Software Project Management

Intro to Project Management

Course Objectives

- Understand the fundamental principles of Software Project management & will also have a good knowledge of responsibilities of project manager and how to handle these.
- Be familiar with the different methods and techniques used for project management.
- By the end of this course student will have good knowledge of the issues and challenges faced while doing the Software project Management and will also be able to understand why majority of the software projects fails and how that failure probability can be reduced effectively. Will be able to do the Project Scheduling, tracking, Risk analysis, Quality management and Project Cost estimation using different techniques

Unit I

Introduction

Project – Definition

- In the broadest sense, a **project** is a specific, finite task to be accomplished. Any activity that results in a deliverable or a product.
- Projects always begin with a problem. The project is to provide the solution to this problem.
- When the project is finished it must be evaluated to determine whether it satisfies the objectives and goals.

What is Management?

- Management can be defined as all activities and tasks undertaken by one or more persons for the purpose of planning and controlling the activities of others in order to achieve objectives or complete an activity that could not be achieved by others acting independently.
  - Management functions can be categorized as
    - Planning
    - Organizing
    - Staffing
    - Directing
    - Controlling

Management Functions

- **Planning**
  Predetermining a course of action for accomplishing organizational Objectives
- **Organizing**
  Arranging the relationships among work units for accomplishment of objectives and the granting of responsibility and authority to obtain those objectives
- **Staffing**
  Selecting and training people for positions in the organization
- **Directing**
  Creating an atmosphere that will assist and motivate people to achieve desired end results
- **Controlling**
  Establishing, measuring, and evaluating performance of activities toward planned objectives

What is Project Management

- "The application of knowledge, skills, tools and techniques to project activities in order to meet project requirements"

What is Project Management

- Project management is a system of
  - management procedures,
  - practices,
  - technologies,
  - skills, and
  - experience

  that are necessary to successfully manage a project.

Software Project Management

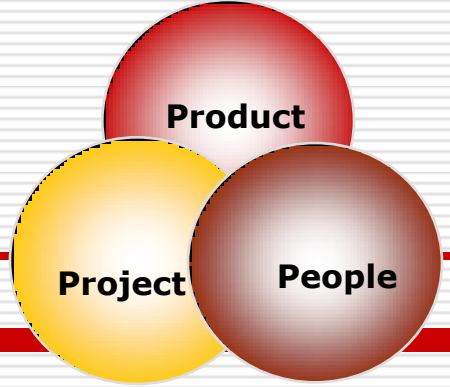
- Concerned with activities involved in ensuring
  that software is delivered:
  - on time
  - on schedule
  - in accordance with the requirements of the organization developing and procuring the software

Project Stakeholders

- Stakeholders are the people involved in or affected by the project actives
  - Stakeholders include
    - The project sponsor and project team
    - Support staff
    - Customers
    - Users
    - Suppliers
    - Opponents to the project

Project Characteristics

- One clear objective
  - A well defined set of end results
  - Goal oriented
  - End product or service must result
- Finite
  - Fixed timeline, start date, end date, milestone dates
- Limited
  - Budget, Resources, Time
- Life Cycle
  - Recognizable sequence of phases



Software Project Management

Product

1. Assessing Processes
2. Awareness of Process Standards
3. Defining the Product
4. Evaluating Alternative Processes
5. Managing Requirements
6. Managing Subcontractors
7. Performing the Initial Assessment
8. Selecting Methods and Tools
9. Tailoring Processes
10. Tracking Product Quality
11. Understanding Development Activities

Project

12. Building a WBS
13. Documenting Plans
14. Estimating Costs
15. Estimating Effort
16. Managing Risks
17. Monitoring Development
18. Scheduling
19. Selecting Metrics
20. Selecting Project Mgmt Tools
21. Tracking Process
22. Tracking Project Progress

People

23. Appraising Performance
24. Handling Intellectual Property
25. Holding Effective Meetings
26. Interaction and Communication
27. Leadership
28. Managing Change
29. Negotiating Successfully
30. Planning Careers
31. Presenting Effectively
32. Recruiting
33. Selecting a Team
34. Teambuilding

34 Competencies Every Software Project Manager Needs to Know

Product Life Cycles

- Products also have life cycles
- The Systems Development Life Cycle (SDLC) is a framework for describing the phases involved in developing and maintaining information systems
- Typical SDLC phases include planning, analysis, design, implementation, and support

Steps in SDLC

- Concept Exploration
- System exploration
- Requirements
- Design
- Implementation
- Installation
- Operations and support
- Maintenance
- Retirement

Process & Process Model

- Software Process
  - the set of activities, methods, and practices that are used in the production and evolution of software
- Software Process Model
  - one specific embodiment of a software process architecture

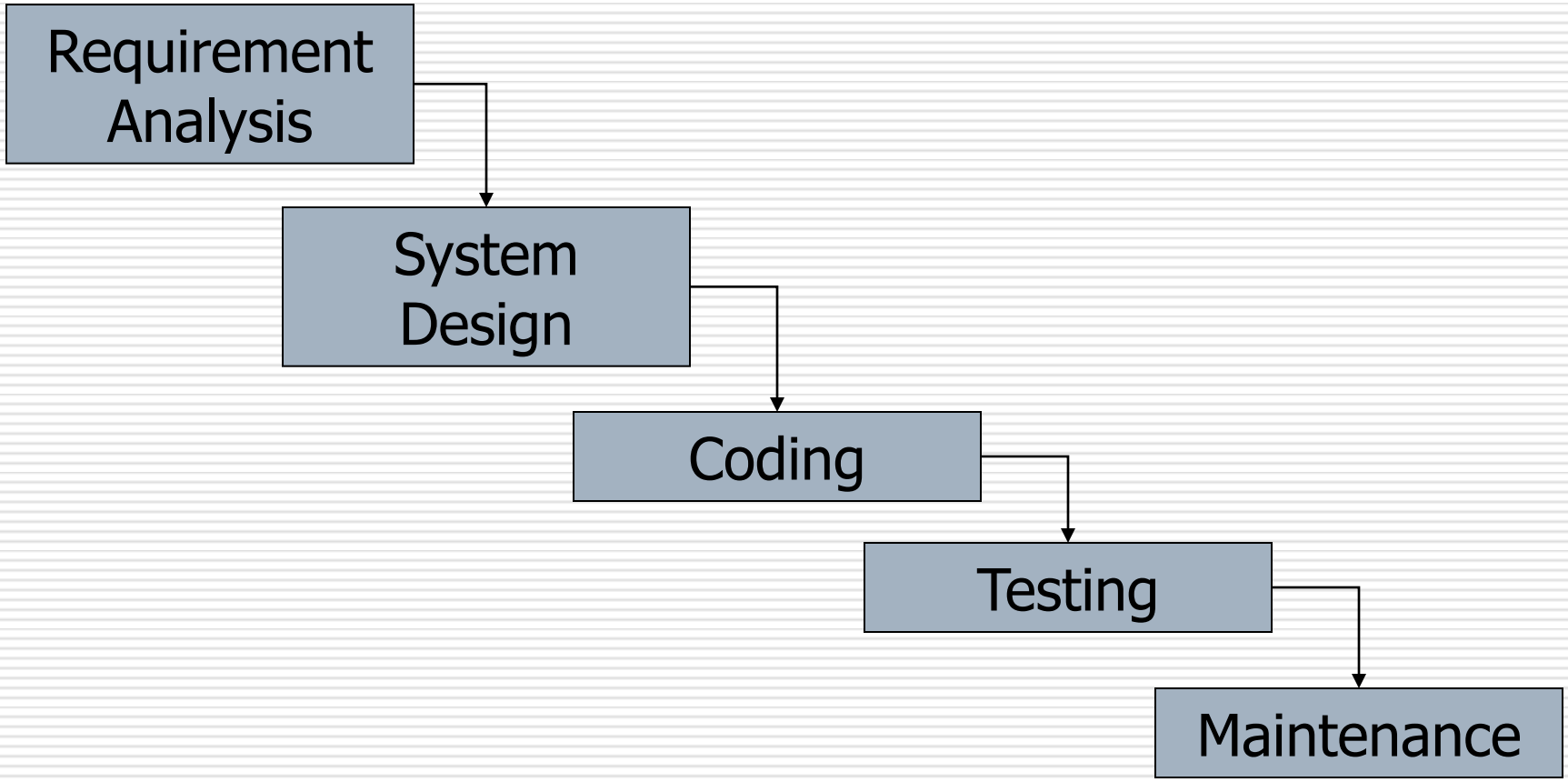
Why Modeling?

