



# **Unit 4- Part (a)**

# **Project Planning and Management**

**Dr. Naveen Saini  
Assistant Professor**

**Department of Information Technology  
Indian Institute of Information Technology  
Allahabad**

# Outline

---

- Introduction
- Project Planning
- Project Scheduling
- Risk Management

# Introduction.

---

- *Software project management* is aimed to ensure that the software is delivered on time, within budget and schedule constraints, and satisfies the requirements of the client
  
- Management of software projects is different from other types of management because:
  - Software is not tangible
  - Software processes are relatively new and still “under trial”
  - Larger software projects are usually “one-off” projects
  - Computer technology evolves very rapidly

# .Introduction

---

## ■ Management activities:

- Writing proposals
- Planning the project
- Scheduling the project
- Estimating the cost of the project
- Monitoring and reviewing the project's progress
- Selecting, hiring, and evaluating personnel
- Writing reports and giving presentations

# Project Planning...

---

- A *project plan* should be drawn at the start of the project. This plan drives the project and needs to be continuously adjusted
- The role of the project manager is to anticipate possible problems and be prepared with solutions for these problems
- Other plans that need be developed:
  - Quality plan
  - Validation and verification plan
  - Configuration management plan
  - Maintenance plan (correction/adaption/enhancement/prevention)
  - Staff development plan

# .Project Planning..

---

## ■ The planning process [Fig 5.2, SE-8]

Establish the project constraints

Make initial assessments of the project parameters

Define project milestones and deliverables

**while** project has not been completed or cancelled**loop**

    Draw up project schedule

    Initiate activities according to schedule

    Wait ( for a while )

    Review project progress

    Revise estimates of project parameters

    Update the project schedule

    Re-negotiate project constraints and deliverables

**if** ( problems arise )**then**

        Initiate technical review and possible revision

**end if**

**end loop**

# ..Project Planning.

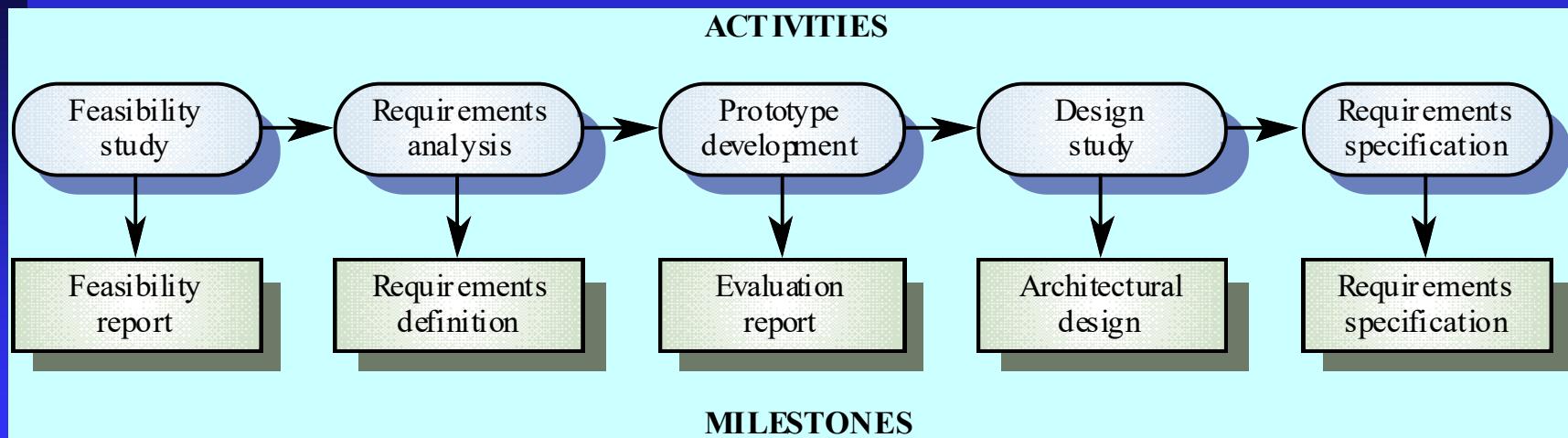
---

- The structure of the project plan:

- Introduction (objectives, constraints)
- Work breakdown (activities, milestones, deliverables)
- Project organization (team structure, personnel involved, roles)
- Hardware and software resources needed (prices, delivery schedule)
- Risk analysis (types of risk, probabilities, solutions to prevent or reduce the risk)
- Project schedule (dependencies between activities/tasks, work assignments, time allocated per task)
- Monitoring and reporting mechanisms (reports, dates)

# ...Project Planning

- **Milestone** = end-point of a specific, distinct software process activity or task (for each milestone a report should be presented to the management)
- **Deliverable** = project result delivered to the client
- In order to establish milestones the phases of the software process need be divided in basic activities/tasks. Example for requirements engineering [Fig. 5.3, SE-8]



# Project Scheduling.....

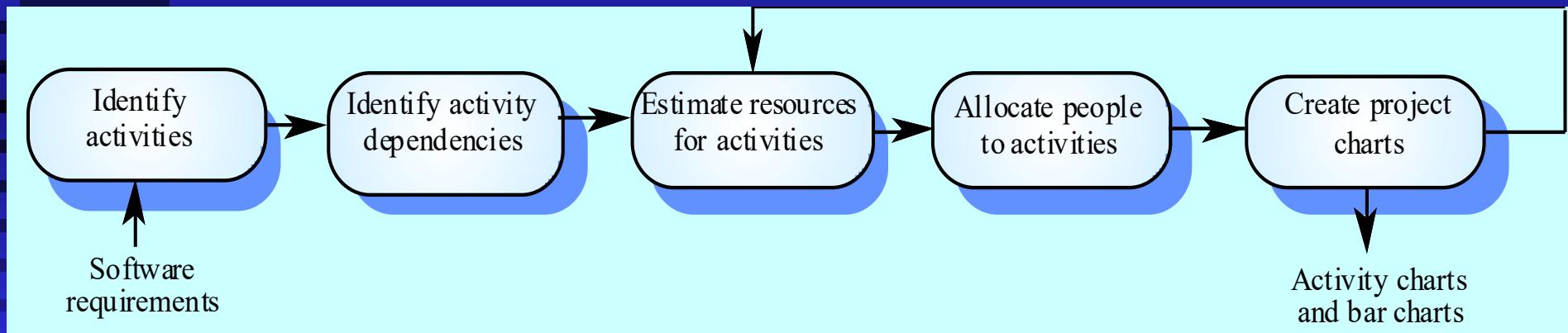
---

## ■ Software managers:

- Divide the project in activities/tasks
- Estimate time and resources needed to finish the project
- Allocate resources to tasks
- Try to employ efficiently all the project personnel
- Minimize dependencies between tasks and teams
- Prepare contingency plans
- Rely on experience and intuition

