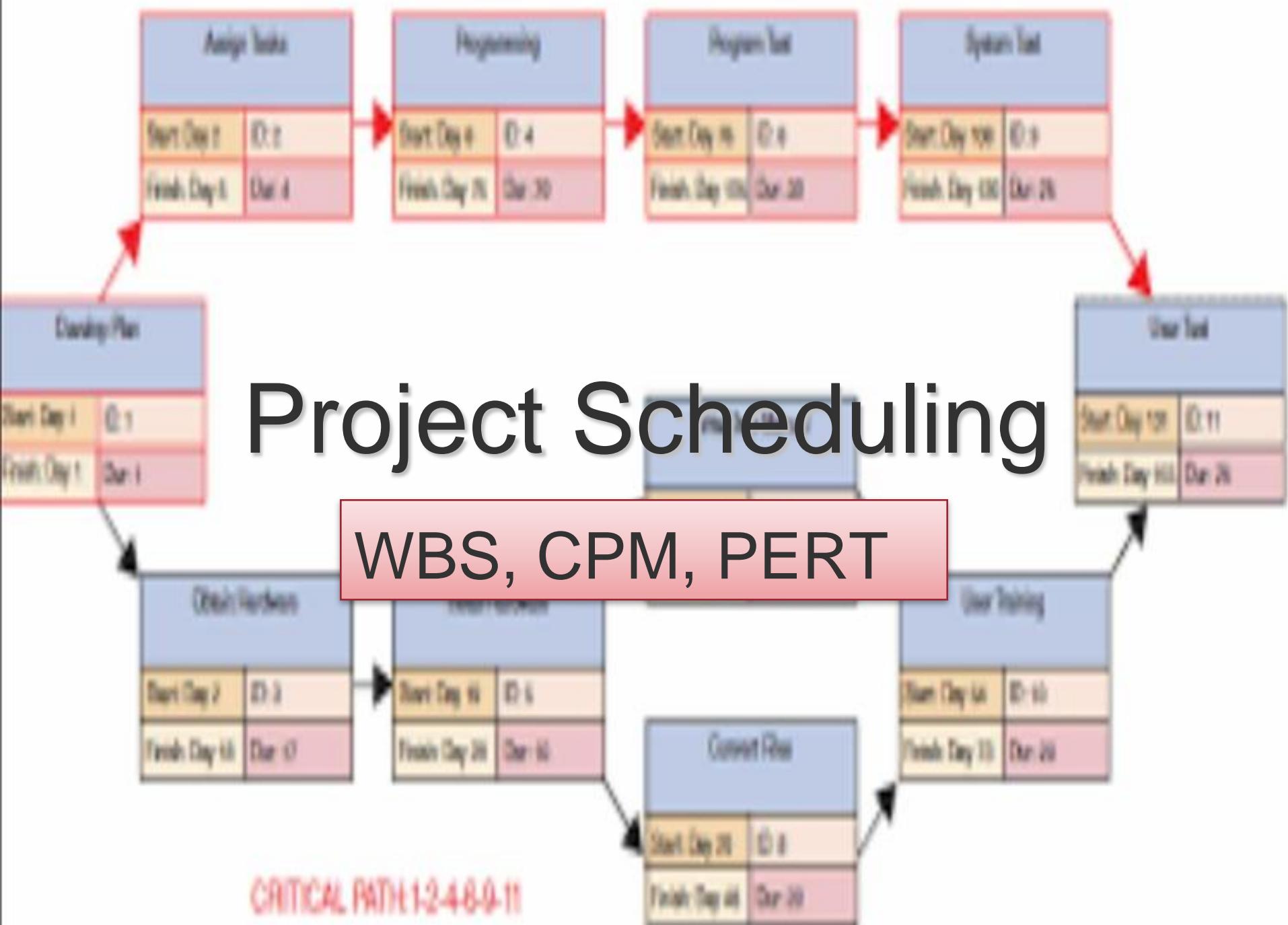
Example

$$Z = \frac{(D - \mu)}{\sqrt{\sigma^2_\mu}} = \frac{(50 - 43)}{\sqrt{33}} = \frac{7}{5.745} = 1.22$$

$$D = \mu + \sigma \cdot Z = 43 + 5.745(1.645) = 52.45$$

Toward Realistic Time Estimates

- Calculations are based on 1% chance of beating estimates
- Calculations can also be based on 5% or 10%
- Changing the percentage requires changing the formulae for variance
- When using 5%, the divisor changes to 3.29
- When using 10%, the divisor changes to 2.56



INTRODUCTION

- Schedule converts action plan into operating time table
- Basis for monitoring and controlling project
- Sometimes customer specified/approved requirement
- Based on Work Breakdown Structure (WBS)

Sequence the Work Activities

Milestone Chart

Gantt chart

Network Techniques

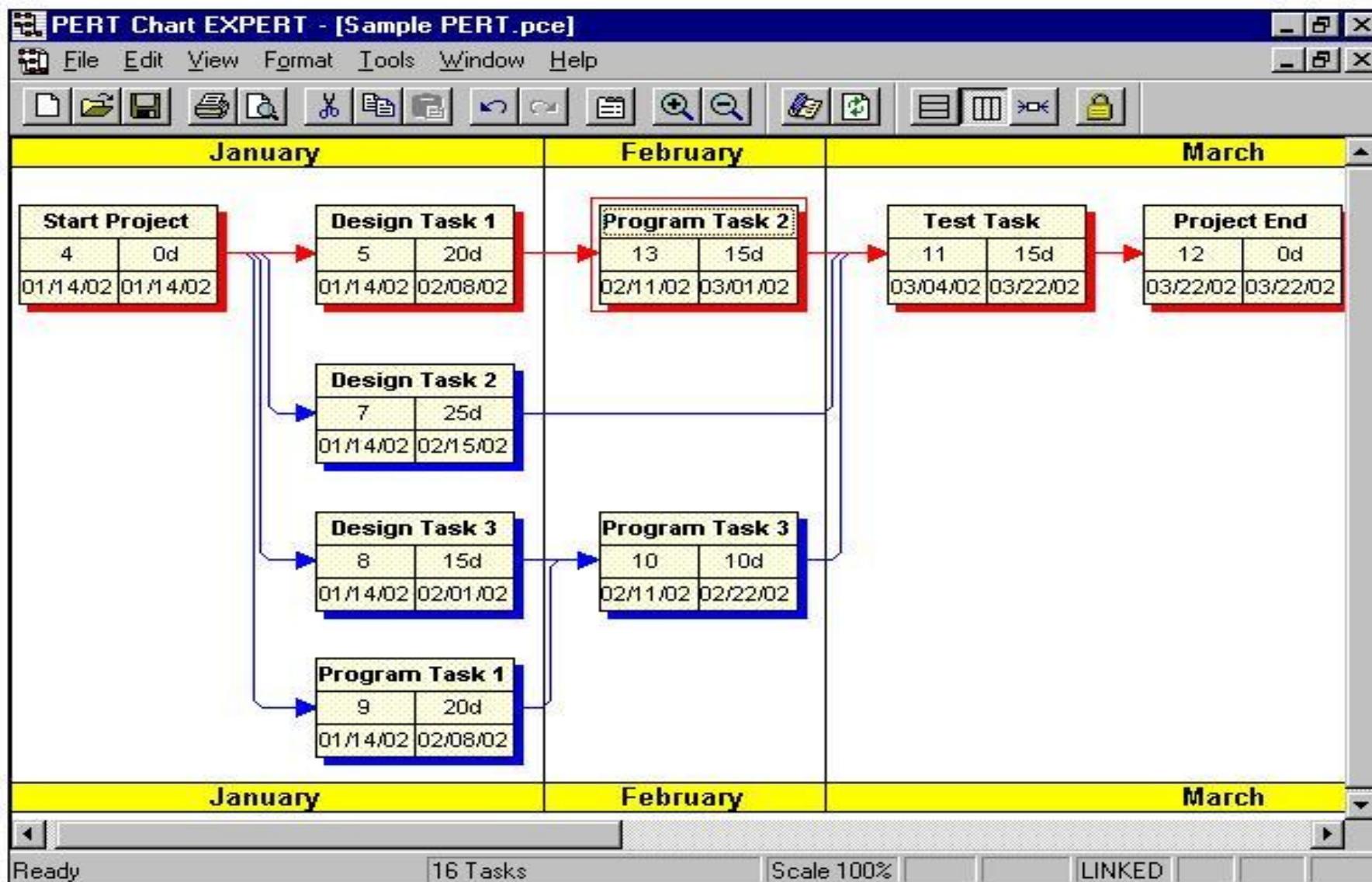
- CPM (Critical Path Method)
- PERT (Program Evaluation and Review Technique)



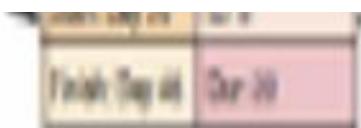
Network Techniques

- A *precedence network* diagram is a graphic model portraying the sequential relationship between key events in a project.
- Initial development of the network requires that the project be defined and thought out.
- The network diagram clearly and precisely communicates the plan of action to the project team and the client.