- To provide a common understanding
- To locate any inconsistencies, redundancies and omissions
- To reflect the development goals and provide early evaluation
- To assist development team to understand any special situation

Sample SDLC Models

- Waterfall model: has well-defined, linear stages of systems development and support
- Spiral model: shows that software is developed using an iterative or spiral approach rather than a linear approach
- Incremental release model: provides for progressive development of operational software
- RAD model: used to produce systems quickly without sacrificing quality
- Prototyping model: used for developing prototypes to clarify user requirements

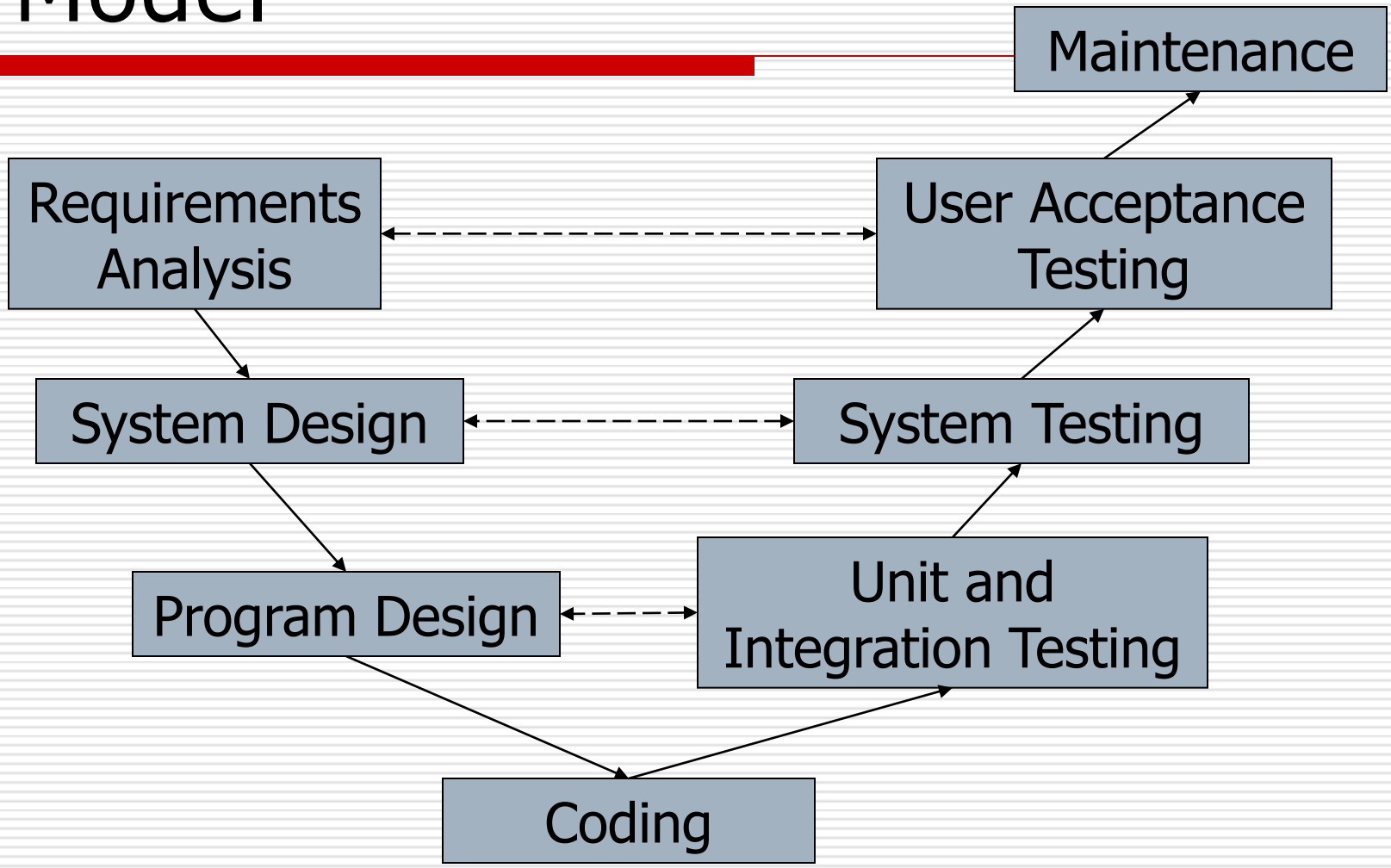
Waterfall Model



Waterfall Model (cont'd)

- classical
- one-shot approach
- effective control
- limited scope of iteration
- long cycle time
- not suitable for system of high uncertainty

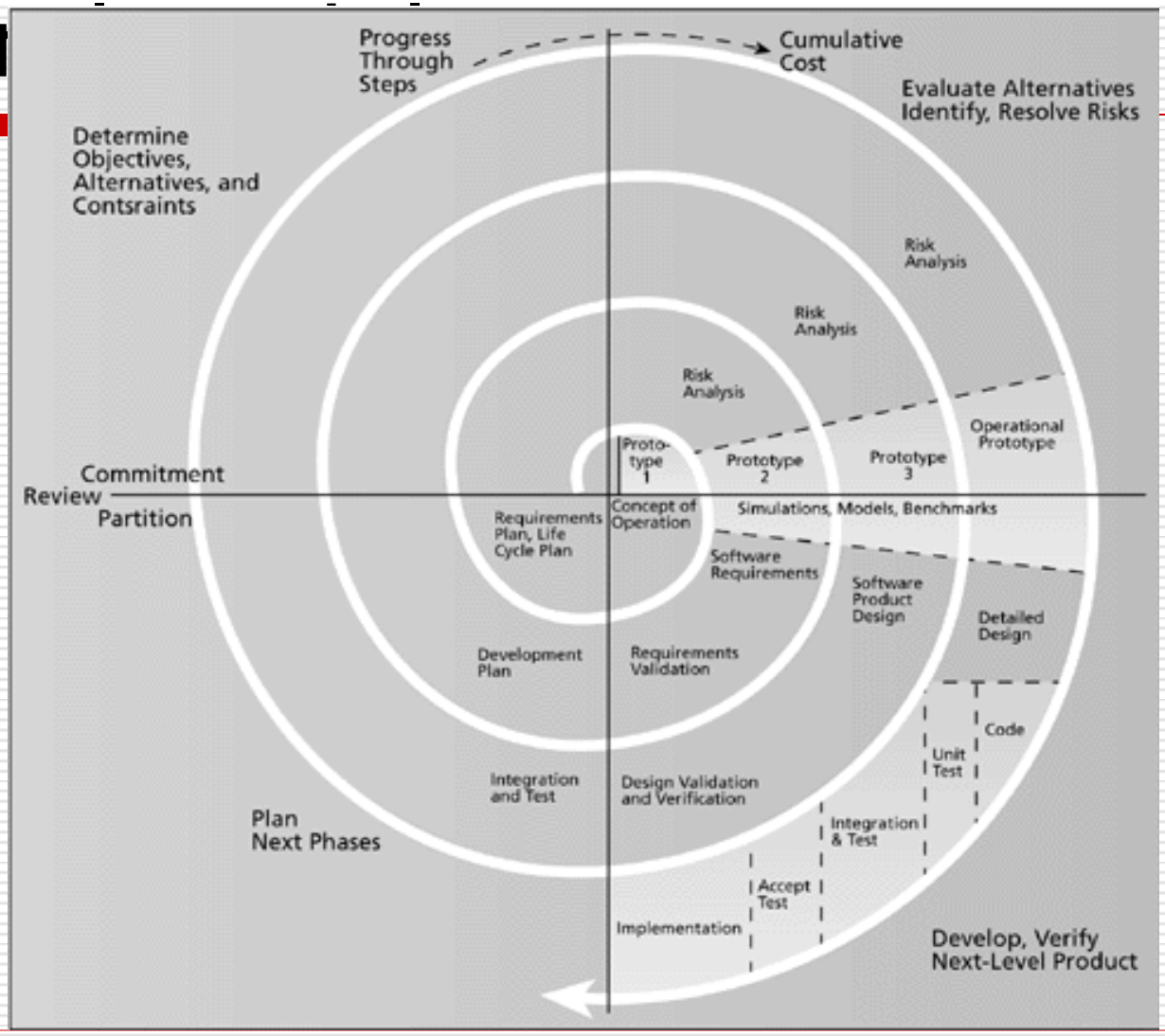
V Model


Maintenance
Requirements Analysis
User Acceptance Testing
System Design
System Testing
Program Design
Unit and Integration Testing
Coding

V Model (cont'd)

- Additional validation process introduced
- Relate testing to analysis and design
- Loop back in case of discrepancy

Spi


Progress Through Steps
Cumulative Cost
Evaluate Alternatives Identify, Resolve Risks
Determine Objectives, Alternatives, and Contsraints
Risk Analysis
Risk Analysis
Risk Analysis
Operational Prototype
Proto-type 1
Prototype 2
Prototype 3
Commitment Review Partition
Concept of Operation
Simulations, Models, Benchmarks
Requirements Plan, Life Cycle Plan
Software Requirements
Software Product Design
Detailed Design
Development Plan
Requirements Validation
Code
Unit Test
Integration and Test
Design Validation and Verification
Integration & Test
Accept Test
Implementation
Plan Next Phases
Develop, Verify Next-Level Product

Spiral Model (cont'd)

- Evolutionary approach
- Iterative development combined with risk management
- Risk analysis results in "go, no-go" decision

Spiral Model (cont'd)

- Four major activities
  - Planning
  - Risk analysis
  - Engineering
  - Customer evaluation