# .Project Scheduling.....

- The scheduling process [Fig. 5.4, SE-8]



# ..Project Scheduling....

---

- Graphical notations used in software project scheduling:
  - *Tables*: summary description of tasks
  - *Bar charts*: show schedule against the time
  - *Activity charts*: graphs that depict dependencies between tasks and indicate the *critical path* (the longest path in the activity graph)

# ...Project Scheduling...

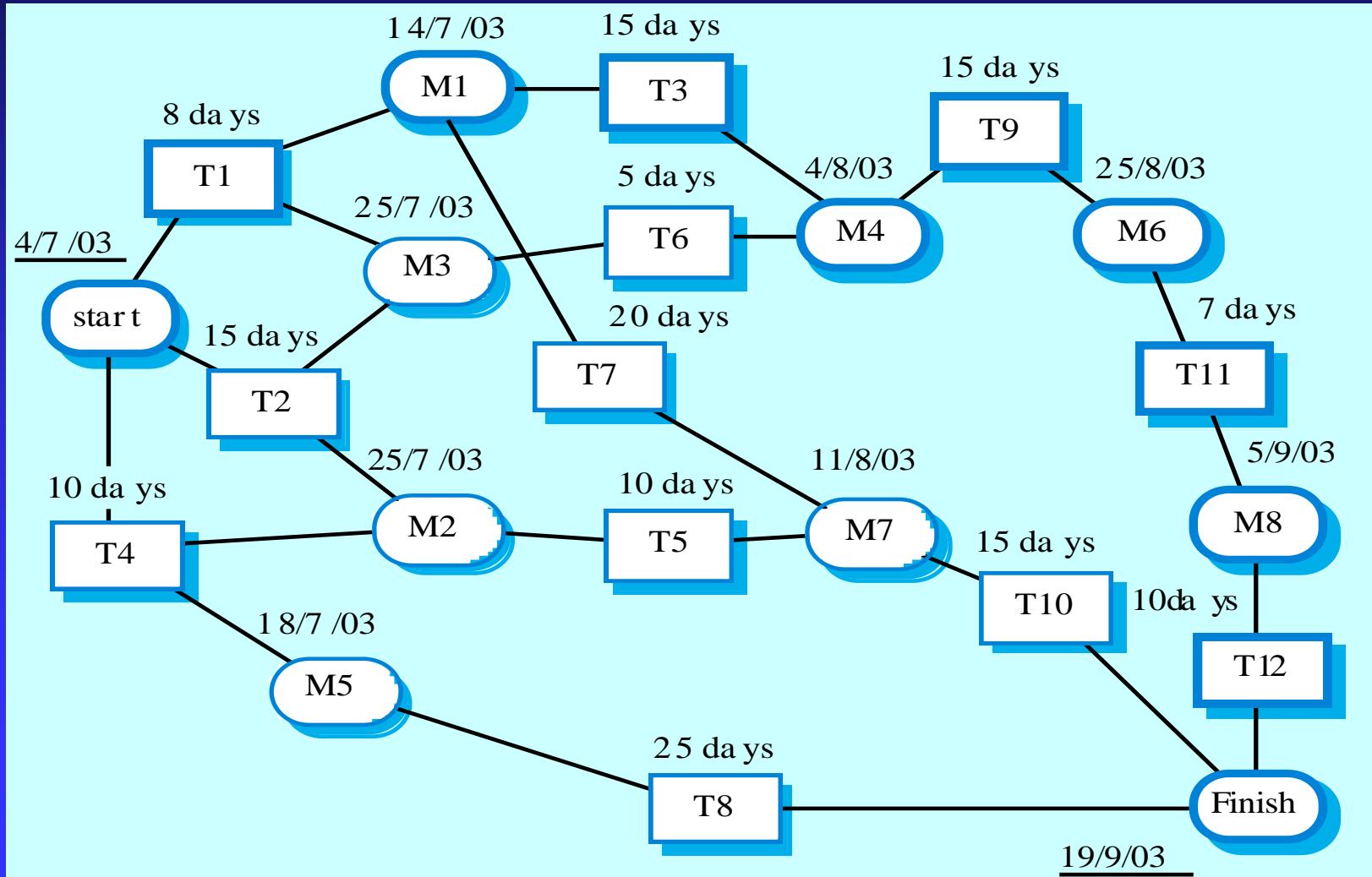
---

- Example of tabular description [Fig. 5.5, SE-8]:

Task	Duration (days)	Dependencies
T1	8	
T2	15	
T3	15	T1 (M1)
T4	10	
T5	10	T2, T4 (M2)
T6	5	T1, T2 (M3)
T7	20	T1 (M1)
T8	25	T4 (M5)
T9	15	T3, T6 (M4)
T10	15	T5, T7 (M7)
T11	7	T9 (M6)
T12	10	T11 (M8)