CRITICAL PATH 1-2-4-6-9-11



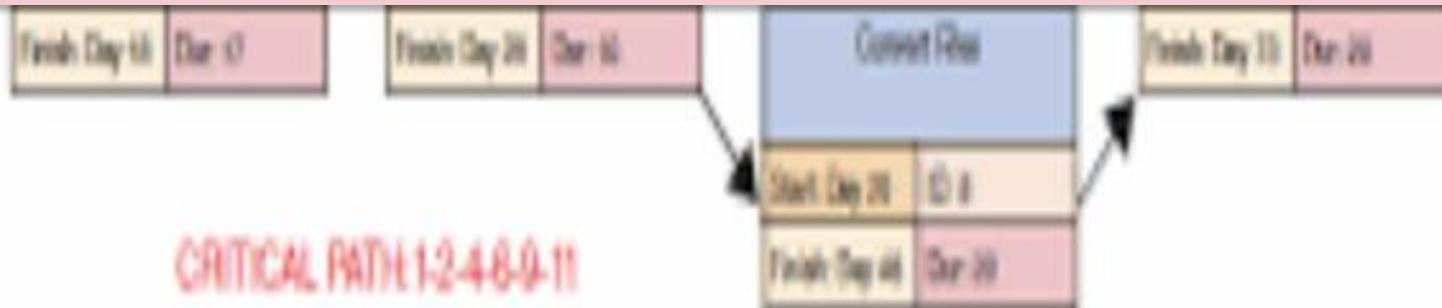
CRITICAL PATH 1-2-4-6-9-11



CPM

Critical Path Method (CPM) tries to answer the following questions:

1. What is the duration of the project?
2. By how much (if at all) will the project be delayed if any one of the activities takes N days longer?
3. How long can certain activities be postponed without increasing the total project duration?



Critical Path

- **Sequence of activities that have to be executed one after another**
- **Duration times of these activities will determine the overall project time, because there is no slack/float time for these activities**
- **If any of the activities on the critical path takes longer than projected, the entire project will be delayed by that same amount**
- **Critical path = Longest path in the precedence network (generally, the longest in time)**

Critical Path 1-2-4-6-9-11

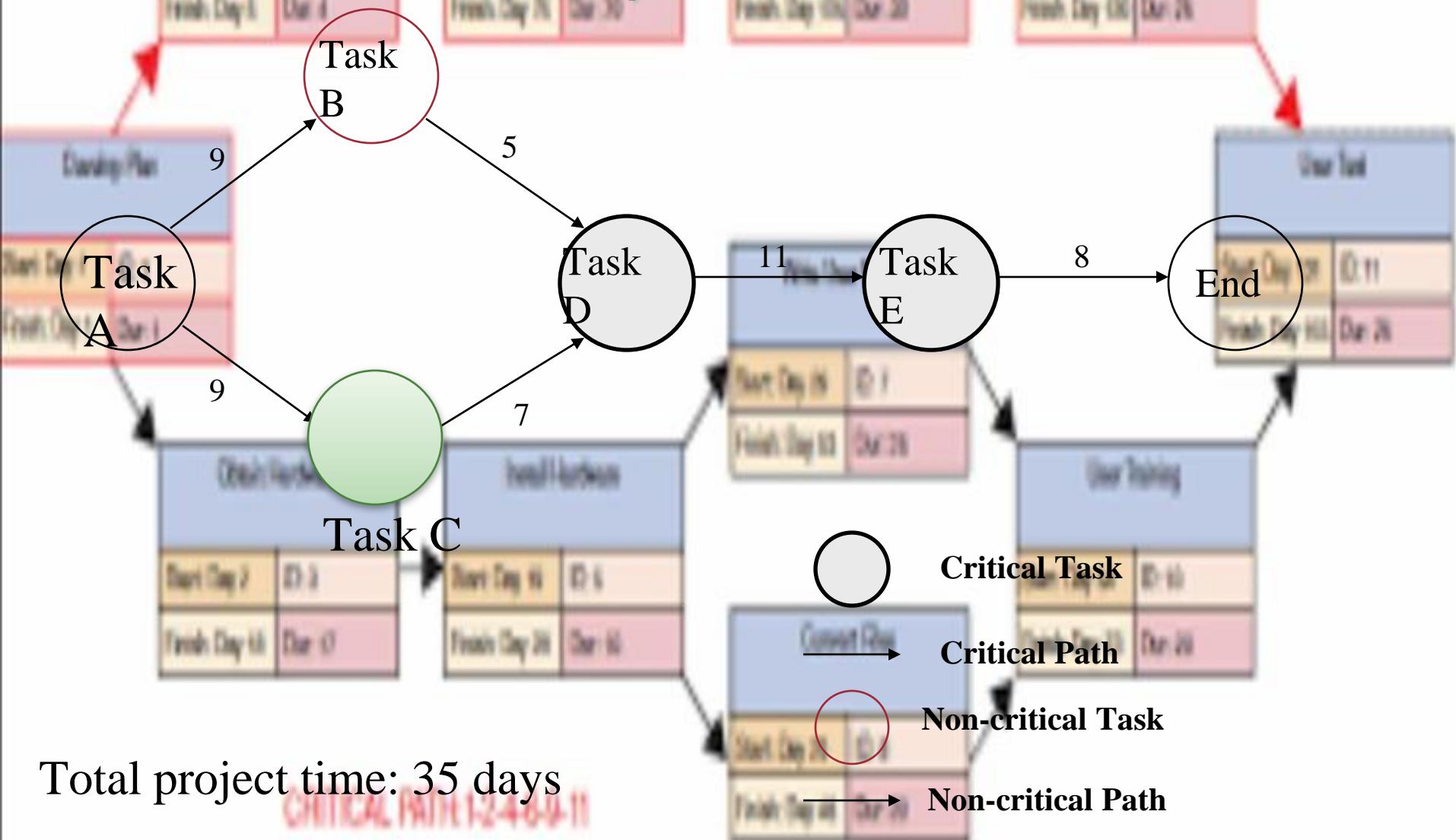
Task sequence/effort table

Task	Immediate prerequisite tasks	Effort (person-days)
A	None	9
B	A	5
C	A	7
D	B,C	11
E	D	8

Total effort required: 40 person-days

CRITICAL PATH 1-2-4-6-9-11

Graphical representation of tasks from previous table



Critical versus Non-Critical Paths

Critical path

- The path that takes the most time to complete.

Critical task (critical activity)

- A task that resides on the critical path.

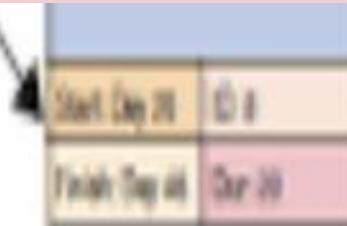
Non-critical path

- Any path that is not a critical path and thus takes less effort to complete than the critical path.

Non-critical task (non-critical activity)

- Any activity that resides on a non-critical path **and** not a critical path. These tasks may accept some delay in completion.

Critical Path 1-2-4-6-9-11



Critical Path Method (CPM)

The critical path method analyses the precedence of activities to predict the total project duration. The method calculates which sequence activities (path) has the least amount of flexibility

CRITICAL PATH 1-2-4-6-9-11

Task Scheduling

- **Forward-pass scheduling**

- Early start time (ES) and early finish (EF)

- **Backward-pass scheduling**

- Late start (LS) and Late finish (LF)

- **Slack time**

CRITICAL PATH 1-2-4-6-9-11

Assign Task

Start Day 0	0:0
Finish Day 0	Day 0

Recovering

Start Day 0	0:0
Finish Day 0	Day 20

Review Test

Start Day 0	0:0
Finish Day 0	Day 20

System Test

Start Day 0	0:0
Finish Day 0	Day 20

Arc with ES & EF time

EF = earliest finish time

Delivery Plan

ES = earliest start time

Activity

A [0,5]
5

2

t = expected activity time

Check Software

Start Day 0	0:0
Finish Day 0	Day 0

Read User Manual

Start Day 0	0:0
Finish Day 0	Day 20

User Training

Start Day 0	0:0
Finish Day 0	Day 20

CRITICAL PATH 1-2-4-6-8-11

Assign Task

Reviewing

Review Test

System Test

Activity, duration, ES, EF, LS, LF

EF = earliest finish time

ES = earliest start time

Activity

C [5,9]
4 [8,12]