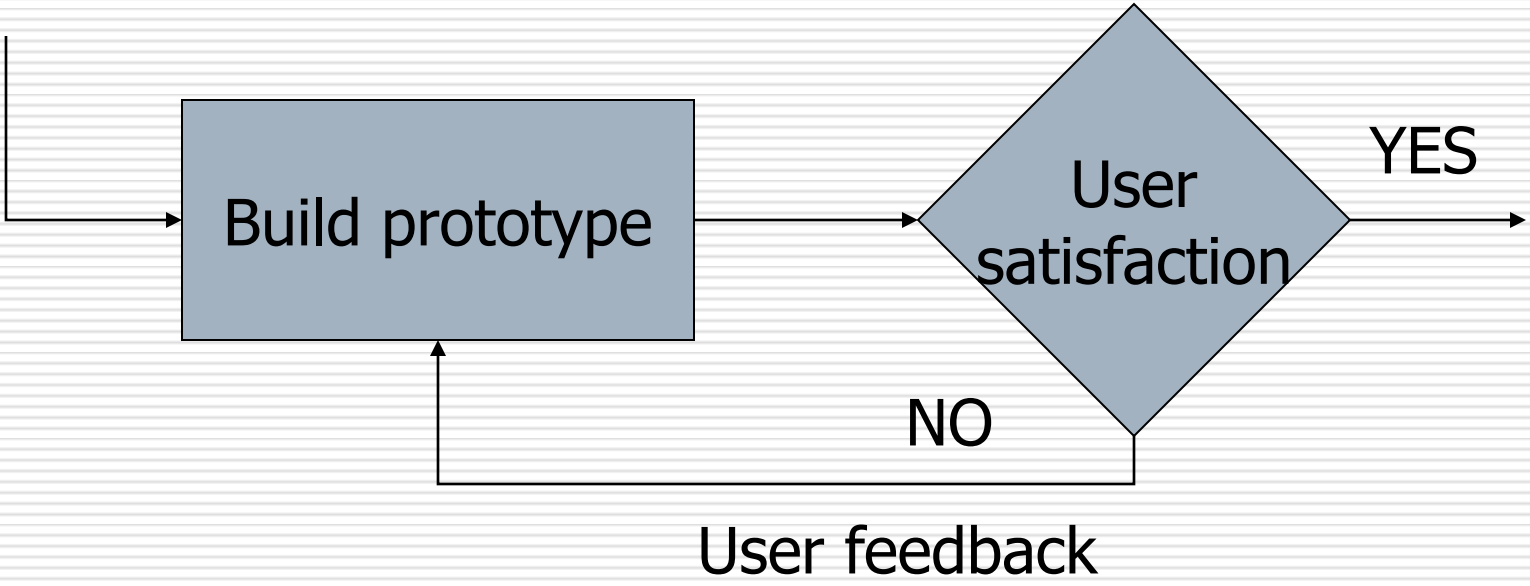
Prototyping Model

- Goals
  - meet users' requirements in early stage
  - reduce risk and uncertainty

Classification of Prototype

- Throw-away
  - After users agree the requirements of the system, the prototype will be discarded.
- Evolutionary
  - Modifications are based on the existing prototype.
- Incremental
  - Functions will be arranged and built accordingly.

Prototyping Model



Benefits of Prototyping

- Learning by doing
- Improved communication
- Improved user involvement
- Clarification of partially-known requirements

Prototyping Sequences

- Requirements gathering
- Quick design
- Prototype construction
- Customer evaluation
- Refinement
- Loop back to quick design for fine tuning
- Product engineering

Benefits of Prototyping

- Demonstration of the consistency and completeness of a specification
- Reduced need for documentation
- Reduced maintenance costs
- Feature constraint
- Production of expected results

Drawbacks of Prototyping

- Users sometimes misunderstand the role of the prototype
- Lack of project standards possible
- Lack of control
- Additional expense
- Close proximity of developers

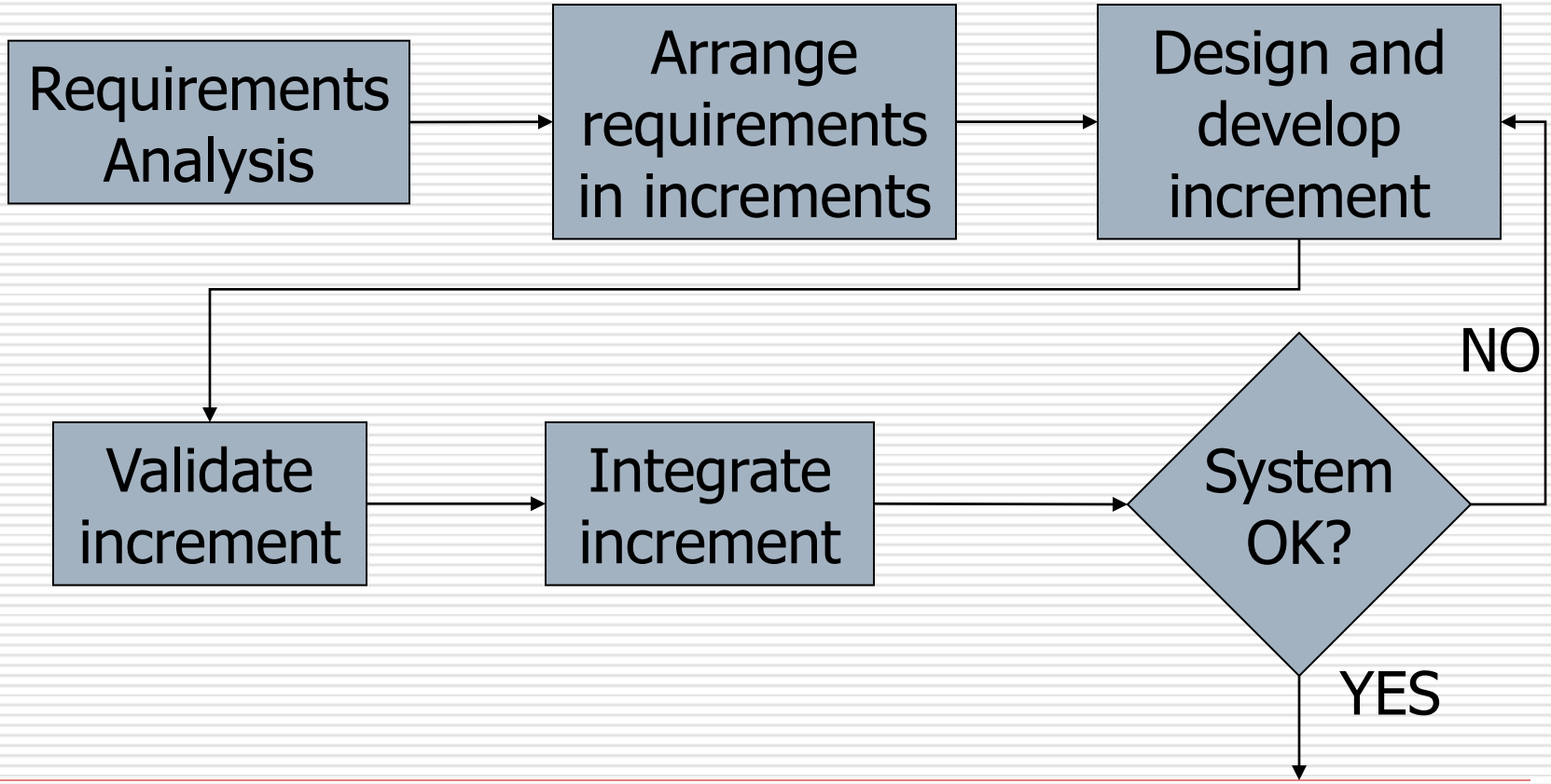
Forms of Prototypes

- Mock-ups
- Simulated interaction
- Partial working model

Incremental Model

- Break system into small components
- Implement and deliver small components in sequence
- Every delivered component provides extra functionality to user

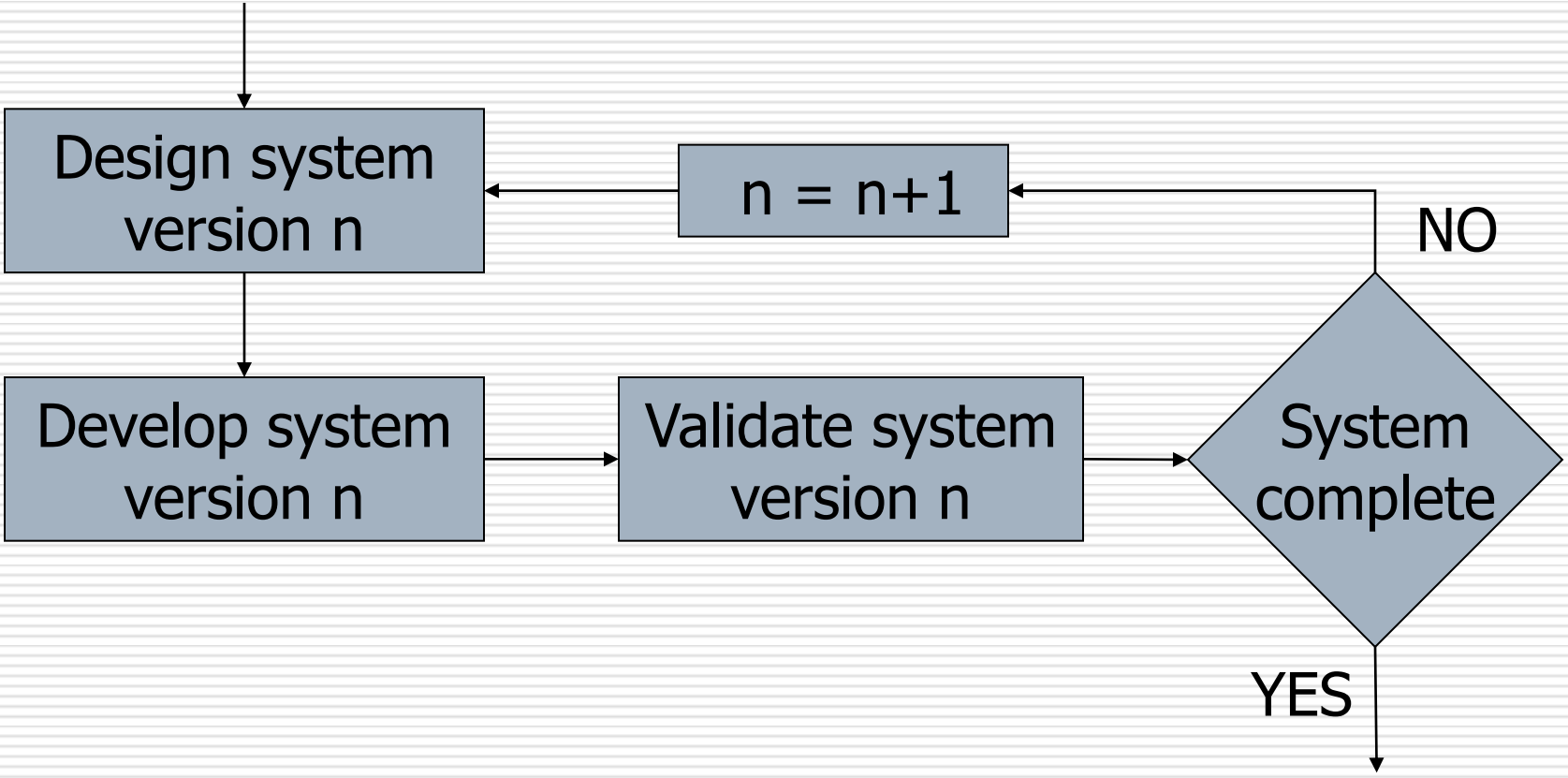
Incremental Model (cont'd)



Iterative Model

- Deliver full system in the beginning
- Enhance functionality in new releases

Iterative Model (cont'd)



Why Have Project Phases and Management Reviews?