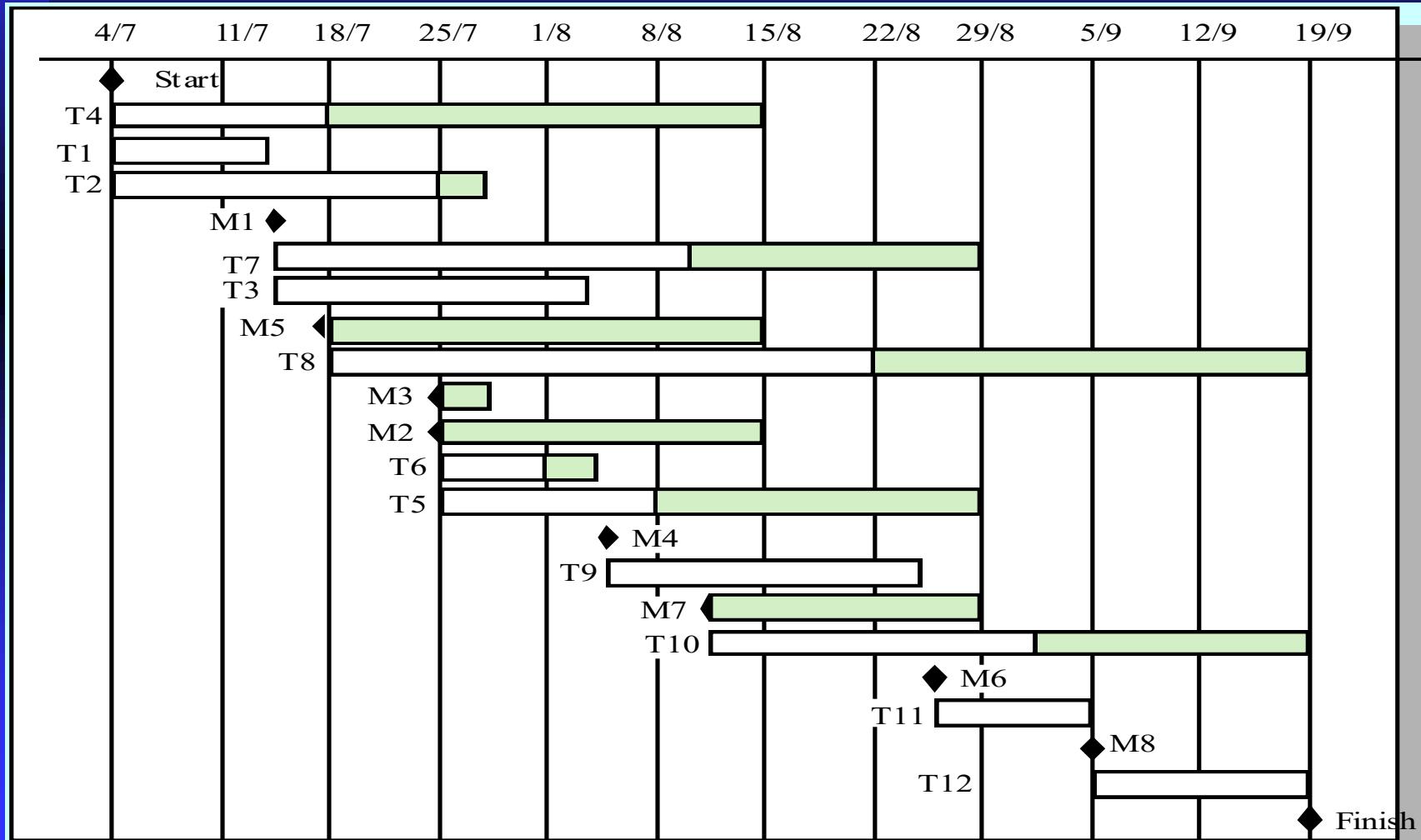
# ....Project Scheduling..

- Example of activity chart [Fig. 5.6, SE-8]



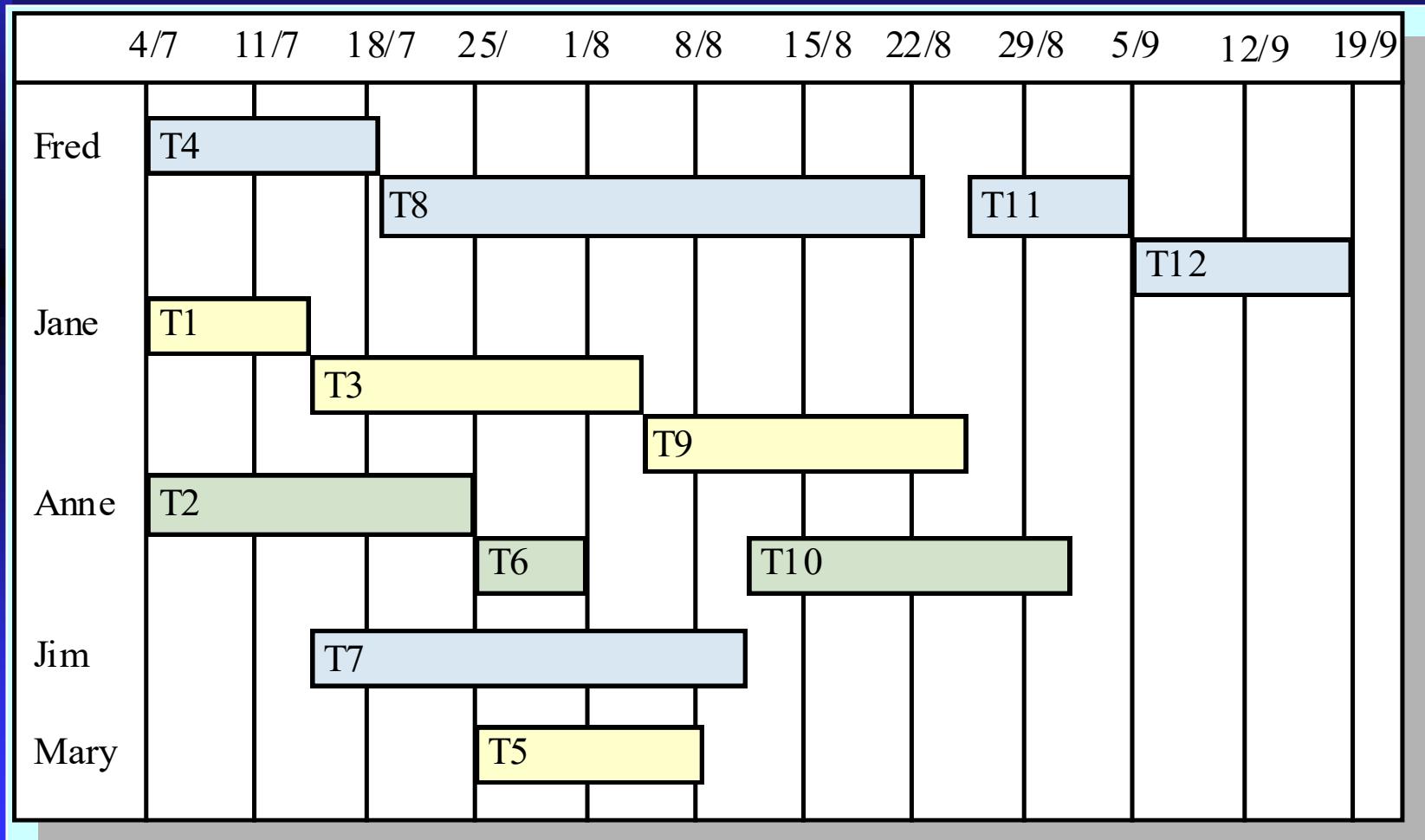
# .....Project Scheduling.