3

LS = latest start time

LF = latest finish time

CRITICAL PATH 1-2-4-6-9-11

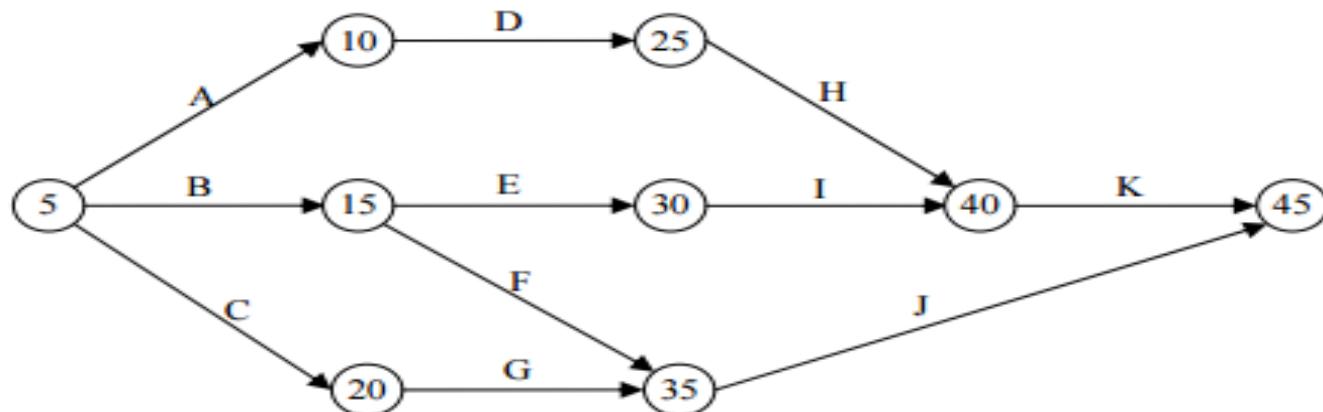


Figure 3.18: AOA network

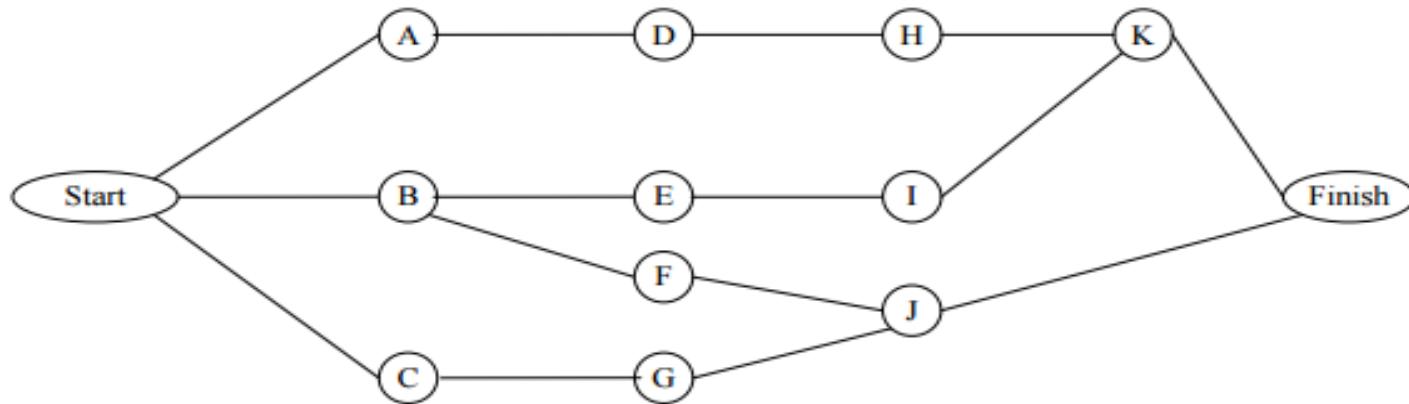
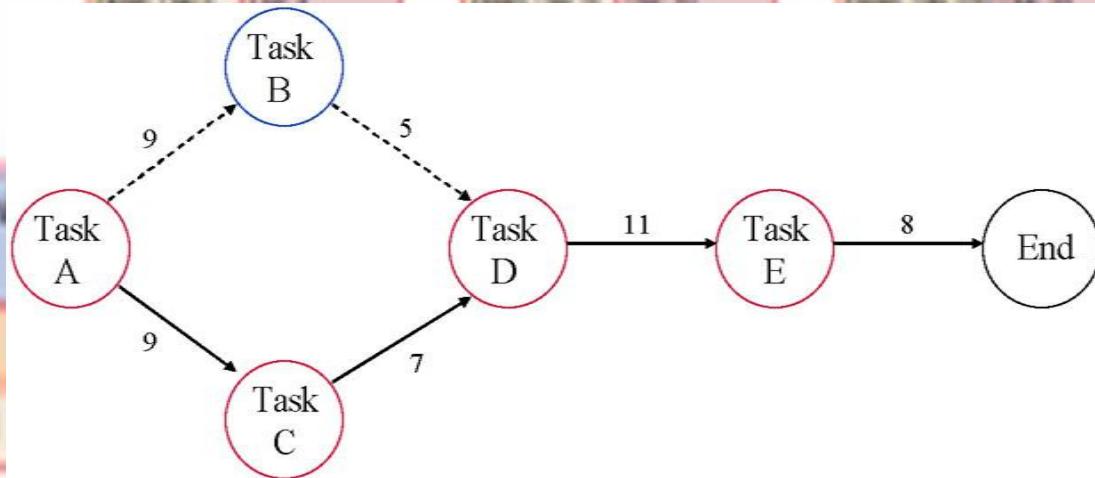


Figure 3.19: AON network

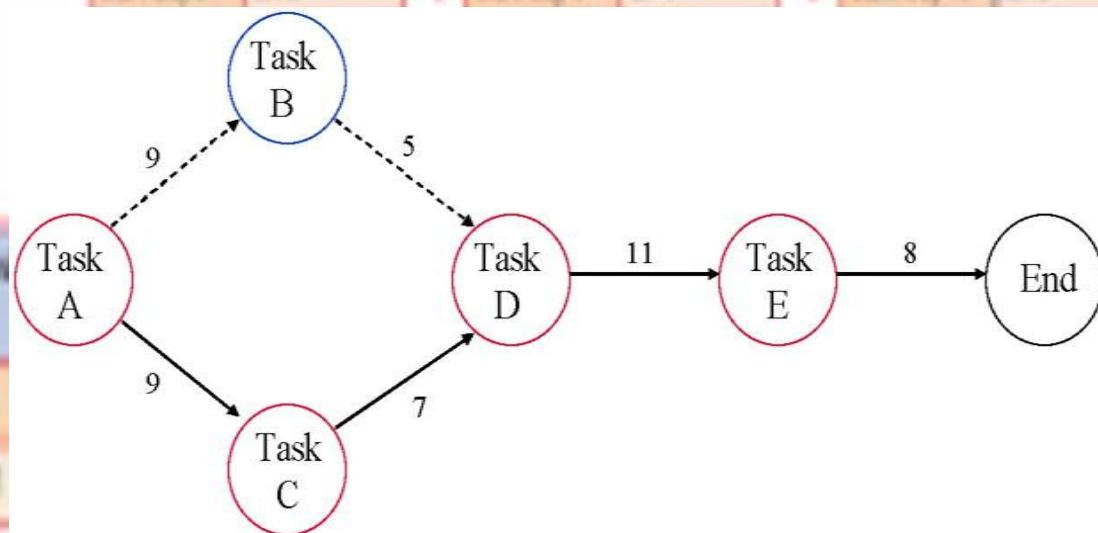
Forward-pass scheduling



Task	Tasks precedence	Task length	Earliest possible start time (ES)	Earliest possible finish time (EF)
A	None	9	0	9
B	A	5	9	14
C		7	9	16
D	B,C	11	16	27
E	D	8	27	35

Critical Path: A-B-C-D-E

Backward-pass scheduling



Task	Tasks precedence	Task length	Late start time (LS)	Late finish time (LF)
A	None	9	0	9
B	A	5	11	16
C	A	7	9	16
D	B,C	11	16	27
E	D	8	27	35

CRITICAL PATH: A-B-C-D-E

Assign Task

Reopen Task

Assign Task

Update Task

Slack time

- **Total slack time of an activity**
 - The difference in start time between a **non-critical task's late start time and its early start time or its late finish time and early finish time.**

