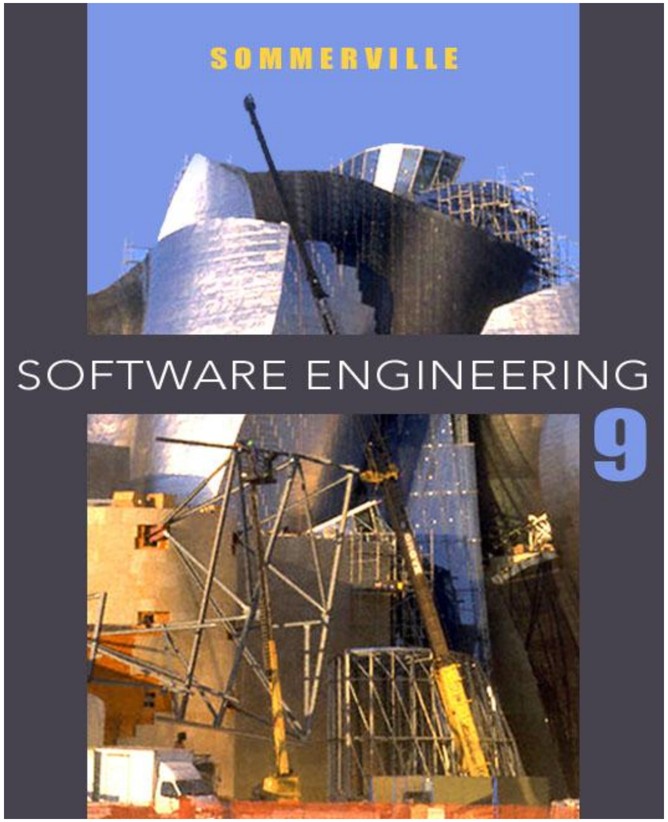
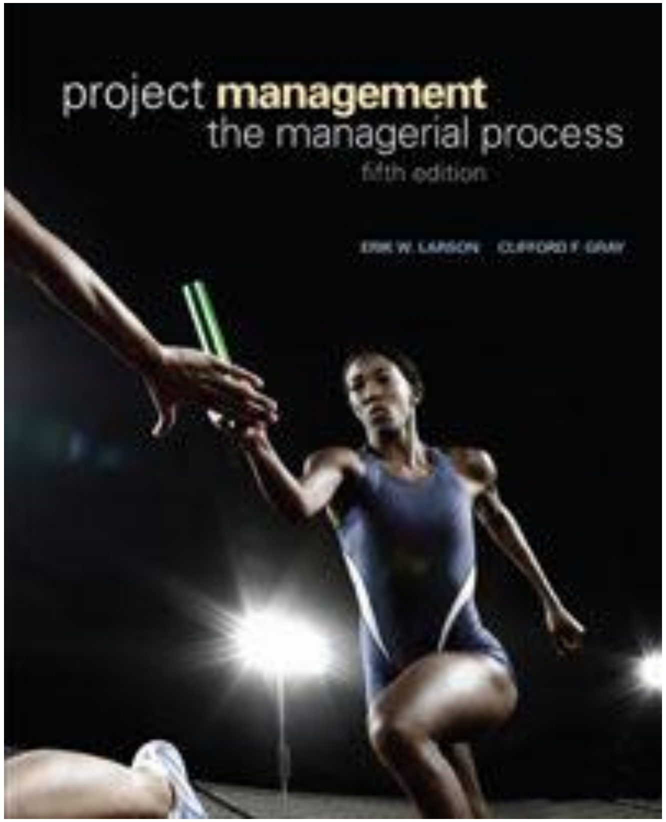
Research Experiences for Undergraduates (REU)

Software Project Planning and Management
Summer 2013—Dr. Straach

Reference Books





What are the attributes of good software?

- Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

What is software engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production.

Software engineering

- Engineering discipline
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
- All aspects of software production
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

What are the fundamental software engineering activities?

- Software specification
- Software development
- Software validation
- Software evolution

Software process activities

- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
- Software development, where the software is designed and programmed.
- Software validation, where the software is checked to ensure that it is what the customer requires.
- Software evolution, where the software is modified to reflect changing customer and market requirements.

What is the difference between software engineering and computer science?

- Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

What are the key challenges facing software engineering?

- Coping with increasing diversity and complexity, demands for reduced delivery times, continual change in requirements and developing trustworthy softwa

Key points

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.
- The high-level activities of specification, development, validation and evolution are part of all software processes.
- The fundamental notions of software engineering are universally applicable to all types of system development.

Importance of software engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

The software process

- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
  - Specification – defining what the system should do;
  - Design and implementation – defining the organization of the system and implementing the system;
  - Validation – checking that it does what the customer wants;
  - Evolution – changing the system in response to changing customer needs.

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

Software process descriptions

- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a data model, designing a user interface, etc. and the ordering of these activities.
- Process descriptions may also include:
  - Products, which are the outcomes of a process activity;
  - Roles, which reflect the responsibilities of the people involved in the process;
  - Pre- and post-conditions, which are statements that are true before and after a process activity has been enacted or a product produced.

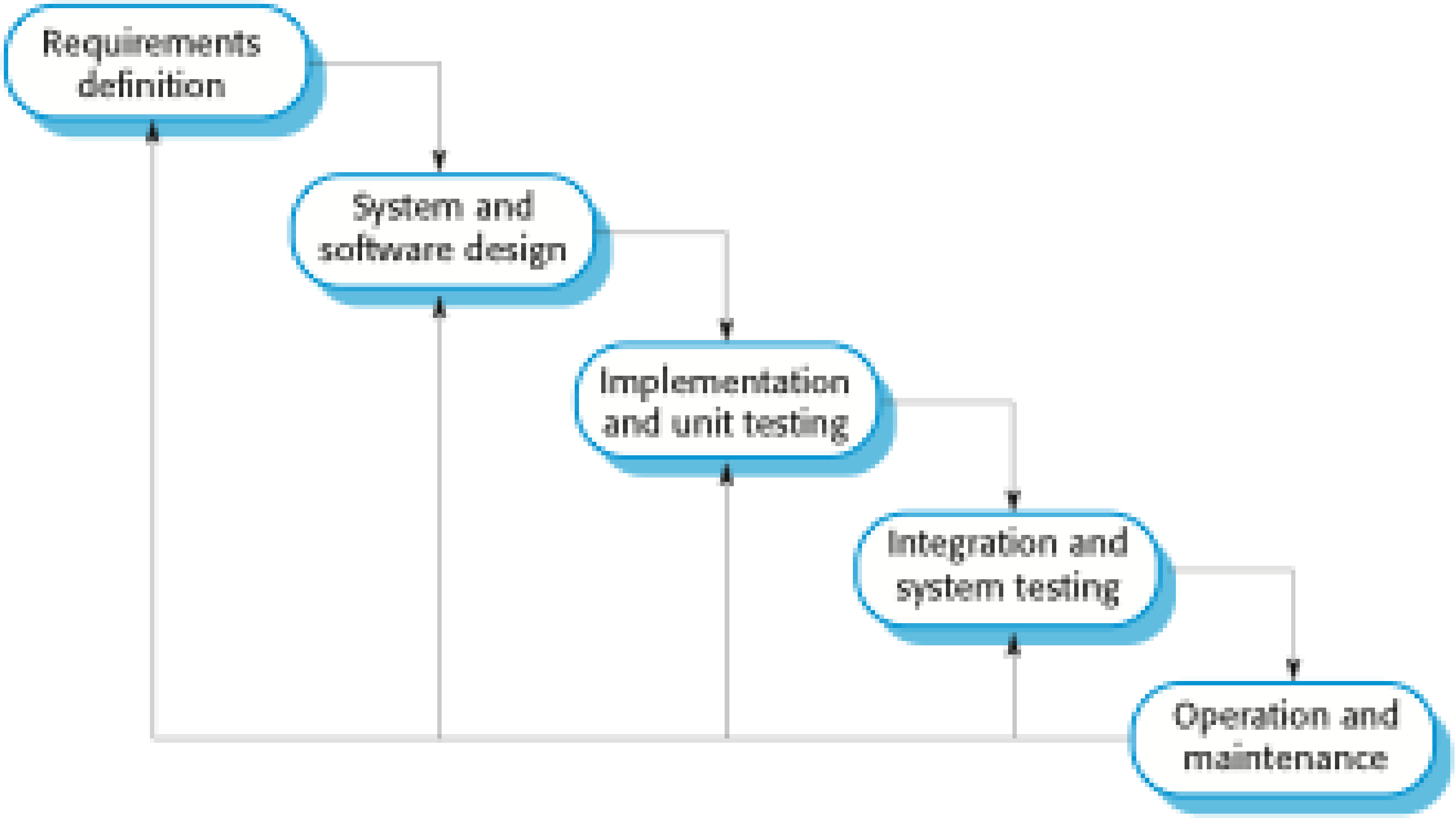
Plan-driven vs. agile processes

- Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.
- In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Software process models

- **The waterfall model**
  - Plan-driven model. Separate and distinct phases of specification and development.
- **Incremental development**
  - Specification, development and validation are interleaved. May be plan-driven or agile.
- **Reuse-oriented software engineering**
  - The system is assembled from existing components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.

The waterfall model


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

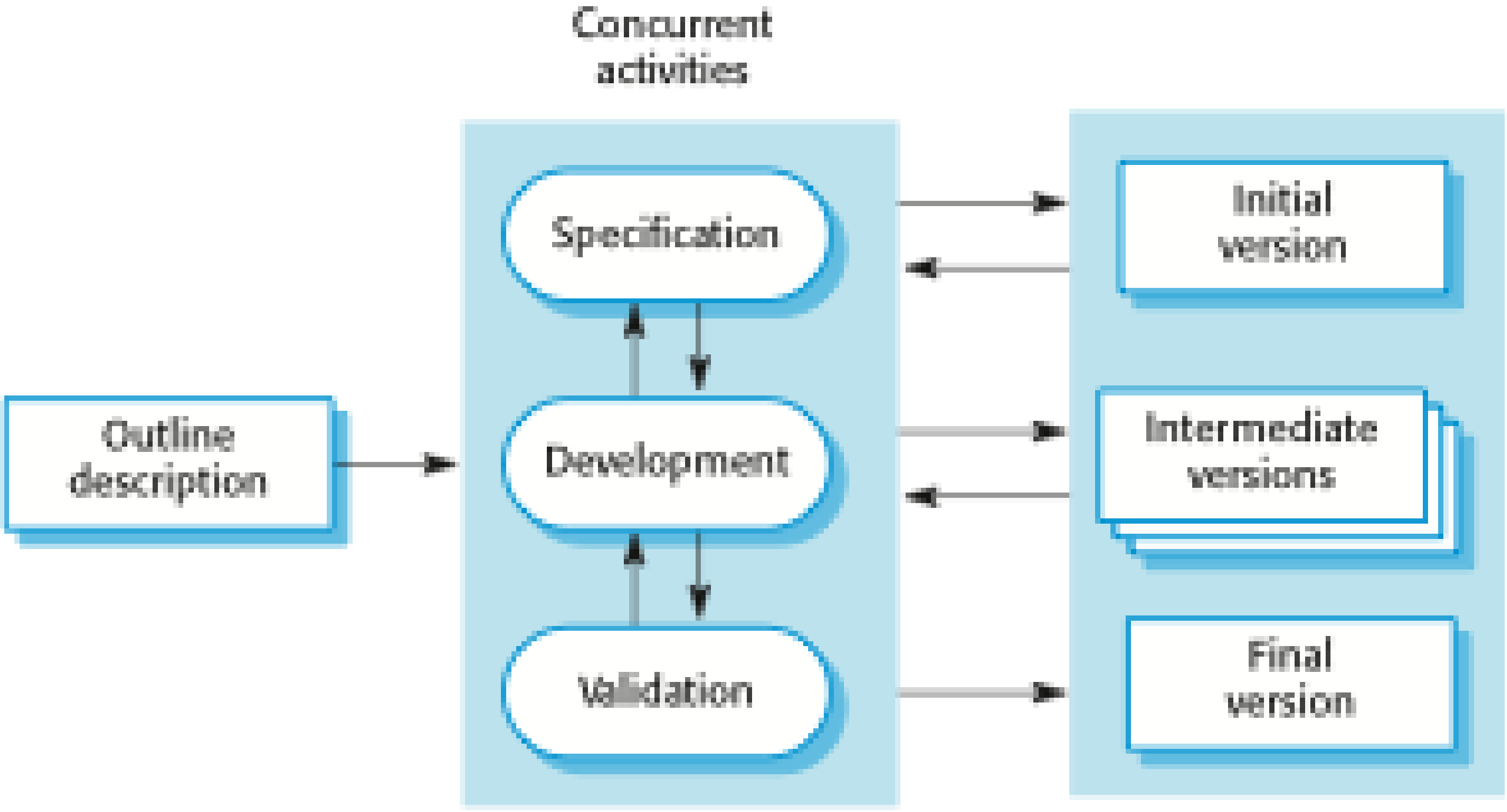
Waterfall model phases

- There are separate identified phases in the waterfall model:
  1) Requirements analysis and definition
  2) System and software design
  3) Implementation and unit testing
  4) Integration and system testing
  5) Operation and maintenance
- The main drawback of the waterfall model is the difficulty of accommodating change after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
  - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.

Incremental development



Incremental development benefits

- The cost of accommodating changing customer requirements is reduced.
    - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
    - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
    - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

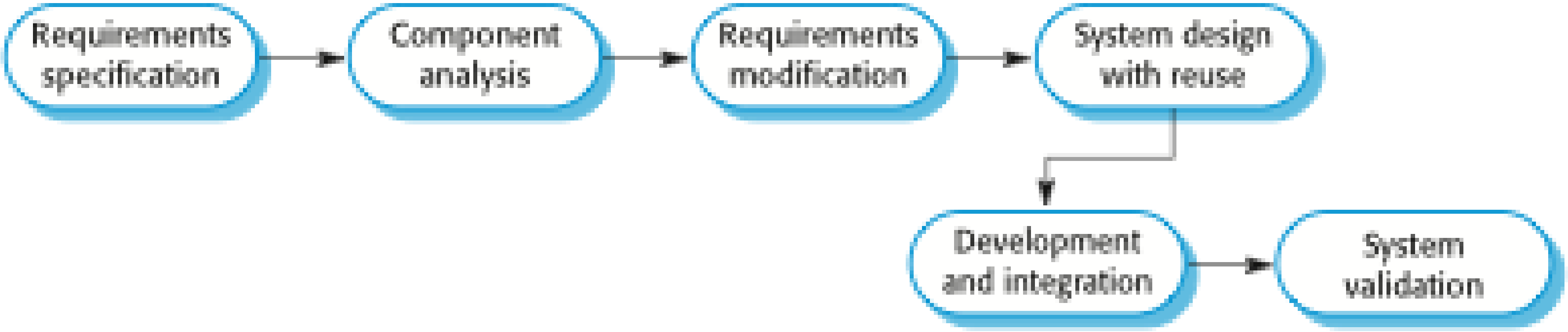
Incremental development problems

- The process is not visible.
  - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.

Reuse-oriented software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Component analysis;
  - Requirements modification;
  - System design with reuse;
  - Development and integration.
- Reuse is now the standard approach for building many types of business system
  - Reuse covered in more depth in Chapter 16.

Reuse-oriented software engineering


Requirements specification
Component analysis
Requirements modification
System design with reuse
Development and integration
System validation

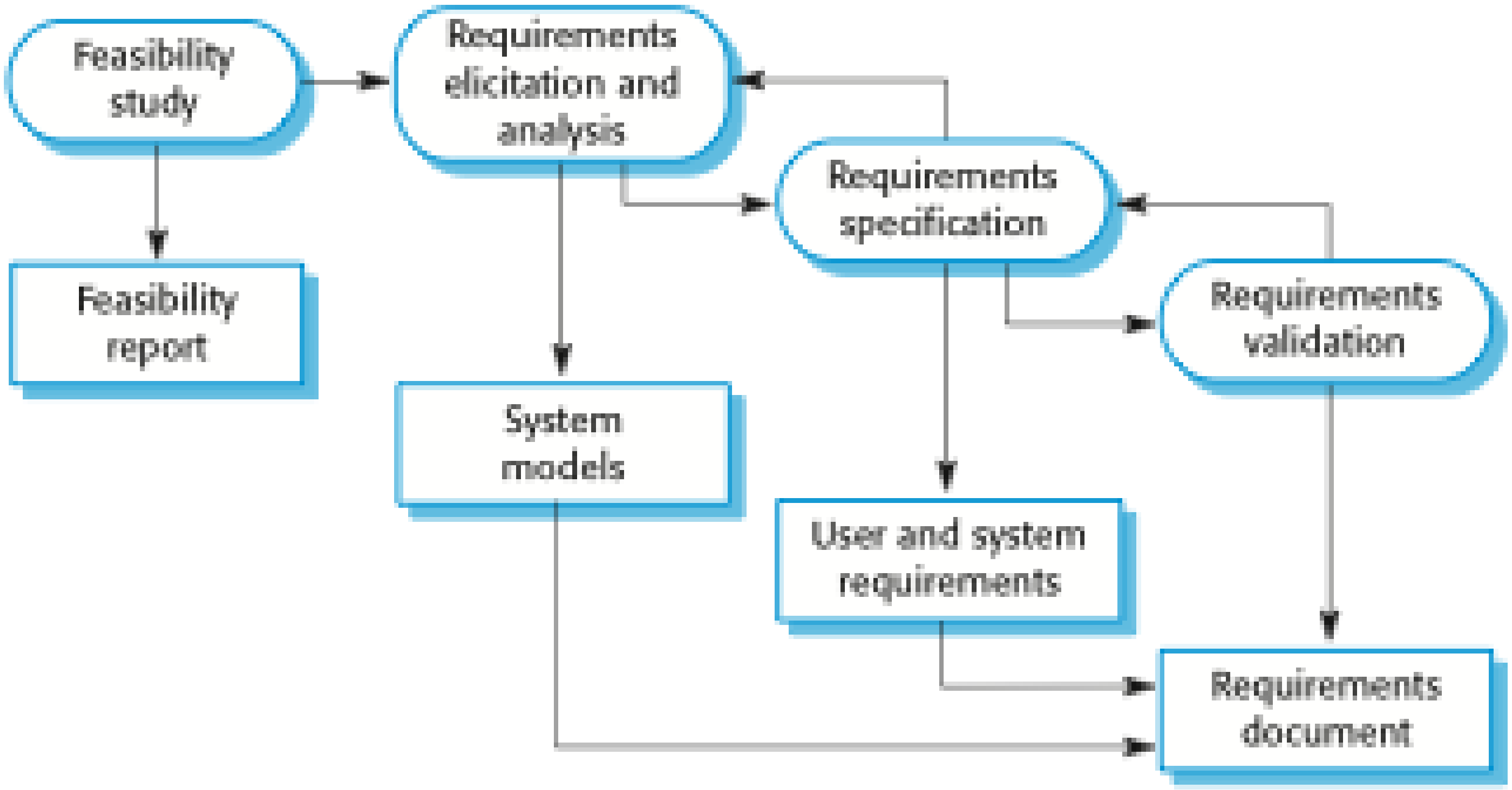
Process activities

- Real software processes are inter-leaved sequences of technical, collaborative and managerial activities with the overall goal of specifying, designing, implementing and testing a software system.
- The four basic process activities of **specification**, **development**, **validation** and **evolution** are organized differently in different development processes. In the waterfall model, they are organized in sequence, whereas in incremental development they are inter-leaved.

Software specification

- The process of establishing what services are required and the constraints on the system‘s operation and development.
- Requirements engineering process
  - Feasibility study
    - Is it technically and financially feasible to build the system?
  - Requirements elicitation and analysis
    - What do the system stakeholders require or expect from the system?
  - Requirements specification
    - Defining the requirements in detail
  - Requirements validation
    - Checking the validity of the requirements

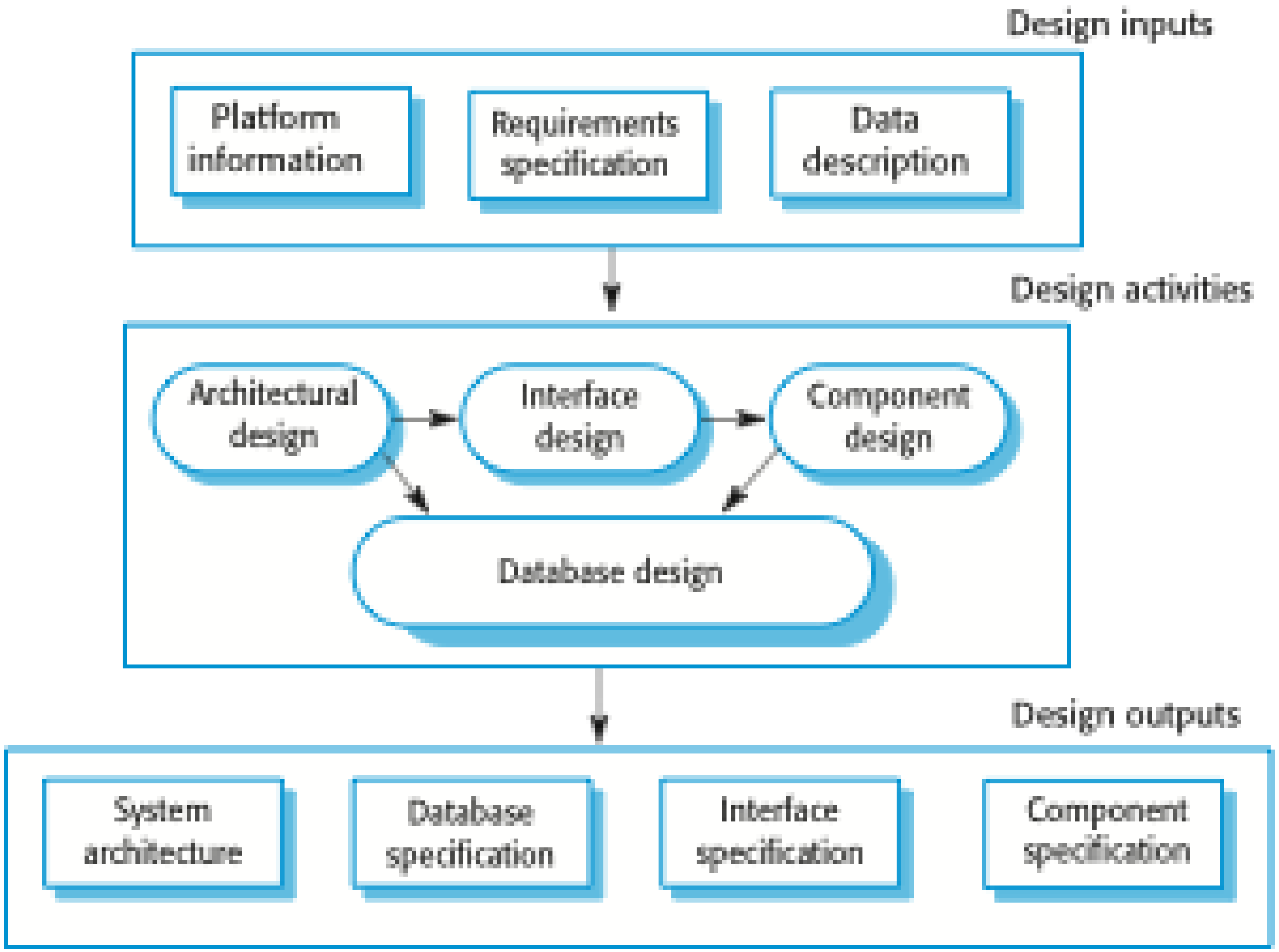
The requirements engineering process



Software design and implementation

- The process of converting the system specification into an executable system.
- Software design
  - Design a software structure that realises the specification;
- Implementation
  - Translate this structure into an executable program;
- The activities of design and implementation are closely related and may be inter-leaved.

A general model of the design process



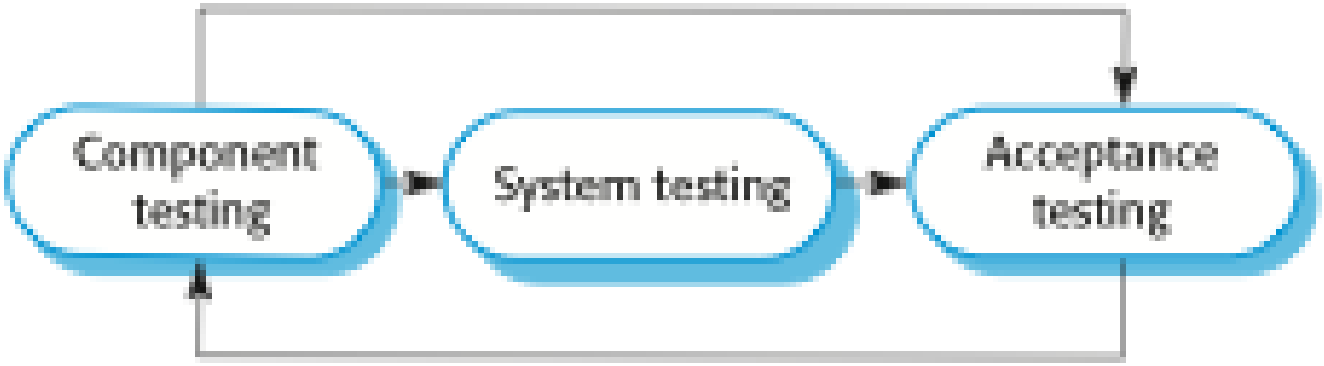
Design activities

- *Architectural design,* where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.
- *Interface design,* where you define the interfaces between system components.
- *Component design,* where you take each system component and design how it will operate.
- *Database design,* where you design the system data structures and how these are to be represented in a database.

Software validation

- Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.
- Involves checking and review processes and system testing.
- System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.
- Testing is the most commonly used V & V activity.

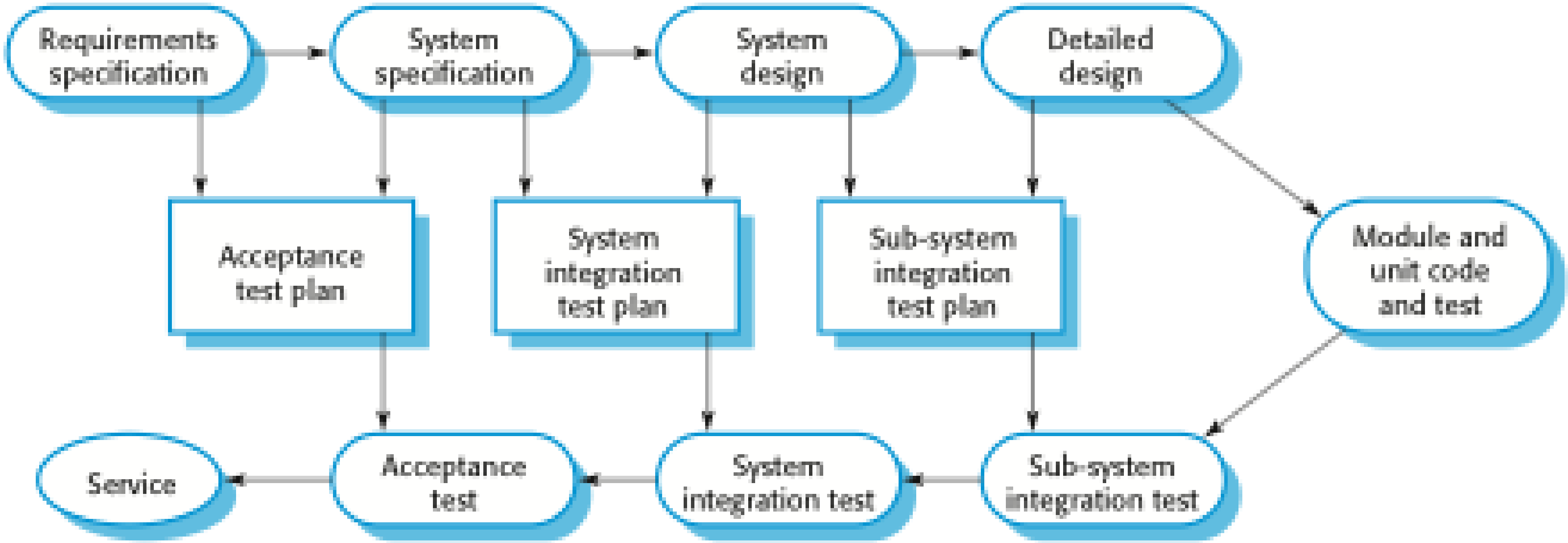
Stages of testing



Testing stages

- Development or component testing
  - Individual components are tested independently;
  - Components may be functions or objects or coherent groupings of these entities.
- System testing
  - Testing of the system as a whole. Testing of emergent properties is particularly important.
- Acceptance testing
  - Testing with customer data to check that the system meets the customer‘s needs.