- Example of bar chart [Fig. 5.7, SE-8]



# .....Project Scheduling

## ■ Staff allocation chart [Fig. 5.8, SE-8]



# Risk Management.....

---

- *Risk* = some adverse circumstance that may happen and affect negatively the project, the product, and/or the business
- Categories of risk:
  - Project risks: threaten the project to deliver it on time and within budget
  - Product risks: threaten the quality and timeliness
  - Business risks: threaten the buy ability of the project
- *Risk management* means anticipating risks and preparing plans to reduce their effect

# .Risk Management.....

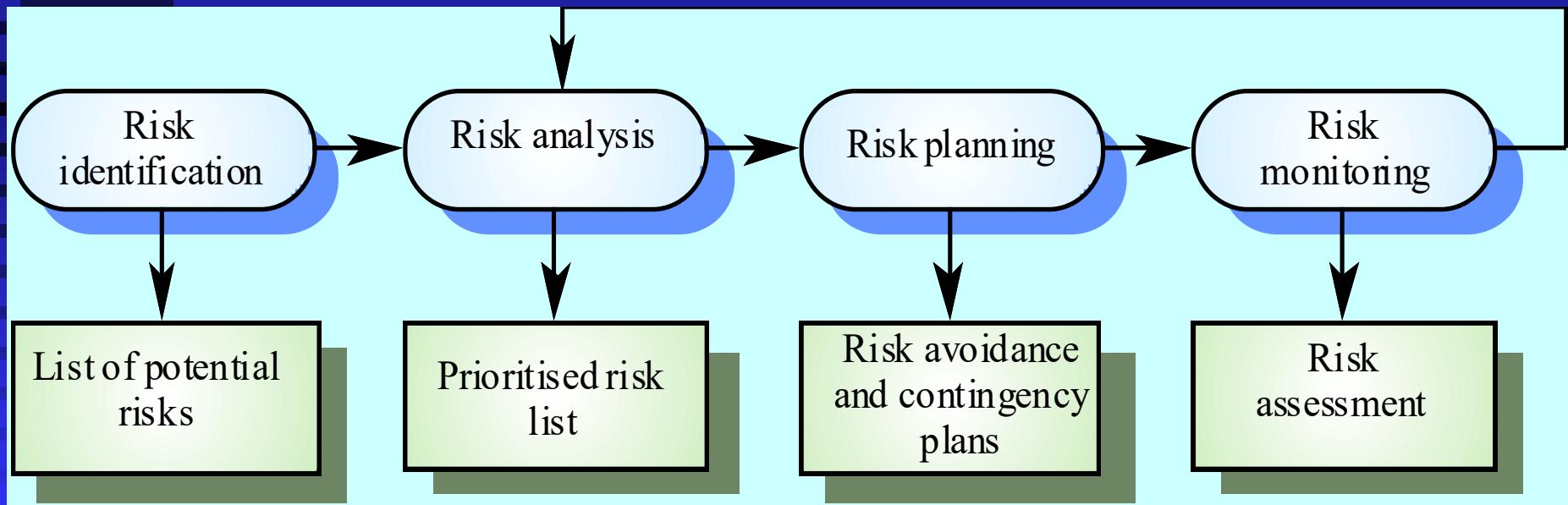
---

- Examples of risks in the software process [Fig. 5.9, SE-8]

<b>Risk</b>	<b>Affects</b>	<b>Description</b>
Staff turnover	Project	Experienced staff will leave the project before it is finished.
Management change	Project	There will be a change of organisational management with different priorities.
Hardware unavailability	Project	Hardware that is essential for the project will not be delivered on schedule.
Requirements change	Project and product	There will be a larger number of changes to the requirements than anticipated.
Specification delays	Project and product	Specifications of essential interfaces are not available on schedule
Size underestimate	Project and product	The size of the system has been underestimated.
CASE tool under-performance	Product	CASE tools which support the project do not perform as anticipated
Technology change	Business	The underlying technology on which the system is built is superseded by new technology.
Product competition	Business	A competitive product is marketed before the system is completed.

# ..Risk Management.....

- The risk management activities [Fig. 5.10, SE-8]



# ...Risk Management....

## ■ Types of risk in *risk identification* [Fig. 5.11, SE-8]

<b>Risk type</b>	<b>Potential indicators</b>
Technology	Late delivery of hardware or support software, many reported technology problems
People	Poor staff morale, poor relationships amongst team member, job availability
Organisational	Organisational gossip, lack of action by senior management
Tools	Reluctance by team members to use tools, complaints about CASE tools, demands for higher-powered work stations
Requirements	Many requirements change requests, customer complaints
Estimation	Failure to meet agreed schedule, failure to clear reported defects

# ....Risk Management...

---

## ■ *Risk analysis:*

- Estimate risk probability:
  - ◆ Very low (< 10%)
  - ◆ Low (10-25%)
  - ◆ Moderate (25-50%)
  - ◆ High (50-75%)
  - ◆ Very high (> 75%)
- Establish risk seriousness:
  - ◆ Insignificant
  - ◆ Tolerable
  - ◆ Serious
  - ◆ Catastrophic

# .....Risk Management..

---

- *Risk planning* means preparing a strategy to deal with each of the risks identified
- Classes of strategies:
  - Avoidance strategies: the probability of the risk will be diminished
  - Minimization strategies: the effect of the risk will be reduced
  - Contingency strategies: plans for the worst case scenarios

# .....Risk Management.

## ■ Examples of risk management strategies [Fig. 5.13, SE-8]

Risk	Strategy
Organisational financial problems	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Recruitment problems	Alert customer of potential difficulties and the possibility of delays, investigate buying-in components.
Staff illness	Reorganise team so that there is more overlap of work and people therefore understand each other's jobs.
Defective components	Replace potentially defective components with bought-in components of known reliability.