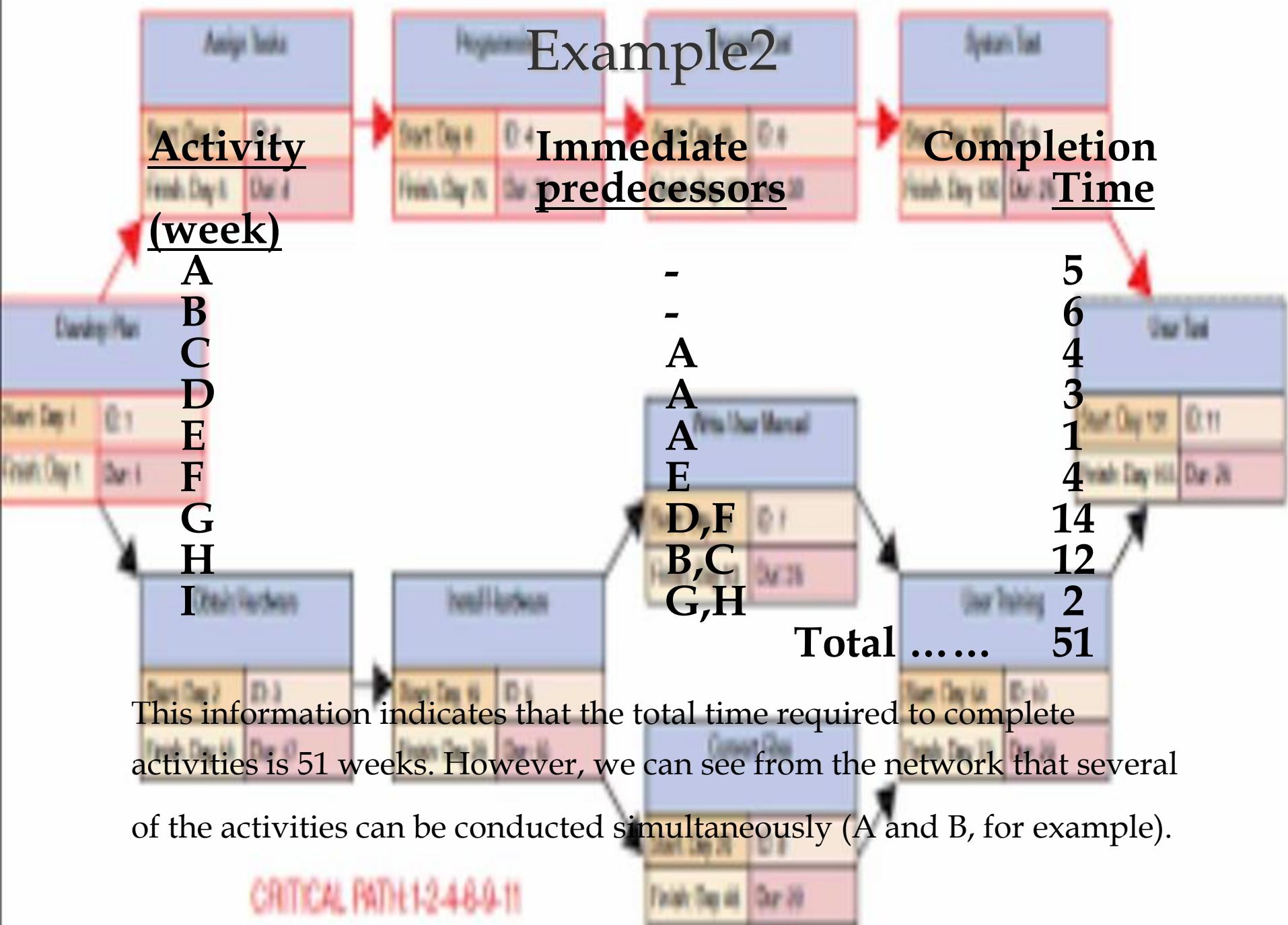
Total slack time of a task = LS - ES

Or

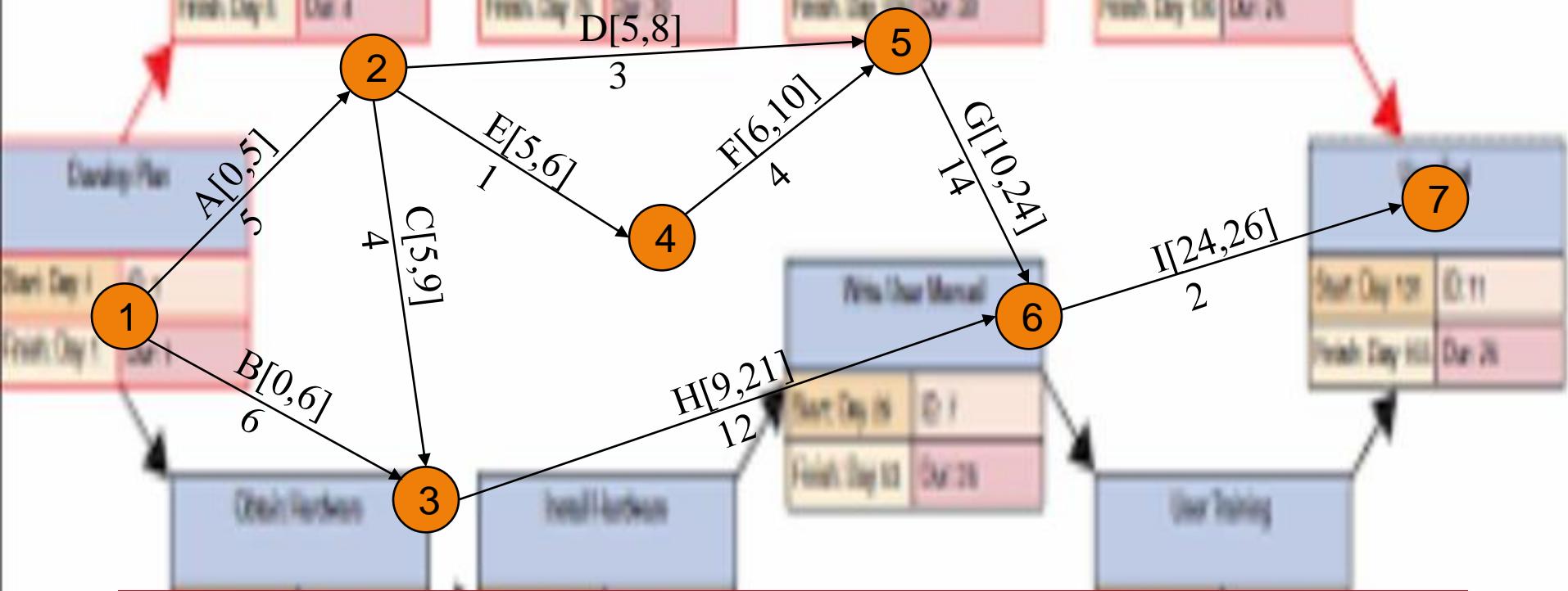
Total slack time of a task = LF – EF

e.g., Activity **B**: $LS - ES (11 - 9)$ or $LF - EF (16 - 14) \Rightarrow 2$

- **Total slack time**
 - The maximum allowable delay for all non-critical activities.



Network with ES & EF time

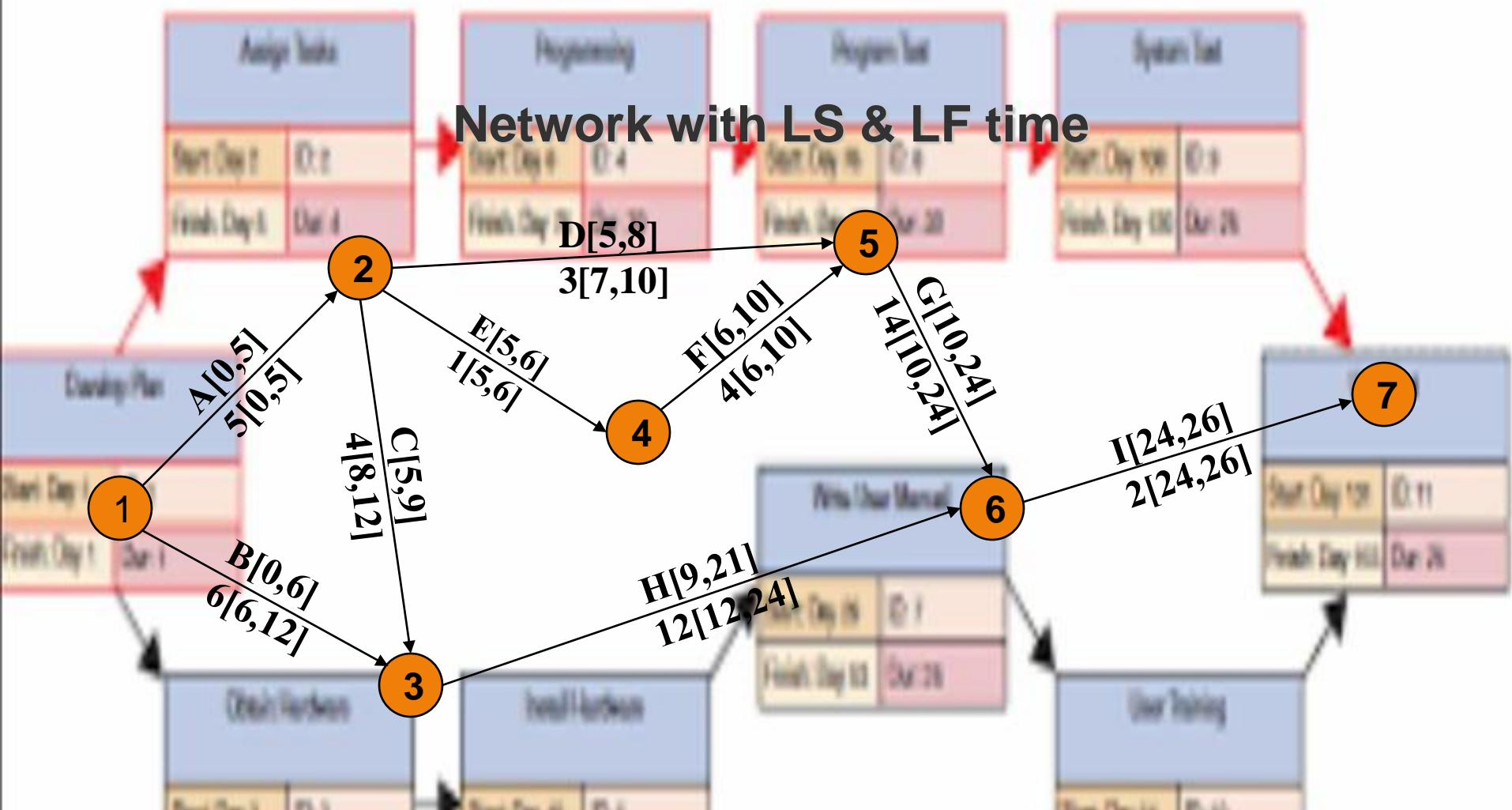


Earliest start time rule:

The *earliest start time* for an activity leaving a particular node is equal to the *largest of the earliest finish times* for all activities entering the node.