- A project should successfully pass through each of the project phases in order to continue on to the next
- Management reviews (also called phase exits or kill points) should occur after each phase to evaluate the project's progress, likely success, and continued compatibility with organizational goals

Distinguishing Project Life Cycles and Product Life Cycles

- The project life cycle applies to all projects, regardless of the products being produced
- Product life cycle models vary considerably based on the nature of the product
- Most large IT products are developed as a series of projects
- Project management is done in all of the product life cycle phases

Capability Maturity Model

- The CMM is a process model based on software best-practices effective in large-scale, multi-person projects.

The CMM has been used to assess the maturity levels of organization areas as diverse as software engineering, system engineering, project management, risk management, system acquisition, information technology (IT) or personnel management, against a scale of five key processes, namely:

Initial,
Repeatable,
Defined,
Managed and
Optimized.

Level 1 - Initial

- At maturity level 1, processes are usually ad hoc, and the organization usually does not provide a stable environment. Success in these organizations depends on the competence and heroics of the people in the organization, and not on the use of proven processes

Level 2 - Repeatable

- At maturity level 2, software development successes are repeatable. The processes may not repeat for all the projects in the organization. The organization may use some basic project management to track cost and schedule.

Level 3 - Defined

- The organization’s set of standard processes, which are the basis for level 3, are established and improved over time. These standard processes are used to establish consistency across the organization. Projects establish their defined processes by applying the organization’s set of standard processes, tailored, if necessary, within similarly standardized guidelines.

Level 4 - Quantitatively Managed

- Using precise measurements, management can effectively control the software development effort. In particular, management can identify ways to adjust and adapt the process to particular projects without measurable losses of quality or deviations from specifications

Level 5 - Optimizing

- Maturity level 5 focuses on continually improving process performance through both incremental and innovative technological improvements. Quantitative process-improvement objectives for the organization are established, continually revised to reflect changing business objectives, and used as criteria in managing process improvement.

International Organization for Standardization(ISO)

12 Engineering Activities

- Systems requirement analysis
- System architectural design
- Software requirement analysis
- Software architectural design
- Software detailed design
- Software coding and testing
- Software Integration
- Software qualification testing
- System integration
- System qualification testing
- Software installation
- Software acceptance test

Unit II

Domain Processes

Scope Planning and the Scope Statement

- A scope statement is a document used to develop and confirm a common understanding of the project scope. It should include
  - a project justification
  - a brief description of the project's products
  - a summary of all project deliverables
  - a statement of what determines project success

Scope Planning and the Work Breakdown Structure

- After completing scope planning, the next step is to further define the work by breaking it into manageable pieces
- Good scope definition
  - helps improve the accuracy of time, cost, and resource estimates
  - defines a baseline for performance measurement and project control
  - aids in communicating clear work responsibilities

The Work Breakdown Structure

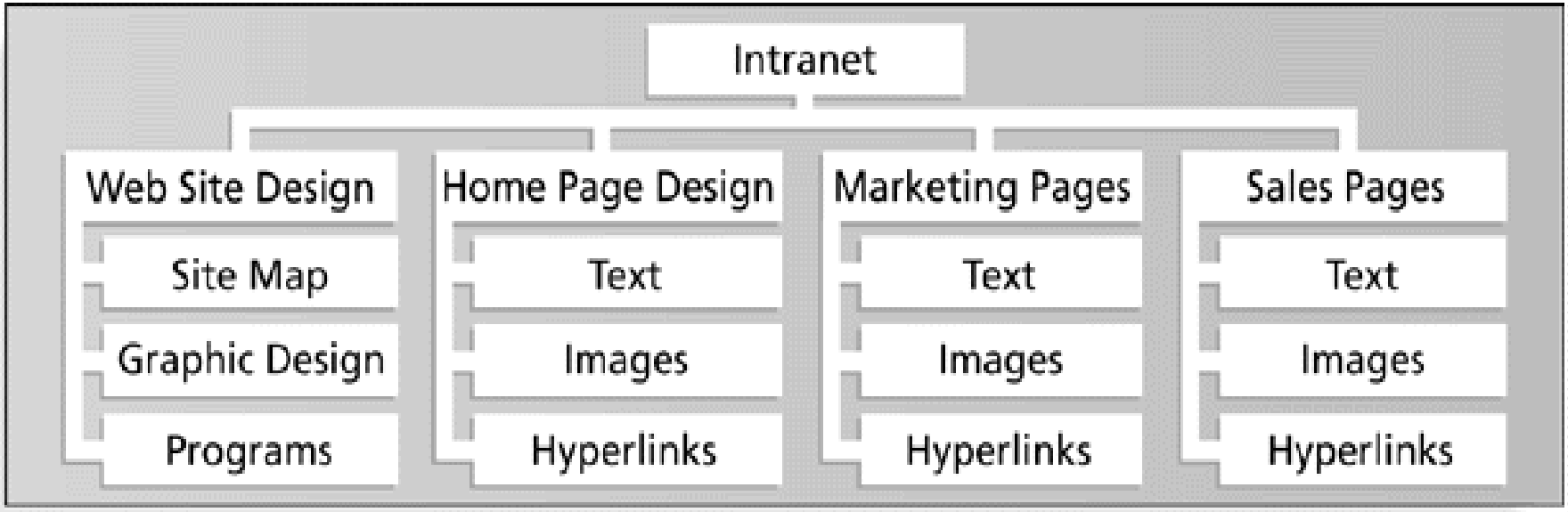
- A work breakdown structure (WBS) is an outcome-oriented analysis of the work involved in a project that defines the total scope of the project
- It is a foundation document in project management because it provides the basis for planning and managing project schedules, costs, and changes

WBS

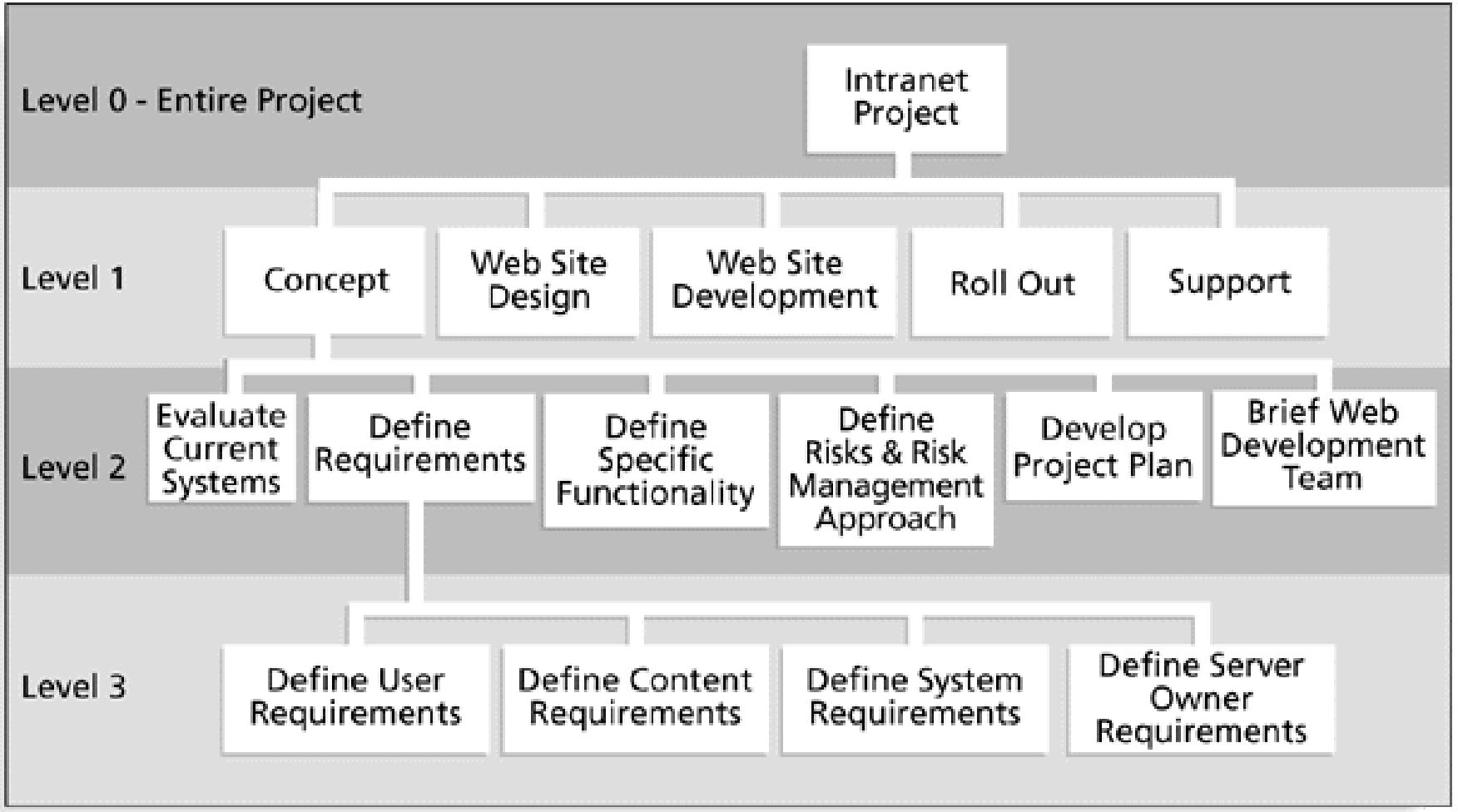
- WBS was initially developed by the U.S. defense establishment, and it is described in Military Standard (MIL-STD) 881B (25 Mar 93) as follows:

- "A work breakdown structure is a product-oriented family tree composed of hardware, software, services, data and facilities .... [it] displays and defines the product(s) to be developed and/or produced and relates the elements of work to be accomplished to each other and to the end product(s)."