Risk	Strategy
Requirements changes	Derive traceability information to assess requirements change impact, maximise information hiding in the design.
Organisational restructuring	Prepare a briefing document for senior management showing how the project is making a very important contribution to the goals of the business.
Database performance	Investigate the possibility of buying a higher-performance database.
Underestimated development time	Investigate buying in components, investigate use of a program generator

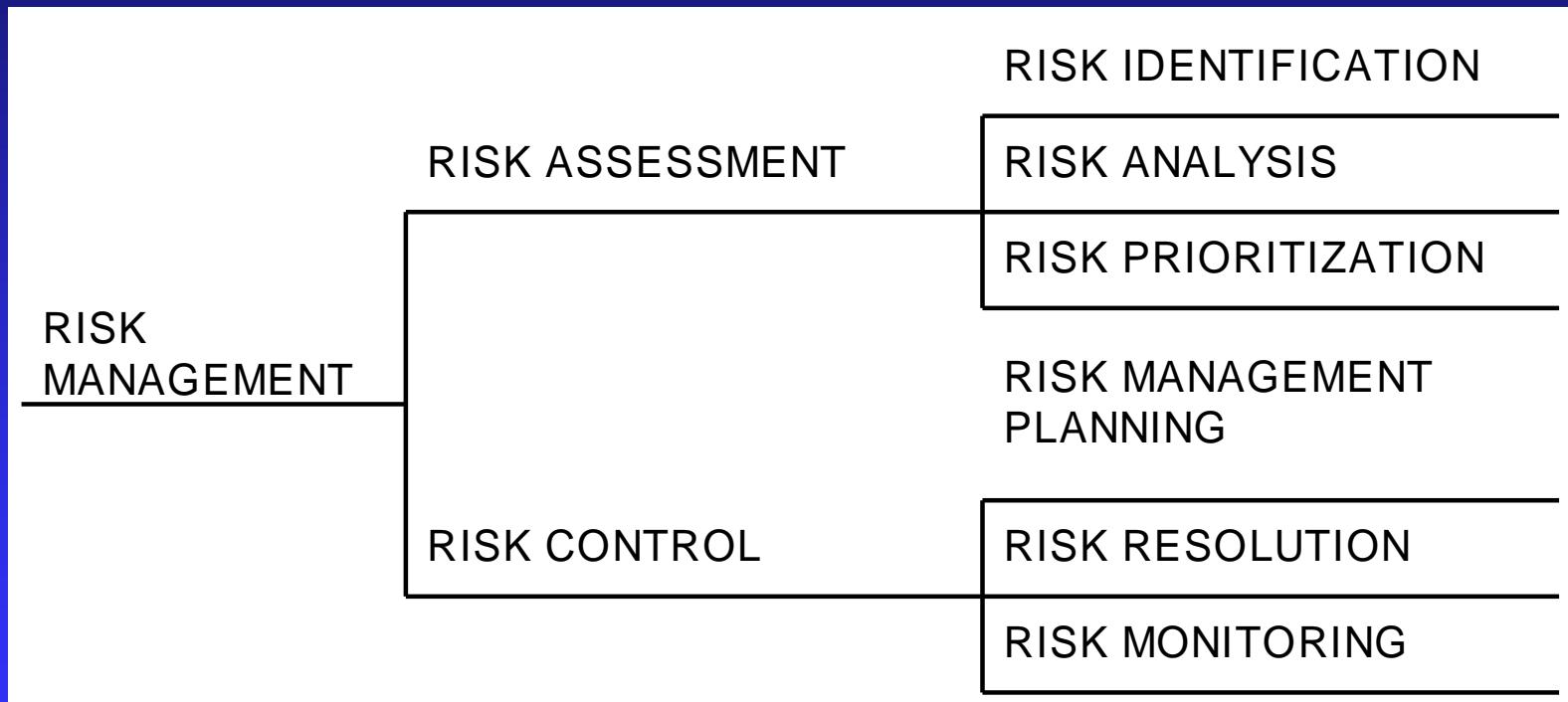
# .....Risk Management

---

## ■ *Risk monitoring:*

- Frequently re-assess the risks
  - ◆ Changes in risk probability?
  - ◆ Changes in risk gravity?
- Take into consideration risk factors
- Discuss key risks at each management project progress meeting

# Risk Management Tasks



# **Project Monitoring Plans**

- A plan is a mere document that can guide and must be executed
- To ensure execution goes as per plan, it must be monitored and controlled
- Monitoring requires measurements and methods for interpreting them
- Monitoring plan has to plan for all the tasks related to monitoring

# Measurements

---

- Must plan for measurements in a project
- Without planning, measurements will not be done
- Main measurements – effort, size, schedule, and defects
  - Effort – as this is the main resource; often tracked through effort reporting tools
  - Defects – as they determine quality; often defect logging and tracking systems used
- During planning – what will be measured, how, tool support, and data management

# **Project Tracking....**

---

- Goal: To get visibility in project execution so corrective actions can be taken when needed to ensure project succeeds
- Different types of monitoring done at projects; measurements provide data for it

# Tracking...

## ■ Activity-level monitoring

- Each activity in detailed schd is getting done
- Often done daily by managers
- A task done marked 100%; tools can determine status of higher level tasks

## ■ Status reports

- Generally done weekly to take stock
- Summary of activities completed, pending
- Issues to be resolved