Latest start & latest finish time

- To find the critical path we need a backward pass calculation.
- Starting at the completion point (node 7) and using a latest finish time (LF) of 26 for activity I, we trace back through the network computing a latest start (LS) and latest finish time for each activity
- The expression $LS = LF - t$ can be used to calculate latest start time for each activity.
- For example, for activity I, $LF = 26$ and
- $t = 2$, thus the latest start time for activity I is
$$LS = 26 - 2 = 24$$



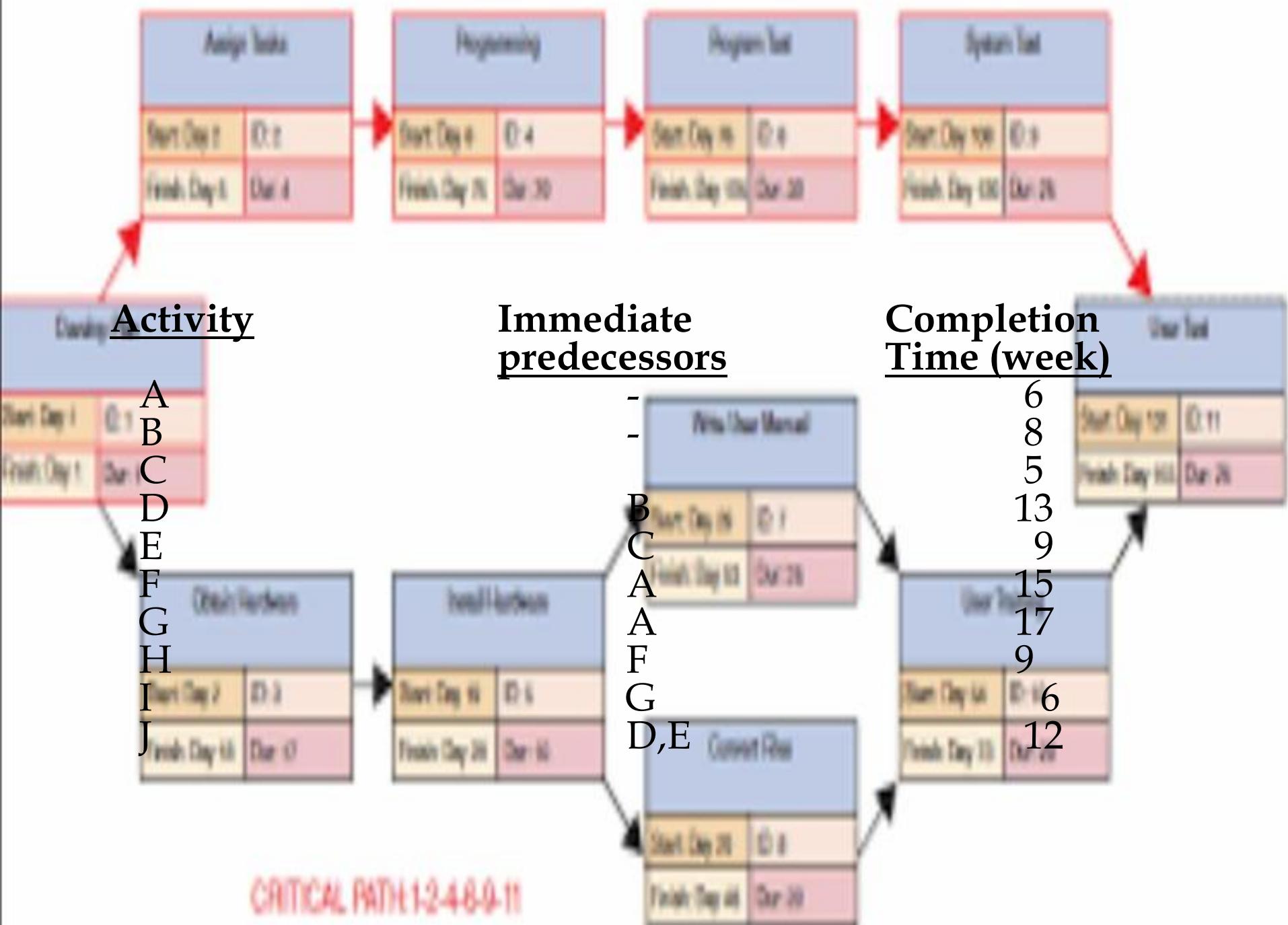
Latest finish time rule:

The latest finish time for an activity entering a particular node is equal to the smallest of the latest start times for all activities leaving the node.

Activity schedule for our example

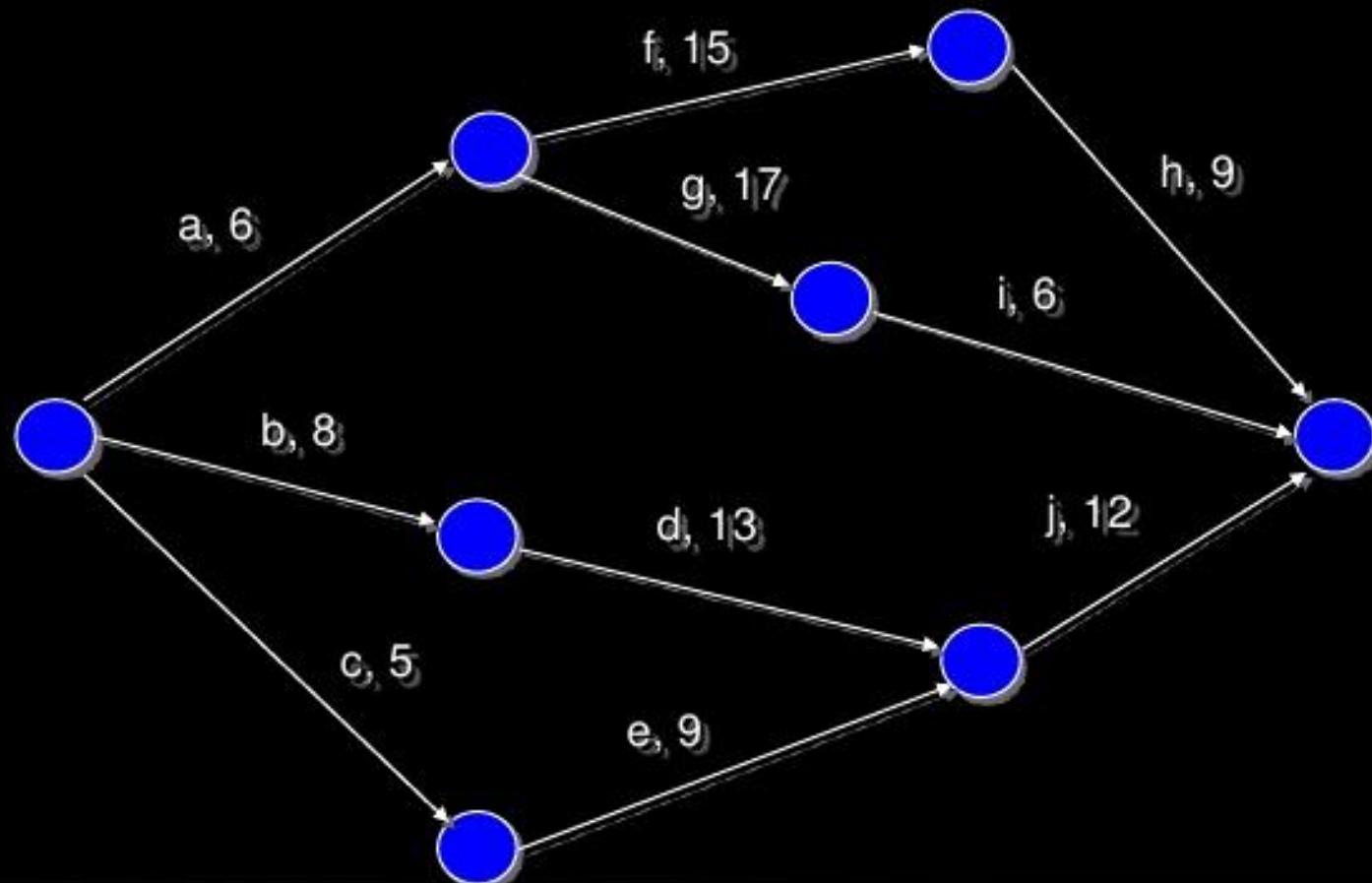
Activity	Earliest start (ES)	Latest start (LS)	Earliest finish (EF)	Latest finish (LF)	Slack (LS-ES)	Critical path
A	0	0	5	5	0	Yes
B	0	6	6	12	6	
C	5	8	9	12	3	
D	5	7	8	10	2	
E	5	5	6	6	0	Yes
F	6	6	10	10	0	Yes
G	10	10	24	24	0	Yes
H	9	12	21	24	3	
I	24	24	26	26	0	Yes