Sample Intranet WBS Organized by Product



Sample Intranet WBS Organized by Phase



Intranet WBS in Tabular Form

- 1.0 Concept
  - 1.1 Evaluate current systems
  - 1.2 Define Requirements
    - 1.2.1 Define user requirements
    - 1.2.2 Define content requirements
    - 1.2.3 Define system requirements
    - 1.2.4 Define server owner requirements
  - 1.3 Define specific functionality
  - 1.4 Define risks and risk management approach
  - 1.5 Develop project plan
  - 1.6 Brief web development team
- 2.0 Web Site Design
- 3.0 Web Site Development
- 4.0 Roll Out
- 5.0 Support

Approaches to Developing WBSs

- Using guidelines: Some organizations, like the DOD, provide guidelines for preparing WBSs
- The analogy approach: It often helps to review WBSs of similar projects
- The top-down approach: Start with the largest items of the project and keep breaking them down
- The bottoms-up approach: Start with the detailed tasks and roll them up

Basic Principles for Creating WBSs*

1. A unit of work should appear at only one place in the WBS.
2. The work content of a WBS item is the sum of the WBS items below it.
3. A WBS item is the responsibility of only one individual, even though many people may be working on it.
4. The WBS must be consistent with the way in which work is actually going to be performed; it should serve the project team first and other purposes only if practical.
5. Project team members should be involved in developing the WBS to ensure consistency and buy-in.
6. Each WBS item must be documented to ensure accurate understanding of the scope of work included and not included in that item.
7. The WBS must be a flexible tool to accommodate inevitable changes while properly maintaining control of the work content in the project according to the scope statement.

*Cleland, David I., *Project Management: Strategic Design and Implementation*, 1994

Scope Verification and Scope Change Control

- It is very difficult to create a good scope statement and WBS for a project
- It is even more difficult to verify project scope and minimize scope changes
- Many IT projects suffer from scope creep and poor scope verification

Factors Causing IT Project Problems*

Factor	Rank
Lack of user input	1
Incomplete requirements and specifications	2
Changing requirements and specifications	3
Lack of executive support	4
Technology incompetence	5
Lack of resources	6
Unrealistic expectations	7
Unclear objectives	8
Unrealistic time frames	9
New Technology	10

*Johnson, Jim, "CHAOS: The Dollar Drain of IT Project Failures," Application Development Trends, January 1995, www.stadishgroup.com/chaos.html

Suggestions for Improving User Input

- Insist that all projects have a sponsor from the user organization
- Have users on the project team
- Have regular meetings
- Deliver something to project users and sponsor on a regular basis
- Co-locate users with the developers

Suggestions for Reducing Incomplete and Changing Requirements

- Develop and follow a requirements management process
- Employ techniques such as prototyping, use case modeling, and Joint Application Design to thoroughly understand user requirements
- Put all requirements in writing and current
- Create a requirements management database
- Provide adequate testing
- Use a process for reviewing requested changes from a systems perspective
- Emphasize completion dates

Unit III

Software Development

Software Size and Reuse Estimating

Software Size and Reuse Estimating

”Predicting the size of a software system becomes progressively easier as the project advances”

”At no other time are the estimates so important than at the beginning of a project”

Product Development Life Cycle

- Estimating size and effort will occur many times during the life cycle
- After requirements, after analysis, after design, and so on...

Learning Objectives

- Explain why the sizing of software is an important estimating technique
- Describe how Work Breakdown Structure can be used to estimate software size
- List, explain and describe several models used to estimate size
- Summarize the advantages and disadvantages of the models
- Explain the impact of reused software components upon the size estimate

Problems with Estimating

- Problem is not well understood
- Little or no historical data
- No standards
- Management uses estimates as performance goals
- Developers are optimistic
- Customer demands quick estimates
- etc...

Risks of Estimating

- If incomplete or incorrect estimate:
  - disappointing the customer
  - possibly losing future business
  - too optimistic fixed-price contract result in the contractor losing money and losing face

How to tackle size-related risks

- Produce a WBS, decomposed to the lowest level possible → smaller is easier to estimate
- Review assumptions with all stakeholders
- Research past organizational experiences and historical data
- Stay in close communication with other developers, common language
- Update estimates
- Use many size estimating methods
- Educate staff in estimation methods

Getting Started

- Sizing is the prediction of product deliverables needed to fulfill requirements
- Estimation is the prediction of effort needed to produce the deliverables
- WBS is a description of the work to be performed, broken down into key elements
- WBS is our TOC, a hierarchical list of the work activities required to complete a project
- Choose a method and stick with it
- Compare actual data to planned estimates
- Everything should be counted, any observable physical piece of software

Size measures

- Lines of Code (LOC)
- Function Points
- Feature Points
- Object Points
- Model Blitz
- Wideband Delphi

Lines of Code

- Very difficult to know how many LOC will be produced before they are written or even designed
- LOC measure has become infamous
- Still the most used metric
- Average programmer productivity rate remains despite of new languages
- Functionality and quality are the real concerns, not the LOC

Lines of Code 2

- WBS should be decomposed to the lowest level possible
- Estimating using expert opinions, asking experts who have developed similar systems
- Estimating using Bottom-Up summing, asking developers to estimate the size of each decomposed level
- Ask for an optimistic (200), pessimistic (400) and realistic size (250) estimate → (200+400+(4*250)) / 6 = 266 LOC
- What exactly is a line of code? Standards and rules needed

Lines of Code 3

- Translate the number of LOC to assembly language line in order to make comparisons between programming languages
- e.g.
  convert 50,000 LOC system written in C to Java
  Assembler level for C = 2.5, Java = 6
  50,000 * 2.5 = 125,000 if written in assembler
  125,000 / 6 = 20,833 LOC if written in Java

Advantages of LOC

- Widely used and accepted
- Allows for comparison between diverse development groups
- Directly relates to the end product
- Easily measured upon project completion
- Measure from the developer’s point of view
- Continuous improvement

Disadvantages of LOC

- Difficult to estimate early in the life cycle
- Source instructions variate with language, design, programmer etc.
- No industry standars for counting LOC
- Fixed costs are not included with coding
- Programmers may be rewarded for large LOC counts
- Distinguish between generated code and hand-crafted code
- Can not be used for normalizing if languages are different
- Only existing products and expert opinions can be used to predict a LOC count

Function Points

- Idea:
  software is better measured in terms of the number and complexity of the functions that it performs
- Function points measure categories of end-user business functions

Function Point Process Steps

1. Count number of functions in each category (outputs, inputs, interfaces etc..)
2. Apply complexity weighting factors (simple, medium, complex)
3. Apply environmental factors
4. Calculate complexity adjustment factor
5. Compute adjusted function points
6. Convert to Lines of Code (optional)

Advantages of Function Point Analysis

- Can be applied early in the life cycle
- Independent of language and technology
- Provide a reliable relationship to effort
- Can be used as a productivity goal
- Users understand better
- Provide a mechanism to track and monitor scope
- Environmental factors are considered

Disadvantages of Function Point Analysis

- Requires subjective evaluations
- Results depend on technology used to implement the analysis
- Many effort and cost models depend on LOC, must be converted
- Best after the creation of a design specification
- Not good to non-MIS applications

Feature Points

- Extension of the function point method
- Designed to deal with different kinds of applications, like embedded or real-time systems
- A feature point is a new category of function point that represent complex algorithms

Areas where Feature Points are used …

- RTS such as missile defense systems
- System software
- Embedded Systems like radar navigation packages
- CAD & CAM

Feature Point Process Steps

1. Count number of feature points in each category
2. Count feature points (algorithms)
3. Weigh Complexity (avg for feature points, simple/avg/complex for algorithms)
4. Evaluate environmental factors
5. Calculate complexity adjustment factor
6. Adjust feature points
7. Convert to LOC (optional)

Advantages of Feature Point Analysis

- Same as in function point analysis
- Additional advantage for algorithmically intensive systems

Disadvantages of Feature Point Analysis

- Primary disadvantage:
  - subjective classification of algorithmic complexity

Object Points

- Method developed for object-oriented technology
- Counting ”object points” to determine software size
- Conducted at a more macro level than function points
- Assigns one object point to each unique class or object
- Otherwise similar to function and feature points