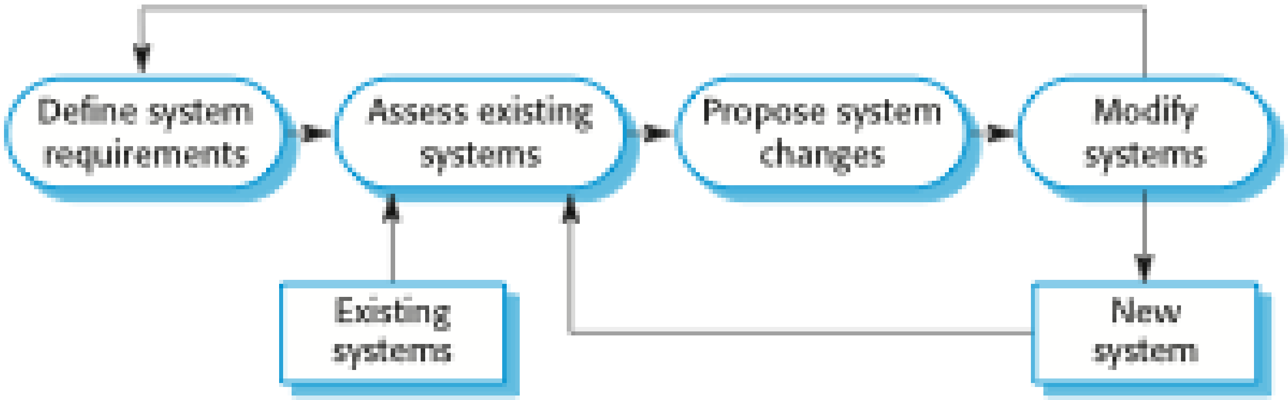
Testing phases in a plan-driven software process


Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and test
Service
Acceptance test
System integration test
Sub-system integration test

Software evolution

- Software is inherently flexible and can change.
- As requirements change through changing business circumstances, the software that supports the business must also evolve and change.
- Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

System evolution



Key points

- Software processes are the activities involved in producing a software system. Software process models are abstract representations of these processes.
- General process models describe the organization of software processes. Examples of these general models include the ‗watefall‘ model, incremental development, and reuse-oriented development.

37

Key points

- Requirements engineering is the process of developing a software specification.
- Design and implementation processes are concerned with transforming a requirements specification into an executable software system.
- Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.
- Software evolution takes place when you change existing software systems to meet new requirements. The software must evolve to remain useful.

Coping with change

- Change is inevitable in all large software projects.
  - Business changes lead to new and changed system requirements
  - New technologies open up new possibilities for improving implementations
  - Changing platforms require application changes
- Change leads to rework so the costs of change include both rework (e.g. re-analyzing requirements) as well as the costs of implementing new functionality

Software project management

- Concerned with activities involved in ensuring that software is delivered on time and on schedule and in accordance with the requirements of the organisations developing and procuring the software.
- Project management is needed because software development is always subject to budget and schedule constraints that are set by the organisation developing the software.

Success criteria

- Deliver the software to the customer at the agreed time.
- Keep overall costs within budget.
- Deliver software that meets the customer‘s expectations.
- Maintain a happy and well-functioning development team.

Software management distinctions

- The product is intangible.
  - Software cannot be seen or touched. Software project managers cannot see progress by simply looking at the artefact that is being constructed.
- Many software projects are 'one-off' projects.
  - Large software projects are usually different in some ways from previous projects. Even managers who have lots of previous experience may find it difficult to anticipate problems.
- Software processes are variable and organization specific.
  - We still cannot reliably predict when a particular software process is likely to lead to development problems.

Management activities

- *Project planning*
  - □ Project managers are responsible for planning. estimating and scheduling project development and assigning people to tasks.
- *Reporting*
  - □ Project managers are usually responsible for reporting on the progress of a project to customers and to the managers of the company developing the software.
- *Risk management*
  - □ Project managers assess the risks that may affect a project, monitor these risks and take action when problems arise.

Management activities

- *People management*
  - Project managers have to choose people for their team and establish ways of working that leads to effective team performance
- *Proposal writing*
  - The first stage in a software project may involve writing a proposal to win a contract to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out.

Project planning

- Project planning involves breaking down the work into parts and assign these to project team members, anticipate problems that might arise and prepare tentative solutions to those problems.
- The project plan, which is created at the start of a project, is used to communicate how the work will be done to the project team and customers, and to help assess progress on the project.

Planning stages

- At the proposal stage, when you are bidding for a contract to develop or provide a software system.
- During the project startup phase, when you have to plan who will work on the project, how the project will be broken down into increments, how resources will be allocated across your company, etc.
- Periodically throughout the project, when you modify your plan in the light of experience gained and information from monitoring the progress of the work.

The planning process

- Project planning is an iterative process that starts when you create an initial project plan during the project startup phase.
- Plan changes are inevitable.
  - As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule and risk changes.
  - Changing business goals also leads to changes in project plans. As business goals change, this could affect all projects, which may then have to be re-planned.

Project scheduling

- Project scheduling is the process of deciding how the work in a project will be organized as separate tasks, and when and how these tasks will be executed.
- You estimate the calendar time needed to complete each task, the effort required and who will work on the tasks that have been identified.
- You also have to estimate the resources needed to complete each task, such as the disk space required on a server, the time required on specialized hardware, such as a simulator, and what the travel budget will be.

Milestones and deliverables

- Milestones are points in the schedule against which you can assess progress, for example, the handover of the system for testing.
- Deliverables are work products that are delivered to the customer, e.g. a requirements document for the system.

Scheduling problems

- Estimating the difficulty of problems and hence the cost of developing a solution is hard.
- Productivity is not proportional to the number of people working on a task.
- Adding people to a late project makes it later because of communication overheads.
- The unexpected always happens. Always allow contingency in planning.

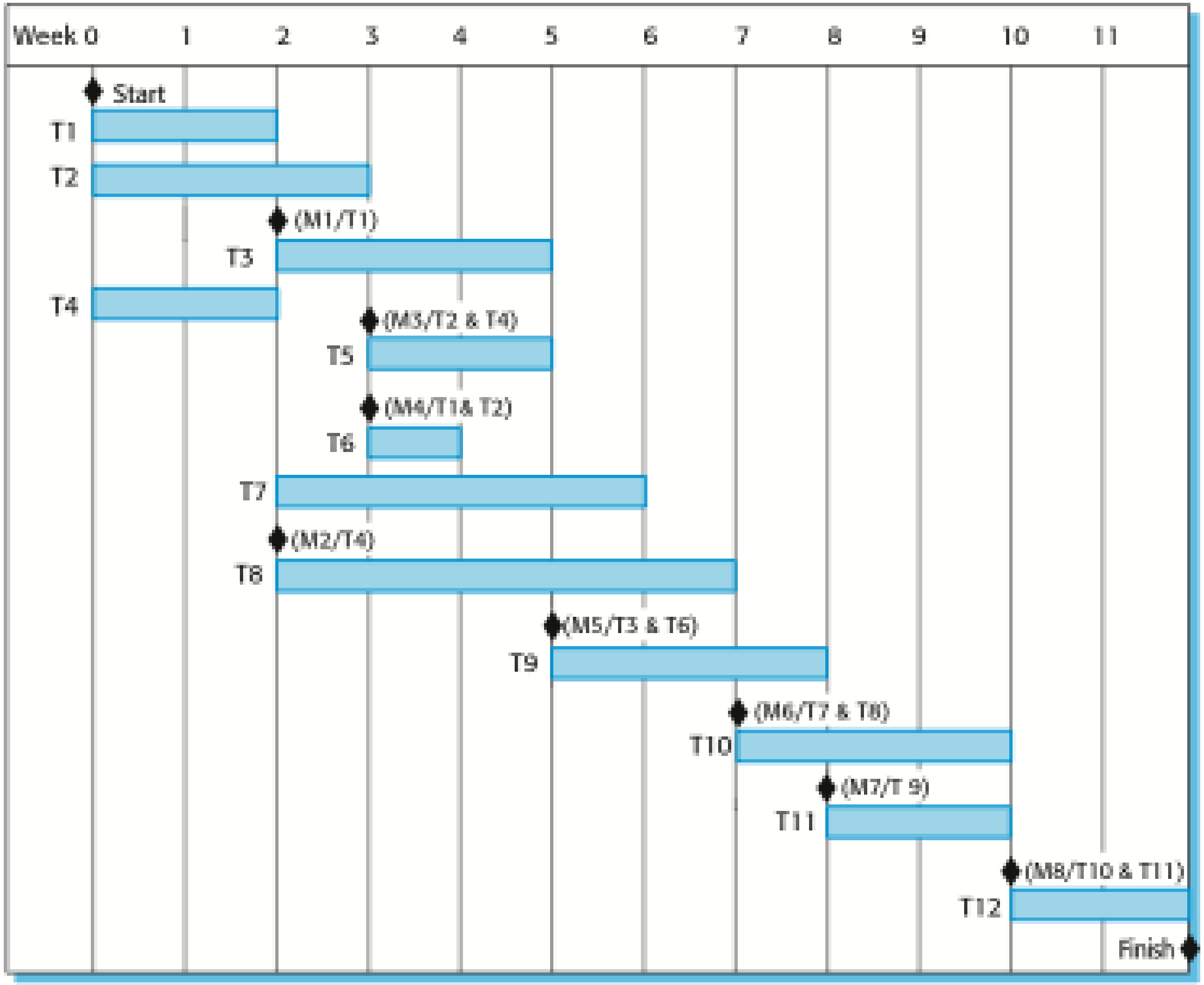
Schedule representation

- Graphical notations are normally used to illustrate the project schedule.
- These show the project breakdown into tasks. Tasks should not be too small. They should take about a week or two.
- Bar charts are the most commonly used representation for project schedules. They show the schedule as activities or resources against time.

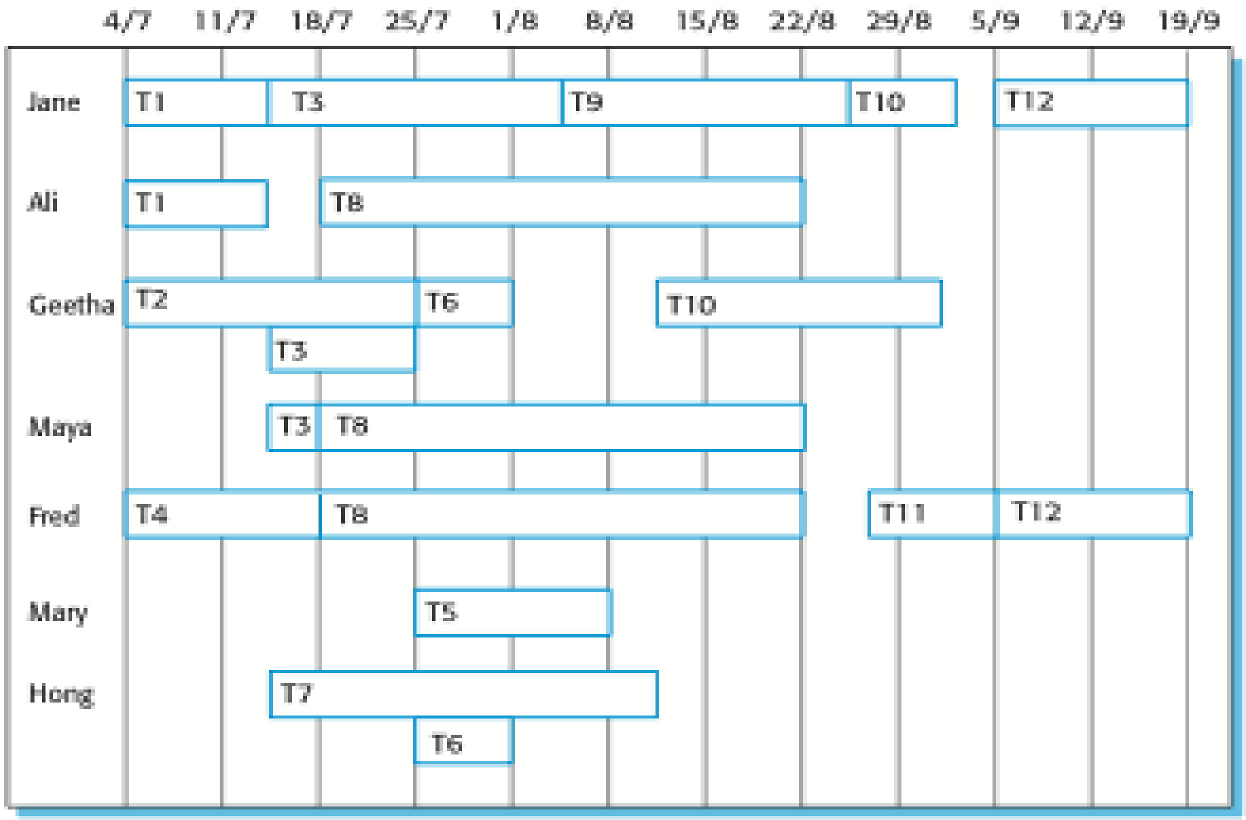
Tasks, durations, and dependencies

Task	Effort (person-days)	Duration (days)	Dependencies
T1	15	10	
T2	8	15	
T3	20	15	T1 (M1)
T4	5	10	
T5	5	10	T2, T4 (M3)
T6	10	5	T1, T2 (M4)
T7	25	20	T1 (M1)
T8	75	25	T4 (M2)
T9	10	15	T3, T6 (M5)
T10	20	15	T7, T8 (M6)
T11	10	10	T9 (M7)
T12	20	10	T10, T11 (M8)

Activity bar chart



Staff allocation chart



What Is a Project?

- Project
  - A complex, nonroutine, one-time effort limited by time, budget, resources, and performance specifications designed to meet customer needs.
- Major Characteristics of a Project
  - Has an established objective.
  - Has a defined life span with a beginning and an end.
  - Typically requires across-the-organizational participation.
  - Involves doing something never done before.
  - Has specific time, cost, and performance requirements.

Comparison of Routine Work with Projects

Routine, Repetitive Work	**Projects**
Taking class notes	Writing a term paper
Daily entering sales receipts into the accounting ledger	Setting up a sales kiosk for a professional accounting meeting
Responding to a supply-chain request	Developing a supply-chain information system
Practicing scales on the piano	Writing a new piano piece
Routine manufacture of an Apple iPod	Designing an iPod that is approximately 2 X 4 inches, interfaces with PC, and stores 10,000 songs
Attaching tags on a manufactured product	Wire-tag projects for GE and Wal-Mart

Value of Project Management

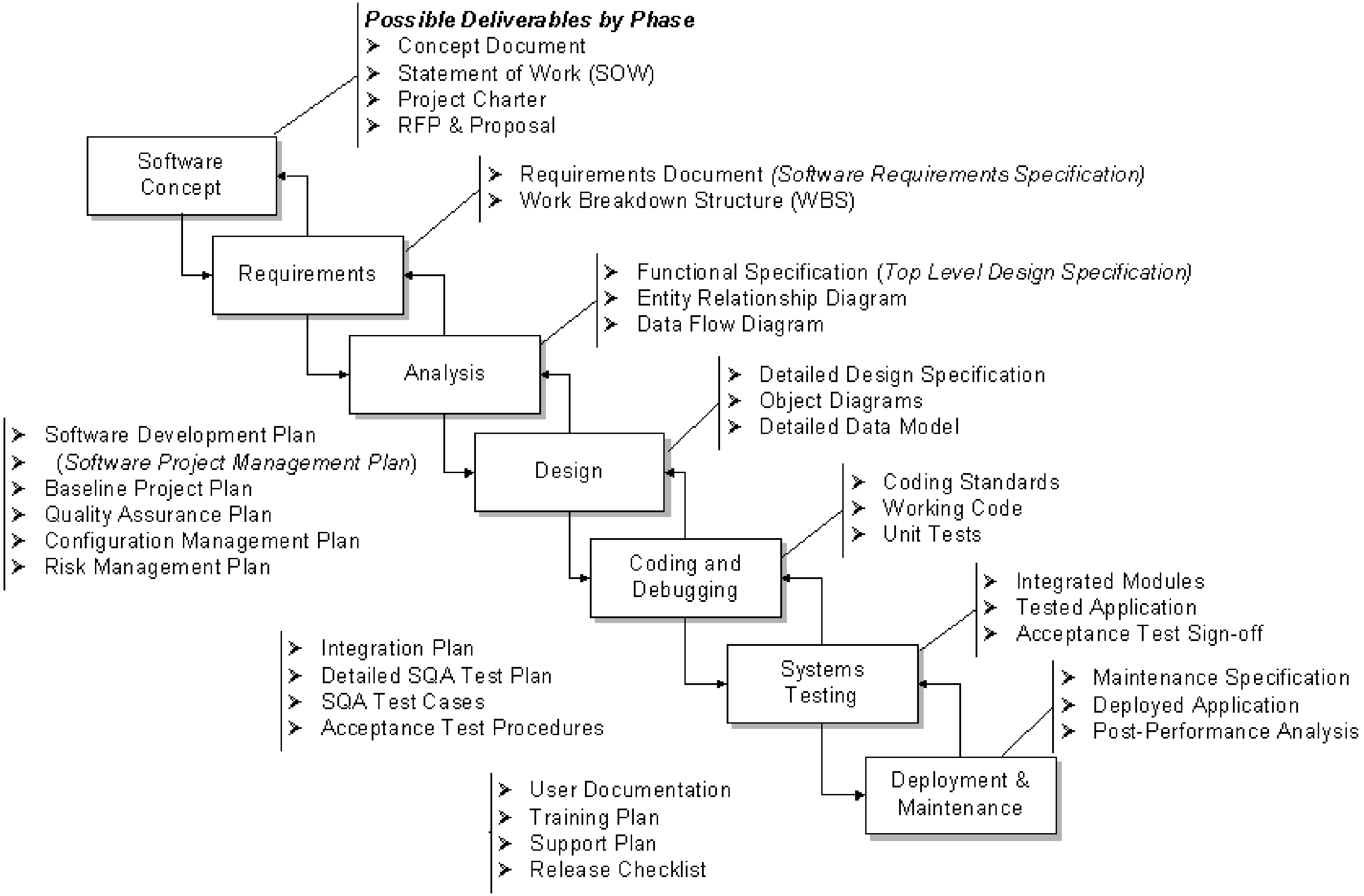
- Allows for excellent organization and tracking
- Better control and use of resources
- Reduces complexity of inter-related tasks
- Allows measurement of outcome versus plans
- Early identification of problems and quick correction

Time Allocation by Phase

Activity	Small Project (2.5K LOC)	Large Project (500K LOC)
Analysis	10%	30%
Design	20%	20%
Code	25%	10%
Unit Test	20%	5%
Integration	15%	20%
System test	10%	15%

McConnell, Steve, “Rapid Development”

Potential Deliverables by Phase



Possible Deliverables by Phase

- Concept Document
- Statement of Work (SOW)
- Project Charter
- RFP & Proposal

Software Concept

- Requirements Document *(Software Requirements Specification)*
- Work Breakdown Structure (WBS)

Requirements

- Functional Specification (*Top Level Design Specification)*
- Entity Relationship Diagram
- Data Flow Diagram

Analysis

- Detailed Design Specification
- Object Diagrams
- Detailed Data Model

- Software Development Plan
- (*Software Project Management Plan*)
- Baseline Project Plan
- Quality Assurance Plan
- Configuration Management Plan
- Risk Management Plan

Design

- Coding Standards
- Working Code
- Unit Tests

Coding and Debugging

- Integrated Modules
- Tested Application
- Acceptance Test Sign-off

- Integration Plan
- Detailed SQA Test Plan
- SQA Test Cases
- Acceptance Test Procedures

Systems Testing

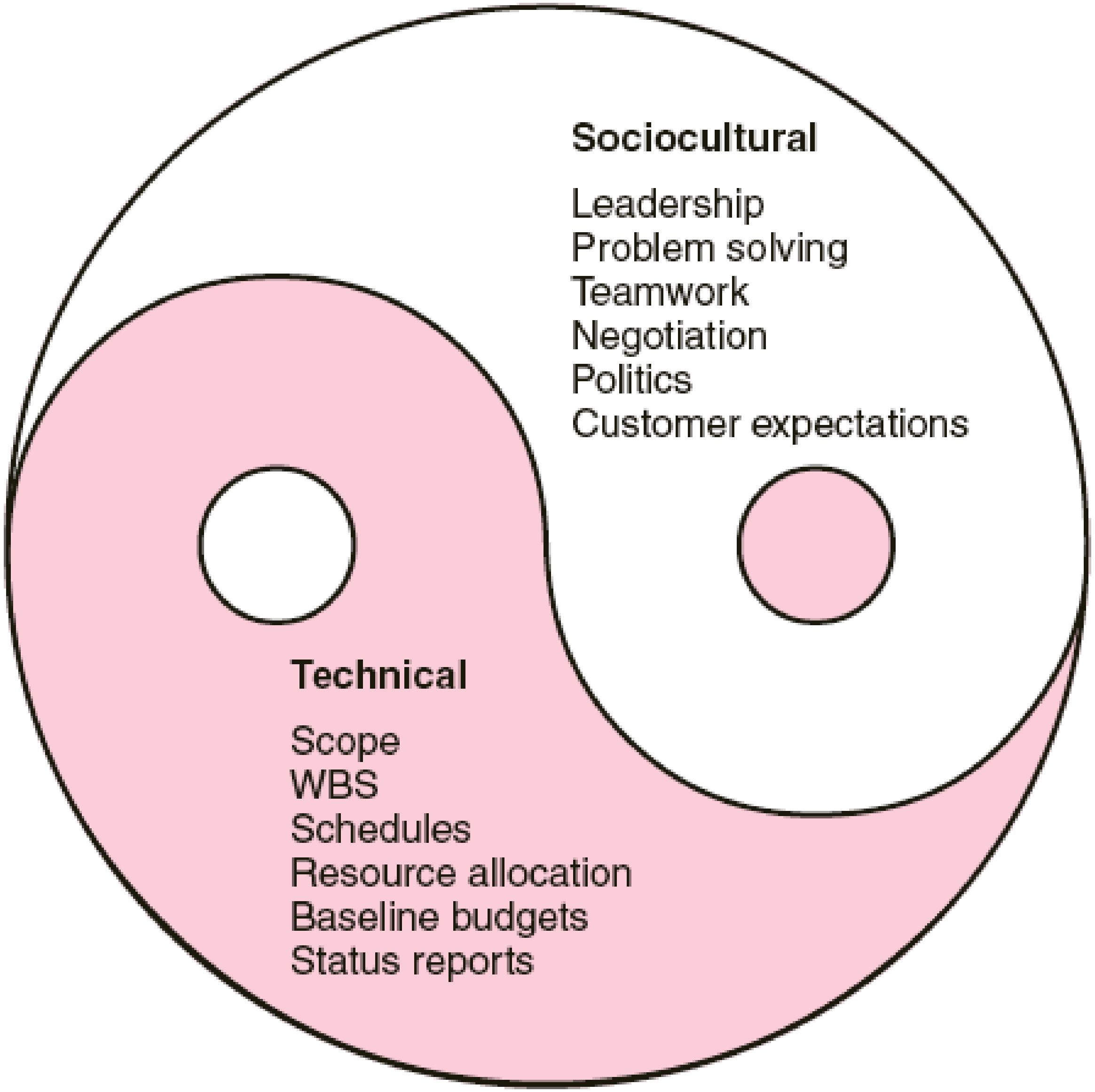
- Maintenance Specification
- Deployed Application
- Post-Performance Analysis

Deployment & Maintenance

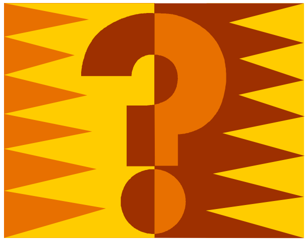
- User Documentation
- Training Plan
- Support Plan
- Release Checklist

The Importance of Project Management

- Factors Leading to the Increased Use of Project Management:
  - □ Compression of the product life cycle
  - □ Global competition
  - □ Knowledge explosion
  - □ Corporate downsizing
  - □ Increased customer focus
  - □ Small projects that represent big problems



The Technical and Sociocultural Dimensions of the Project Management Process



Portfolio of Projects by Type

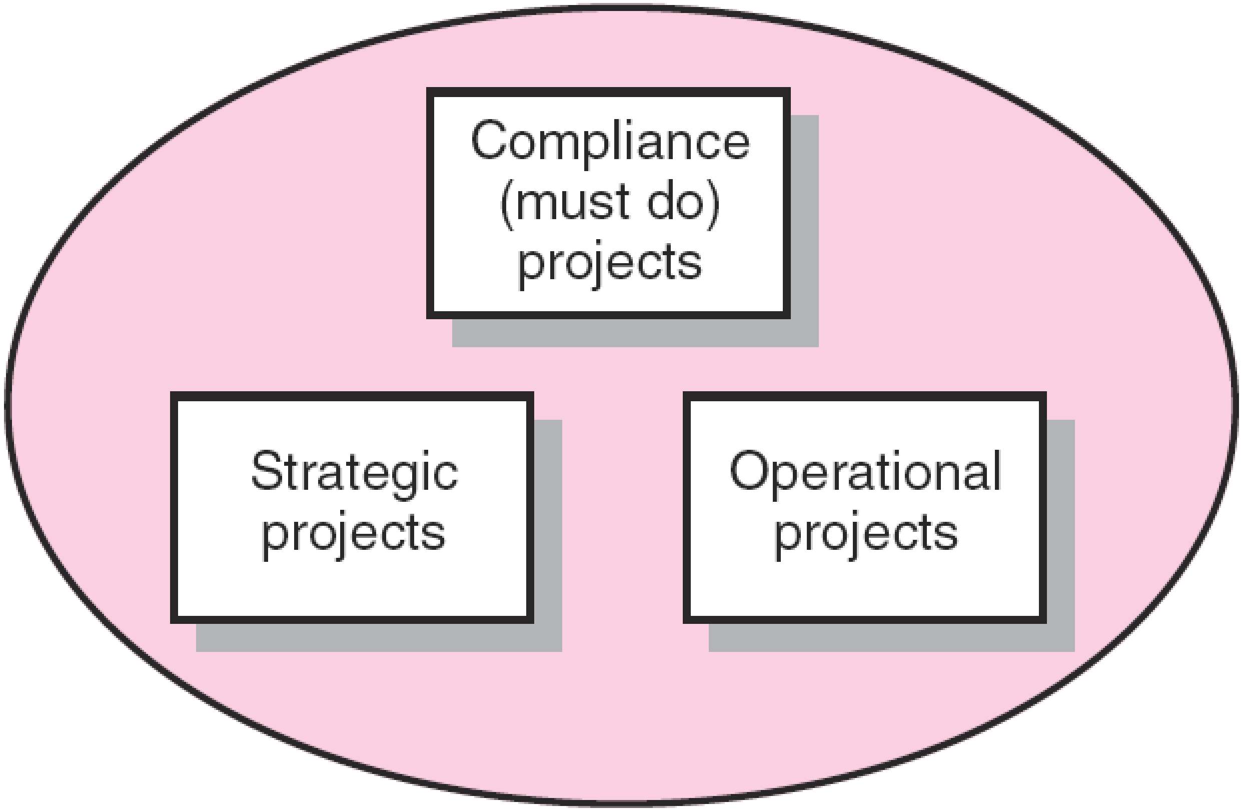


FIGURE 2.2

Multi-Criteria Selection Models

- Checklist Model
  - Uses a list of questions to review potential projects and to determine their acceptance or rejection.
  - Fails to answer the relative importance or value of a potential project and doesn‘t to allow for comparison with other potential projects.
- Multi-Weighted Scoring Model
  - Uses several weighted qualitative and/or quantitative selection criteria to evaluate project proposals.
  - Allows for comparison of projects with other potential projects

Sample Selection Questions Used in Practice

Topic	Question
Strategy/alignment	What specific strategy does this project align with?
Driver	What business problem does the project solve?
Success metrics	How will we measure success?
Sponsorship	Who is the project sponsor?
Risk	What is the impact of not doing this project?
Risk	What is the project risk to our organization?
Risk	Where does the proposed project fit in our risk profile?
Benefits, value, ROI	What is the value of the project to this organization?
Benefits, value, ROI	When will the project show results?
Objectives	What are the project objectives?