CRITICAL PATH 1-2-4-6-8-9-11



CPM Example:-

□ CPM Network:-



Assign Task

Reviewing

Assign Task

Update Task

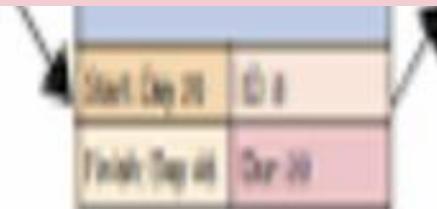
IMPORTANT QUESTIONS

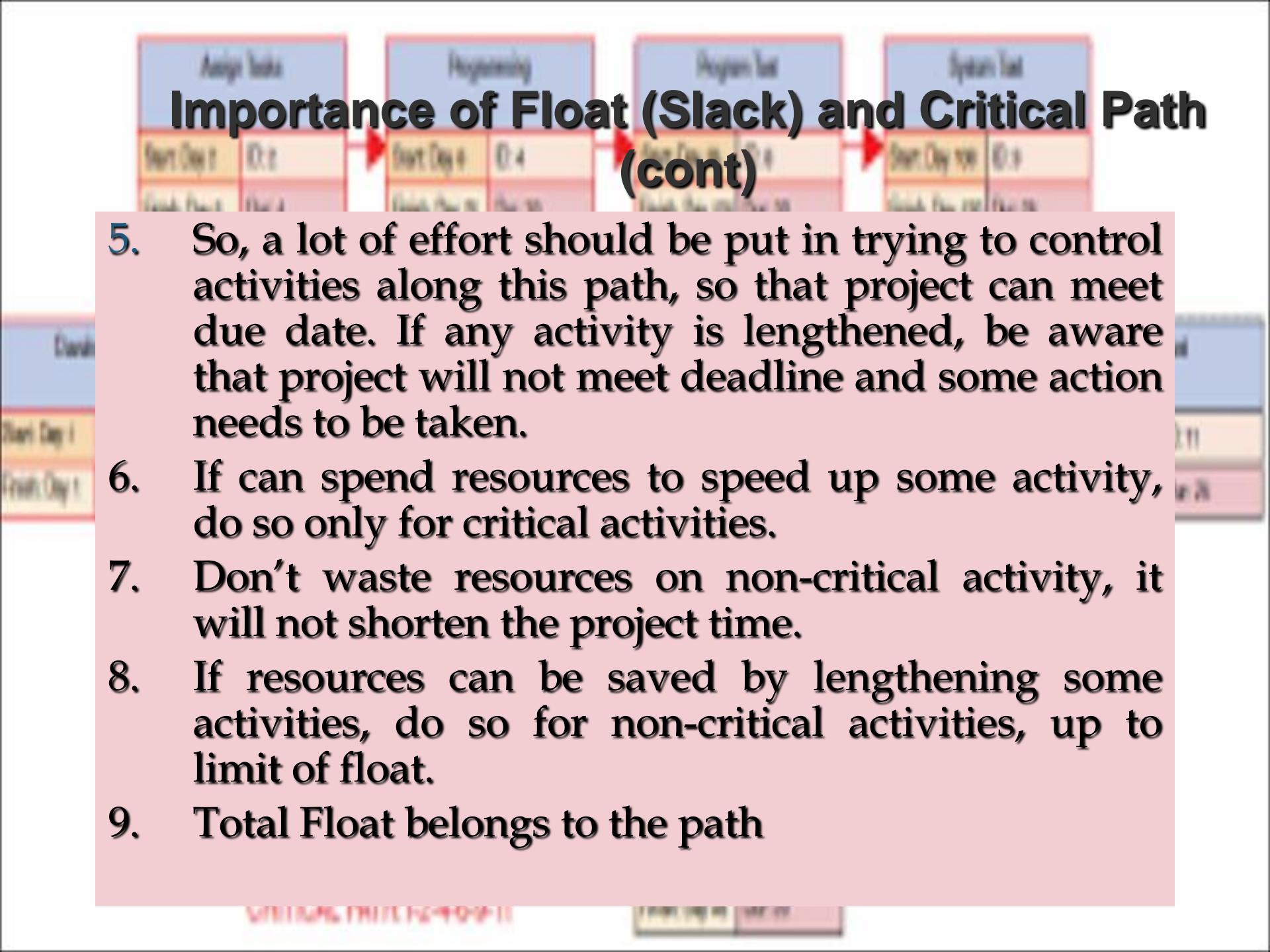
- **What is the total time to complete the project?**
 - 26 weeks if the individual activities are completed on schedule.
- **What are the scheduled start and completion times for each activity?**
 - ES, EF, LS, LF are given for each activity.
- **What activities are *critical* and must be completed as scheduled in order to keep the project on time?**
 - Critical path activities: A, E, F, G, and I.
- **How long can *non-critical* activities be delayed before they cause a delay in the project's completion time**
 - Slack time available for all activities are given.

Importance of Float (Slack) and Critical Path

1. Slack or Float shows how much allowance each activity has, i.e how long it can be delayed without affecting completion date of project.
2. Critical path is a sequence of activities from start to finish with zero slack. Critical activities are activities on the critical path.
3. Critical path identifies the minimum time to complete project
4. If any activity on the critical path is shortened or extended, project time will be shortened or extended accordingly

Critical Path 1-2-4-6-8-11





Importance of Float (Slack) and Critical Path (cont)

5. So, a lot of effort should be put in trying to control activities along this path, so that project can meet due date. If any activity is lengthened, be aware that project will not meet deadline and some action needs to be taken.
6. If can spend resources to speed up some activity, do so only for critical activities.
7. Don't waste resources on non-critical activity, it will not shorten the project time.
8. If resources can be saved by lengthening some activities, do so for non-critical activities, up to limit of float.
9. Total Float belongs to the path

PERT For Dealing With Uncertainty

- So far, times can be estimated with relative certainty, confidence
- For many situations this is not possible, e.g Research, development, new products and projects etc.
- Use 3 time estimates
 - m = most likely time estimate, mode.
 - a = optimistic time estimate,
 - b = pessimistic time estimate, and