Model Blitz

- Estimating gets better with each passing phase
- Concept of blitz modelling is counting number of process (object classes) * number of programs per class * avg program size = Estimated size (LOC)
- A historical database is essential
- Function-strong systems and data-strong systems are calculated separately

Contd…

- 20 object classes implemented to 5 procedural programs & on an avg 75 LOC per procedural prg
- No. of Process X no. of programs per class X Avg Prg Size = Estimated Size
- 20 X 5 X 75 = ?
- 7500 LOC

Advantages of Model Blitz

- Easy to use with structured methods and with object-oriented classes
- Accuracy increases with historical data
- Continuous improvement activities used for estimation techniques

Disadvantages of Model Blitz

- Requires use of design methodology
- Estimation can not begin until design is complete
- Requires historical data
- Does not evaluate environmental factors

Wideband Delphi

- Popular and simple technique to estimate size and effort
- Group consensus approach
- Uses experience of several people to reach an estimate

Wideband Delphi steps

1. Present experts with the problem and a response form
2. Group discussion
3. Collect opinions anonymously
4. Feed back a summary of results
5. Another group discussion
6. Iterate until consensus

Advantages of Wideband Delphi

- Easy and inexpensive
- Expertise of several people
- Participants become better educated about the software and project
- Does not require historical data
- Used for high-level and detailed estimation
- Results more accurate than in LOC

Disadvantages of Wideband Delphi

- Difficult to repeat with different group of experts
- Possible to reach consensus on an incorrect estimate, people may not be skeptical enough
- Can develop a false sense of confidence
- May fail to reach a consensus
- Experts may be biased in the same subjective direction

Effects of Reuse on Software Size

- Many softwares are derived from previous programs
- Result in savings of cost and time, increased quality
- Can also cost more, take longer time and yield lower quality
- First step in code reuse is to separate new code from modified and reused code

Effects of Reuse continues

- If the unit has changed, it is modified
- If more than 50% of the unit is changed, it is considered to be ”new”
- Reused code will be converted to equivalent new code
- Conversion factor reflects the amount of effort saved by reuse
- Reuse factors come from experience (e.g. 30% for reused, 60% for modified)
- Can also be done on more accurate level

Unit IV

Scheduling Activities

Project management resource activities.

- Inference management is the process of managing all the different interfaces with other people and groups related to the product and the project, it includes identifying, documenting, scheduling, communicating, and monitoring these interfaces throughout the course of project.

Kinds of interfaces:

- Personal
- Organizational
- System

- PERSONAL

  It deals with all conflicts involving people in or related to the project.

- ORGANIZATIONAL

  It deals with conflicts due to varying organizational goals and the different management styles of the
  project and resource supplying organizations.

- SYSTEM

  It deals with the product, facility, or other non people interfaces developed by the project.

Organizational structure:

- "An organization is a system that exchanges materials, personnel, manpower, and energy with the environment“

TYPES OF ORGANIZATIONS:

- Functional organizations
- Matrix organizations
- Projectized organizations
- Combination of all the above

FUNCTIONAL ORGANIZATIONS

- The functional organization is a standard old-fashioned organization. It is the style in which people are divided into their functional specialties and report to a functional area manager.

MATRIX ORGANIZATIONS

- In the matrix organizations there is a balance of power established between the functional and project managers.
  · The project worker in a matrix organization has a multiple command system of accountability and responsibility.

PROJECTIZED ORGANISATIONS:

- In projectized organizations the project manger has total authority and acts like a mini-CEO.
  · All personnel assigned to the project report to project manager.

Software development dependencies

- "Dependencies are any relationship connections between activities in a project that may impact their scheduling".

- External vs Internal Dependencies
- Resource vs Activity Dependencies

DEPENDENCY RELATIONSHIPS:

- Finish-to-start(FS)
  Start-to-start(SS)
  Finish-to-finish(FF)
  Start-to-finish(SF)

Scheduling fundamentals:

- The three most common forms of presenting a project schedule are:
  Ø Table.
  Ø Gantt chart.
  Ø Network diagram.

Critical Path Method (CPM)

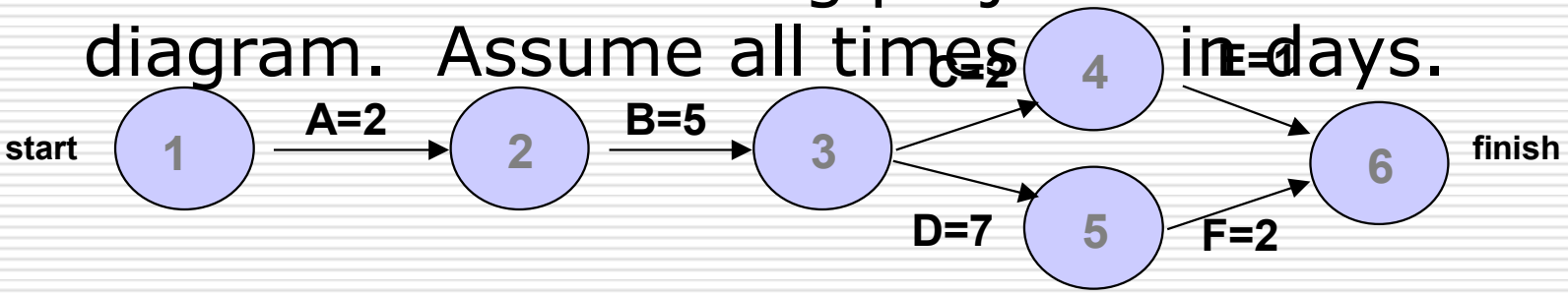
- CPM is a project network analysis technique used to predict total project duration
- A critical path for a project is the series of activities that determines the *earliest time* by which the project can be completed
- The critical path is the *longest path* through the network diagram and has the least amount of slack or float

Finding the Critical Path

- First develop a good project network diagram
- Add the durations for all activities on each path through the project network diagram
- The longest path is the critical path

Simple Example of Determining the Critical Path

- Consider the following project network diagram. Assume all times in days.



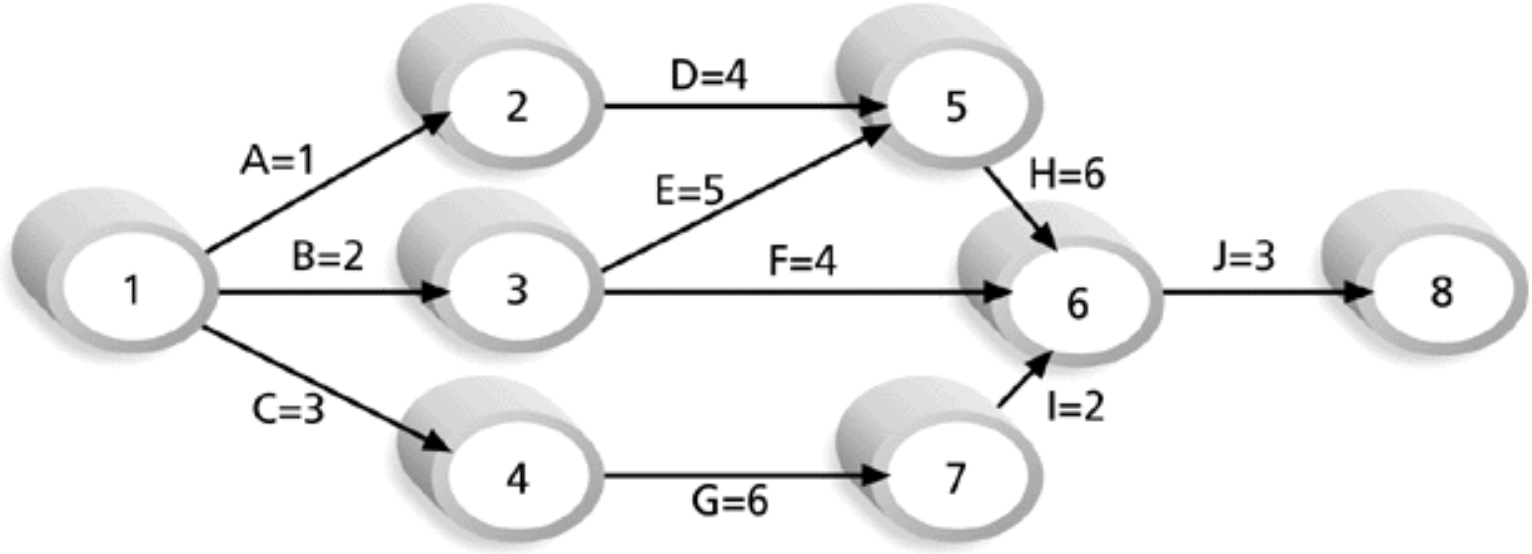
a. How many paths are on this network diagram?

b. How long is each path?

c. Which is the critical path?

d. What is the shortest amount of time needed to complete this project?

Determining the Critical Path for Project X



Note: Assume all durations are in days.

Path 1:	A-D-H-J	Length = 1+4+6+3 = 14 days
Path 2:	**B-E-H-J**	**Length = 2+5+6+3 = 16 days**
Path 3:	B-F-J	Length = 2+4+3 = 9 days
Path 4:	C-G-I-J	Length = 3+6+2+3 = 14 days

Since the critical path is the longest path through the network diagram, Path 2, B-E-H-J, is the critical path for Project X.