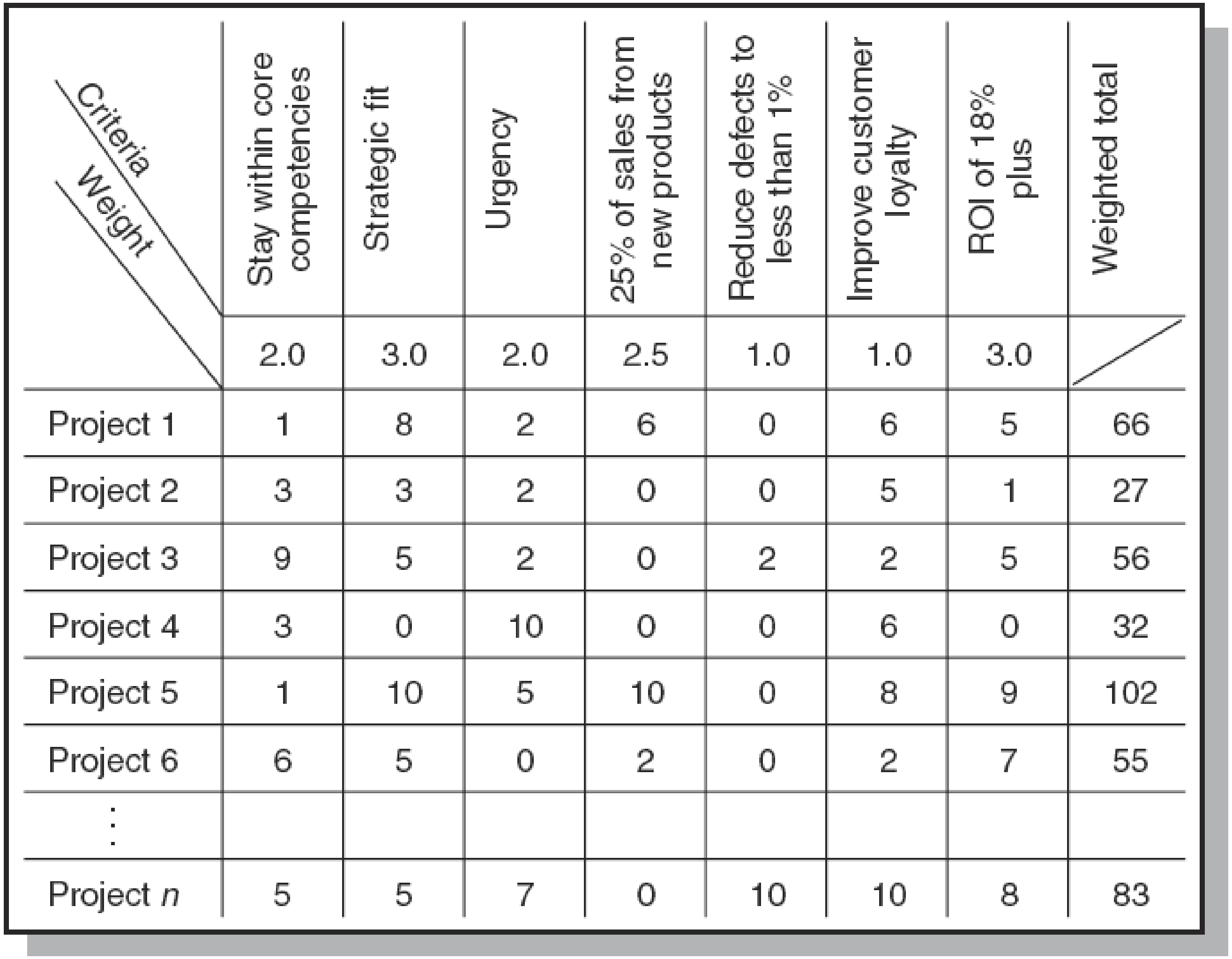
EXHIBIT 2.4

Sample Selection Questions Used in Practice

Topic	Question
Organization culture	Is our organization culture right for this type of project?
Resources	Will internal resources be available for this project?
Approach	Will we build or buy?
Schedule	How long will this project take?
Schedule	Is the time line realistic?
Training/resources	Will staff training be required?
Finance/portfolio	What is the estimated cost of the project?
Portfolio	Is this a new initiative or part of an existing initiative?
Portfolio	How does this project interact with current projects?
Technology	Is the technology available or new?

EXHIBIT 2.4 cont'd

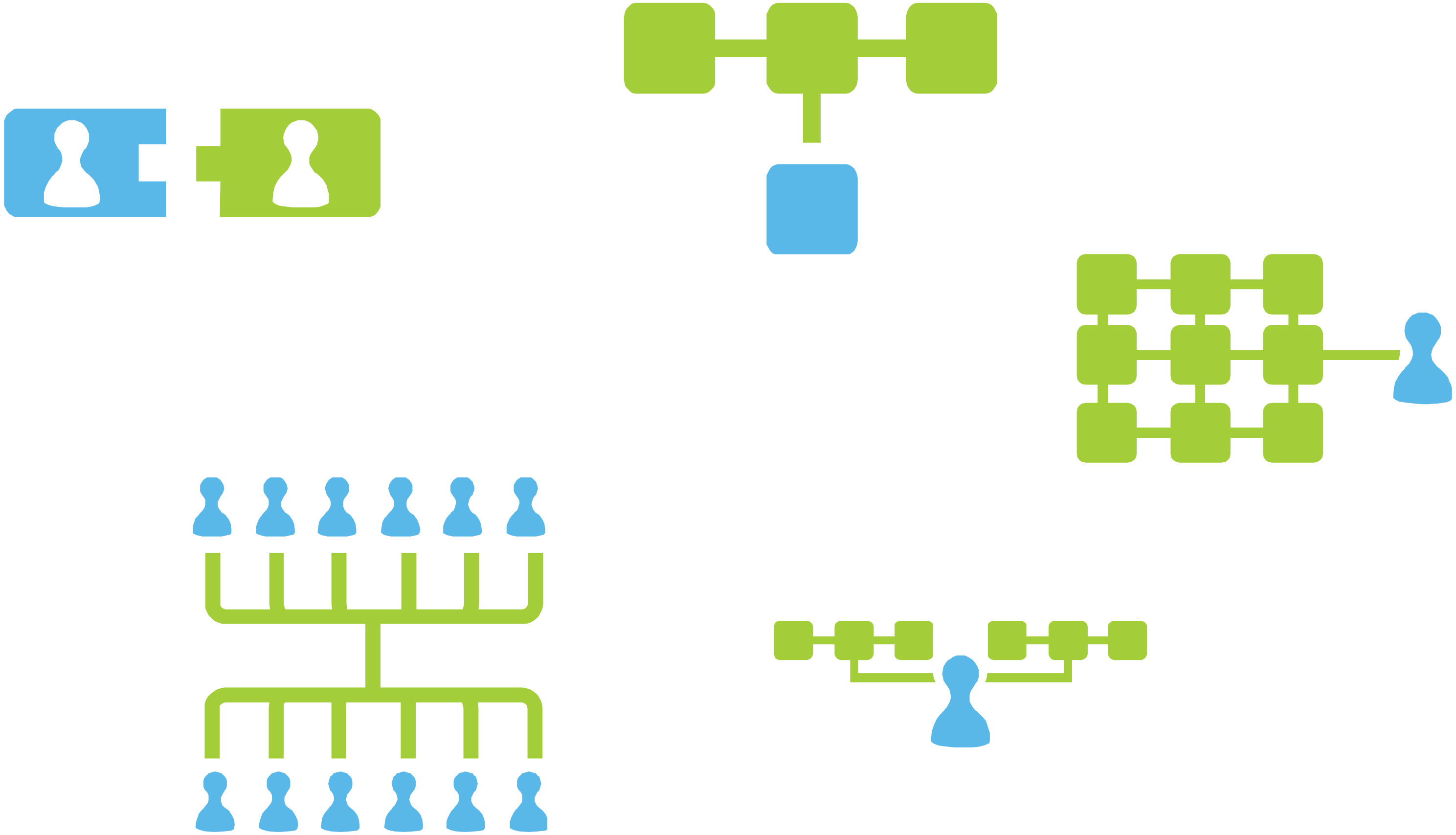
Project Screening Matrix



Criteria Weight	Stay within core competencies	Strategic fit	Urgency	25% of sales from new products	Reduce defects to less than 1%	Improve customer loyalty	ROI of 18% plus	Weighted total
	2.0	3.0	2.0	2.5	1.0	1.0	3.0	
Project 1	1	8	2	6	0	6	5	66
Project 2	3	3	2	0	0	5	1	27
Project 3	9	5	2	0	2	2	5	56
Project 4	3	0	10	0	0	6	0	32
Project 5	1	10	5	10	0	8	9	102
Project 6	6	5	0	2	0	2	7	55
⋮								
Project *n*	5	5	7	0	10	10	8	83

FIGURE 2.3

How do you organize a project team?



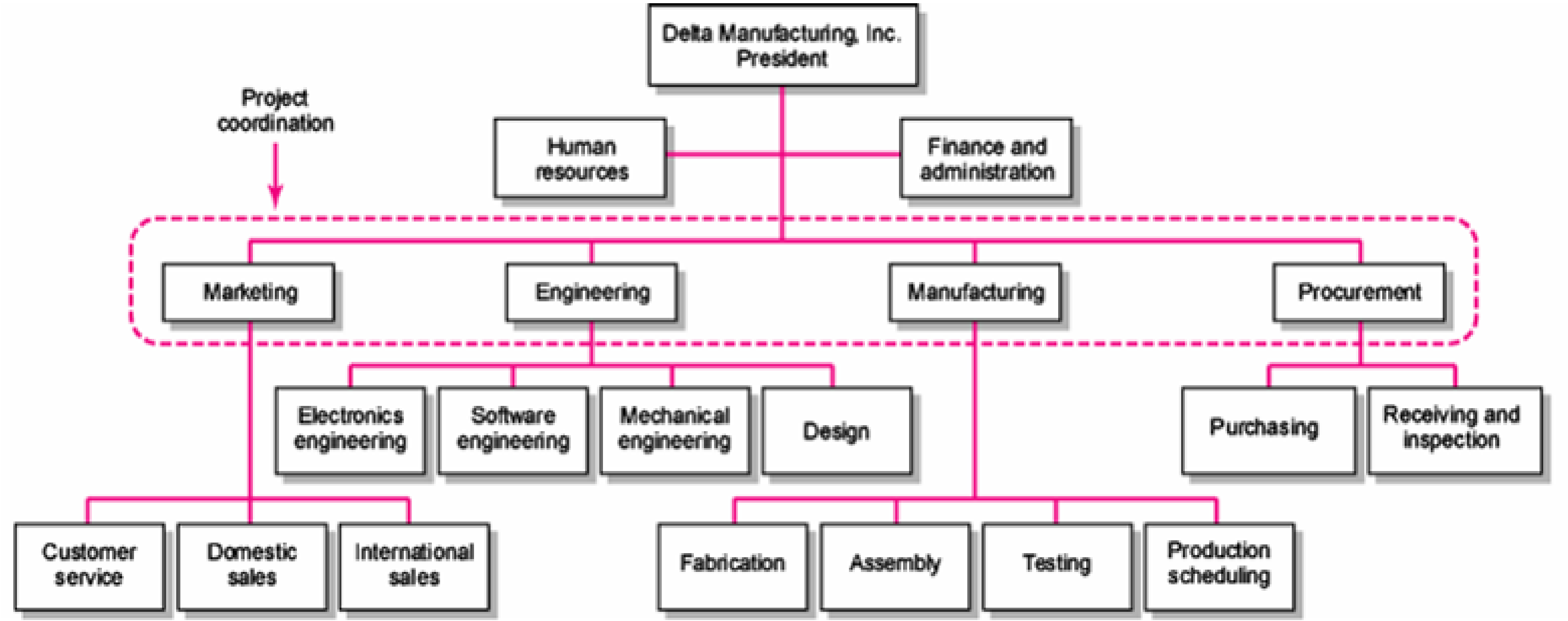
Most Common Structures

- Functional
- Dedicated
- Matrix

Functional Organization

- Different segments of the project are delegated to respective functional units.
- Coordination is maintained through normal management channels.
- Used when the interest of one functional area dominates the project or one functional area has a dominant interest in the project‘s success.

Functional Organizations



Functional Organization of Projects

- **Advantages**
  - **No Structural Change**
  - **Flexibility**
  - **In-Depth Expertise**
  - **Easy Post-Project Transition**

- **Disadvantages**
  - **Lack of Focus**
  - **Poor Integration**
  - **Slow**
  - **Lack of Ownership**

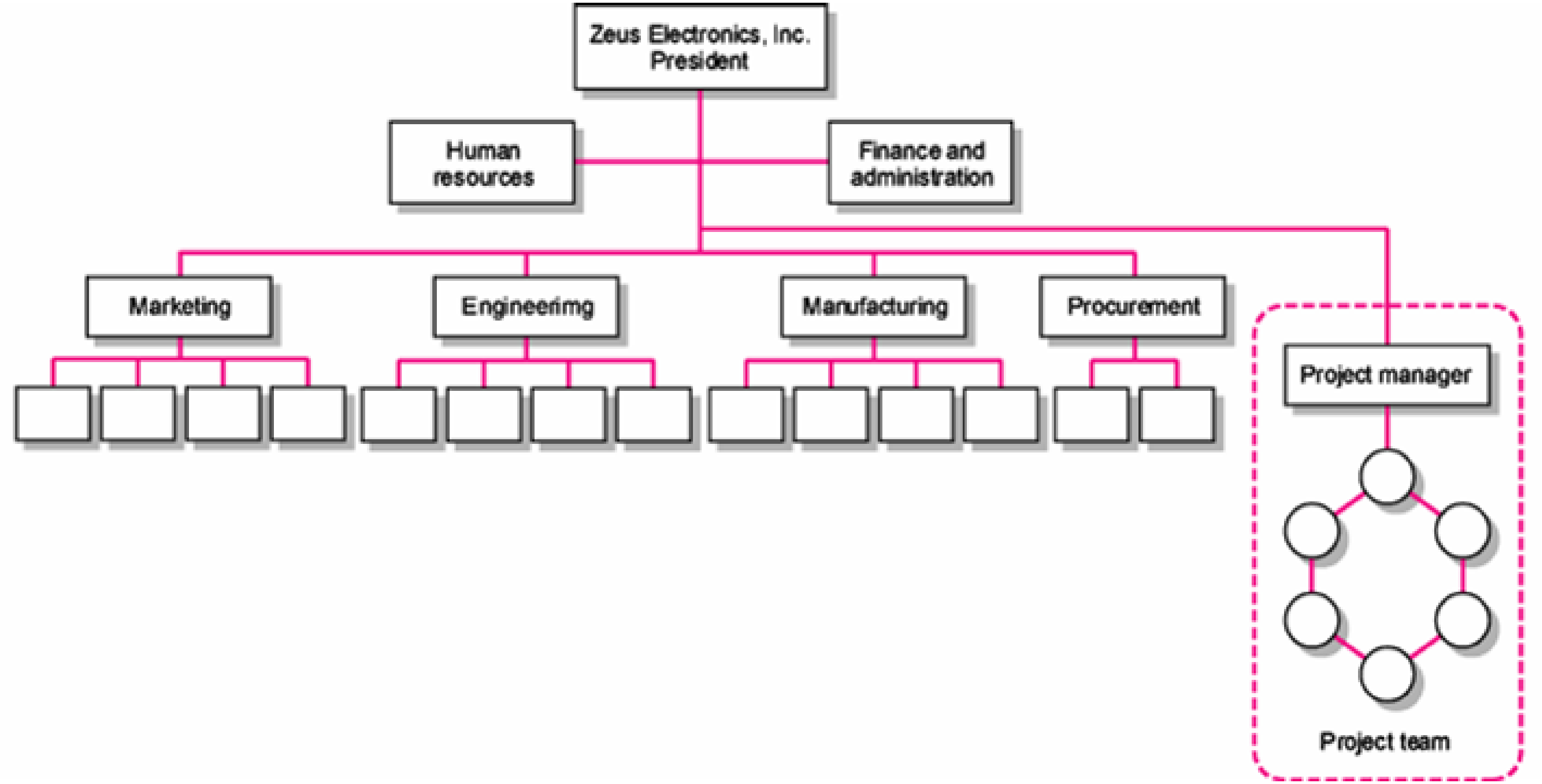
Most Common Structures

- Functional
- Dedicated
- Matrix

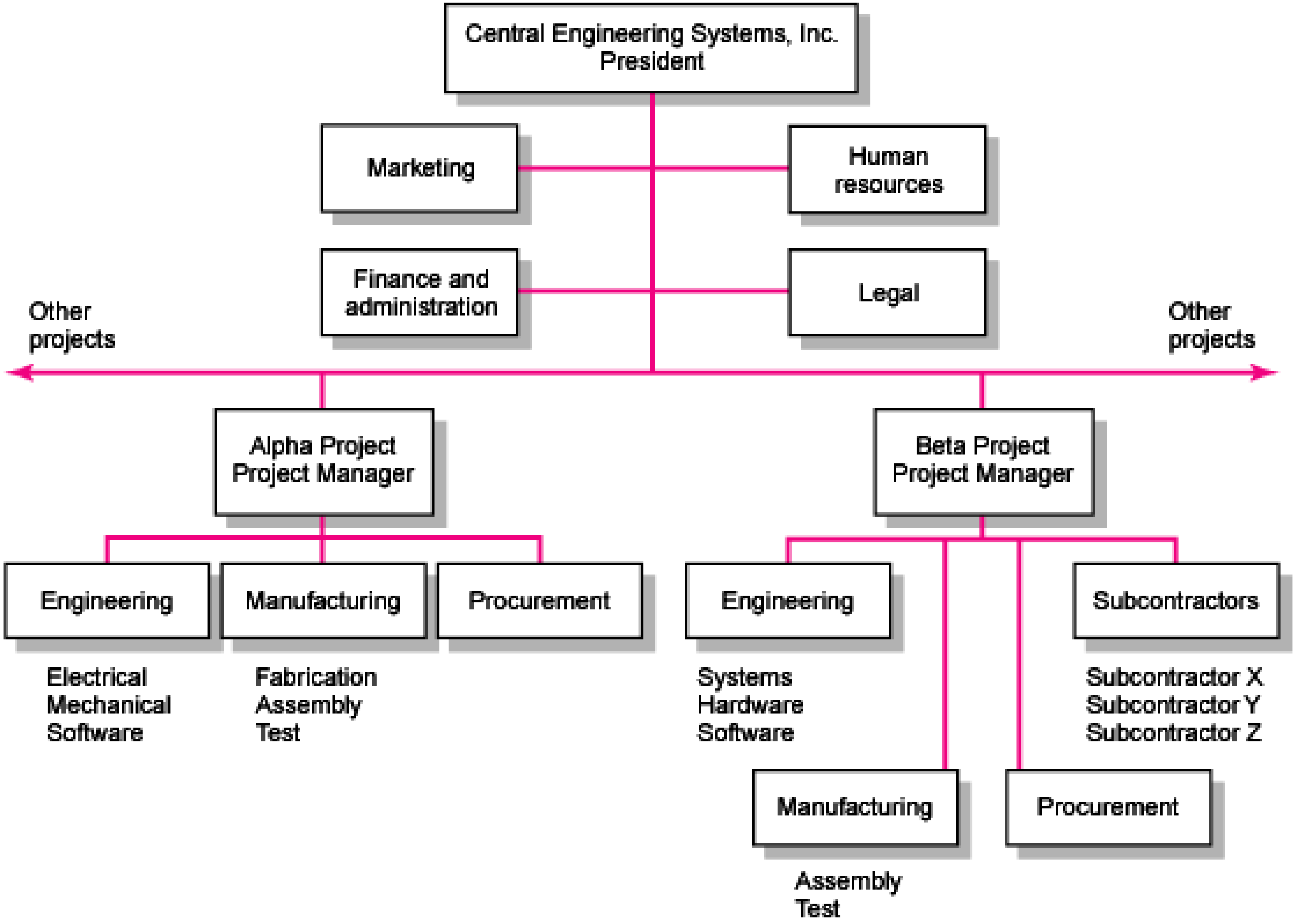
Dedicated Team Structure

- Teams operate as separate units under the leadership of a full-time project manager.
- In a ***projectized*** organization where projects are the dominant form of business, functional departments are responsible for providing support for its teams.

Dedicated Project Team



Project Organizational Structure



Project Organization: Dedicated Team

- **Advantages**
  - **Simple**
  - **Fast**
  - **Cohesive**
  - **Cross-Functional Integration**

- **Disadvantages**
  - **Expensive**
  - **Internal Strife**
  - **Limited Technological Expertise**
  - **Difficult Post-Project Transition**

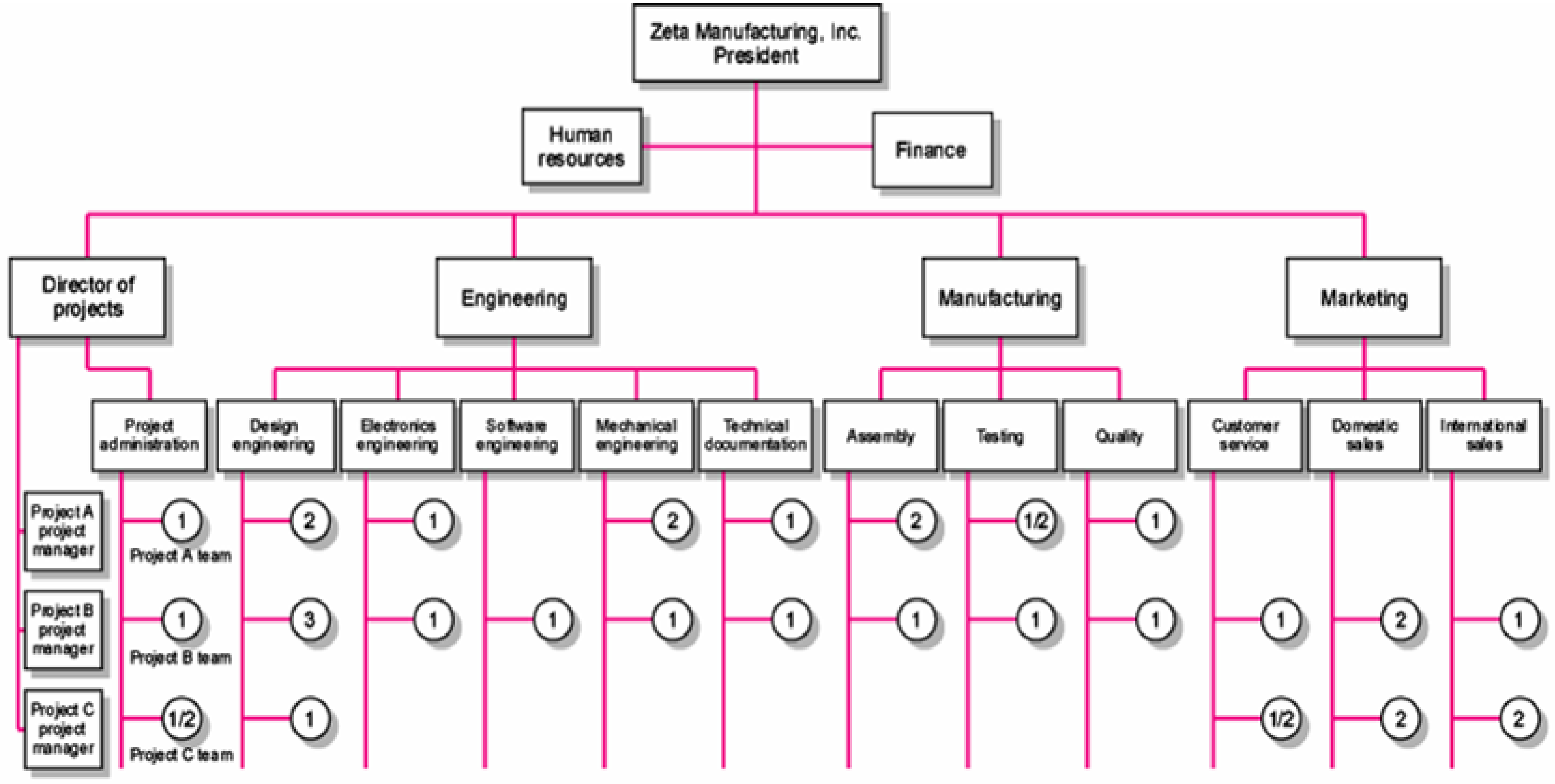
Most Common Structures

- Functional
- Dedicated
- Matrix

Matrix Structure

- Organizing Projects: Matrix Structure
  - Hybrid organizational structure (matrix) is overlaid on the normal functional structure.
    - Two chains of command (functional and project)
    - Project participants report simultaneously to both functional and project managers.
  - Matrix structure optimizes the use of resources.
    - Allows for participation on multiple projects while performing normal functional duties
    - Achieves a greater integration of expertise and project requirements

Matrix Organization Structure



Zeta Manufacturing, Inc.
President

Human resources

Finance

Director of projects

Engineering

Manufacturing

Marketing

Project A project manager

Project B project manager

Project C project manager

	Project administration	Design engineering	Electronics engineering	Software engineering	Mechanical engineering	Technical documentation	Assembly	Testing	Quality	Customer service	Domestic sales	International sales
Project A team	1	2	1		2	1	2	1/2	1			
Project B team	1	3	1	1	1	1	1	1	1	1	2	1
Project C team	1/2	1								1/2	2	2

Project Organization: Matrix Form

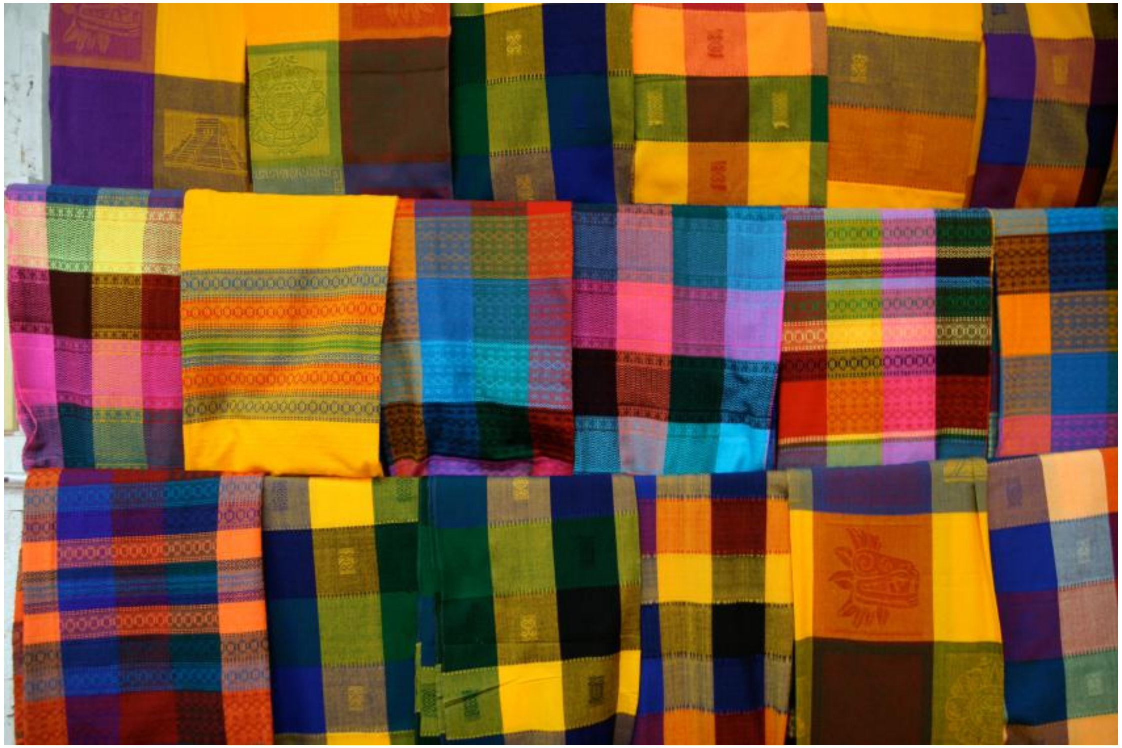
- **Advantages**
  - **Efficient**
  - **Strong Project Focus**
  - **Easier Post-Project Transition**
  - **Flexible**

- **Disadvantages**
  - **Dysfunctional Conflict**
  - **Infighting**
  - **Stressful**
  - **Slow**

Group Discussion

- Going to college is analogous to working in a matrix environment in that most students take more than one class and must distribute their time across multiple classes.
- What problems does this situation create for you?
- How does it affect your performance?
- How could the system be better managed to make your life less difficult and more productive?