$$\text{Expected Value (TE)} = (a + 4m + b) / 6$$

$$\text{Variance (V)} = ((b - a) / 6)^2$$

$$\text{Std Deviation } (\delta) = \text{SQRT (V)}$$

Assign Tasks

Progressing

Progress Test

System Test

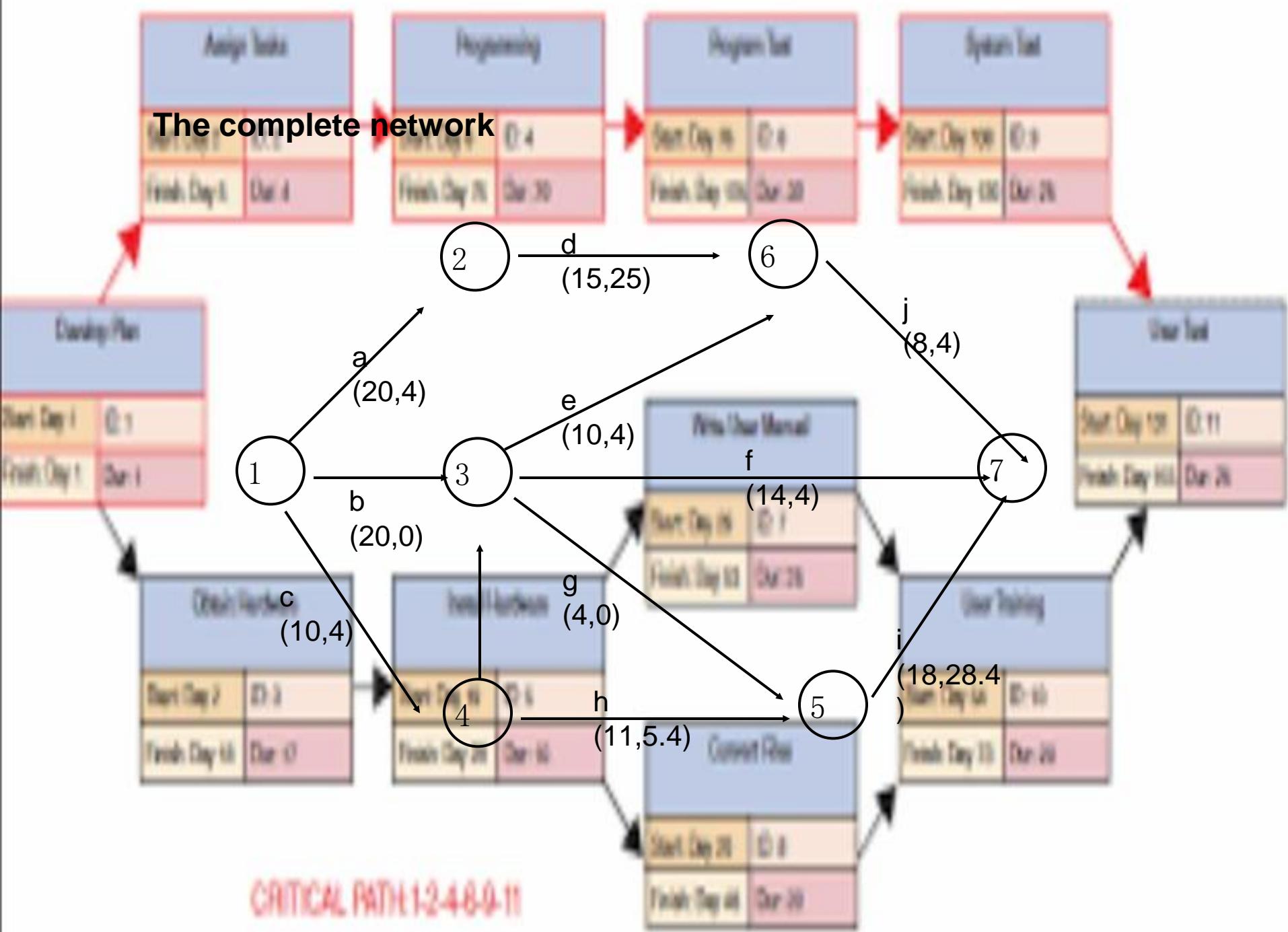
Precedences And Project Activity Times

	Immediate Activity	Optimistic Time	Most Likely Time	Pessimistic Time	EXP	Var	S.Dev
	Predecessor				TE	V	σ
a	-	10	22	22	20	4	2
b	-	20	20	20	20	0	0
c	-	4	10	16	10	4	2
d	a	2	14	32	15	25	5
e	b,c	8	8	20	10	4	2
f	b,c	8	14	20	14	4	2
g	b,c	4	4	4	4	0	0
h	c	2	12	16	11	5.4	2.32
i	g,h	6	16	38	18	28.4	5.33
j	d,e	2	8	14	8	4	2

CRITICAL PATH 1-2-4-6-9-11

Finish Day 11 Sat 11/11

- Now, following is the process we follow with the two values:
- For every activity in the critical path, E and V are calculated.
- Then, the total of all Es are taken. This is the overall expected completion time for the project.
- Now, the corresponding V is added to each activity of the critical path. This is the variance for the entire project.
- This is done only for the activities in the critical path as only the critical path activities can accelerate or delay the project duration.
- Then, standard deviation of the project is calculated. This equals to the square root of the variance (V).



Assign Tasks

Reviewing

Review Test

System Test

Planning Phase

Start Day 1 01/1
Finish Day 1 01/1

Design Work

Start Day 2 01/2
Finish Day 3 01/3

EF=20

a

(20,4)

b

(20,0)

c

(10,4)

20

10

2

d

(15,25)

e

(10,4)

g

(4,0)

h

(11,5.4)

i

(18,28.4)

35

24

6

j

(8,4)

f

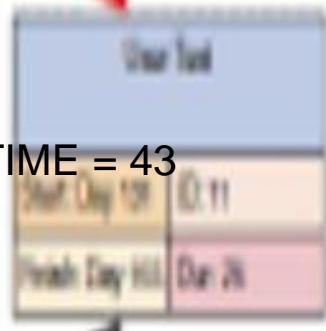
(14,4)

5

43

7

CRIT. TIME = 43



Review



Figure 8-13 The complete Network

Critical Path 1-2-4-6-8-11

Critical Path Analysis (PERT)

Activity	LS	ES	Slacks	Critical ?
a	0	0	0	Yes
b	1	0	1	
c	4	0	4	
d	20	20	0	Yes
e	25	20	5	
f	29	20	9	
g	21	20	1	
h	14	10	4	
i	25	24	1	
j	35	35	0	Yes

Critical Path 1-2-4-6-9-11

Comparison Between CPM and PERT

	CPM	PERT
1	Uses network, calculate float or slack, identify critical path and activities, guides to monitor and controlling project	Same as CPM
2	Uses one value of activity time	Requires 3 estimates of activity time Calculates mean and variance of time
3	Used where times can be estimated with confidence, familiar activities	Used where times cannot be estimated with confidence. Unfamiliar or new activities
4	Minimizing cost is more important	Meeting time target or estimating percent completion is more important
5	Example: construction projects, building one off machines, ships, etc	Example: Involving new activities or products, research and development etc

CRITICAL PATH 1-2-4-6-9-11



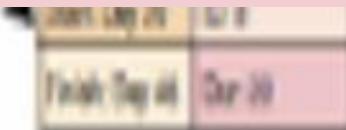
Consistent framework for planning, scheduling, monitoring, and controlling project.

- Shows interdependence of all tasks, work packages, and work units.
- Helps proper communications between departments and functions.
- Determines expected project completion date.
- Identifies so-called critical activities, which can delay the project completion time.

BENEFITS OFCPM / PERT NETWORK (cont.)

- Identified activities with slacks that can be delayed for specified periods without penalty, or from which resources may be temporarily borrowed
- Determines the dates on which tasks may be started or must be started if the project is to stay in schedule.
- Shows which tasks must be coordinated to avoid resource or timing conflicts.
- Shows which tasks may run in parallel to meet project completion date