More on the Critical Path

- If one of more activities on the critical path takes longer than planned, the whole project schedule will slip *unless* corrective action is taken
- Misconceptions:
  - The critical path is not the one with all the critical activities; it only accounts for time
  - There can be more than one critical path if the lengths of two or more paths are the same
  - The critical path can change as the project progresses

Using Critical Path Analysis to Make Schedule Trade-offs

- Knowing the critical path helps you make schedule trade-offs
- *Free slack or free float* is the amount of time an activity can be delayed without delaying the early start of any immediately following activities
- *Total slack or total float* is the amount of time an activity may be delayed from its early start without delaying the planned project finish date

Techniques for Shortening a Project Schedule

- Shortening durations of critical tasks for adding more resources or changing their scope
- *Crashing* tasks by obtaining the greatest amount of schedule compression for the least incremental cost
- *Fast tracking* tasks by doing them in parallel or overlapping them

Importance of Updating Critical Path Data

- It is important to update project schedule information
- The critical path may change as you enter actual start and finish dates
- If you know the project completion date will slip, negotiate with the project sponsor

Critical Chain Scheduling

- Technique that addresses the challenge of meeting or beating project finish dates and an application of the Theory of Constraints (TOC)
- Developed by Eliyahu Goldratt in his books *The Goal* and *Critical Chain*
- Critical chain scheduling is a method of scheduling that takes limited resources into account when creating a project schedule and includes buffers to protect the project completion date
- Critical chain scheduling assumes resources do not multitask because it often delays task completions and increases total durations

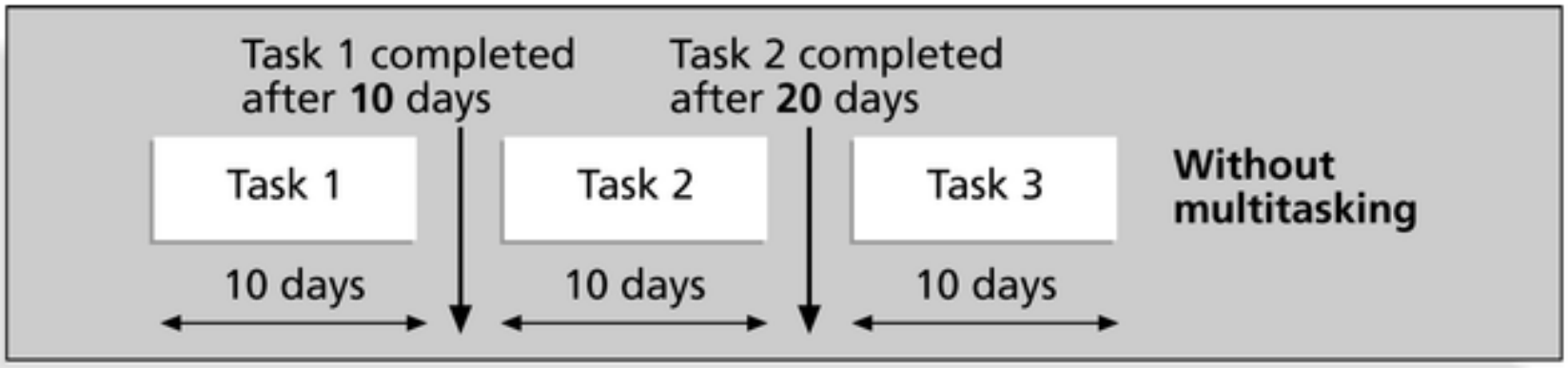


Figure 5-9a. Three Tasks Without Multitasking

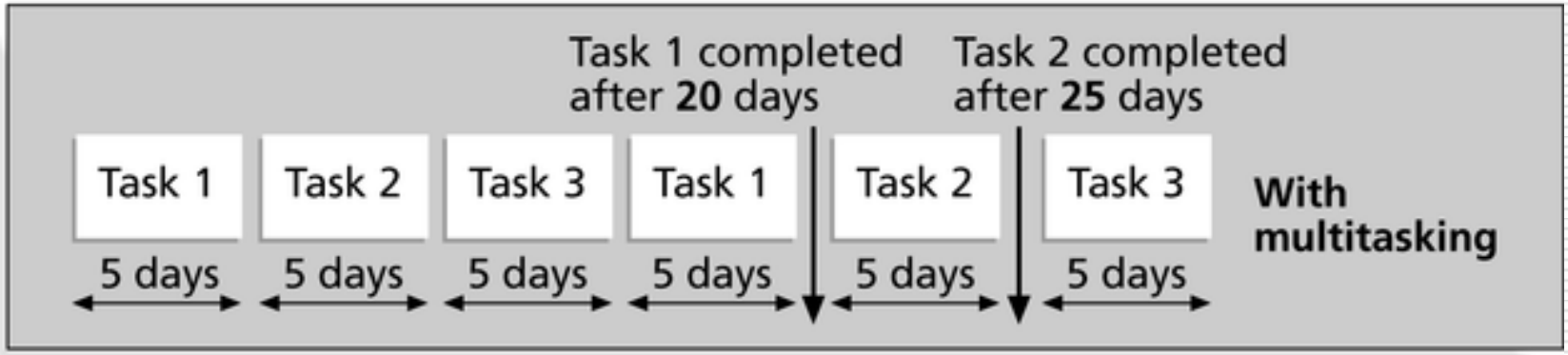


Figure 5-9b. Three Tasks with Multitasking

Buffers and Critical Chain

- A buffer is additional time to complete a task
- Murphy's Law states that if something can go wrong, it will, and Parkinson's Law states that work expands to fill the time allowed. In traditional estimates, people often add a buffer and use it if it's needed or not
- Critical chain schedule removes buffers from individual tasks and instead creates
  - A project buffer, which is additional time added before the project's due date
  - Feeding buffers, which are addition time added before tasks on the critical path

Program Evaluation and Review Technique (PERT)

- PERT is a network analysis technique used to estimate project duration when there is a high degree of uncertainty about the individual activity duration estimates
- PERT uses probabilistic time estimates based on using optimistic, most likely, and pessimistic estimates of activity durations

PERT Formula and Example

- PERT weighted average formula:

$$\frac{\text{optimistic time} + 4\text{X most likely time} + \text{pessimistic time}}{6}$$

- Example:

PERT weighted average =

$$\frac{8 \text{ workdays} + 4 \text{ X } 10 \text{ workdays} + 24 \text{ workdays}}{6} = 12 \text{ days}$$

where 8 = optimistic time, 10 = most likely time, and 24 = pessimistic time

Controlling Changes to the Project Schedule

- Perform reality checks on schedules
- Allow for contingencies
- Don’t plan for everyone to work at 100% capacity all the time
- Hold progress meetings with stakeholders and be clear and honest in communicating schedule issues

Working with People Issues

- Strong leadership helps projects succeed more than good PERT charts
- Project managers should use
  - empowerment
  - incentives
  - discipline
  - negotiation

Using Software to Assist in Time Management

- Software for facilitating communications helps people exchange schedule-related information
- Decision support models help analyze trade-offs that can be made
- Project management software can help in various time management areas

Words of Caution on Using Project Management Software

- Many people misuse project management software because they don’t understand important concepts and have not had good training
- You must enter dependencies to have dates adjust automatically and to determine the critical path
- You must enter actual schedule information to compare planned and actual progress

Unit V

Quality Assurance

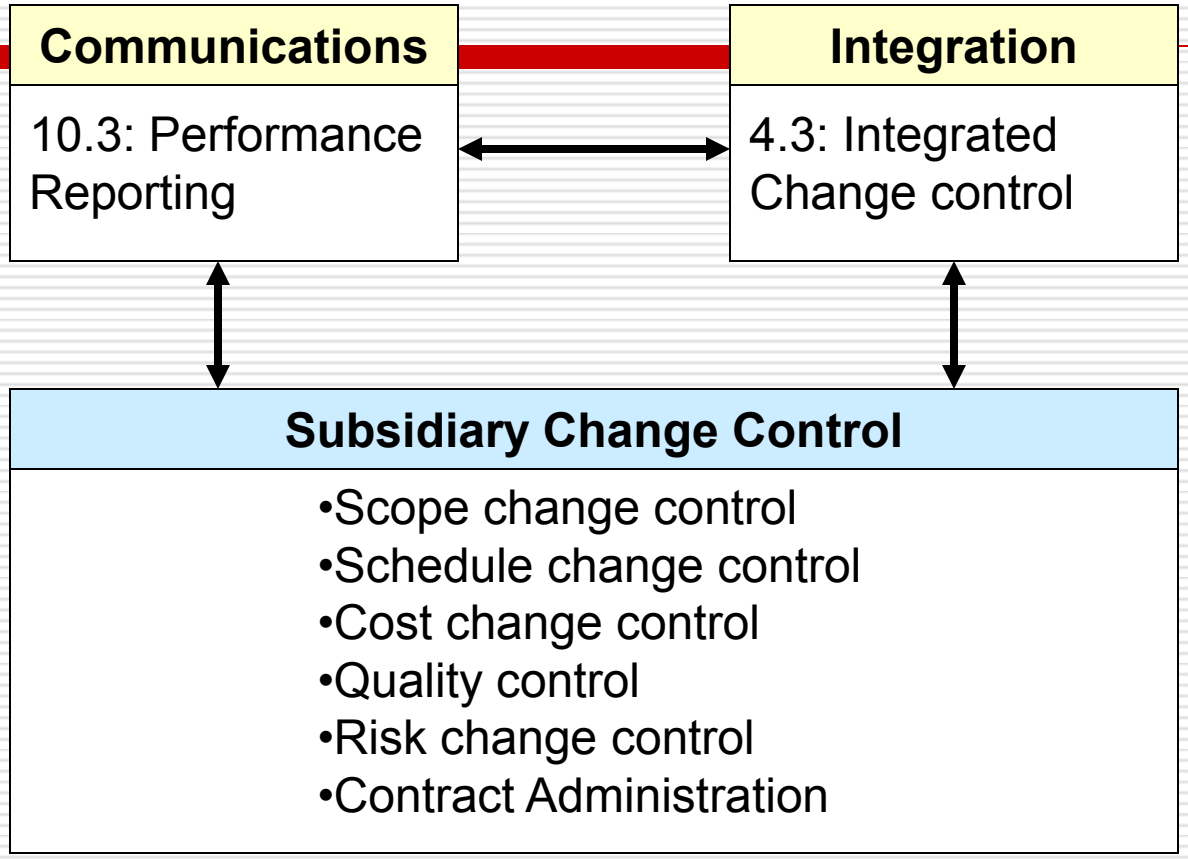
Integrated Change Control

- Integrated change control involves identifying, evaluating, and managing changes throughout the project life cycle (Note: 1996 PMBOK called this process "overall change control")
- Three main objectives of change control:
  - Influence the factors that create changes to ensure they are agreed upon
  - Determine that a change has occurred
  - Manage actual changes when and as