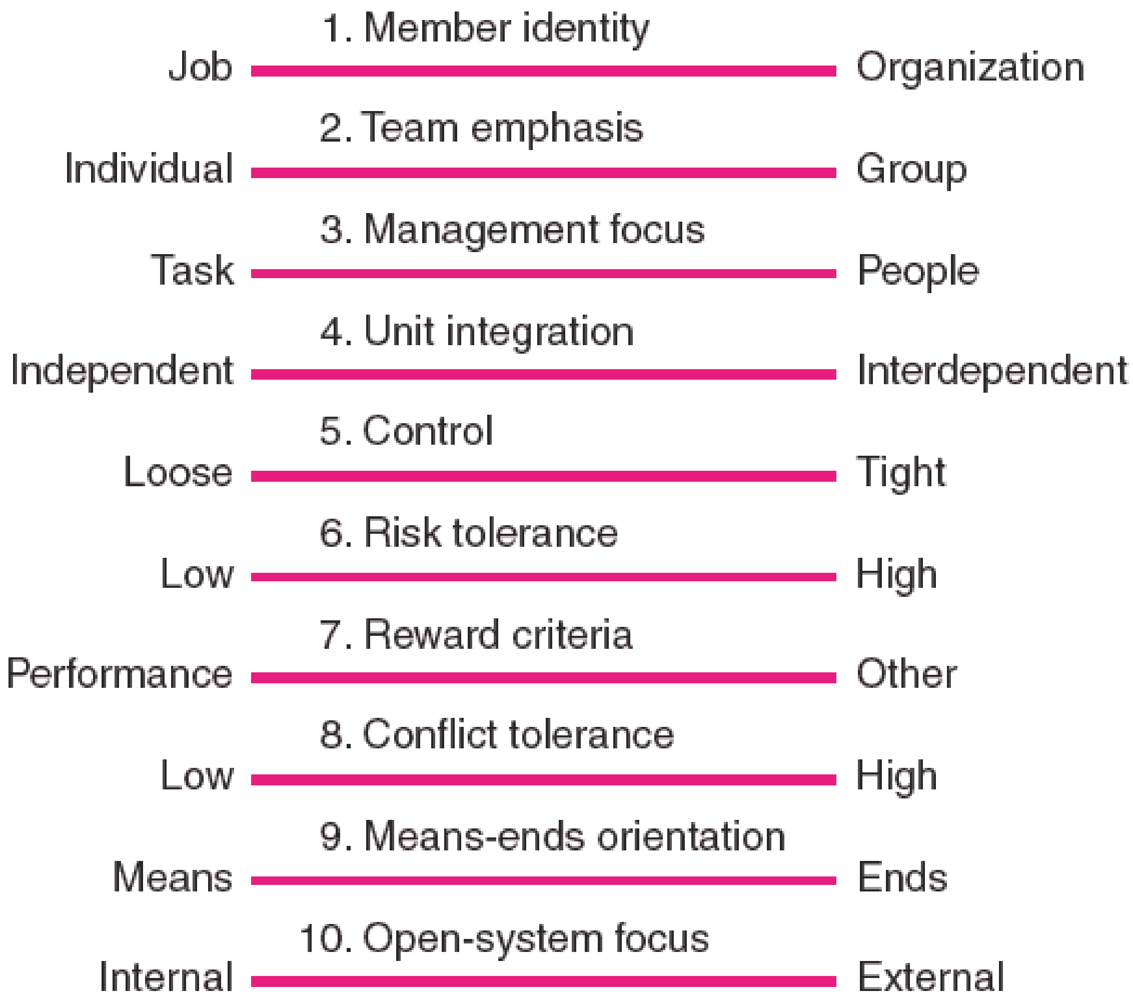
What is an organization‘s culture?



Organizational Culture

- Organizational Culture Defined
  - □ A system of shared norms, beliefs, values, and assumptions which bind people together, thereby creating shared meanings
  - □ The —personality” of the organization that sets it apart from other organizations.
    - Provides a sense of identify to its members
    - Helps legitimize the management system of the organization
    - Clarifies and reinforces standards of behavior

Key Dimensions Defining an Organization's Culture

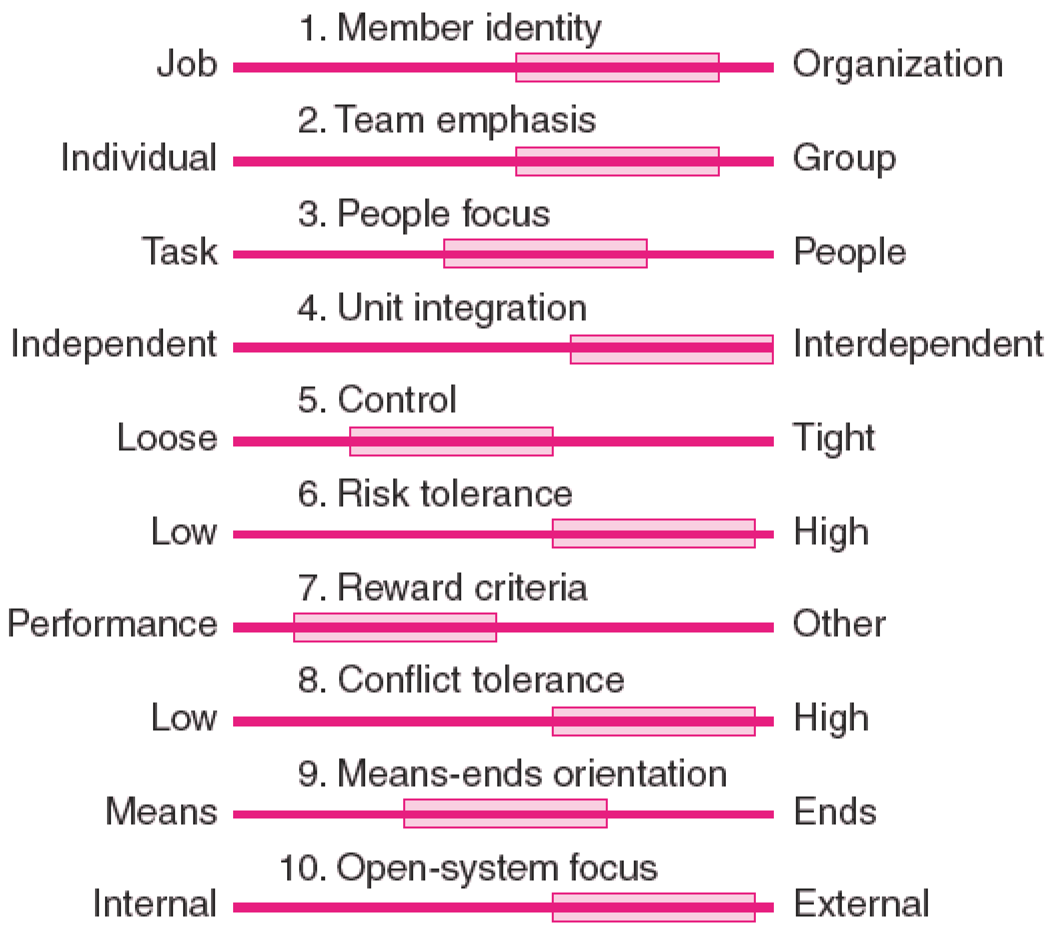


	1. Member identity	
Job		Organization
	2. Team emphasis	
Individual		Group
	3. Management focus	
Task		People
	4. Unit integration	
Independent		Interdependent
	5. Control	
Loose		Tight
	6. Risk tolerance	
Low		High
	7. Reward criteria	
Performance		Other
	8. Conflict tolerance	
Low		High
	9. Means-ends orientation	
Means		Ends
	10. Open-system focus	
Internal		External

Group Discussion

- How would we map the culture of UTD to the previous mapping

Cultural Dimensions of an Organization Supportive of Project Management



Small Team Discussion

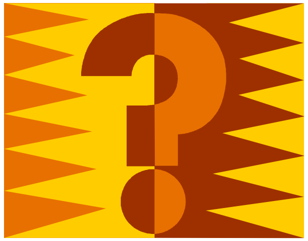
- **What do you believe is more important for successfully completing a project – the formal project management structure or the culture of the parent organization?**

Some thoughts on previous question

- Both are important and an argument can be made for either structure or culture.
- Culture tends to be more important than structure since it more directly impacts behavior. A positive organizational culture can compensate for the inherent weaknesses of the formal structure.
  - For example a functional matrix can be effective if the norms and customs of the organization value teamwork and effective problem-solving. Conversely, a functional matrix is likely to be disastrous in a negative culture that encourages competition and looking out only for yourself.

Some thoughts on previous question

- Alternatively, one could argue that an organization can circumvent a negative culture by creating an independent project team or a strong project matrix. In either case, the strategy is to insulate the project team from the dominant organizational culture and create a unique project subculture.



Small Team Hands-On

- **Team size of 3-4**
- Build a house using paper and tape
- You can only request the tape once during the exercise and can only use it for at most 30 seconds
- If the tape is in use, form a queue for the tape
- You will need to transport your house to the front
- 8 minutes

Getting started on a project…

- Define it
- Get agreement on the definition
- Improperly defining a project leads to project failure a high percentage of the time

Defining the Project

Step 1: Defining the Project Scope

Step 2: Establishing Project Priorities

Step 3: Creating the Work Breakdown Structure (not covered in this course)

Step 4: Integrating the WBS with the Organization(not covered in this course)

Step 5: Coding the WBS for the Information System(not covered in this course)

Step 1: Defining the Project Scope

- Project Scope
  - A definition of the end result or mission of the project—a product or service for the client/customer—in specific, tangible, and measurable terms.
- Purpose of the Scope Statement
  - To clearly define the deliverable(s) for the end user.
  - To focus the project on successful completion of its goals.
  - To be used by the project owner and participants as a planning tool and for measuring project success.

Project Scope Checklist

1. Project objective
2. Deliverables
3. Milestones
4. Technical requirements
5. Limits and exclusions
6. Reviews with customer

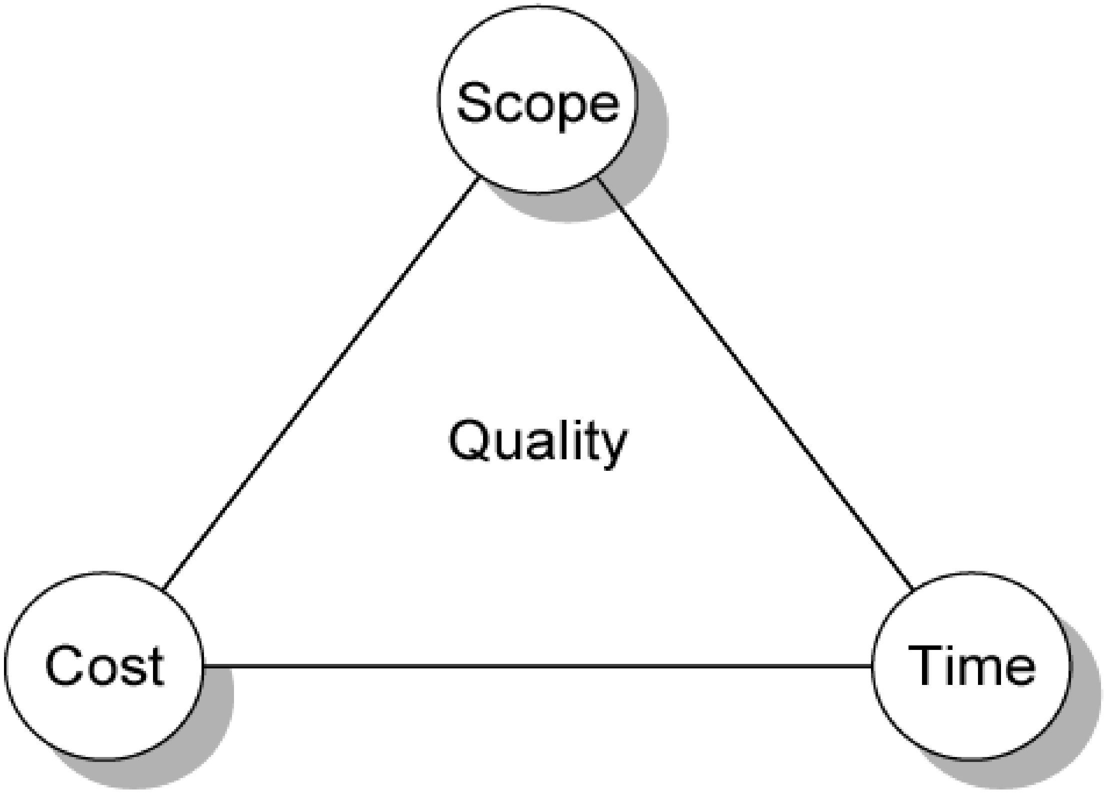
Project Scope: Terms and Definitions

- Scope Statements
  - Also called statements of work (SOW)
- Project Charter
  - Can contain an expanded version of scope statement
  - A document authorizing the project manager to initiate and lead the project.
- Scope Creep
  - The tendency for the project scope to expand over time due to changing requirements, specifications, and priorities.

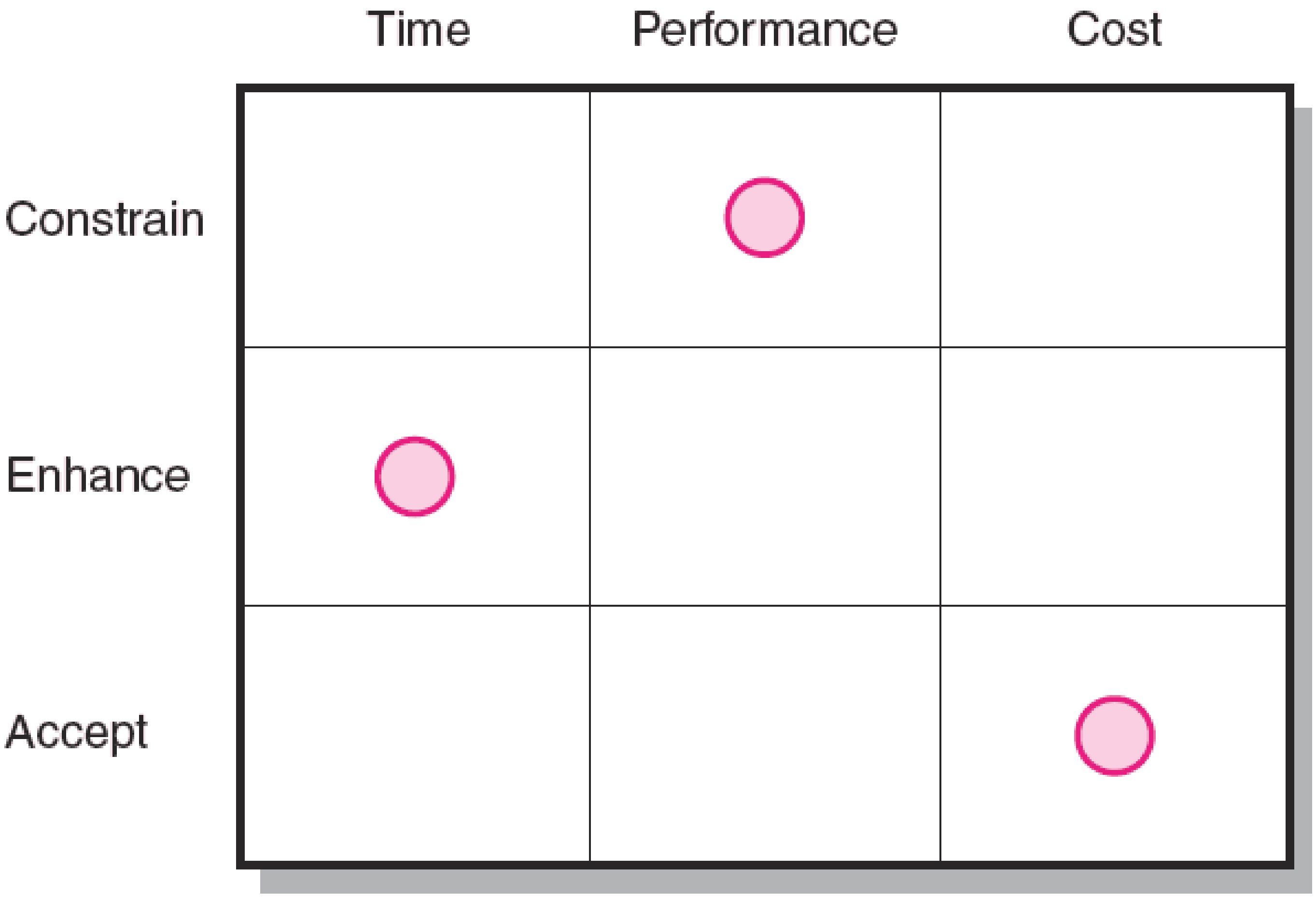
Step 2: Establishing Project Priorities

- Causes of Project Trade-offs
  - Shifts in the relative importance of criterions related to cost, time, and performance parameters
    - Budget–Cost
    - Schedule–Time
    - Performance–Scope
- Managing the Priorities of Project Trade-offs
  - Constrain: a parameter is a fixed requirement.
  - Enhance: optimizing a parameter over others.
  - Accept: reducing (or not meeting) a parameter requirement.

Project Management Trade-offs



Project Priority Matrix

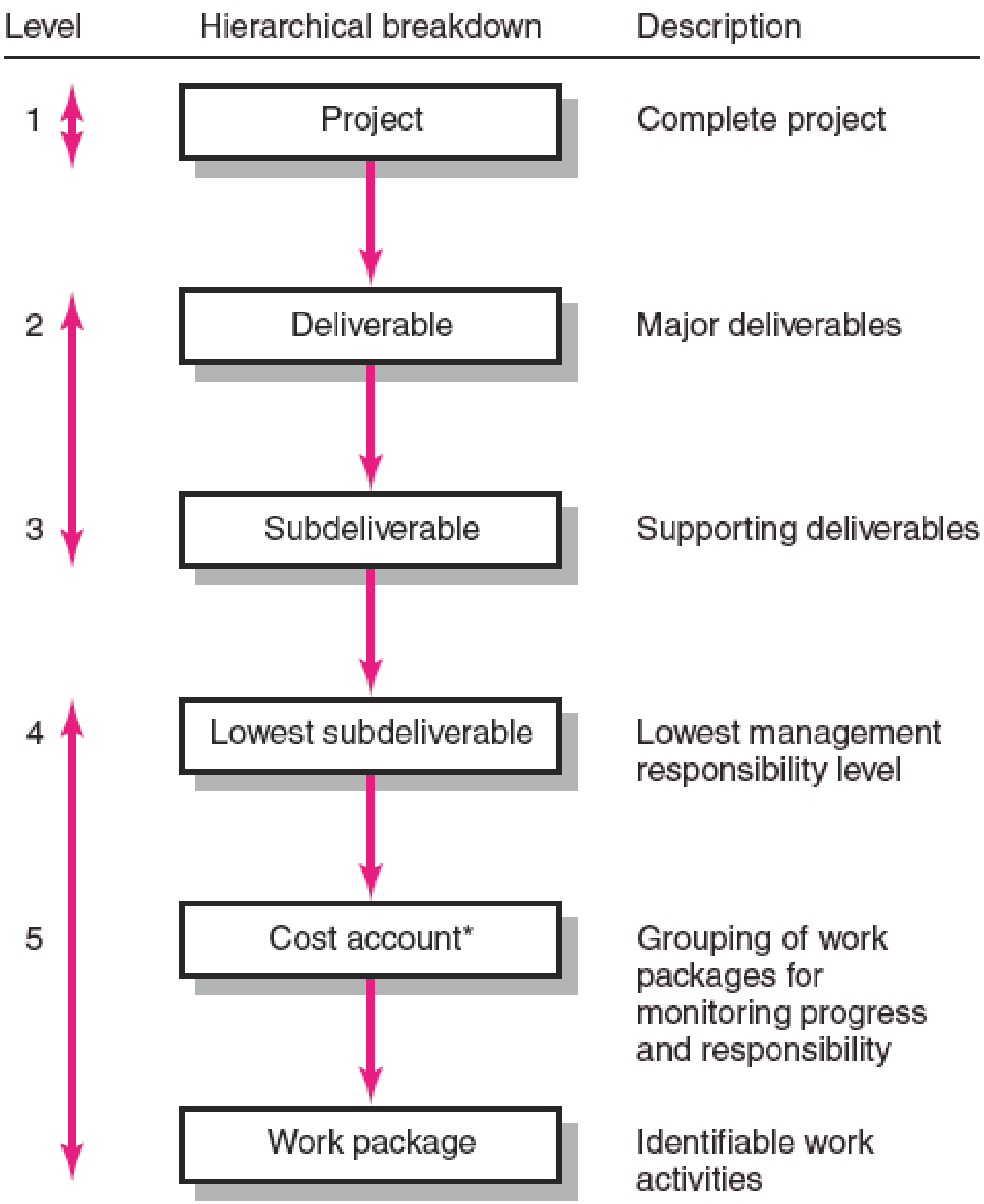


	Time	Performance	Cost
Constrain		●	
Enhance	●		
Accept			●

Example Projects

- **Time constrain, Scope enhance, cost accept**
  - Wealthy New Year‘s Eve Party
  - Political campaign
- **Time enhance, Scope constrain**
  - New line of bulletproof clothing
  - Public construction of a bridge
- **Time constrain, cost enhance**
  - Fuel efficient engine
  - Longer lasting battery for laptop computers

Hierarchical Breakdown of the WBS

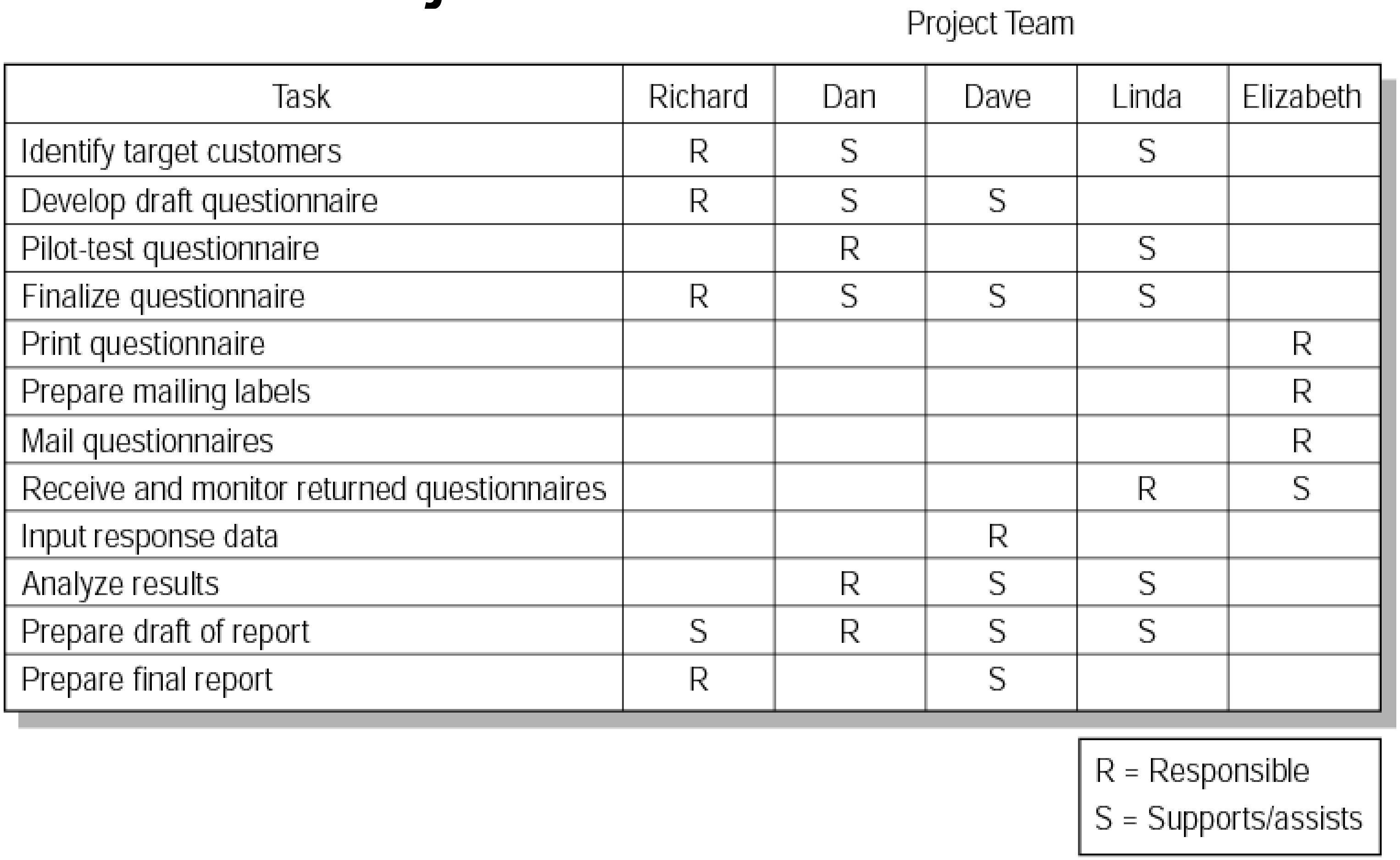


Level	Hierarchical breakdown	Description
1	Project	Complete project
2	Deliverable	Major deliverables
3	Subdeliverable	Supporting deliverables
4	Lowest subdeliverable	Lowest management responsibility level
5	Cost account*	Grouping of work packages for monitoring progress and responsibility
	Work package	Identifiable work activities

Responsibility Matrices

- Responsibility Matrix (RM)
  - Also called a linear responsibility chart
  - Summarizes the tasks to be accomplished and who is responsible for what on the project
    - Lists project activities and participants
    - Clarifies critical interfaces between units and individuals that need coordination
    - Provide an means for all participants to view their responsibilities and agree on their assignments
    - Clarifies the extent or type of authority that can be exercised by each participant

Responsibility Matrix for a Market Research Project

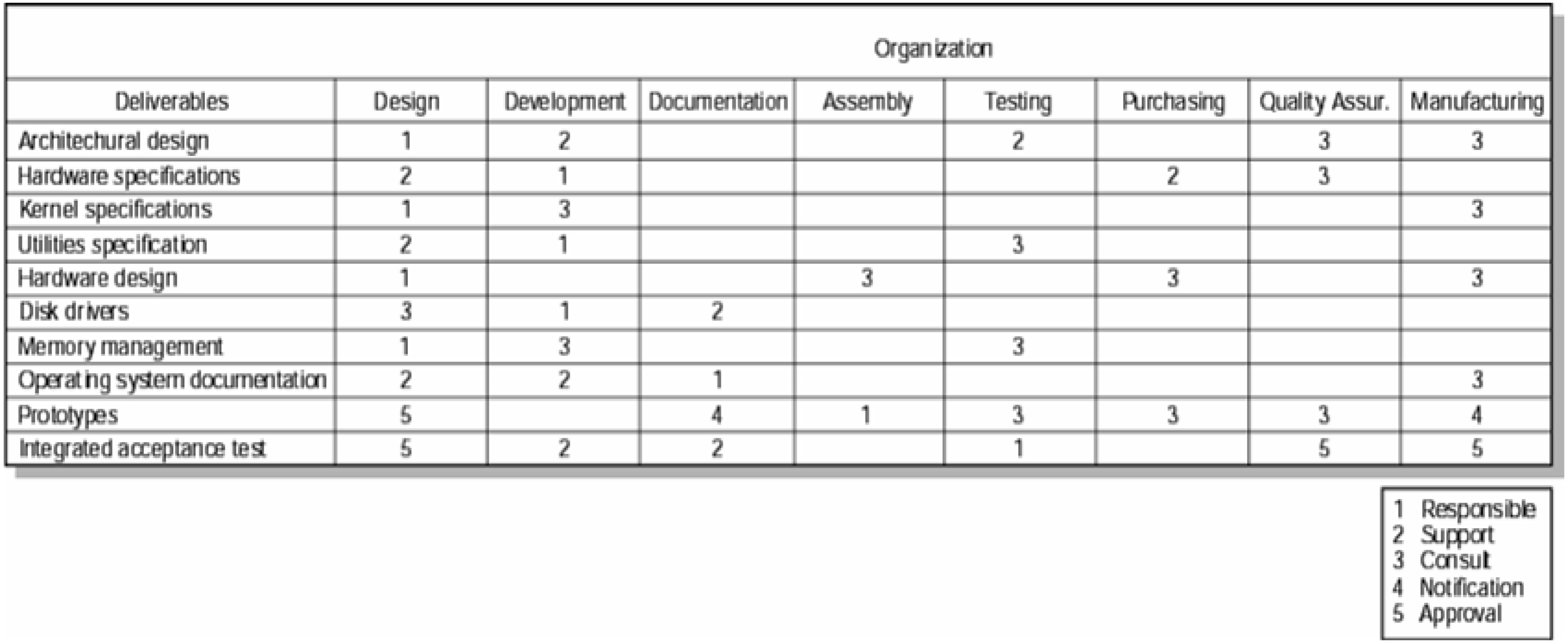


Project Team

Task	Richard	Dan	Dave	Linda	Elizabeth
Identify target customers	R	S		S	
Develop draft questionnaire	R	S	S		
Pilot-test questionnaire		R		S	
Finalize questionnaire	R	S	S	S	
Print questionnaire					R
Prepare mailing labels					R
Mail questionnaires					R
Receive and monitor returned questionnaires				R	S
Input response data			R		
Analyze results		R	S	S	
Prepare draft of report	S	R	S	S	
Prepare final report	R		S		

R = Responsible
S = Supports/assists

Responsibility Matrix for the Conveyor Belt Project



	Organization							
Deliverables	Design	Development	Documentation	Assembly	Testing	Purchasing	Quality Assur.	Manufacturing
Architectural design	1	2			2		3	3
Hardware specifications	2	1				2	3	
Kernel specifications	1	3						3
Utilities specification	2	1			3			
Hardware design	1			3		3		3
Disk drivers	3	1	2					
Memory management	1	3			3			
Operating system documentation	2	2	1					3
Prototypes	5		4	1	3	3	3	4
Integrated acceptance test	5	2	2		1		5	5

1 Responsible
2 Support
3 Consult
4 Notification
5 Approval

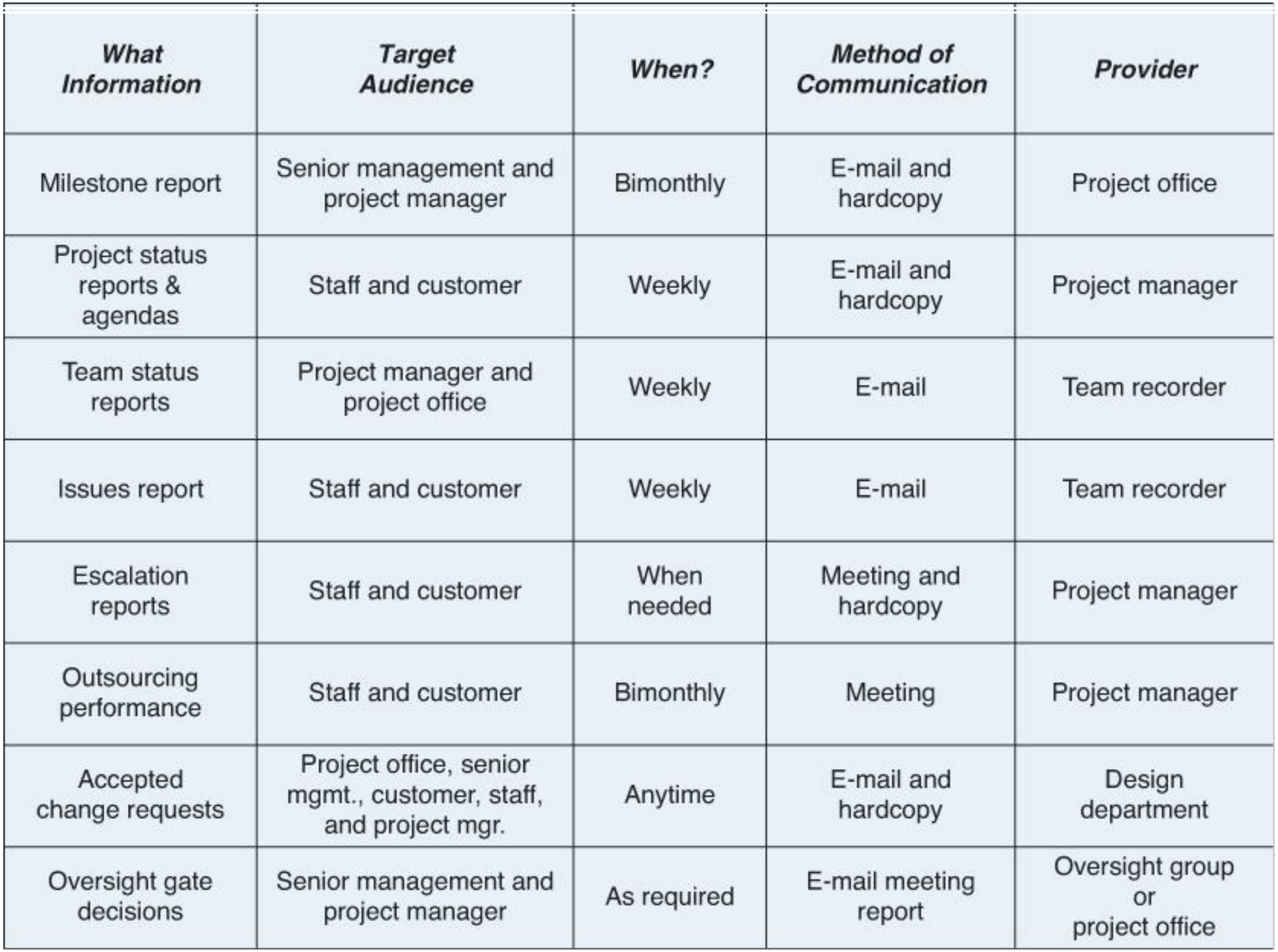
Project Communication Plan

- What information needs to be collected?
- Who will receive information?
- What information methods will be used?
- What are the access restrictions?
- When will information be communicated?
- How will information be communicated?