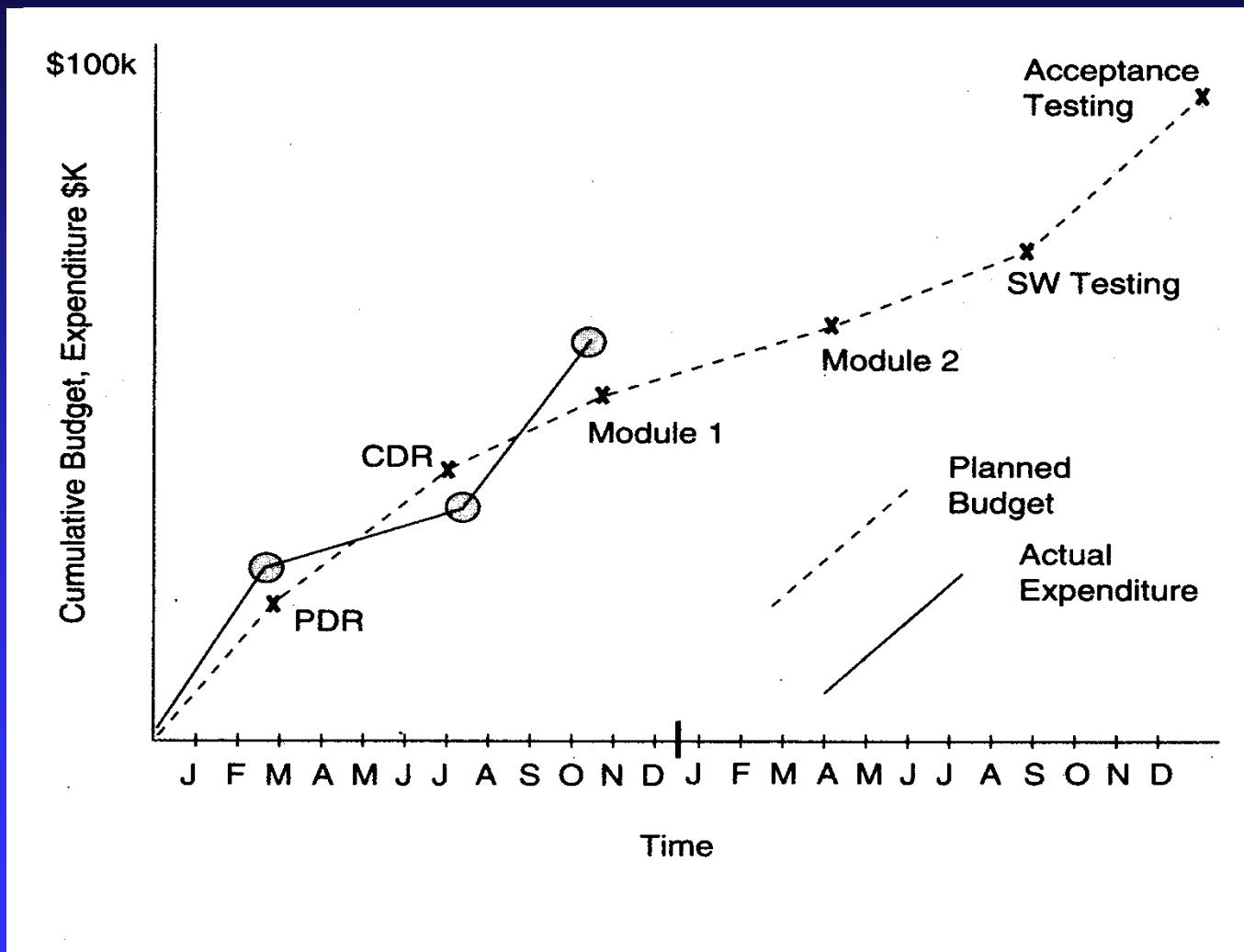
# Tracking...

## ■ Milestone analysis

- A bigger review at milestones
- Actual vs estimated for effort and schedule is done
- Risks are revisited
- Changes to product and their impact may be analyzed

## ■ Cost-schedule milestone graph is another way of doing this

# Cost-schedule milestone graph



# Project Management Plan

- The project management plan (PMP) contains outcome of all planning activities - focuses on overall project management
  
- Besides PMP, a project schedule is needed
  - Reflects what activities get done in the project
  - Microsoft project (MSP) can be used for this
  - Based on project planning; is essential for day-to-day management
  - Does not replace PMP !

# PMP Structure - Example

---

- Project overview - customer, start and end date, overall effort, overall value, main contact persons, project milestones, development environment..
- Project planning - process and tailoring, requirements change management, effort estimation, quality goals and plan, risk management plan, ..

# PMP Example ...

- Project tracking - data collection, analysis frequency, escalation procedures, status reporting, customer complaints, ...
- Project team, its organization, roles and responsibility, ...

# **Project Planning - Summary**

---

- Project planning forms the foundation of project management
- Key aspects: effort and schedule estimation, quality planning, risk mgmt., ...
- Outputs of all can be documented in a PMP, which carries all relevant info about project
- Besides PMP, a detailed project schedule maintains tasks to be done in the project

# Estimation for Software Projects

(Source: Pressman, R. *Software Engineering: A Practitioner's Approach*. McGraw-Hill, 2005)

# **Estimation for Software Projects**

---

- Project planning
- Scope and feasibility
- Project resources
- Estimation of project cost and effort
- Decomposition techniques
- Empirical estimation models

# **Project Planning**

# Software Project Planning

---

- Software project planning encompasses five major activities
  - Estimation, scheduling, risk analysis, quality management planning, and change management planning
- Estimation determines how much money, effort, resources, and time it will take to build a specific system or product
- The software team first estimates
  - The work to be done
  - The resources required
  - The time that will elapse from start to finish
- Then they establish a project schedule that
  - Defines tasks and milestones
  - Identifies who is responsible for conducting each task
  - Specifies the inter-task dependencies

# Observations on Estimation

---

- Planning requires *technical managers* and the software team to make an initial commitment
- Process and project metrics can provide a historical perspective and valuable input for generation of quantitative estimates
- Past experience can aid greatly
- Estimation carries inherent risk, and this risk leads to uncertainty
- The availability of historical information has a strong influence on estimation risk

(More on next slide)

# Observations on Estimation (continued)

---

- When software metrics are available from past projects
  - Estimates can be made with greater assurance
  - Schedules can be established to avoid past difficulties
  - Overall risk is reduced
- Estimation risk is measured by the **degree of uncertainty** in the **quantitative estimates for cost, schedule, and resources**
- Nevertheless, a project manager should not become obsessive about estimation
  - Plans should be iterative and allow adjustments as time passes and more is made certain

"It is the mark of an instructed mind to rest satisfied with the degree of precision that the nature of the subject admits, and not to seek exactness when only an approximation of the truth is possible."      ARISTOTLE