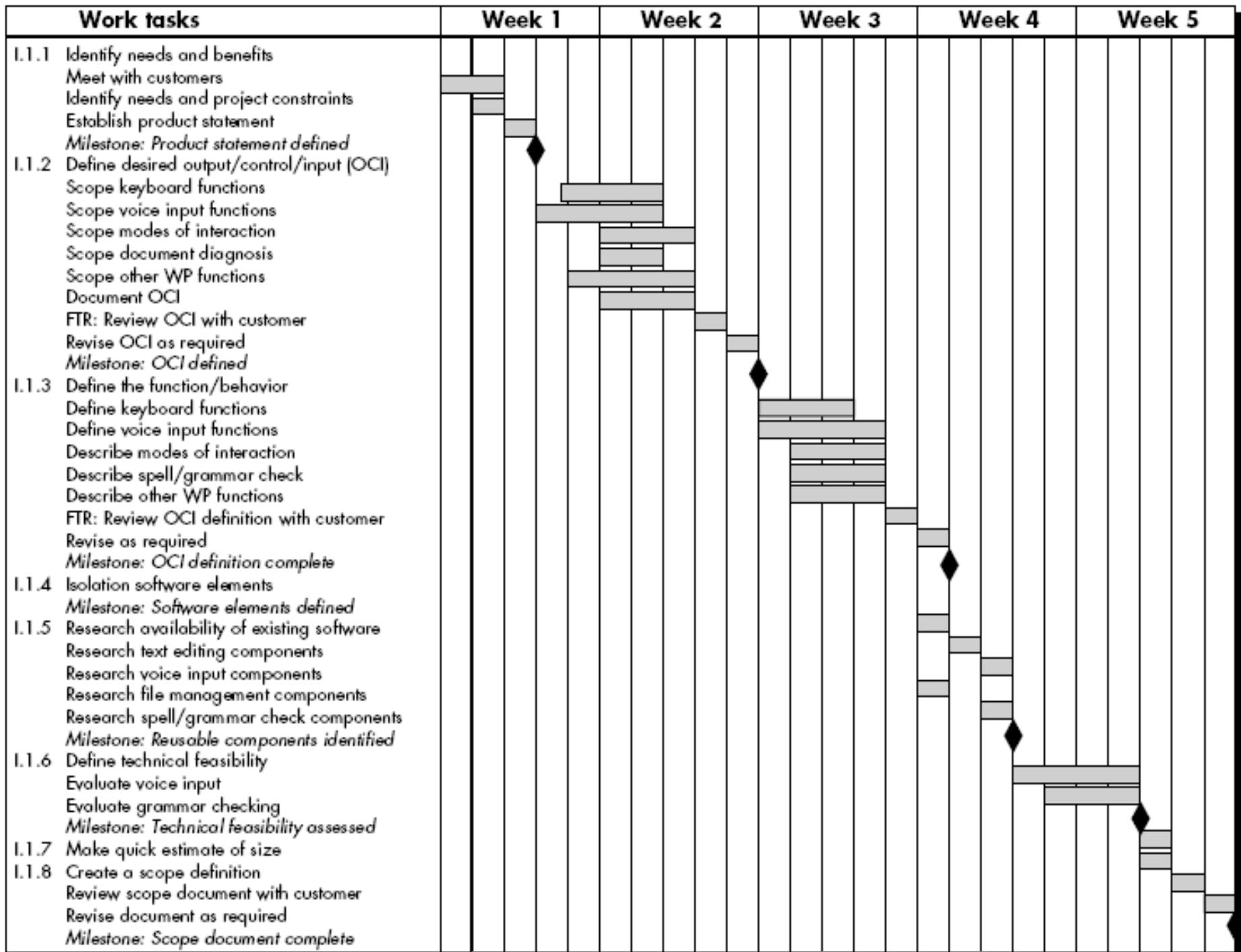
CRITICAL PATH 1-2-4-6-9-11



Timeline Charts

- **The planner always begins with a set of tasks (the work breakdown structure).**
- **If automated tools are used, the work breakdown is input as a task network or task outline.**
- **Effort, duration, and start date are then input for each task. In addition, tasks may be assigned to specific individuals.**
- **A timeline chart enables you to determine what tasks will be conducted at a given point in time.**
- **A timeline chart can be developed for the entire project.**
- **Alternatively, separate charts can be developed for each project function or for each individual working on the project.**

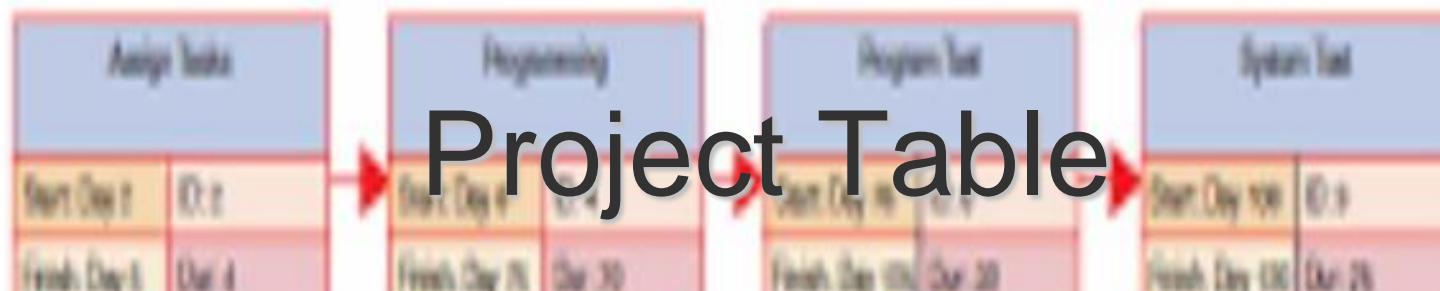
CRITICAL PATH 1-2-4-6-8-11



Contd.

- All project tasks are listed in the left-hand column.
- The horizontal bars indicate the duration of each task.
- When multiple bars occur at the same time on the calendar, task concurrency is implied.
- The diamonds indicate milestones.
- Once the information necessary for the generation of a timeline chart has been input, the majority of software project scheduling tools produce **project tables**
- It is a tabular listing of all project tasks, their planned and actual start- and end-dates, and a variety of related information.
- Project tables enable the project manager to track progress.

CRITICAL PATH 1-2-4-6-9-11



Project Table

Work tasks	Planned start	Actual start	Planned complete	Actual complete	Assigned person	Effort allocated	Notes
I.1.1 Identify needs and benefits Meet with customers Identify needs and project constraints Establish product statement <i>Milestone: Product statement defined</i>	wk1, d1 wk1, d2 wk1, d3 wk1, d3	wk1, d1 wk1, d2 wk1, d3 wk1, d3	wk1, d2 wk1, d2 wk1, d3 wk1, d3	wk1, d2 wk1, d2 wk1, d3 wk1, d3	BLS JPP BLS/JPP	2 p-d 1 p-d 1 p-d	Scoping will require more effort/time
I.1.2 Define desired output/control/input (OCI) Scope keyboard functions Scope voice input functions Scope modes of interaction Scope document diagnostics Scope other WP functions Document OCI FTR: Review OCI with customer Revise OCI as required <i>Milestone: OCI defined</i>	wk1, d4 wk1, d3 wk2, d1 wk2, d1 wk1, d4 wk2, d1 wk2, d3 wk2, d4	wk1, d4 wk1, d3 wk2, d1 wk2, d2 wk1, d4 wk2, d1 wk2, d3 wk2, d4	wk2, d2 wk2, d2 wk2, d3 wk2, d2 wk2, d3 wk2, d3 wk2, d3 wk2, d4		BLS JPP MLL BLS JPP MLL all all	1.5 p-d 2 p-d 1 p-d 1.5 p-d 2 p-d 3 p-d 3 p-d 3 p-d	
I.1.3 Define the function/behavior	wk2, d5						

CRITICAL PATH 1-2-4-6-8-11

Finish Day 11 | Oct 20

Tracking the Project Schedule

- It is a road map for the Software Project
- It defines the tasks and milestones
- Tracking can be done by:
 - **Conducting periodic project status meetings**
 - **Evaluating the results of all reviews**
 - **Determining whether milestones were reached by the scheduled date**
 - **Compare actual start date to planned start date**
 - **Meeting informally with professionals to get their subjective opinion**

Tracking Earned Value

- The earned value system provides a common value scale for every task, regardless of the type of work being performed.
- The total hours to do the whole project are estimated, and every task is given an earned value based on its estimated percentage of the total.
- Basically, earned value is useful as it provides a quantitative technique of assessing progress on the project as a whole

EARNED VALUE ANALYSIS

- The earned value system provides a common value scale for every task, regardless of the type of work being performed.
- Simply stated, earned value is a measure of progress.
- It enables us to assess the “percent of completeness” of a project using quantitative analysis
- The total hours to do the whole project are estimated, and every task is given an earned value based on its estimated percentage of the total.

CRITICAL PATH 1-2-4-6-9-11

Tracking a Project with Earned Value

Presented by: [Name] | Date: [Date]

Project Name: [Project Name] | Version: [Version]

Page: [Page Number] / Total Pages: [Total Pages]



In the example below, compiled at the end of the second week, we see that the group has completed the following tasks with their associated earned value:

- Planning - 7.81%
- Scope - 2.64%
- Product Features - 2.64%
- User Profile - 1.98%
- Assumptions - 2.64%
- UI requirements - 5.21%

By summing the earned values for the completed tasks we determine that at the end of week 2, the cumulative earned value (the percent complete) is $7.81+2.64+2.64+1.98+2.64+5.21=22.91\%$.

Now, knowing the earned value the project group can determine if they are ahead, behind or on schedule to complete on time

Critical Path 1-2-4-8-9-11

Tracking a Project with Earned Value

Knowing this allows the project team to take action.

Such action may involve

- adjusting work practices,
- negotiating with the client for an extension to the deadline,
- etc...

It is argued that earned value provides accurate and reliable readings of performance as early as 15% into a project. This kind of quantitative project management is essential if projects are to be consistently delivered on time.

CRITICAL PATH 1-2-4-6-8-11

To determine the earned value

- The **budgeted cost of work scheduled (BCWS)** is determined for each work task represented in the schedule.
 - **BCWS_i** is the effort planned for work task *i*.
 - To determine progress at a given point along the project schedule, the value of BCWS is the sum of the BCWS_i values for all work tasks for that point in time.
- The BCWS values for all work tasks are summed to derive the **budget at completion, BAC**.
- Hence, $BAC = \sum (BCWS_k)$ for all tasks *k*

CRITICAL PATH 1-2-4-6-8-11

- Next, the value for **budgeted cost of work performed (BCWP)** is computed.
- The value for BCWP is the sum of the BCWS values for all work tasks that have actually been completed by a point in time on the project schedule.

- BCWS - budget of the activities that were planned to be completed
- BCWP - budget of the activities that actually were completed.”

- Given values for BCWS, BAC, and BCWP, important progress indicators can be computed:

- Schedule performance index, $SPI = BCWP/BCWS$
- Schedule variance, $SV = BCWP - BCWS$

SPI is an indication of the efficiency with which the project is utilizing scheduled resources.

- SPI value close to 1.0 indicates efficient execution of the project schedule.

CRITICAL PATH 1-2-4-8-9-11

- Percent scheduled for completion = BCWS/BAC

- . Indication of the percentage of work that should have been completed by time t .

- Percent complete = BCWP/BAC

- . provides a quantitative indication of the percent of completeness of the project at a given point in time, t .

- Actual cost of work performed, ACWP, is the sum of the effort actually expended on work tasks that have been completed by a point in time on the project schedule. It is then possible to compute

- . Cost performance index, CPI = BCWP/ACWP
 - . Cost variance, CV = BCWP – ACWP

- CPI value close to 1.0 provides a strong indication that the project is within its defined budget

CRITICAL PATH 1-2-4-6-9-11