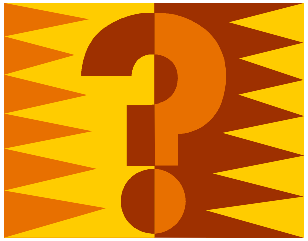
Communication Plan

- Stakeholder analysis
- Information needs
- Sources of information
- Dissemination modes
- Responsibility and timing

Communication Plan:



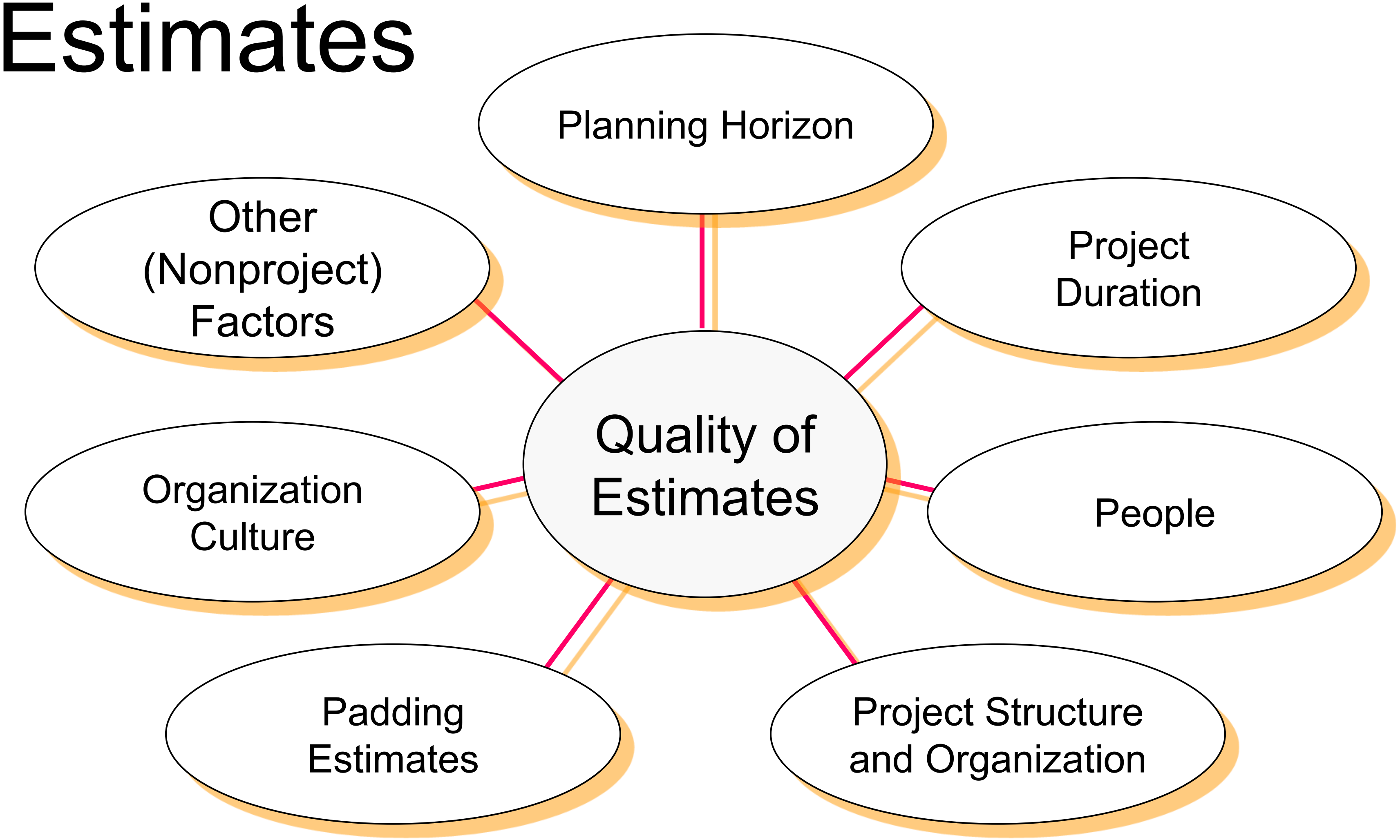
What Information	*Target Audience*	*When?*	*Method of Communication*	*Provider*
Milestone report	Senior management and project manager	Bimonthly	E-mail and hardcopy	Project office
Project status reports & agendas	Staff and customer	Weekly	E-mail and hardcopy	Project manager
Team status reports	Project manager and project office	Weekly	E-mail	Team recorder
Issues report	Staff and customer	Weekly	E-mail	Team recorder
Escalation reports	Staff and customer	When needed	Meeting and hardcopy	Project manager
Outsourcing performance	Staff and customer	Bimonthly	Meeting	Project manager
Accepted change requests	Project office, senior mgmt., customer, staff, and project mgr.	Anytime	E-mail and hardcopy	Design department
Oversight gate decisions	Senior management and project manager	As required	E-mail meeting report	Oversight group or project office



Estimating Projects

- Estimating
  - The process of forecasting or approximating the time and cost of completing project deliverables.
  - The task of balancing expectations of stakeholders and need for control while the project is implemented.
- Types of Estimates
  - Top-down (macro) estimates: analogy, group consensus, or mathematical relationships
  - Bottom-up (micro) estimates: estimates of elements of the work breakdown structure

Factors Influencing the Quality of Estimates



Why Estimating Time and Cost Are Important

- support good decisions.
- schedule work.
- determine how long the project should take and its cost.
- determine whether the project is worth doing.
- develop cash flow needs.
- determine how well the project is progressing.
- develop time-phased budgets and establish the project baseline.

Why are accurate estimates critical to effective project management?

- Without accurate time and cost estimates project control is ineffective. Inaccurate estimates can make the difference between profit and loss.
- Time and cost estimates are major inputs to project planning.
- Project control is completely dependent on accuracy of estimates.
- Estimates are needed to support good decisions.
- Estimates are used to determine project duration and cost.
- Estimates are used to develop cash flow needs.
- Estimates are used to develop time-phased budgets and establish the project baseline.
- Absence of estimates results in inaccuracies which result in time and cost under/overruns.
- The activity of estimating reduces error.

Estimating Guidelines for Times, Costs, and Resources

1. Have people familiar with the tasks make the estimate.
2. Use several people to make estimates.
3. Base estimates on normal conditions, efficient methods, and a normal level of resources.
4. Use consistent time units in estimating task times.
5. Treat each task as independent, don‘t aggregate.
6. Don‘t make allowances for contingencies.
7. Adding a risk assessment helps avoid surprises to stakeholders.

Hands On Activity

- ES-DTU company plans to develop software and needs you to create an estimate
- The software has
  - 15 inputs that are rather low complexity
  - 5 outputs that are average complexity
  - 10 inquiries that are average complexity
  - 30 files that are rather complex
  - 20 interfaces with other systems that are average complexity
- How can you estimate this new software development?

Types of Estimates

- Top-down (macro) estimates: analogy, group consensus, or mathematical relationships
- Bottom-up (micro) estimates: estimates of elements of the work breakdown structure

Top-Down versus Bottom-Up Estimating

- Top-Down Estimates
  - □ Are usually are derived from someone who uses experience and/or information to determine the project duration and total cost.
  - □ Are made by top managers who have little knowledge of the processes used to complete the project.
- Bottom-Up Approach
  - □ Can serve as a check on cost elements in the WBS by rolling up the work packages and associated cost accounts to major deliverables at the work package level.

Top-Down versus Bottom-Up Estimating

Conditions for Preferring Top-Down or Bottom-up Time and Cost Estimates

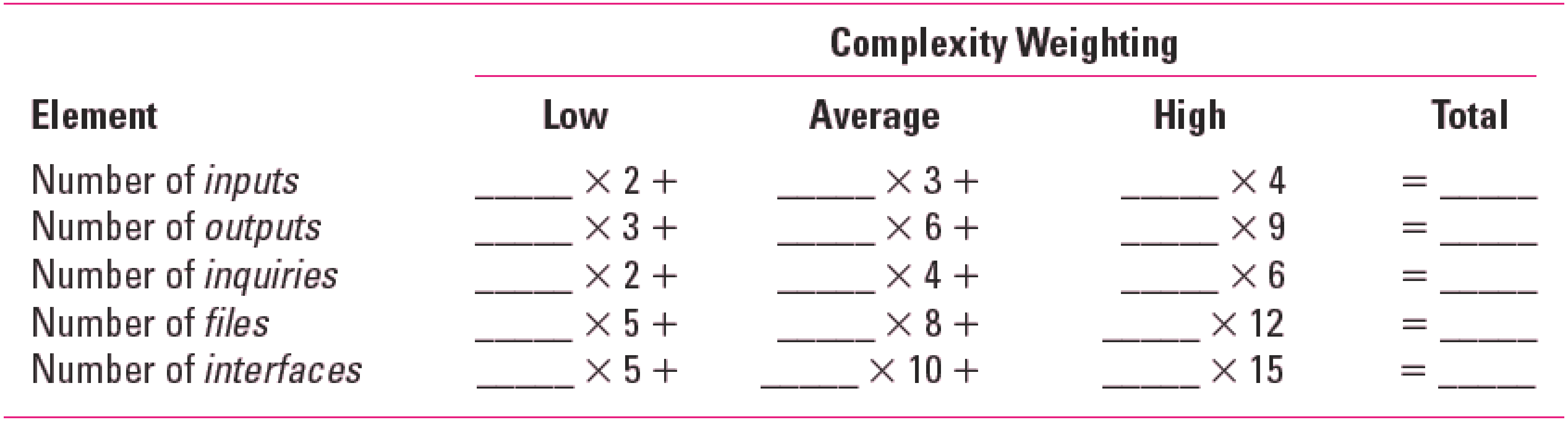
Condition	Macro Estimates	Micro Estimates
Strategic decision making	X	
Cost and time important		X
High uncertainty	X	
Internal, small project	X	
Fixed-price contract		X
Customer wants details		X
Unstable scope	X	

TABLE 5.1

Estimating Projects: Preferred Approach

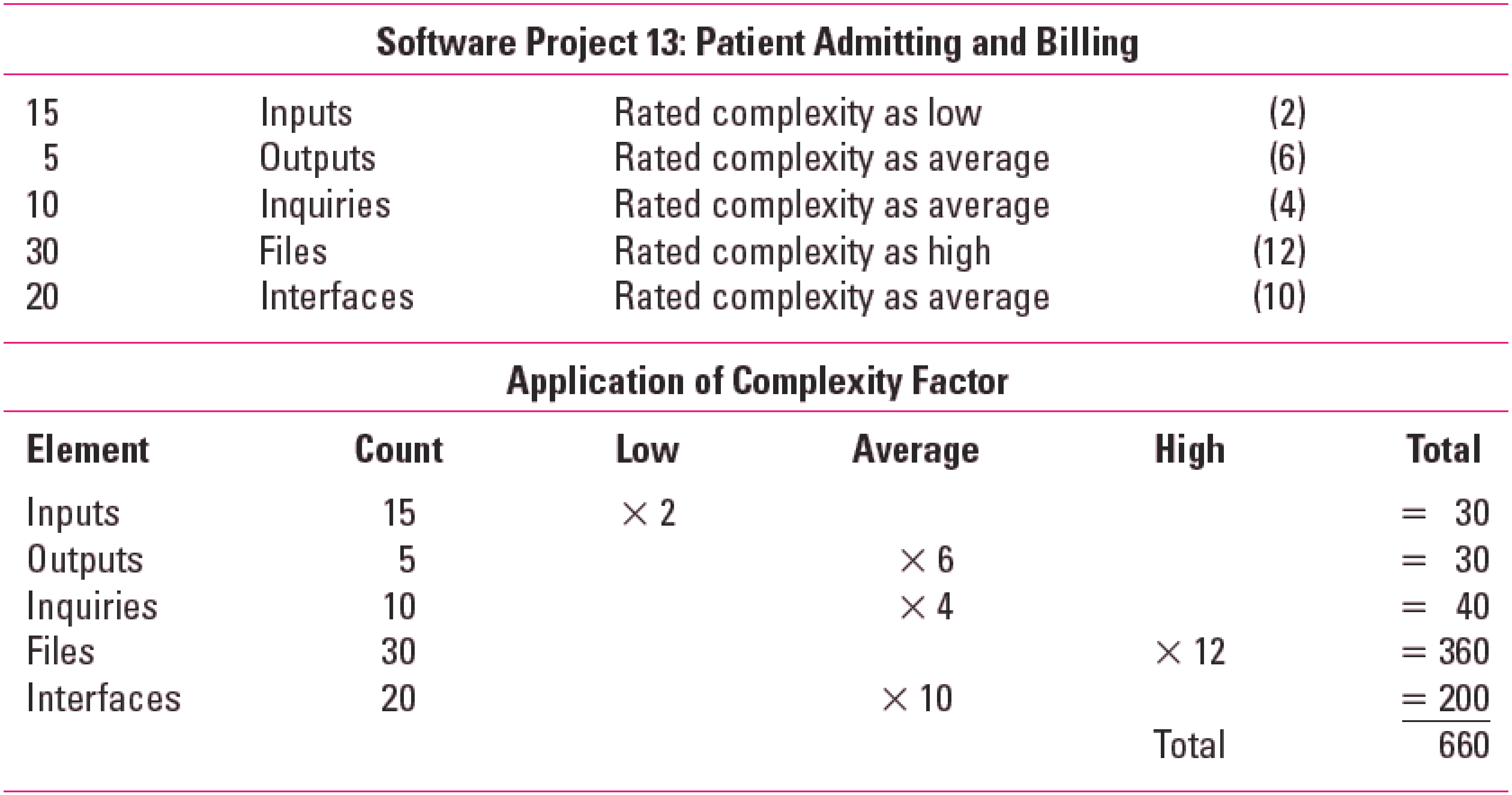
- Make rough top-down estimates
- Make bottom-up estimates
- Develop schedules and budgets
- Reconcile differences between top-down and bottom-up estimates

Simplified Basic Function Point Count Process for a Prospective Project or Deliverable



	Complexity Weighting			
Element	Low	Average	High	Total
Number of *inputs*	_____ × 2 +	_____ × 3 +	_____ × 4	= _____
Number of *outputs*	_____ × 3 +	_____ × 6 +	_____ × 9	= _____
Number of *inquiries*	_____ × 2 +	_____ × 4 +	_____ × 6	= _____
Number of *files*	_____ × 5 +	_____ × 8 +	_____ × 12	= _____
Number of *interfaces*	_____ × 5 +	_____ × 10 +	_____ × 15	= _____

Example:
Function Point Count Method



Software Project 13: Patient Admitting and Billing

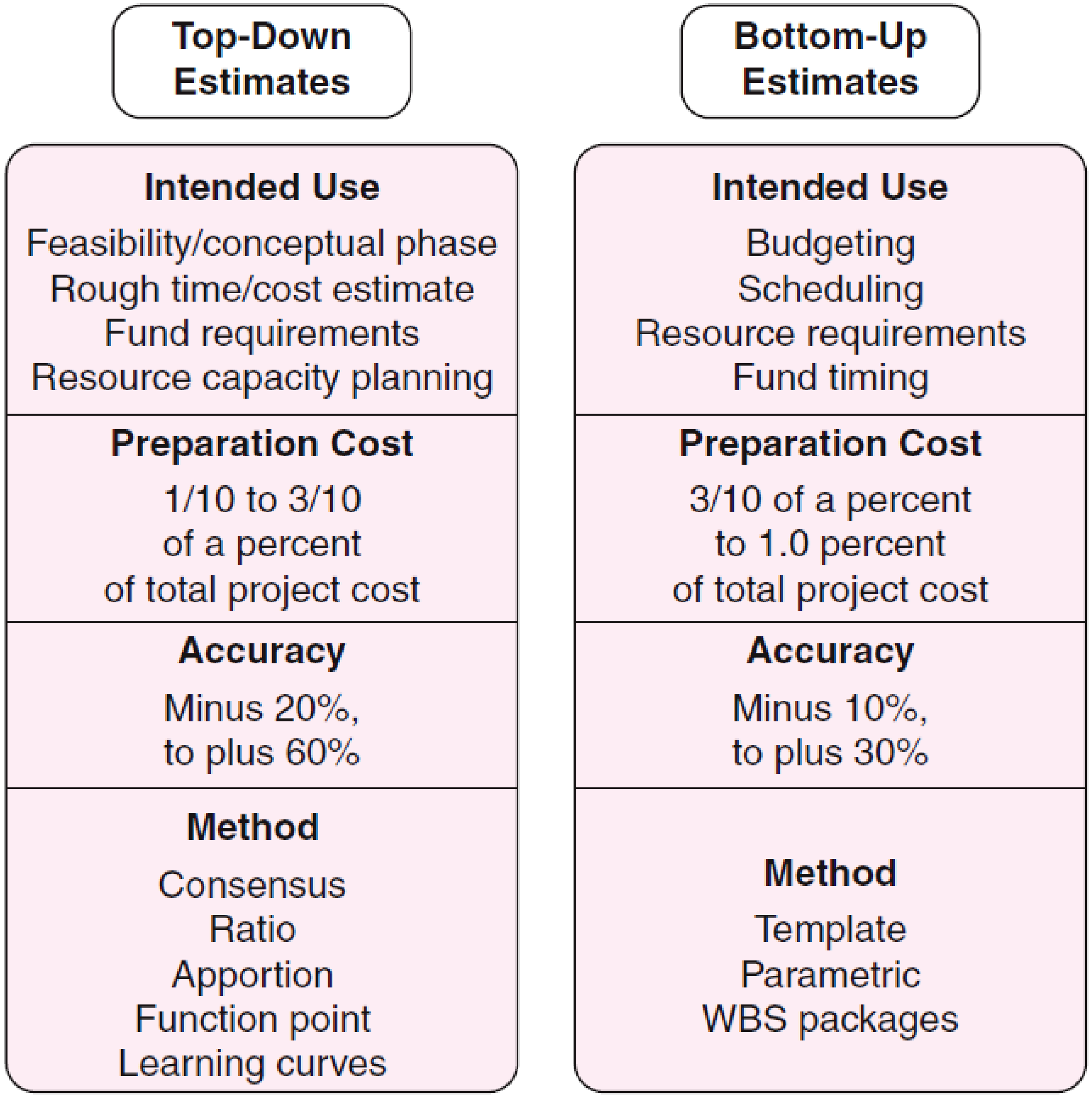
15	Inputs	Rated complexity as low	(2)
5	Outputs	Rated complexity as average	(6)
10	Inquiries	Rated complexity as average	(4)
30	Files	Rated complexity as high	(12)
20	Interfaces	Rated complexity as average	(10)

Application of Complexity Factor

Element	**Count**	**Low**	**Average**	**High**	**Total**
Inputs	15	× 2			= 30
Outputs	5		× 6		= 30
Inquiries	10		× 4		= 40
Files	30			× 12	= 360
Interfaces	20		× 10		= 200
				Total	660

Historically one person month = 5 function points

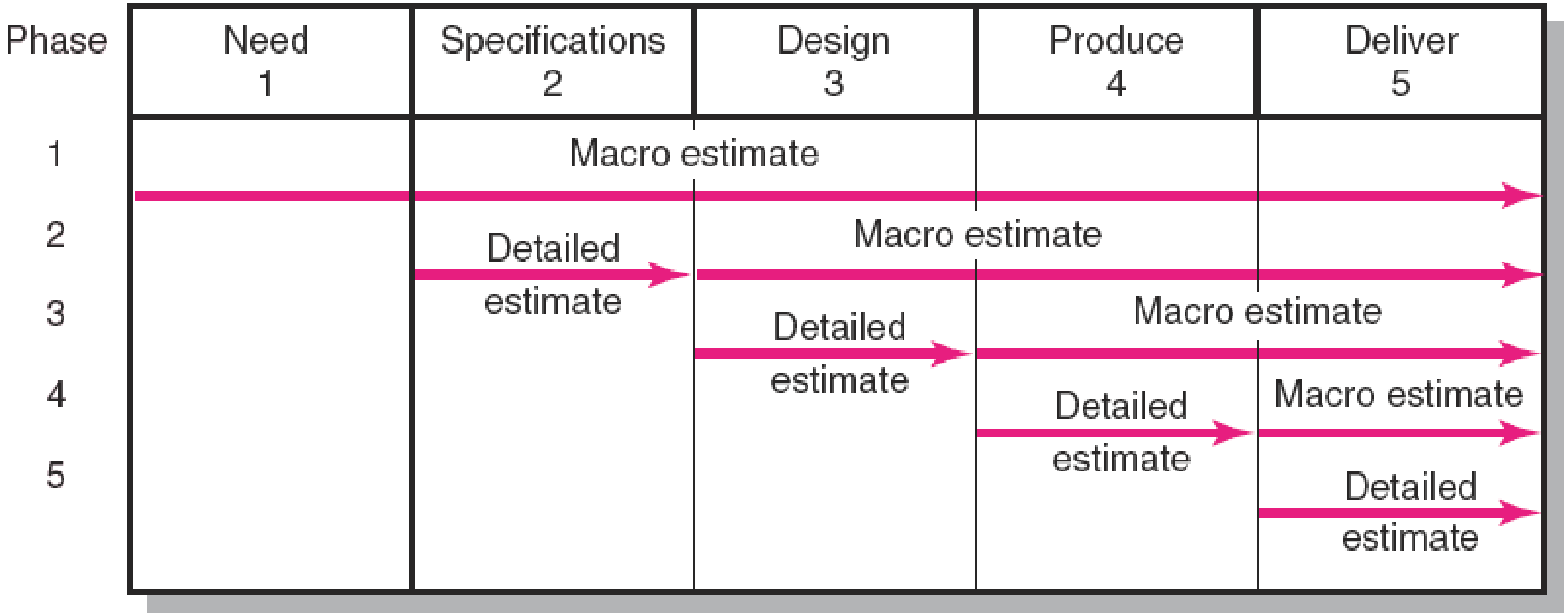
Top-Down and Bottom-Up Estimates



Top-Down Estimates	Bottom-Up Estimates
Intended Use Feasibility/conceptual phase Rough time/cost estimate Fund requirements Resource capacity planning	**Intended Use** Budgeting Scheduling Resource requirements Fund timing
Preparation Cost 1/10 to 3/10 of a percent of total project cost	**Preparation Cost** 3/10 of a percent to 1.0 percent of total project cost
Accuracy Minus 20%, to plus 60%	**Accuracy** Minus 10%, to plus 30%
Method Consensus Ratio Apportion Function point Learning curves	**Method** Template Parametric WBS packages

FIGURE 5.4

Phase Estimating over Product Life Cycle



Phase

Need
1

Specifications
2

Design
3

Produce
4

Deliver
5

1 Macro estimate

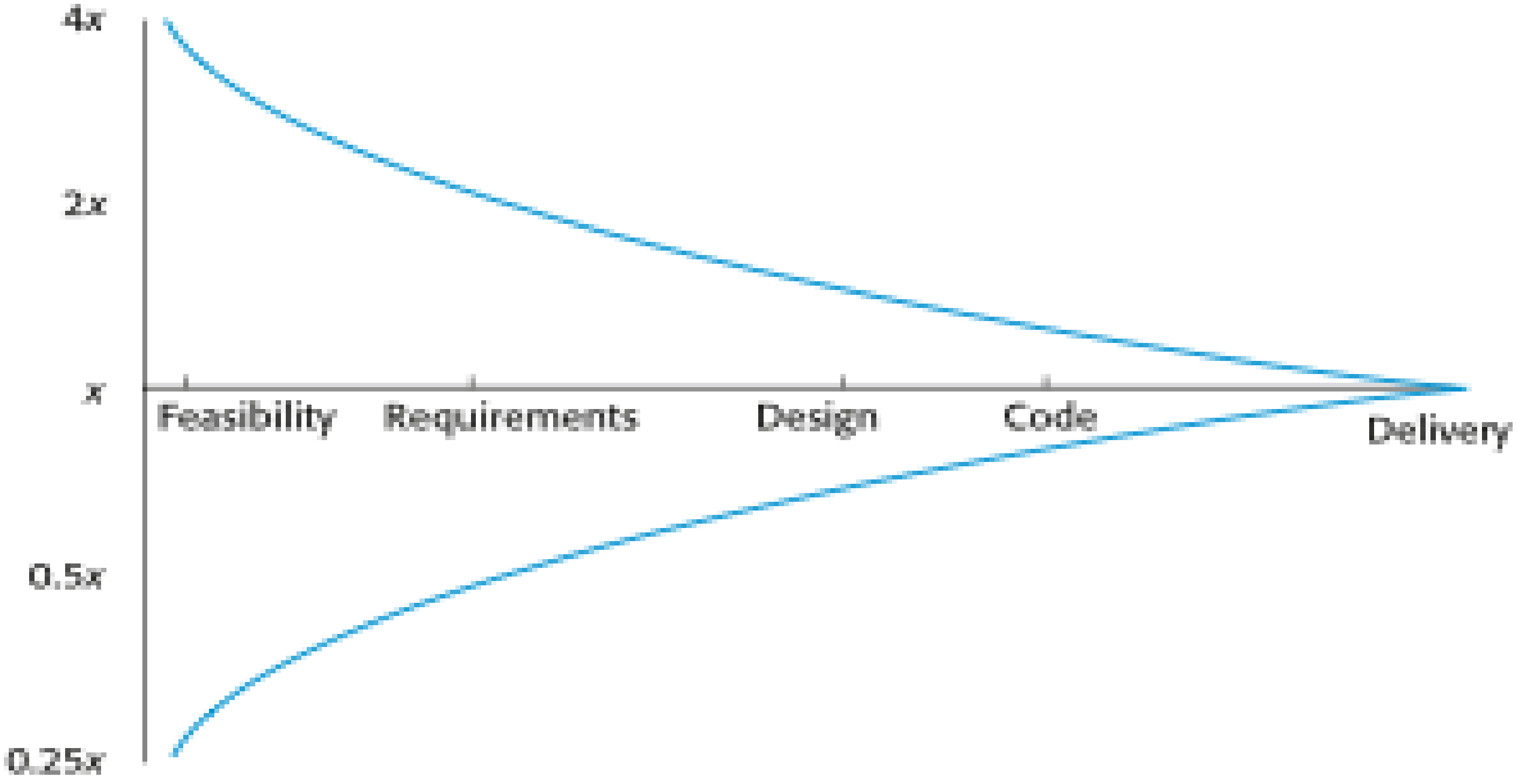
2 Detailed estimate Macro estimate

3 Detailed estimate Macro estimate

4 Detailed estimate Macro estimate

5 Detailed estimate

Estimate uncertainty



Refining Estimates

- Reasons for Adjusting Estimates
  - Interaction costs are hidden in estimates.
  - Normal conditions do not apply.
  - Things go wrong on projects.
  - Changes in project scope and plans.
- Adjusting Estimates
  - Time and cost estimates of specific activities are adjusted as the risks, resources, and situation particulars become more clearly defined.