Integrated Change Control

- Integrated change control requires
  - Maintaining the integrity of the performance measurement baseline
  - Ensuring that changes to the product scope are reflected in the definition of the project scope
  - Coordinating changes across the knowledge areas

Integrated Change Control -- Coordination

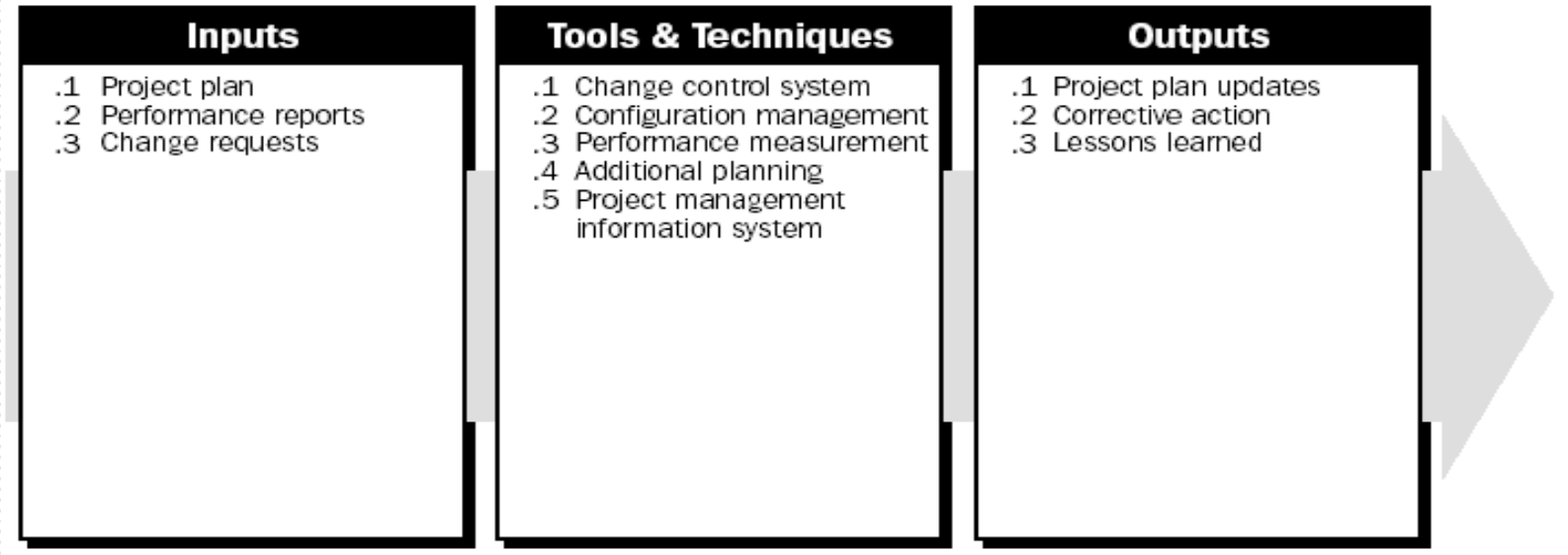


Coordinating changes Across the Entire Project

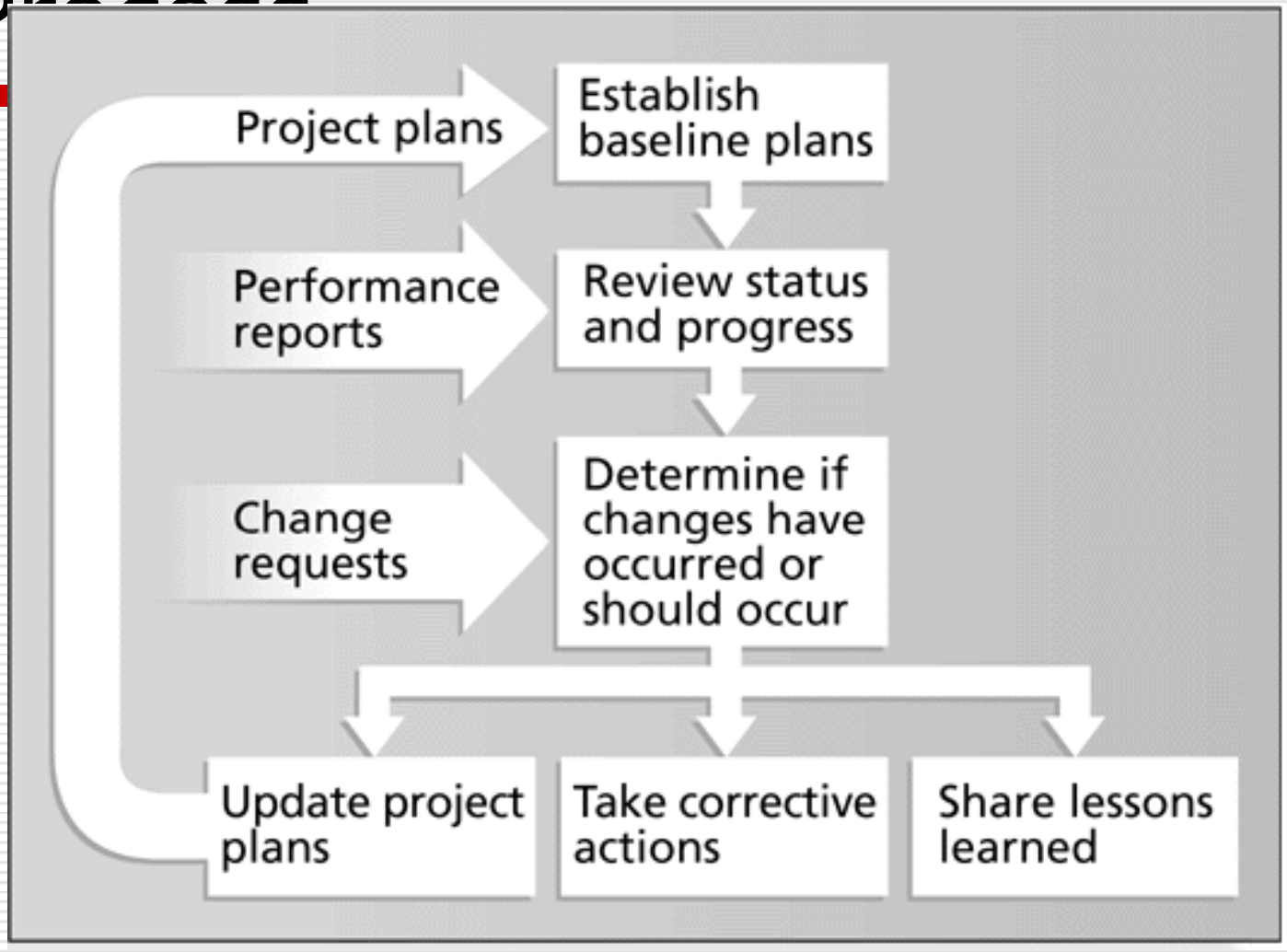
Change Control on Information Technology Projects

- Former view: The project team should strive to do exactly what was planned on time and within budget
- Problem: Stakeholders rarely agreed up-front on the project scope, and time and cost estimates were inaccurate
- Modern view: Project management is a process of constant communication and negotiation
- Solution: Changes are often beneficial, and the project team should plan for them

Integrated Change Control -- Process



Integrated Change Control Process



Change Control System

- A formal, documented process that describes when and how official project documents and work may be changed
- Describes who is authorized to make changes and how to make them
- Often includes a change control board (CCB), configuration management, and a process for communicating changes

Change Control Boards (CCBs)

- A formal group of people responsible for approving or rejecting changes on a project
- Provides guidelines for preparing change requests, evaluates them, and manages the implementation of approved changes
- Includes stakeholders from the entire organization

Making Timely Changes

- Some CCBs only meet occasionally, so it may take too long for changes to occur
- Some organizations have policies in place for time-sensitive changes
  - "48 hour policy" allowed project team members to make decisions, then they had 48 hours reverse the decision pending senior management approval
  - Delegate changes to the lowest level possible, but keep everyone informed of

Configuration Management

- Ensures that the products and their descriptions are correct and complete
- Concentrates on the management of technology by identifying and controlling the functional and physical design characteristics of products
- Configuration management specialists identify and document configuration requirements, control changes, record and report changes, and audit the products to verify conformance to requirements

Suggestions for Managing Integrated Change Control

- View project management as a process of constant communications and negotiations
- Plan for change
- Establish a formal change control system, including a Change Control Board (CCB)
- Use good configuration management
- Define procedures for making timely decisions on smaller changes
- Use written and oral performance reports to help identify and manage change
- Use project management and other software to help manage and communicate changes