# Task Set for Project Planning

---

- 1) Establish project scope
- 2) Determine feasibility
- 3) Analyze risks
- 4) Define required resources
  - a) Determine human resources required
  - b) Define reusable software resources
  - c) Identify environmental resources
- 5) Estimate cost and effort
  - a) Decompose the problem
  - b) Develop two or more estimates using different approaches
  - c) Reconcile the estimates
- 6) Develop a project schedule
  - a) Establish a meaningful task set
  - b) Define a task network
  - c) Use scheduling tools to develop a timeline chart
  - d) Define schedule tracking mechanisms

# **Example Project: Campus Information Access Kiosk**

- Both podium-high and desk-high terminals located throughout the campus in all classroom buildings, admin buildings, labs, and dormitories
- Hand/Palm-login and logout (seamlessly)
- Voice input
- Optional audio/visual or just visual output
- Immediate access to all campus information plus
  - E-mail
  - Cell phone voice messaging
  - Text messaging

# **Scope and Feasibility**

# Software Scope

- Software scope describes
  - The functions and features that are to be delivered to end users
  - The data that are input to and output from the system
  - The "content" that is presented to users as a consequence of using the software
  - The performance, constraints, interfaces, and reliability that bound the system
- Scope can be define using two techniques
  - A narrative description of software scope is developed after communication with all stakeholders
  - A set of use cases is developed by end users

(More on next slide)

# Software Scope (continued)

- After the scope has been identified, two questions are asked
  - Can we build software to meet this scope?
  - Is the project feasible?
- Software engineers too often rush (or are pushed) past these questions
- Later they become mired in a project that is doomed from the onset

# Feasibility

- After the scope is resolved, feasibility is addressed
- Software feasibility has four dimensions
  - Technology – Is the project technically feasible? Is it within the state of the art? Can defects be reduced to a level matching the application's needs?
  - Finance – Is it financially feasible? Can development be completed at a cost that the software organization, its client, or the market can afford?
  - Time – Will the project's time-to-market beat the competition?
  - Resources – Does the software organization have the resources needed to succeed in doing the project?

Another view recommends the following feasibility dimensions: technological, economical, legal, operational, and schedule issues (TELOS)

# Project Resources

# Resource Estimation

- Three major categories of software engineering resources
  - People
  - Development environment
  - Reusable software components
    - ◆ Often neglected during planning but become a paramount concern during the construction phase of the software process
- Each resource is specified with
  - A description of the resource
  - A statement of availability
  - The time when the resource will be required
  - The duration of time that the resource will be applied

Time window

# Categories of Resources

## People

- Number required
- Skills required
- Geographical location

## Development Environment

- Software tools
- Computer hardware
- Network resources

The  
Project

## Reusable Software Components

- Off-the-shelf components
- Full-experience components
- Partial-experience components
- New components

# Human Resources

- Planners need to select the number and the kind of people skills needed to complete the project
- They need to specify the organizational position and job specialty for each person
- Small projects of a few person-months may only need one individual
- Large projects spanning many person-months or years require the location of the person to be specified also
- The number of people required can be determined only after an estimate of the development effort

# Development Environment Resources

- A software engineering environment (SEE) incorporates hardware, software, and network resources that provide platforms and tools to develop and test software work products
- Most software organizations have many projects that require access to the SEE provided by the organization
- Planners must identify the time window required for hardware and software and verify that these resources will be available

# Reusable Software Resources

- **Off-the-shelf components**
  - Components are from a third party or were developed for a previous project
  - Ready to use; fully validated and documented; virtually no risk
- **Full-experience components**
  - Components are similar to the software that needs to be built
  - Software team has full experience in the application area of these components
  - Modification of components will incur relatively low risk
- **Partial-experience components**
  - Components are related somehow to the software that needs to be built but will require substantial modification
  - Software team has only limited experience in the application area of these components
  - Modifications that are required have a fair degree of risk
- **New components**
  - Components must be built from scratch by the software team specifically for the needs of the current project
  - Software team has no practical experience in the application area
  - Software development of components has a high degree of risk

# **Estimation of Project Cost and Effort**

# Factors Affecting Project Estimation

- The accuracy of a software project estimate is predicated on
  - The degree to which the planner has properly estimated the size (e.g., KLOC) of the product to be built
  - The ability to translate the size estimate into human effort, calendar time, and money
  - The degree to which the project plan reflects the abilities of the software team
  - The stability of both the product requirements and the environment that supports the software engineering effort

# Project Estimation Options

- Options for achieving reliable cost and effort estimates
  - 1) Delay estimation until late in the project (we should be able to achieve 100% accurate estimates after the project is complete)
  - 2) Base estimates on similar projects that have already been completed
  - 3) Use relatively simple decomposition techniques to generate project cost and effort estimates
  - 4) Use one or more empirical estimation models for software cost and effort estimation
- Option #1 is not practical, but results in good numbers
- Option #2 can work reasonably well, but it also relies on other project influences being roughly equivalent
- Options #3 and #4 can be done in tandem to cross check each other

# Project Estimation Approaches

- **Decomposition techniques**
  - These take a "**divide and conquer**" approach
  - Cost and effort estimation are performed in a stepwise fashion by breaking down a project into major functions and related software engineering activities
- **Empirical estimation models**
  - Offer a potentially valuable estimation approach if the historical data used to seed the estimate is good

# **Decomposition Techniques**

# Introduction

- Before an estimate can be made and decomposition techniques applied, the planner must
  - Understand the scope of the software to be built
  - Generate an estimate of the software's size
- Then one of two approaches are used
  - Problem-based estimation
    - ◆ Based on either source lines of code or function point estimates
  - Process-based estimation
    - ◆ Based on the effort required to accomplish each task

# Approaches to Software Sizing

- Function point sizing
  - Develop estimates of the information domain characteristics
- Standard component sizing
  - Estimate the number of occurrences of each standard component
  - Use historical project data to determine the delivered LOC size per standard component
- Change sizing
  - Used when changes are being made to existing software
  - Estimate the number and type of modifications that must be accomplished
  - Types of modifications include reuse, adding code, changing code, and deleting code
  - An effort ratio is then used to estimate each type of change and the size of the change

The results of these estimates are used to compute an optimistic (low), a most likely, and a pessimistic (high) value for software size

# Problem-Based Estimation

- 1) Start with a bounded statement of scope
- 2) Decompose the software into problem functions that can each be estimated individually
- 3) Compute an LOC or FP value for each function
- 4) Derive cost or effort estimates by applying the LOC or FP values to your baseline productivity metrics (e.g., LOC/person-month or FP/person-month)
- 5) Combine function estimates to produce an overall estimate for the entire project