Estimating: more than predicting the future

- Rules of Thumbs
- Techniques for software

Rules of Thumb for SE –Part 1

- Projects can be broken down by
  - Features
  - Phase
  - Combination of the two
- Every project can be broken down into 10 to 20 tasks for the WBS/PBS
  - Create sub-WBS/PBS as needed
- WBS/PBS key
  - Get it wrong and waste time going down wrong path
- Most accurate estimates rely on prior experience
  - Postmortem reports from past projects key

Rules of Thumb for SE –Part 2

- No estimate is guaranteed to be accurate
  - Things happen
- Goal is not to predict the future but gauge an honest, well-informed opinion of the effort
- Disagreements about estimates are likely due to assumptions
  - Assumptions are used to deal with incomplete information
  - To reduce disagreements → **<u>document assumptions</u>**
  - If assumption incorrect → adjust, explain
    - Keep senior management aware of assumptions
  - Use experts to brainstorm assumptions

Brainstorming Questions

- Are there project goals that are known to the team and not written in any documentation?
- Are there any concepts, terms, definitions that need to be clarified?
- Are there standards that must be met but will be expensive to comply with?
- How will the development of this project differ from that of previous projects? Will there be new task added that were not performed previously?
- Are there technology and architecture decisions that have already been made?
- What changes are likely to occur elsewhere in the organization that could cause this estimate to be inaccurate?
- Are there any issues that the team is know to disagree on that will affect the project?

Team Motivation

- Brainstorming brings team together
- Begins the process of ownership
- Eliminates distrust
- Reduces —I didn‘t estimate it but have to live with it”
- Provides accurate estimates in the future
- Allows team pressure to not —pad” estimate
- Goal: reach common understanding between engineers, managers, stakeholders.

Estimation Techniques-Part 1

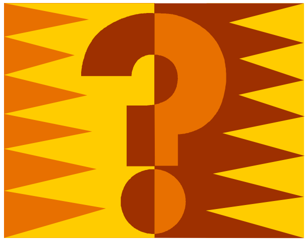
- Delphi
  - Moderator + estimation team
  - RAND corporation
  - Team corrects one another in a way that helps avoid errors and poor estimation
- PROBE
  - Proxy Based Estimating
    - CMU as part of personal software process (discipline that helps individual software engineers monitor, test, and improve their own work)
  - Database of historical components of types and size
    - Type examples: calculation, data, logic, etc
    - Size examples: very small, small….very large
  - Uses linear regression

Estimation Techniques-Part 2

- COCOMO II
  - Constructive Cost Model
    - Barry Boehm
    - Empirical study of 63 software projects
    - Statisitical analysis
  - 15 cost drivers
  - Variables that must be entered into model
    - e.g. computer, personnel, project attributes

Estimation Techniques-Part 3

- The Planning Game
  - XP (Extreme Programming)
  - Negotiate between engineering team and stakeholders
  - Create emotional distance by treating as game where playing pieces are ―user stories”
  - User stories get assigned a value and put into production eventually
  - Combines estimation with scope creation
  - Highly iterative

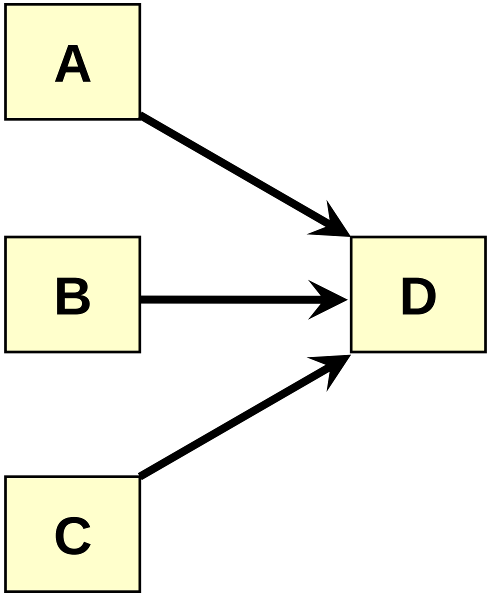


Developing the Project Plan

- The Project Network
  - A flow chart that graphically depicts the sequence, interdependencies, and start and finish times of the project job plan of activities that is the ***critical path*** through the network
    - Provides the basis for scheduling labor and equipment
    - Provides an estimate of the project‘s duration
    - Provides a basis for budgeting cash flow
    - Highlights activities that are —critical” and should not be delayed
    - Help managers get and stay on plan

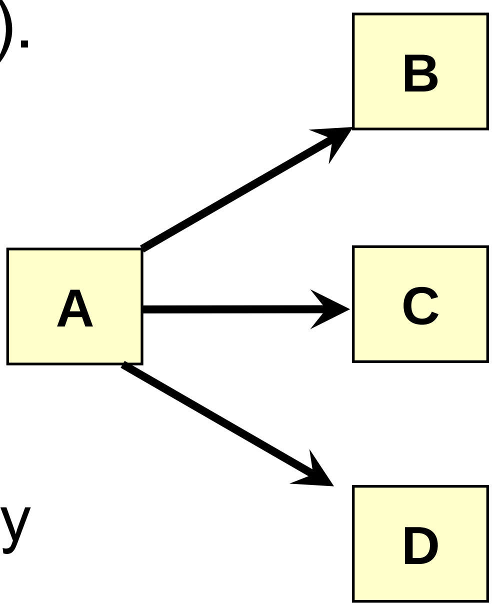
Constructing a Project Network

- Terminology
  - □ **Activity:** an element of the project that requires time.
  - □ **Merge activity:** an activity that has two or more preceding activities on which it depends.
  - □ **Parallel (concurrent) activities:** Activities that can occur independently and, if desired, not at the same time.



Constructing a Project Network (cont‘d)

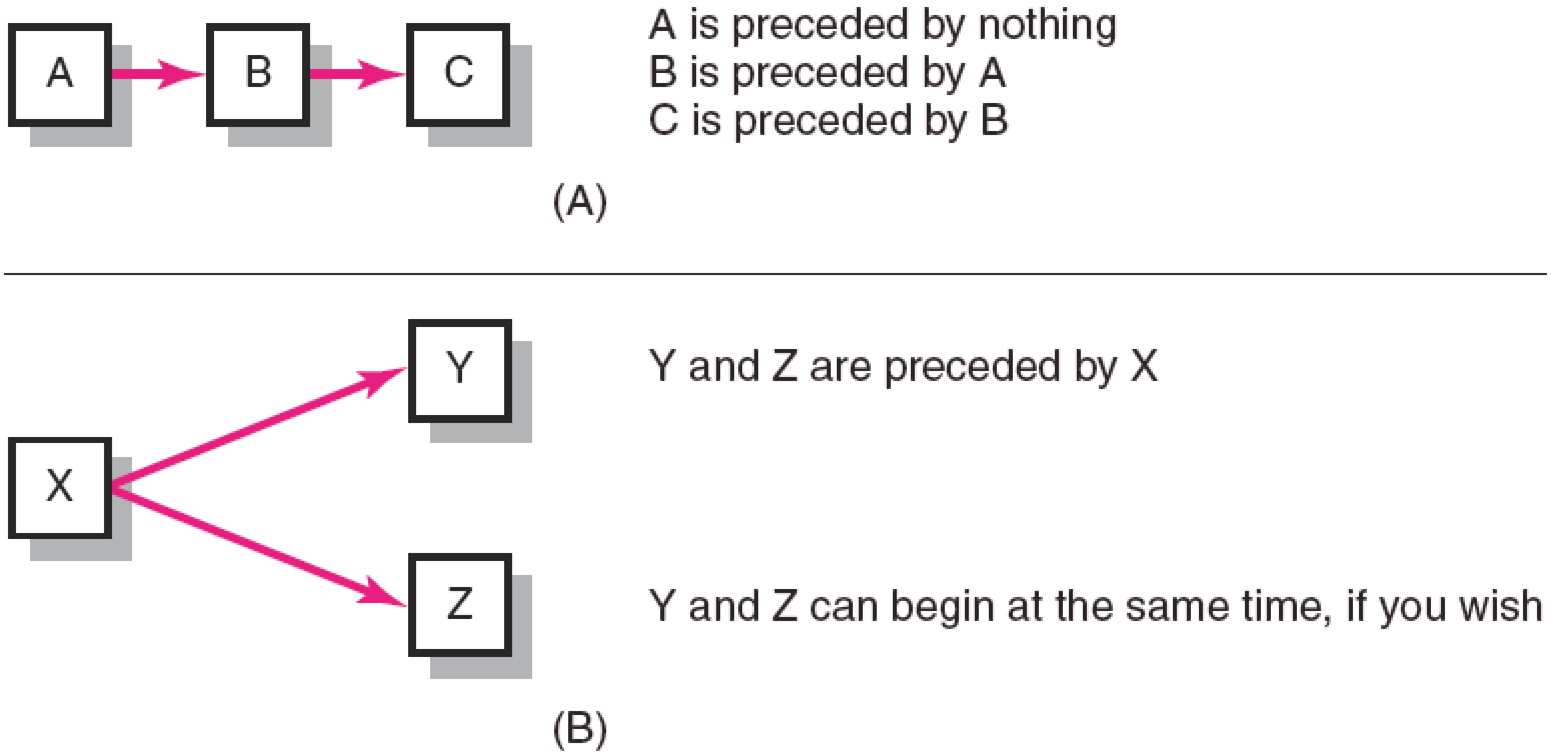
- Terminology
  - **Event:** a point in time when an activity is started or completed. It does not consume time.
  - **Burst activity:** an activity that has more than one activity immediately following it (more than one dependency arrow flowing from it).
- Two Approaches
  - Activity-on-Node (AON)
    - Uses a node to depict an activity
  - Activity-on-Arrow (AOA)
    - Uses an arrow to depict an activity



Basic Rules to Follow in Developing Project Networks

- Networks typically flow from left to right.
- An activity cannot begin until all of its dependent prior activities are complete.
- Arrows indicate precedence and flow and can cross over each other.
- Identify each activity with a unique number; this number must be greater than its predecessors.
- Looping is not allowed.
- Conditional statements are not allowed.
- Use common start and stop nodes.

Activity-on-Node Fundamentals



A is preceded by nothing
B is preceded by A
C is preceded by B

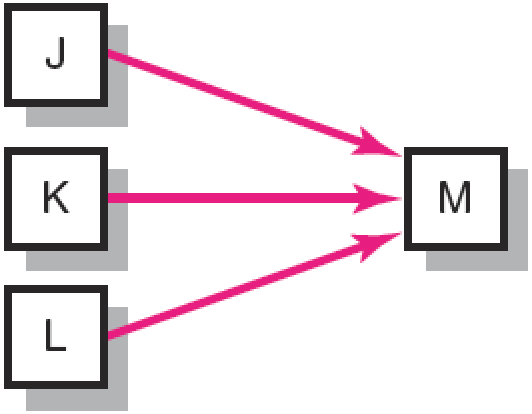
(A)

Y and Z are preceded by X

Y and Z can begin at the same time, if you wish

(B)

Activity-on-Node Fundamentals (cont‘d)

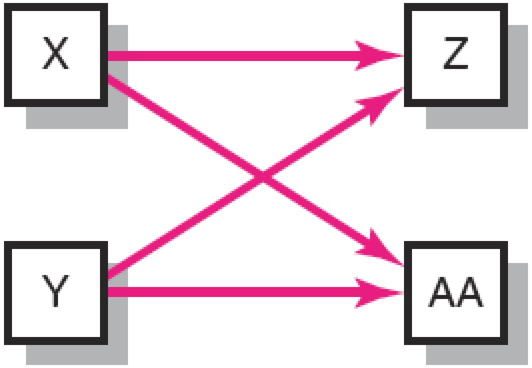


J, K, & L can all begin at the same time, if you wish (they need not occur simultaneously)

but

All (J, K, L) must be completed before M can begin

(C)

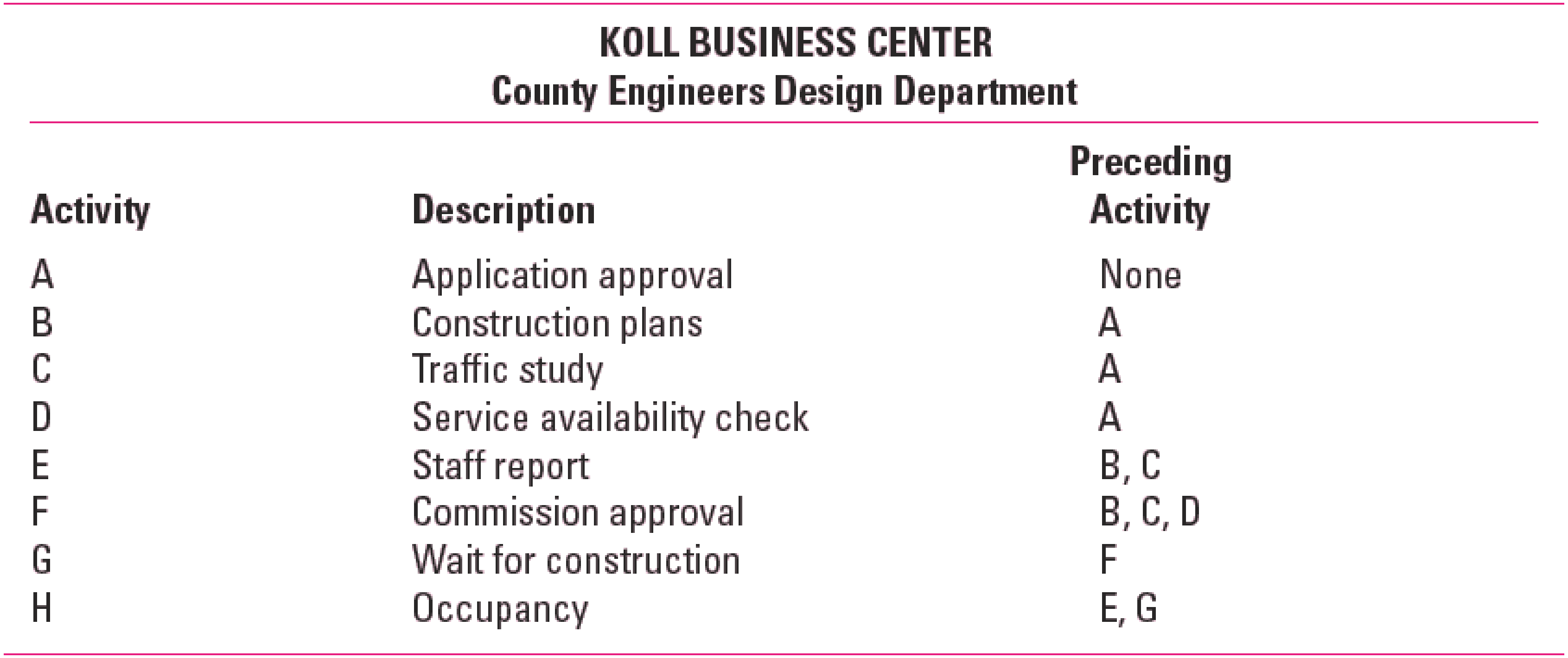


Z is preceded by X and Y

AA is preceded by X and Y

(D)

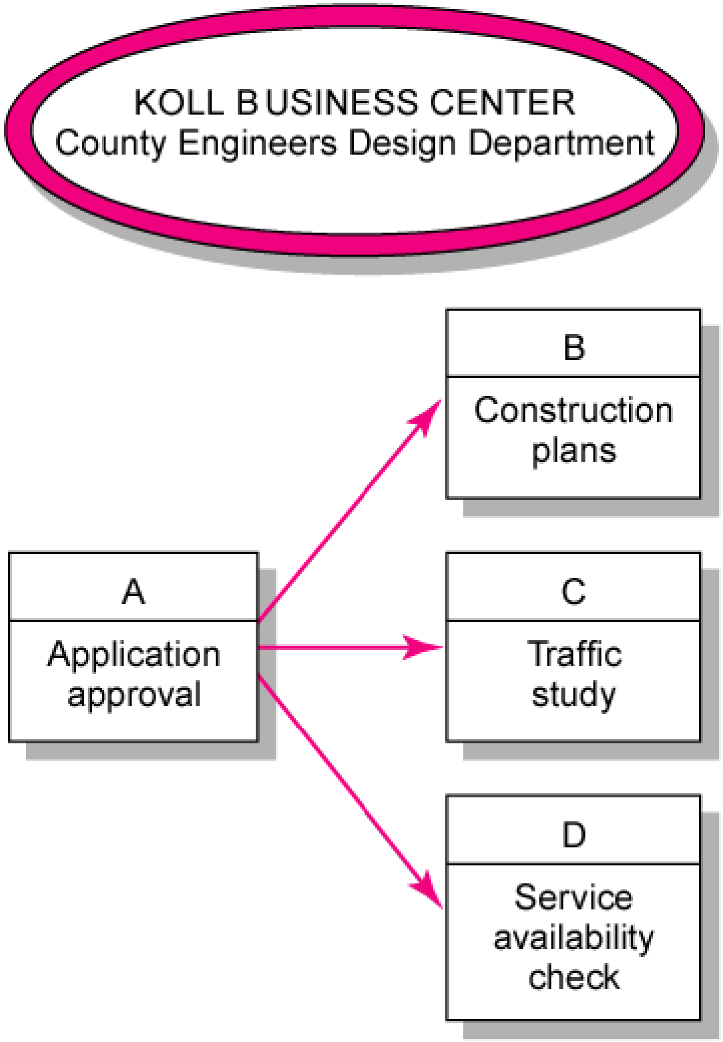
Network Information



KOLL BUSINESS CENTER
County Engineers Design Department

Activity	Description	Preceding Activity
A	Application approval	None
B	Construction plans	A
C	Traffic study	A
D	Service availability check	A
E	Staff report	B, C
F	Commission approval	B, C, D
G	Wait for construction	F
H	Occupancy	E, G

Koll Business Center—Partial Network



KOLL BUSINESS CENTER
County Engineers Design Department

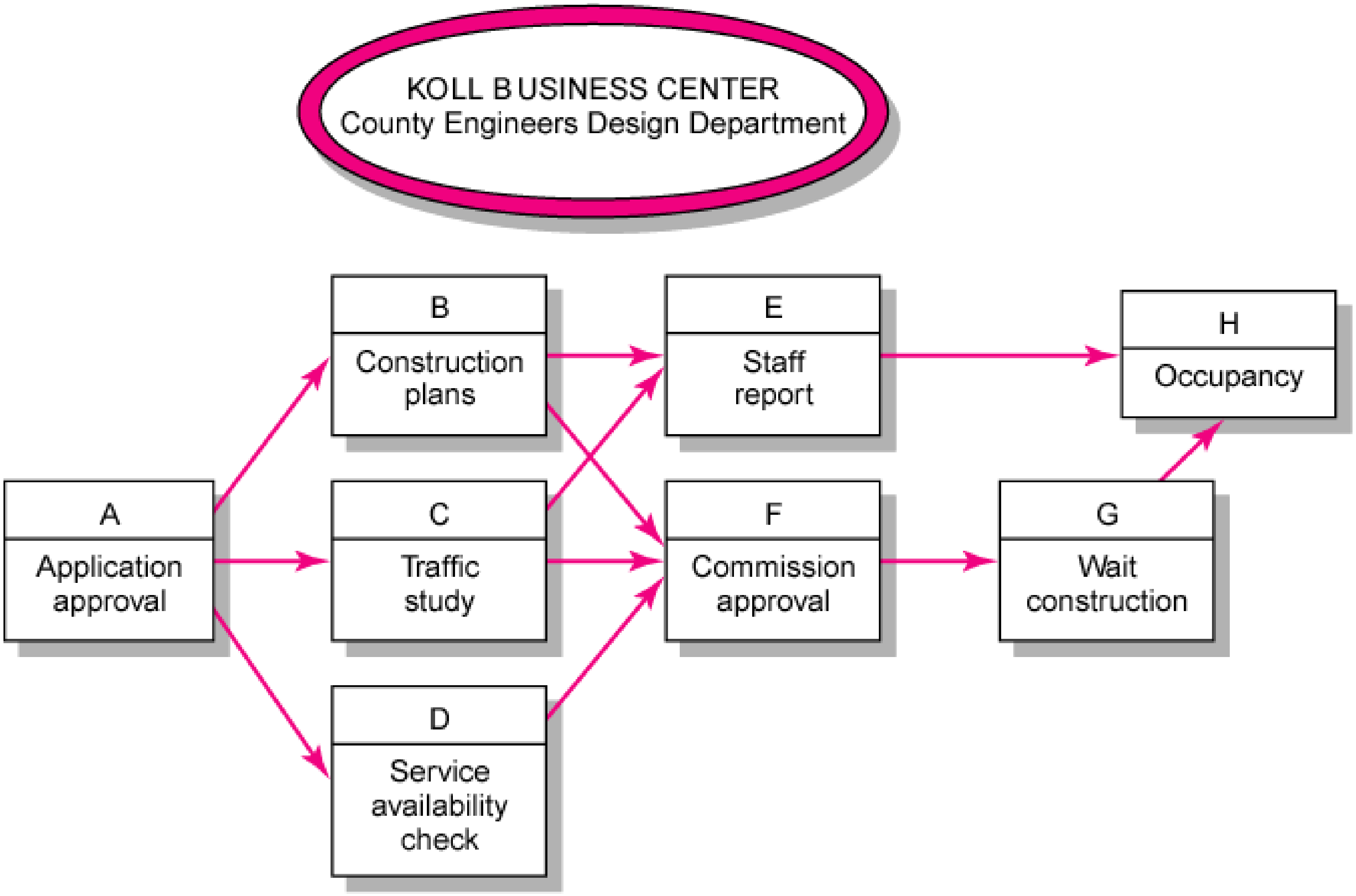
A
Application approval

B
Construction plans

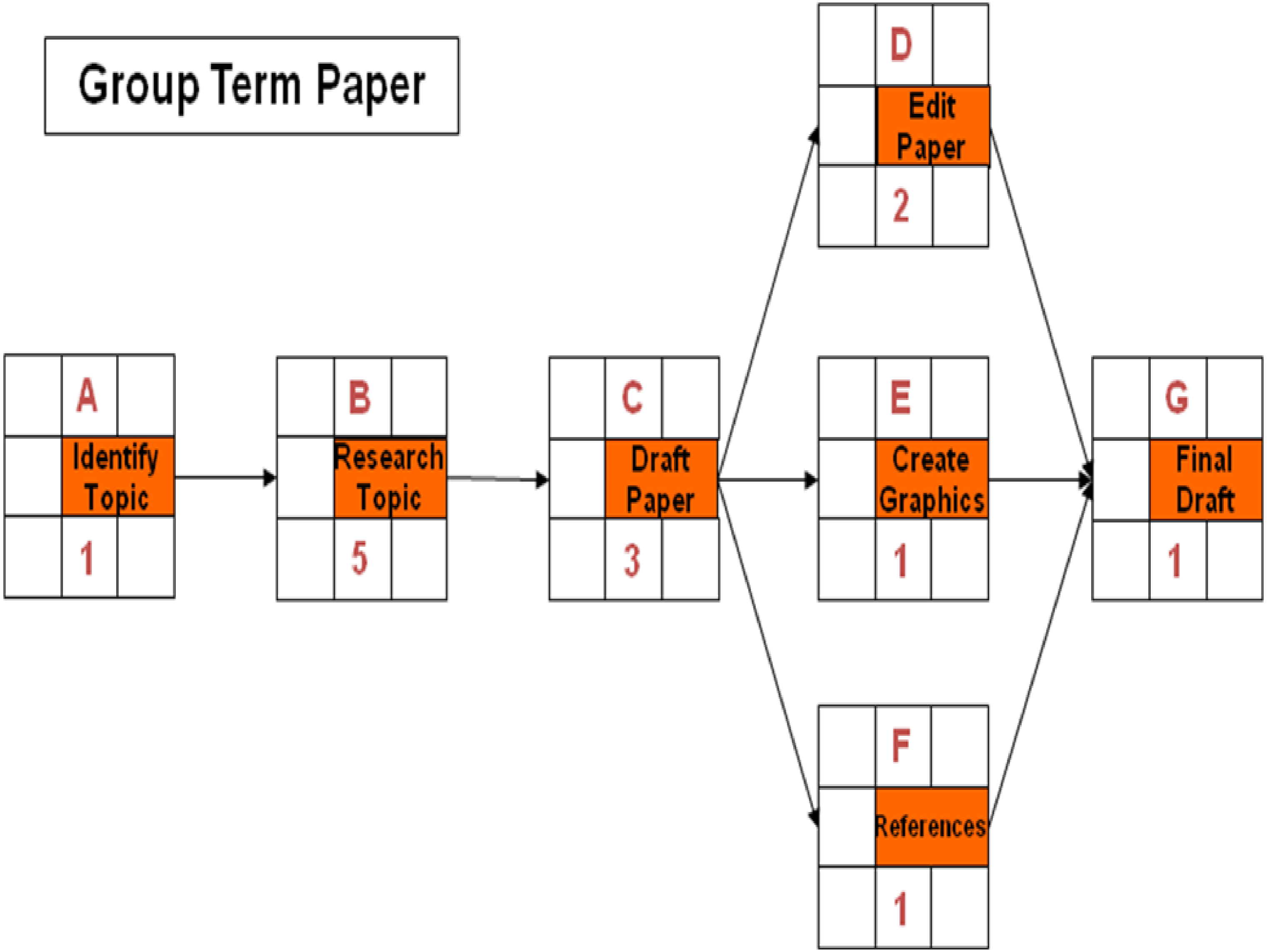
C
Traffic study

D
Service availability check

Koll Business Center—Complete Network



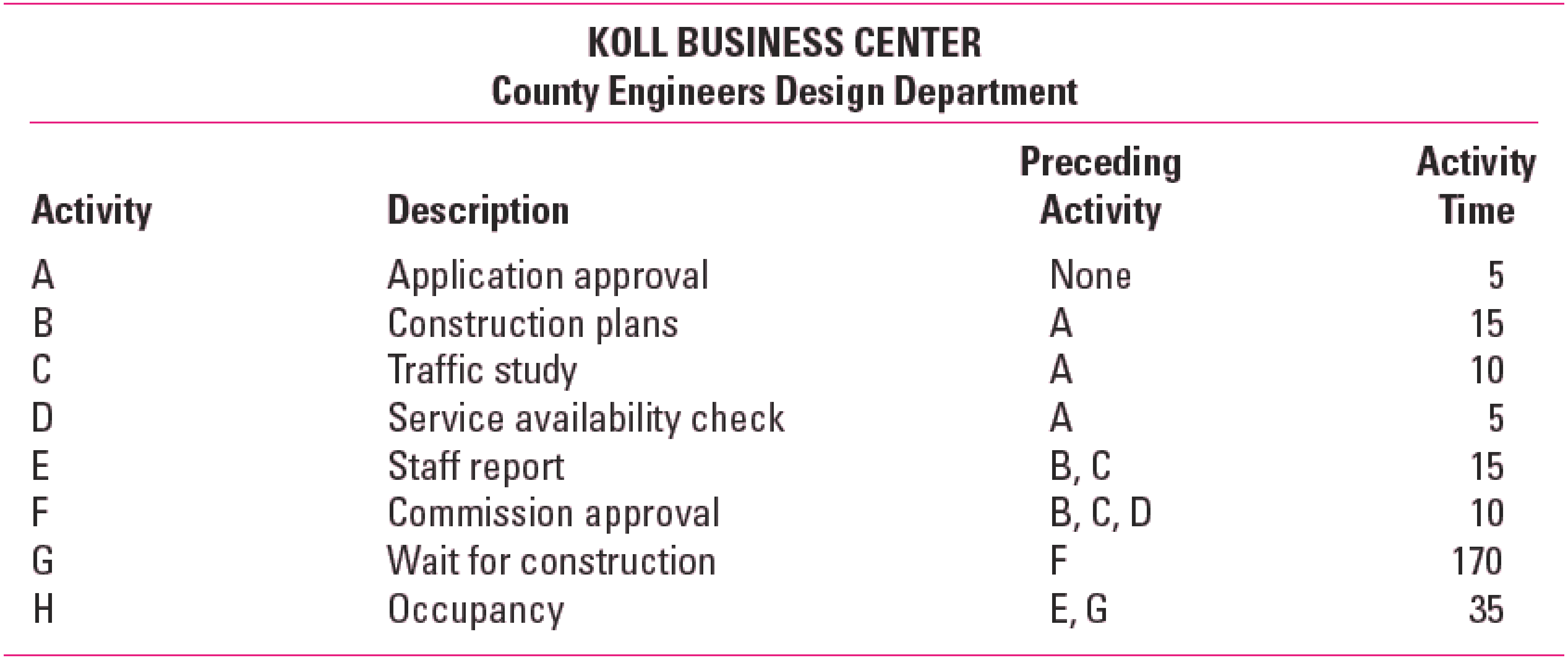
Group Term Paper



Network Computation Process

- Forward Pass—Earliest Times
  - □ How soon can the activity start? (early start—ES)
  - □ How soon can the activity finish? (early finish—EF)
  - □ How soon can the project finish? (expected time—ET)
- Backward Pass—Latest Times
  - □ How late can the activity start? (late start—LS)
  - □ How late can the activity finish? (late finish—LF)
  - □ Which activities represent the critical path?
  - □ How long can it be delayed? (slack or float—SL)

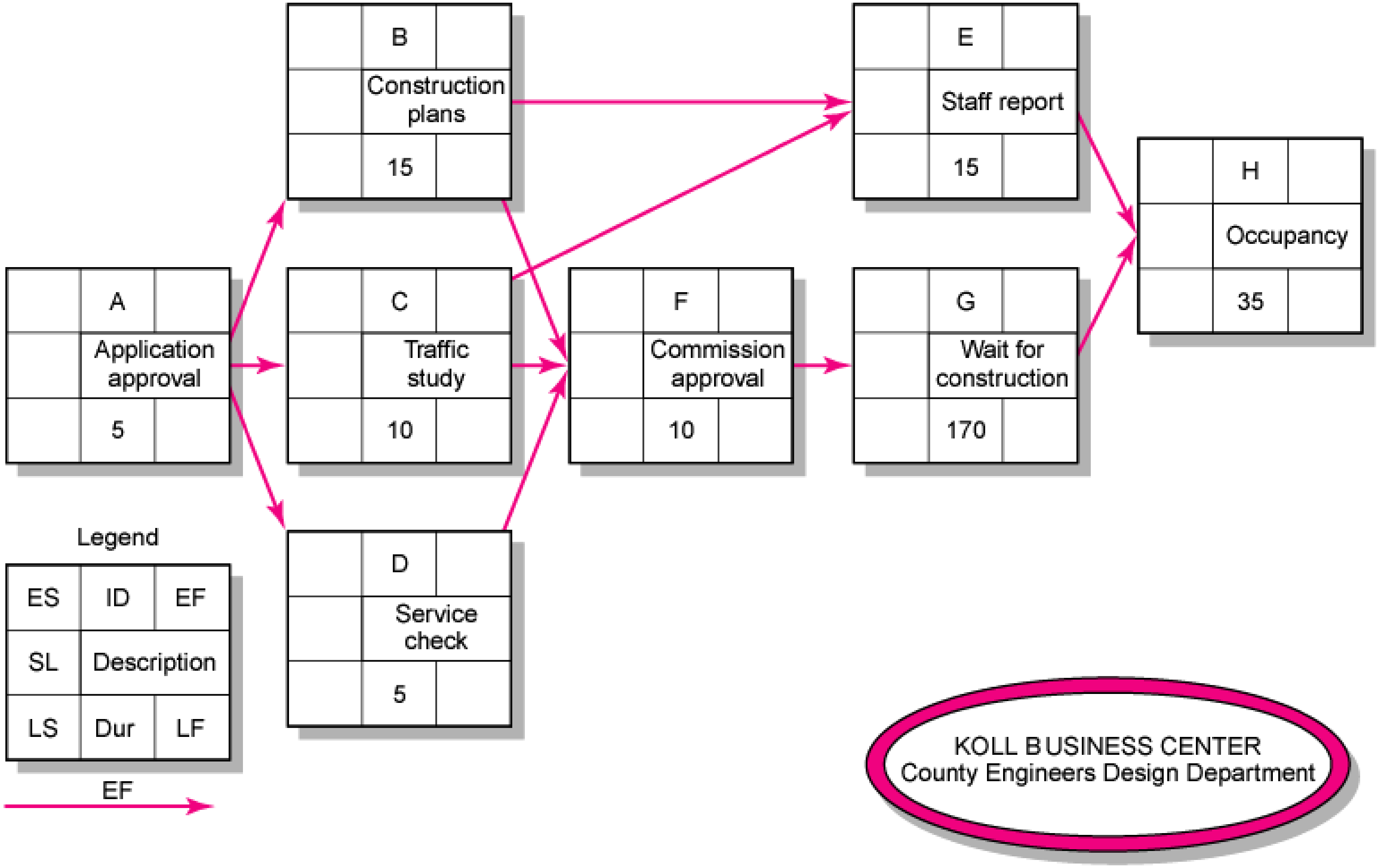
Network Information



KOLL BUSINESS CENTER
County Engineers Design Department

Activity	Description	Preceding Activity	Activity Time
A	Application approval	None	5
B	Construction plans	A	15
C	Traffic study	A	10
D	Service availability check	A	5
E	Staff report	B, C	15
F	Commission approval	B, C, D	10
G	Wait for construction	F	170
H	Occupancy	E, G	35

Activity-on-Node Network



A Application approval 5

B Construction plans 15

C Traffic study 10

D Service check 5

E Staff report 15

F Commission approval 10

G Wait for construction 170

H Occupancy 35

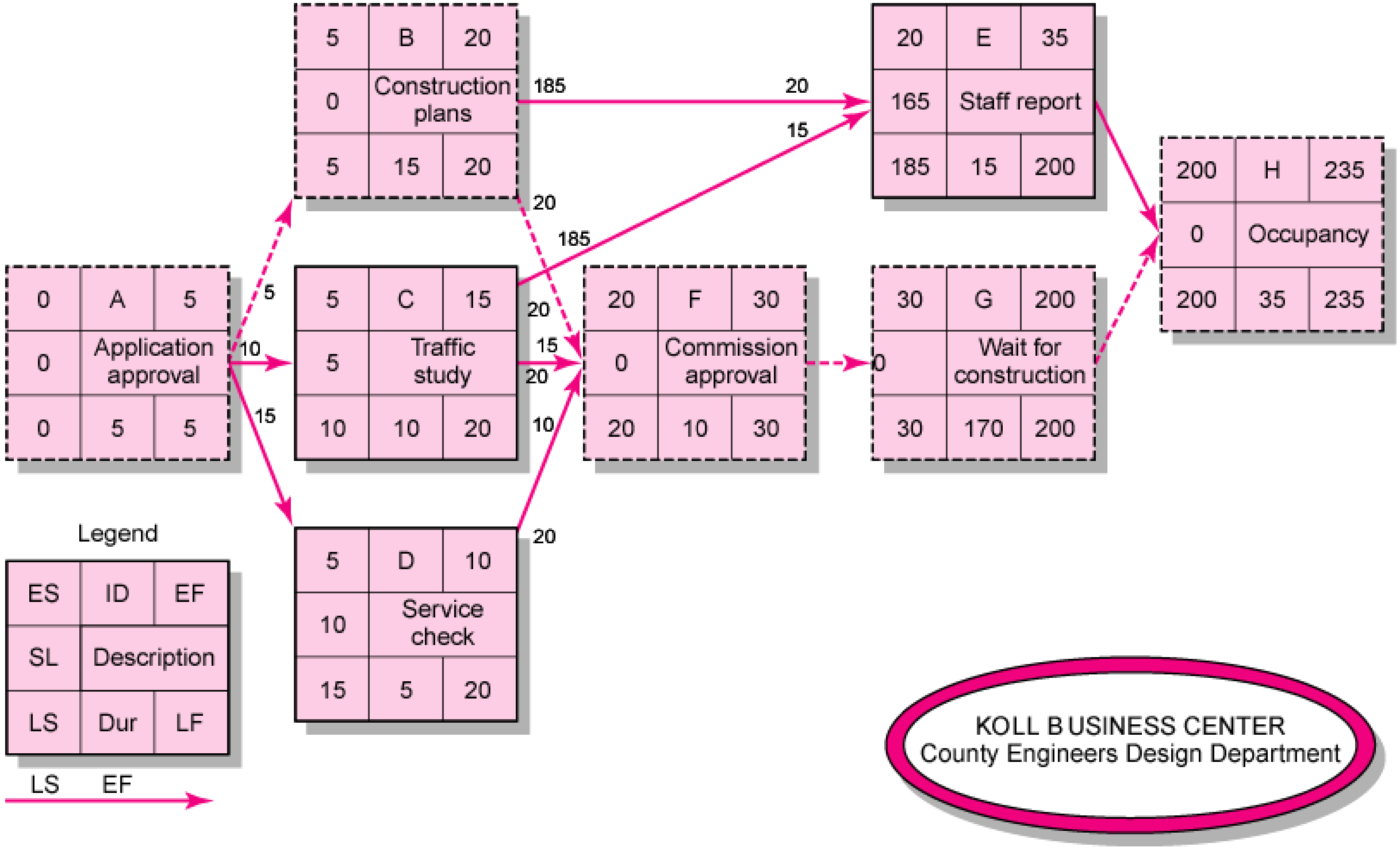
Legend

ES	ID	EF
SL	Description	
LS	Dur	LF

EF

KOLL BUSINESS CENTER
County Engineers Design Department

Activity-on-Node Network with Slack



5 B 20
0 Construction plans
5 15 20

20 E 35
165 Staff report
185 15 200

200 H 235
0 Occupancy
200 35 235

0 A 5
0 Application approval
0 5 5

5 C 15
5 Traffic study
10 10 20

20 F 30
0 Commission approval
20 10 30

30 G 200
0 Wait for construction
30 170 200

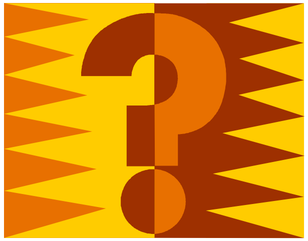
5 D 10
10 Service check
15 5 20

Legend

ES ID EF
SL Description
LS Dur LF

LS EF

KOLL BUSINESS CENTER
County Engineers Design Department

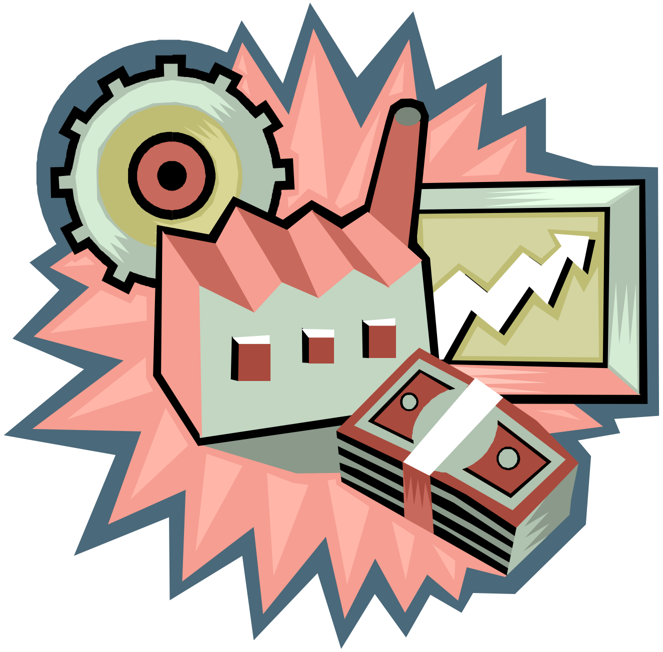


The Resource Problem

- Resources and Priorities
    - Project network times are not a schedule until resources have been assigned.
        - The implicit assumption is that resources will be available in the required amounts when needed.
        - Adding new projects requires making realistic judgments of resource availability and project durations.
- Resource-Constrained Scheduling
    - Resource leveling (or smoothing) involves attempting to even out demands on resources by using slack (delaying noncritical activities) to manage resource utilization.

Kinds of Resource Constraints

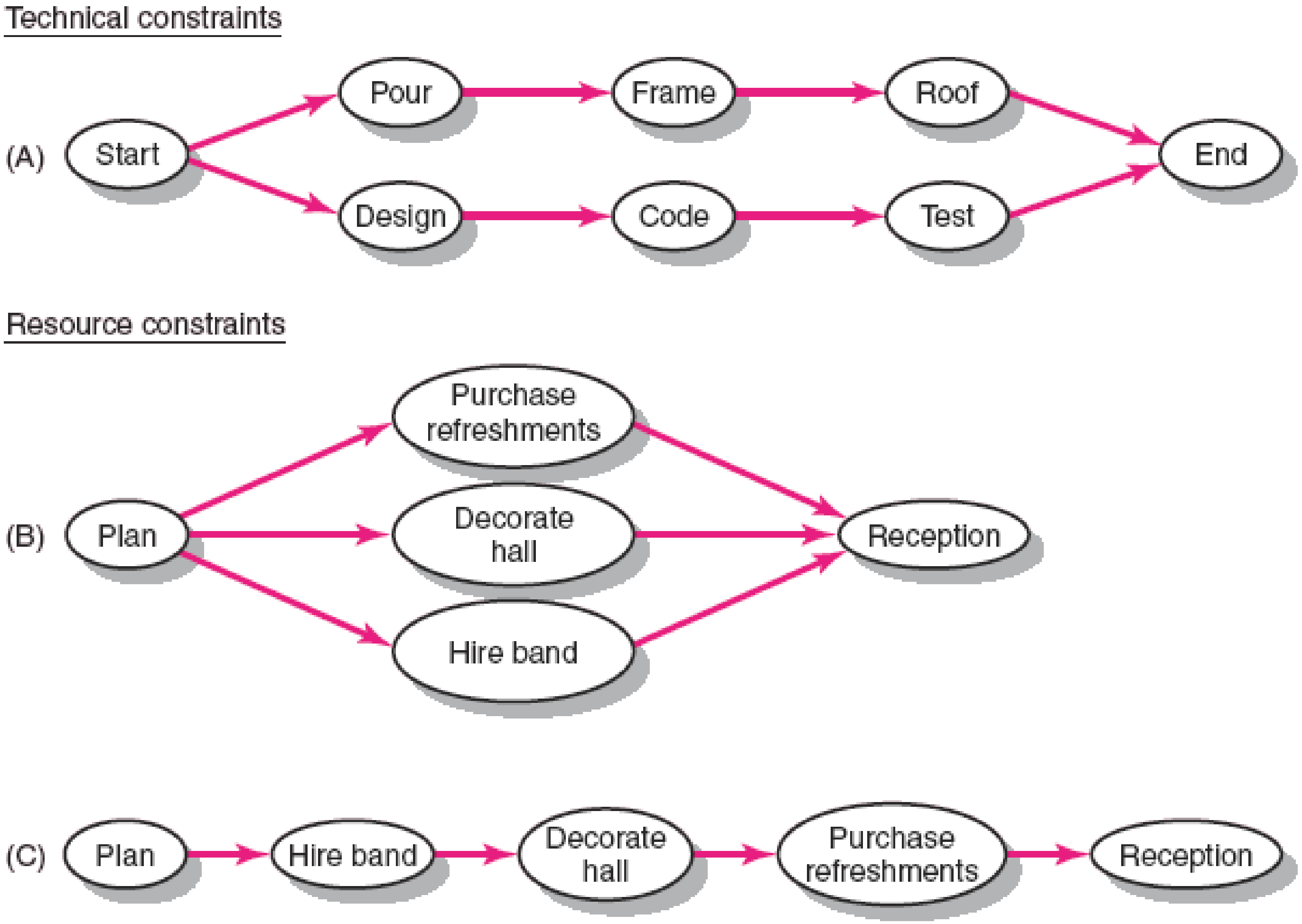
- **People**
- **Materials**
- **Equipment**
- **Working Capital**



Types of Project Constraints

- Technical or Logic Constraints
  - Constraints related to the networked sequence in which project activities must occur
- Resource Constraints
  - The absence, shortage, or unique interrelationship and interaction characteristics of resources that require a particular sequencing of project activities

Constraint Examples



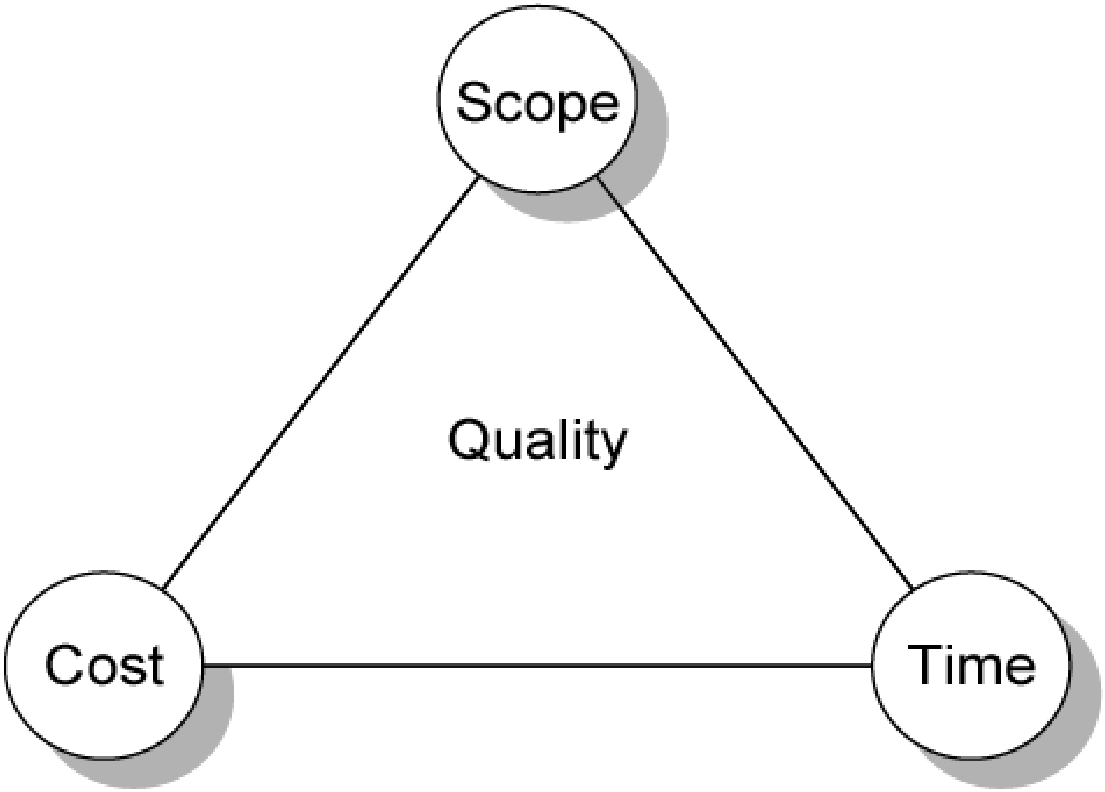
Resource Allocation Methods

- Limiting Assumptions
  - Splitting activities is not allowed—once an activity is start, it is carried to completion.
  - Level of resource used for an activity cannot be changed.
  - Activities with the most slack pose the least risk.
  - Reduction of flexibility does not increase risk.
  - The nature of an activity (easy, complex) doesn‘t increase risk.

Classification of a Scheduling Problem

- Time Constrained Project
  - A project that must be completed by an imposed date
    - Time is fixed, resources are flexible: additional resources are required to ensure project meets schedule.
- Resource Constrained Project
  - A project in which the level of resources available cannot be exceeded
    - Resources are fixed, time is flexible: inadequate resources will delay the project.

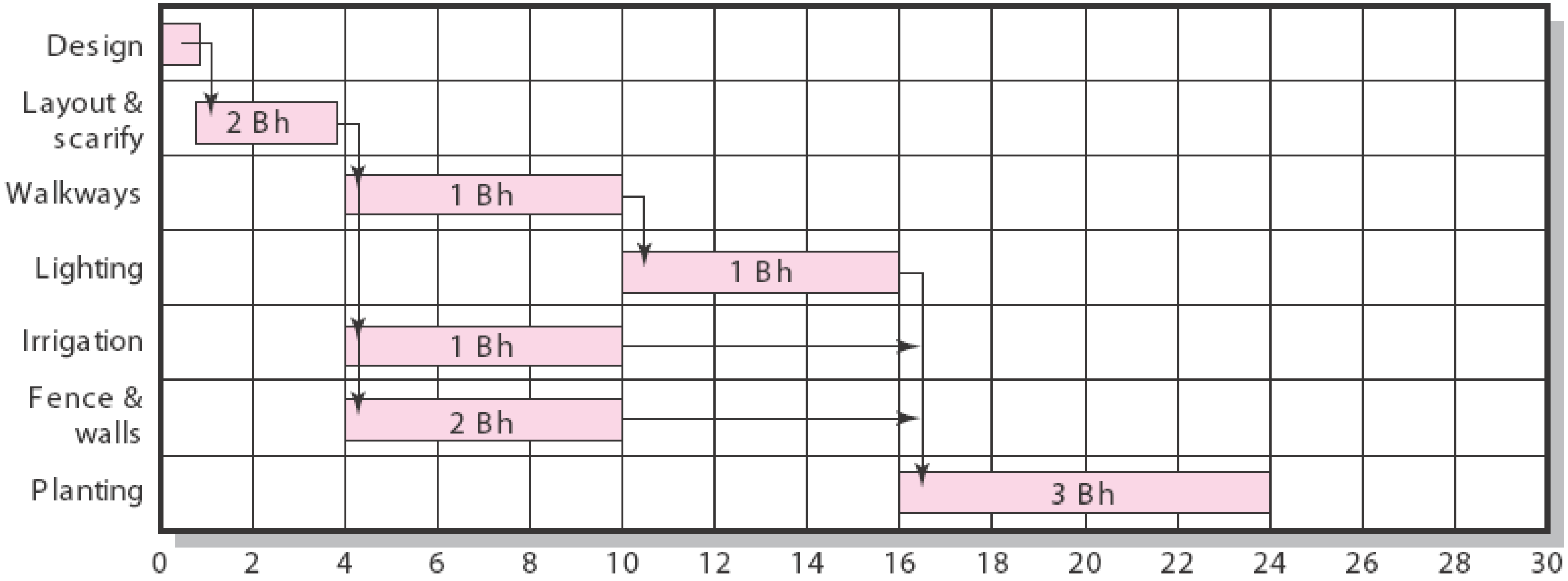
Project Management Trade-offs



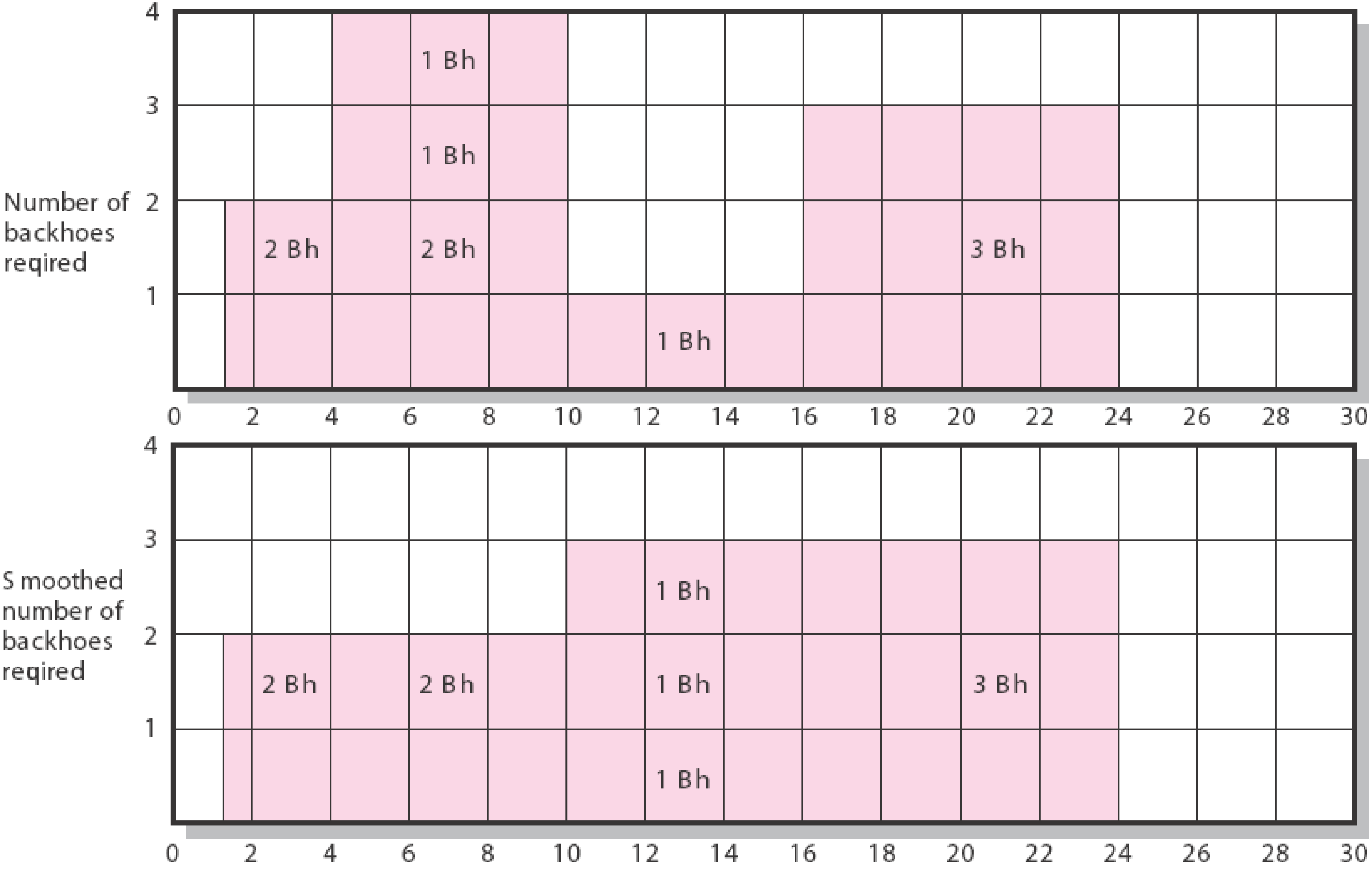
Time-Constrained Projects

- Projects that must be completed by an imposed date
- Require the use of leveling techniques that focus on balancing or smoothing resource demands by using positive slack (delaying noncritical activities) to manage resource utilization over the duration of the project
  - Peak resource demands are reduced.
  - Resources over the life of the project are reduced.
  - Fluctuation in resource demand is minimized.