# Problem-Based Estimation (continued)

- In general, the LOC/pm and FP/pm metrics should be computed by project domain
  - Important factors are team size, application area, and complexity
- LOC and FP estimation differ in the level of detail required for decomposition with each value
  - For LOC, decomposition of functions is essential and should go into considerable detail (the more detail, the more accurate the estimate)
  - For FP, decomposition occurs for the five information domain characteristics and the 14 adjustment factors
    - ◆ External inputs, external outputs, external inquiries, internal logical files, external interface files

# Problem-Based Estimation (continued)

- For both approaches, the planner uses lessons learned to estimate an optimistic, most likely, and pessimistic size value for each function or count (for each information domain value)
- Then the expected size value S is computed as follows:

$$S = (S_{\text{opt}} + 4S_m + S_{\text{pess}}) / 6$$

- Historical LOC or FP data is then compared to S in order to cross-check it

# Process-Based Estimation

- 1) Identify the set of functions that the software needs to perform as obtained from the project scope
- 2) Identify the series of framework activities that need to be performed for each function
- 3) Estimate the effort (in person months) that will be required to accomplish each software process activity for each function

# Process-Based Estimation (continued)

- 4) Apply average labor rates (i.e., cost/unit effort) to the effort estimated for each process activity
- 5) Compute the total cost and effort for each function and each framework activity (See table in Pressman, p. 655)
- 6) Compare the resulting values to those obtained by way of the LOC and FP estimates
  - If both sets of estimates agree, then your numbers are highly reliable
  - Otherwise, conduct further investigation and analysis concerning the function and activity breakdown

This is the most commonly used of the two estimation techniques  
(problem and process)

# Reconciling Estimates

- The results gathered from the various estimation techniques must be reconciled to produce a single estimate of effort, project duration, and cost
- If widely divergent estimates occur, investigate the following causes
  - The scope of the project is not adequately understood or has been misinterpreted by the planner
  - Productivity data used for problem-based estimation techniques is inappropriate for the application, obsolete (i.e., outdated for the current organization), or has been misapplied
- The planner must determine the cause of divergence and then reconcile the estimates

# **Empirical Estimation Models**

# Introduction

- Estimation models for computer software use empirically derived formulas to predict effort as a function of LOC or FP
- Resultant values computed for LOC or FP are entered into an estimation model
- The empirical data for these models are derived from a limited sample of projects
  - Consequently, the models should be calibrated to reflect local software development conditions

# COCOMO

- Stands for COnstructive COst MOdel
- Introduced by Barry Boehm in 1981 in his book “Software Engineering Economics”
- Became one of the well-known and widely-used estimation models in the industry
- It has evolved into a more comprehensive estimation model called COCOMO II
- COCOMO II is actually a hierarchy of three estimation models
- As with all estimation models, it requires sizing information and accepts it in three forms: object points, function points, and lines of source code

# COCOMO Model

- Application composition model - Used during the early stages of software engineering when the following are important
  - Prototyping of user interfaces
  - Consideration of software and system interaction
  - Assessment of performance
  - Evaluation of technology maturity
- Early design stage model – Used once requirements have been stabilized and basic software architecture has been established
- Post-architecture stage model – Used during the construction of the software

# COCOMO Cost Drivers

## ■ Personnel Factors

- Applications experience
- Programming language experience
- Virtual machine experience
- Personnel capability
- Personnel experience
- Personnel continuity
- Platform experience
- Language and tool experience

## ■ Product Factors

- Required software reliability
- Database size
- Software product complexity
- Required reusability
- Documentation match to life cycle needs
- Product reliability and complexity

# COCOMO Cost Drivers (continued)

## ■ Platform Factors

- Execution time constraint
- Main storage constraint
- Computer turn-around time
- Virtual machine volatility
- Platform volatility
- Platform difficulty

## ■ Project Factors

- Use of software tools
- Use of modern programming practices
- Required development schedule
- Classified security application
- Multi-site development
- Requirements volatility

# Make/Buy Decision

- It is often more cost effective to acquire rather than develop software
- Managers have many acquisition options
  - Software may be purchased (or licensed) off the shelf
  - “Full-experience” or “partial-experience” software components may be acquired and integrated to meet specific needs
  - Software may be custom built by an outside contractor to meet the purchaser’s specifications
- The make/buy decision can be made based on the following conditions
  - Will the software product be available sooner than internally developed software?
  - Will the cost of acquisition plus the cost of customization be less than the cost of developing the software internally?
  - Will the cost of outside support (e.g., a maintenance contract) be less than the cost of internal support?



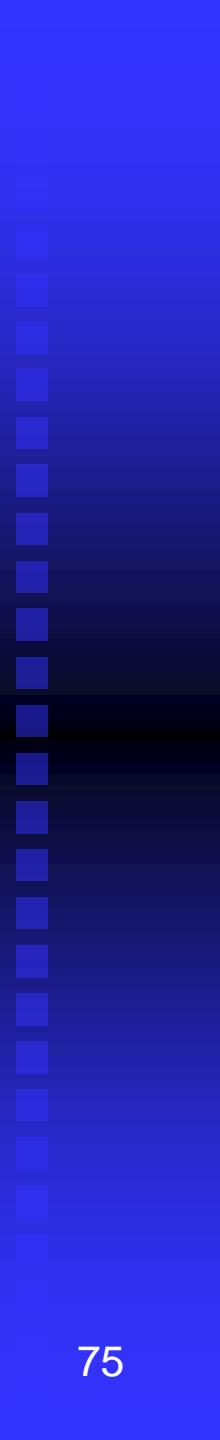
# Next Class Topics (~4 Hrs)

- Estimation models:
  - Size estimation (LOC and FP), effort estimation (SEL model, WF model) [2 Hrs]
  - COCOMO, Halsted size estimation and quality estimation [2 Hrs]

# Reference

---

- Ian Sommerville, Software Engineering, 8th Ed., Addison-Wesley, 2006
- Bruce R. Maxim, Roger S. Pressman, Software Engineering, McGraw Hill Education; Eighth edition
- <http://www.software-engin.com/>



# Thank you!