Botanical Garden



Botanical Garden (cont‘d)

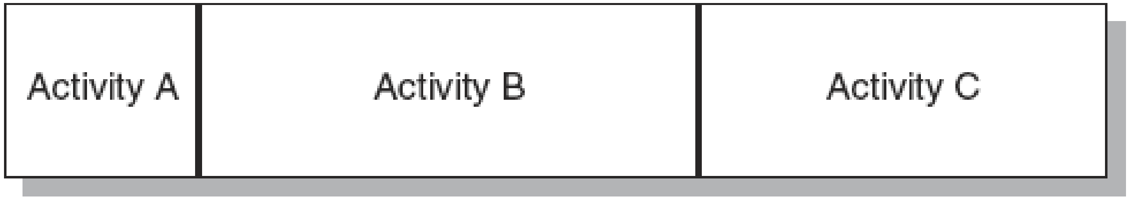


Splitting/Multitasking

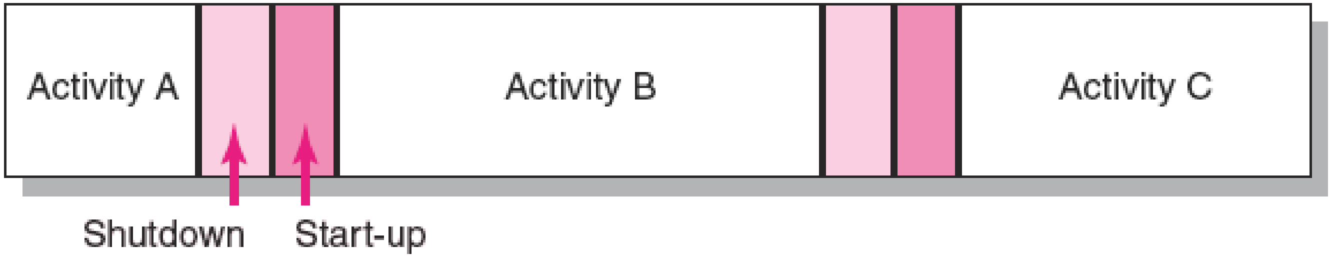
- Splitting/Multitasking
  - □ A scheduling technique use to get a better project schedule and/or increase resource utilization
    - Involves interrupting work on an activity to employ the resource on another activity, then returning the resource to finish the interrupted work
    - Is feasible when startup and shutdown costs are low
    - Is considered the major reason why projects fail to meet schedule

Splitting/Multitasking

Activity duration without splitting



Activity duration split into three segments—A, B, C



Activity duration split with shutdown and start-up

Reducing Project Duration

- Time Is Money: Cost-Time Tradeoffs
  - Reducing the time of a critical activity usually incurs additional direct costs.
    - Cost-time solutions focus on reducing (crashing) activities on the critical path to shorten overall duration of the project.
  - Reasons for imposed project duration dates:
    - Customer requirements and contract commitments
    - Time-to-market pressures
    - Incentive contracts (bonuses for early completion)
    - Unforeseen delays
    - Overhead and goodwill costs
    - Pressure to move resources to other projects

Inclass Group Activity

- Brainstorm list of ways to shorten duration
- Write on board

Options for Accelerating Project Completion

- If resources are not constrained
  - Adding Resources
  - Outsourcing Project Work
  - Scheduling Overtime
  - Establishing a Core Project Team
  - Do It Twice—Fast and Correctly

- If resources are constrained
  - Fast-Tracking
  - Reducing Project Scope
  - Compromise Quality

R
I
S
K

InClass Group Exercise

- Work in teams of 3 to 4
- ESDTU is a company that is developing a new software. They estimate it will take 24 weeks to complete. The project will start December 1st.
- Brainstorm an extensive list of things that could go wrong with the above project
- Write your list on the board

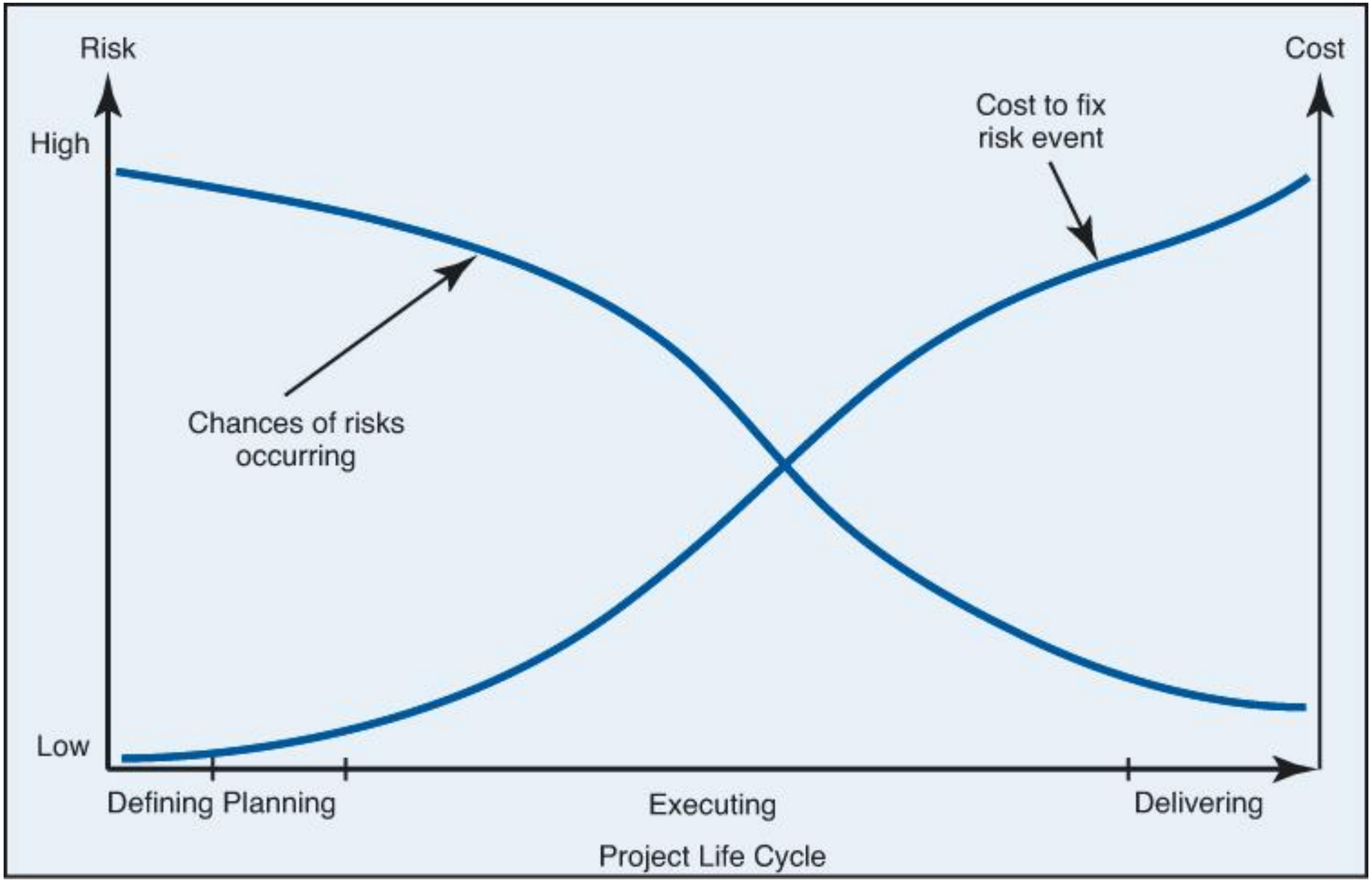
Risk Management Process

- Risk
  - An uncertain event that, if it occurs, has a positive or negative effect on project objectives
  - Uncertain or chance events that planning can not overcome or control.
- Risk Management
  - A proactive attempt to recognize and manage internal events and external threats that affect the likelihood of a project‘s success
    - What can go wrong (risk event)
    - How to minimize the risk event‘s impact (consequences)
    - What can be done before an event occurs (anticipation)
    - What to do when an event occurs (contingency plans)

Thinking About Risks

- What kinds of things could go wrong with your class project in this class?
- What will you do if one of them does?
- How likely are they to happen?

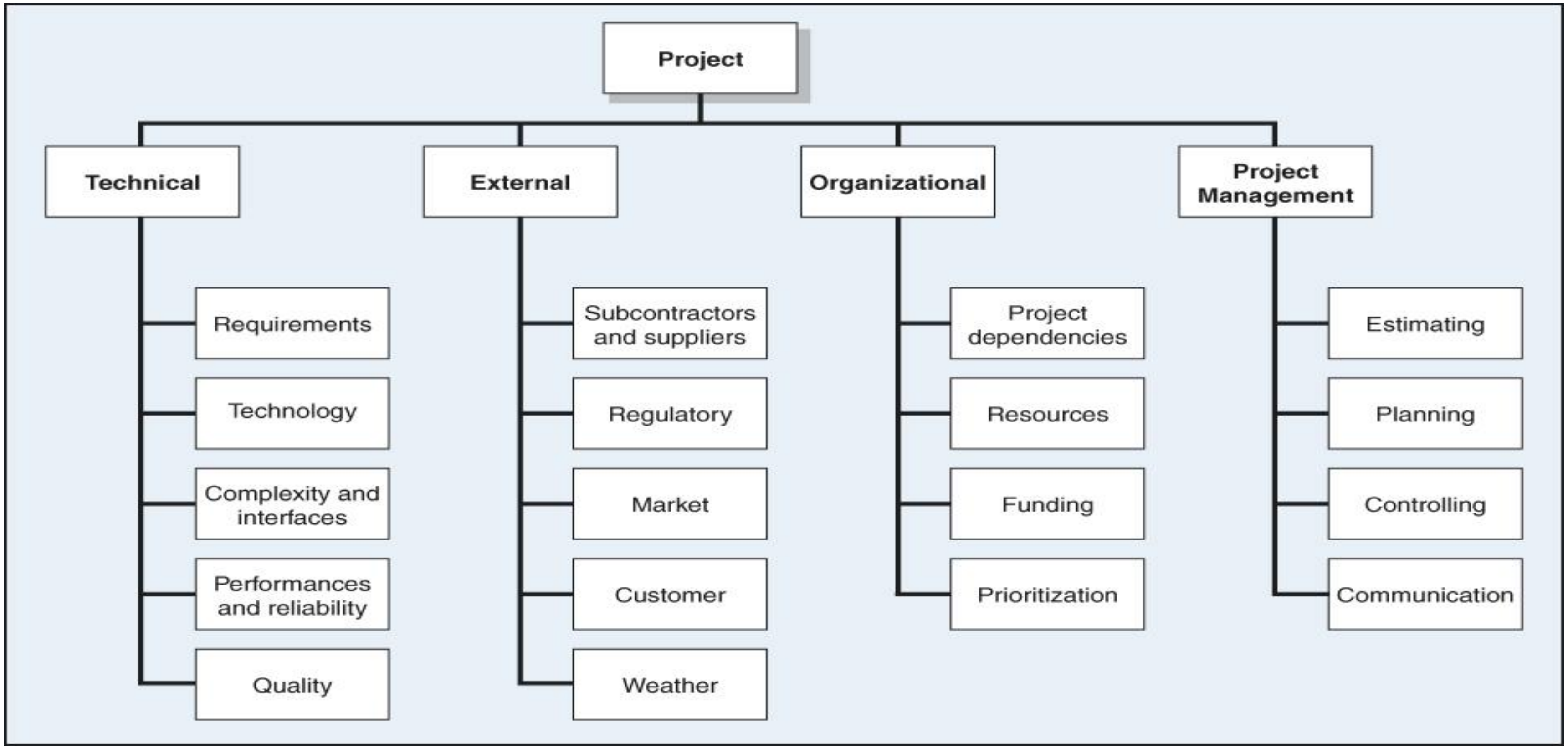
The Risk Event Graph



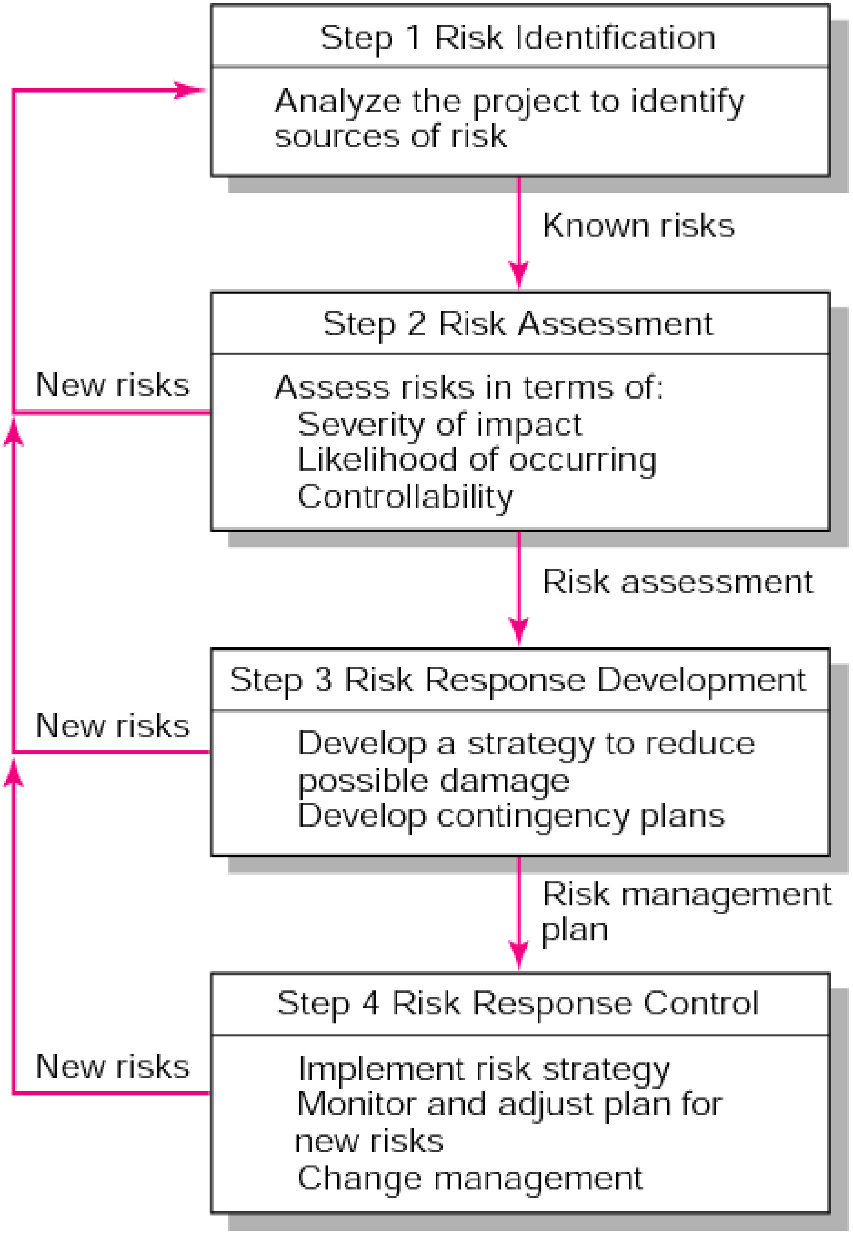
Risk Management‘s Benefits

- A proactive rather than reactive approach
- Reduces surprises and negative consequences
- Prepares the project manager to take advantage of appropriate risks
- Provides better control over the future
- Improves chances of reaching project performance objectives within budget and on time

Risk Breakdown Structure



The Risk Management Process



Step 1 Risk Identification

Analyze the project to identify sources of risk

Known risks

Step 2 Risk Assessment

Assess risks in terms of:
Severity of impact
Likelihood of occurring
Controllability

Risk assessment

Step 3 Risk Response Development

Develop a strategy to reduce possible damage
Develop contingency plans

Risk management plan

Step 4 Risk Response Control

Implement risk strategy
Monitor and adjust plan for new risks
Change management

New risks

New risks

New risks

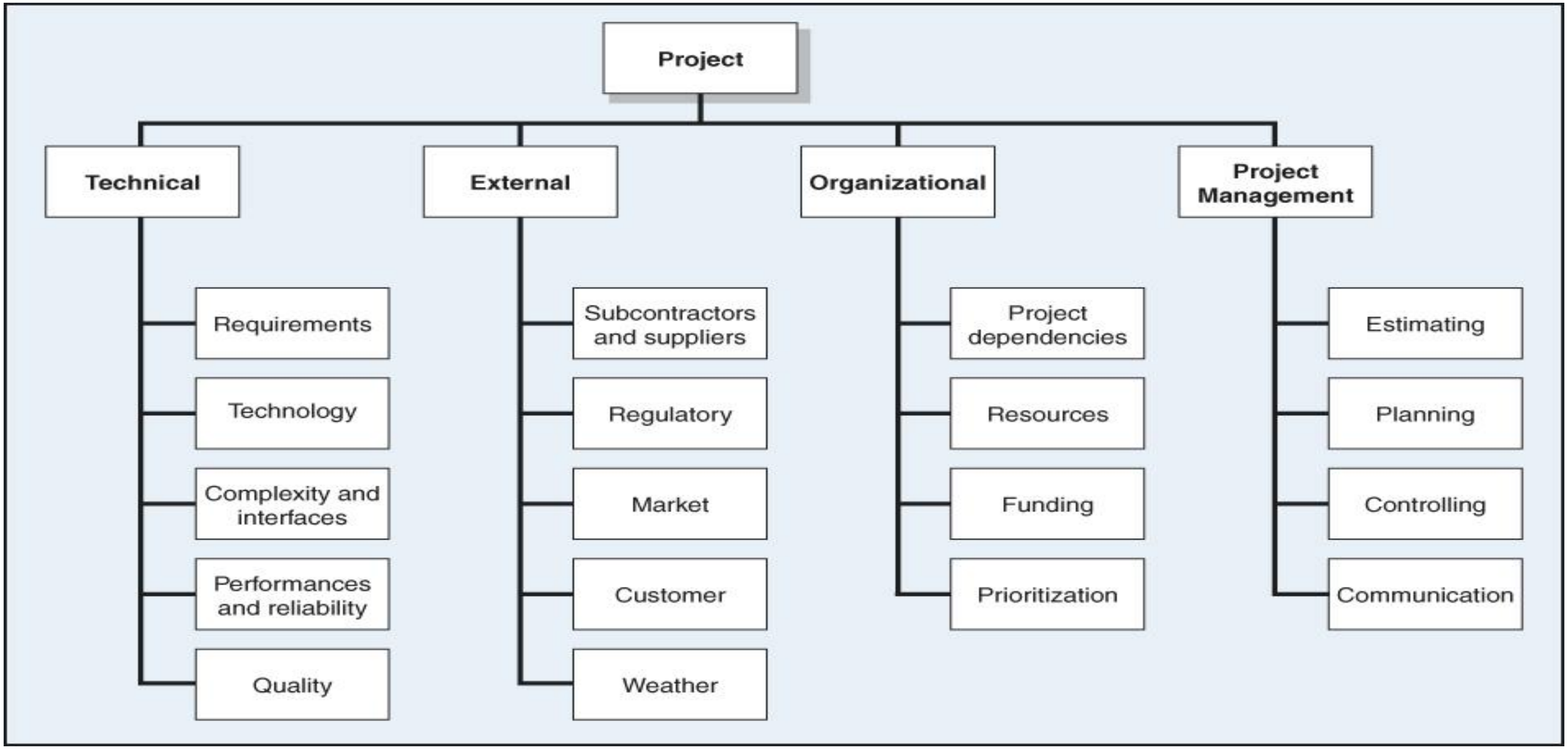
Managing Risk

- Step 1: Risk Identification
  - Generate a list of possible risks through brainstorming, problem identification and risk profiling.
    - Macro risks first, then specific events

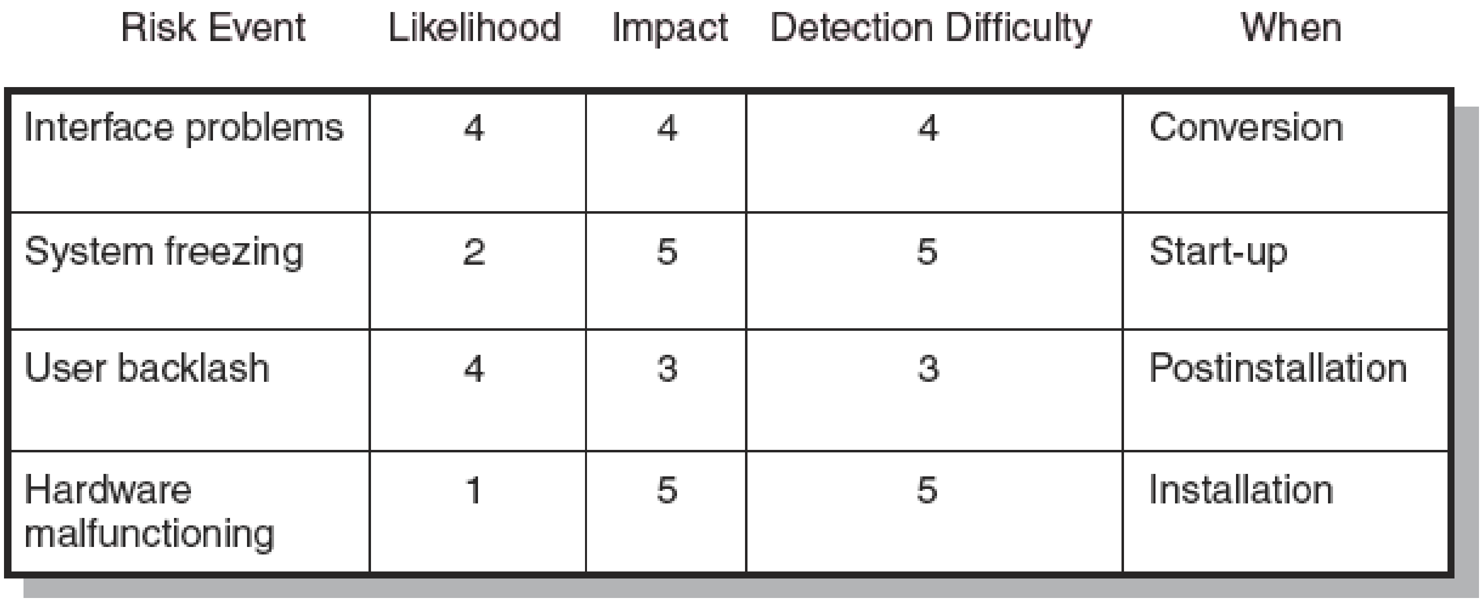
Managing Risk

- Step 2: Risk Assessment
  - Scenario analysis
  - Risk assessment matrix
  - Probability analysis

Risk Breakdown Structure

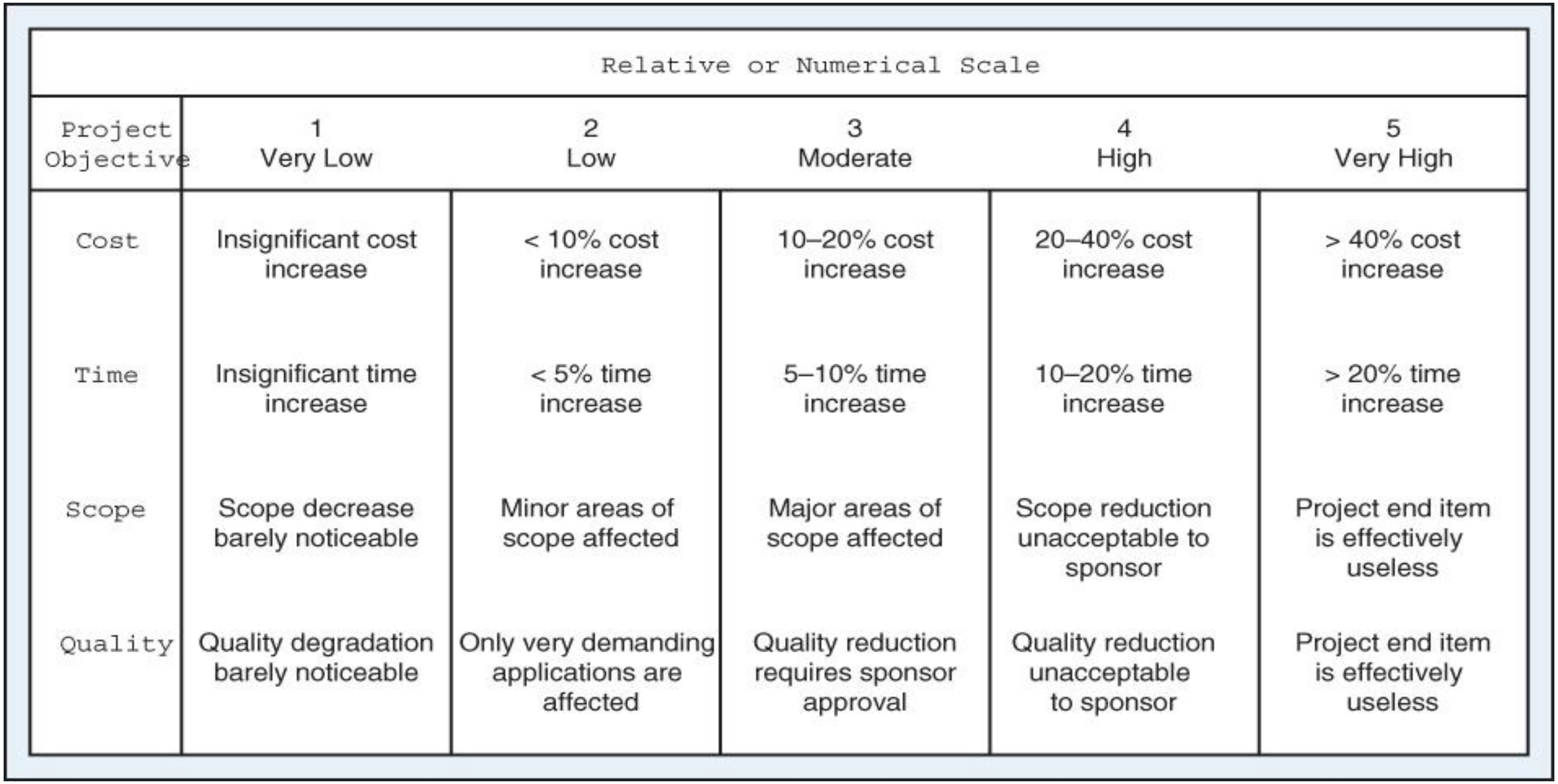


Risk Assessment Form



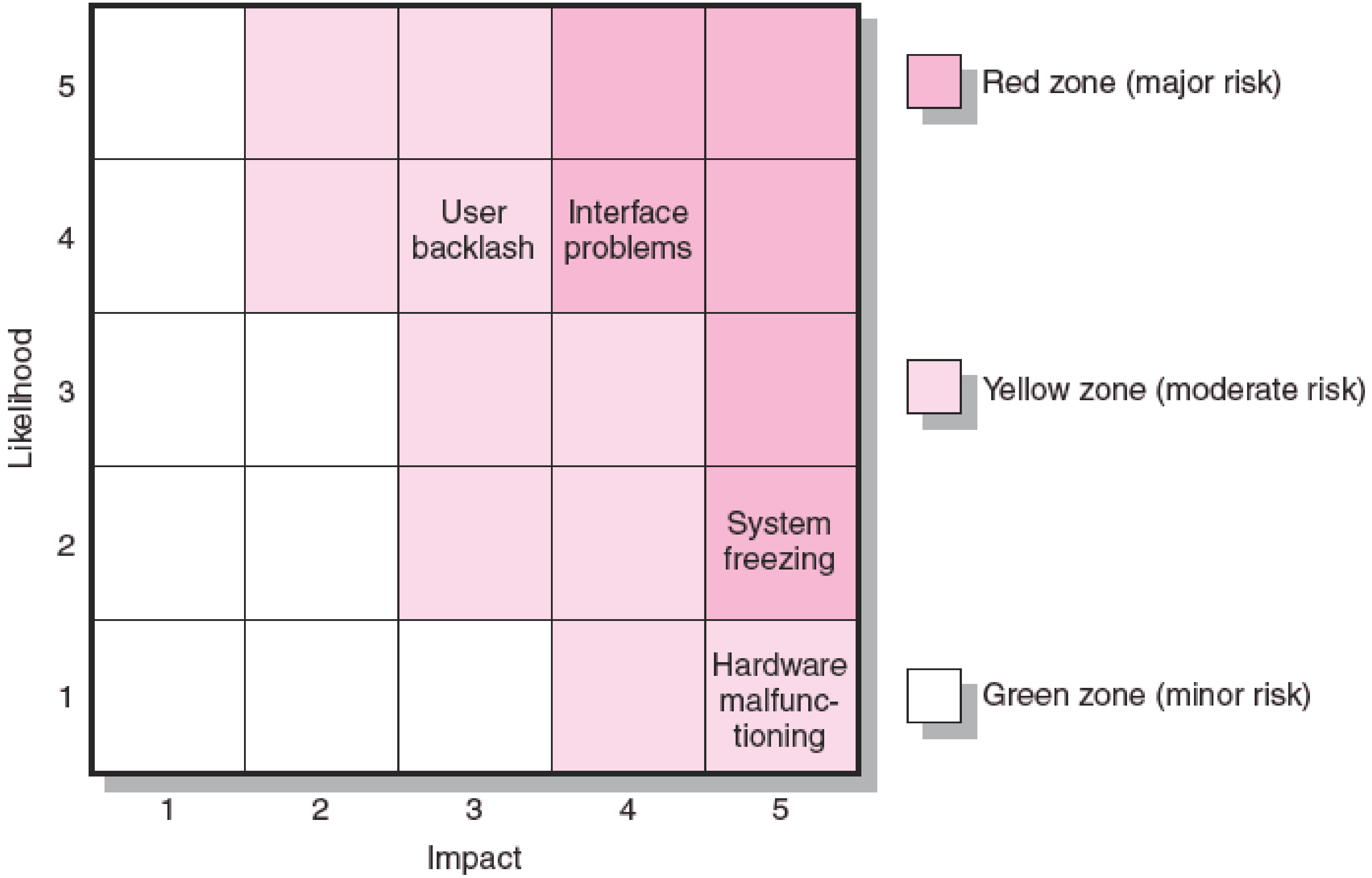
Risk Event	Likelihood	Impact	Detection Difficulty	When
Interface problems	4	4	4	Conversion
System freezing	2	5	5	Start-up
User backlash	4	3	3	Postinstallation
Hardware malfunctioning	1	5	5	Installation

Impact Scales



	Relative or Numerical Scale				
Project Objective	1 Very Low	2 Low	3 Moderate	4 High	5 Very High
Cost	Insignificant cost increase	< 10% cost increase	10–20% cost increase	20–40% cost increase	> 40% cost increase
Time	Insignificant time increase	< 5% time increase	5–10% time increase	10–20% time increase	> 20% time increase
Scope	Scope decrease barely noticeable	Minor areas of scope affected	Major areas of scope affected	Scope reduction unacceptable to sponsor	Project end item is effectively useless
Quality	Quality degradation barely noticeable	Only very demanding applications are affected	Quality reduction requires sponsor approval	Quality reduction unacceptable to sponsor	Project end item is effectively useless

Risk Severity Matrix



Likelihood					
5	🟩	🟨	🟨	🟥	🟥
4	🟩	🟨	🟨 User backlash	🟥 Interface problems	🟥
3	🟩	🟩	🟨	🟨	🟥
2	🟩	🟩	🟨	🟨	🟥 System freezing
1	🟩	🟩	🟩	🟨	🟨 Hardware malfunctioning
	1	2	3	4	5
	Impact				

🟥 Red zone (major risk)

🟨 Yellow zone (moderate risk)

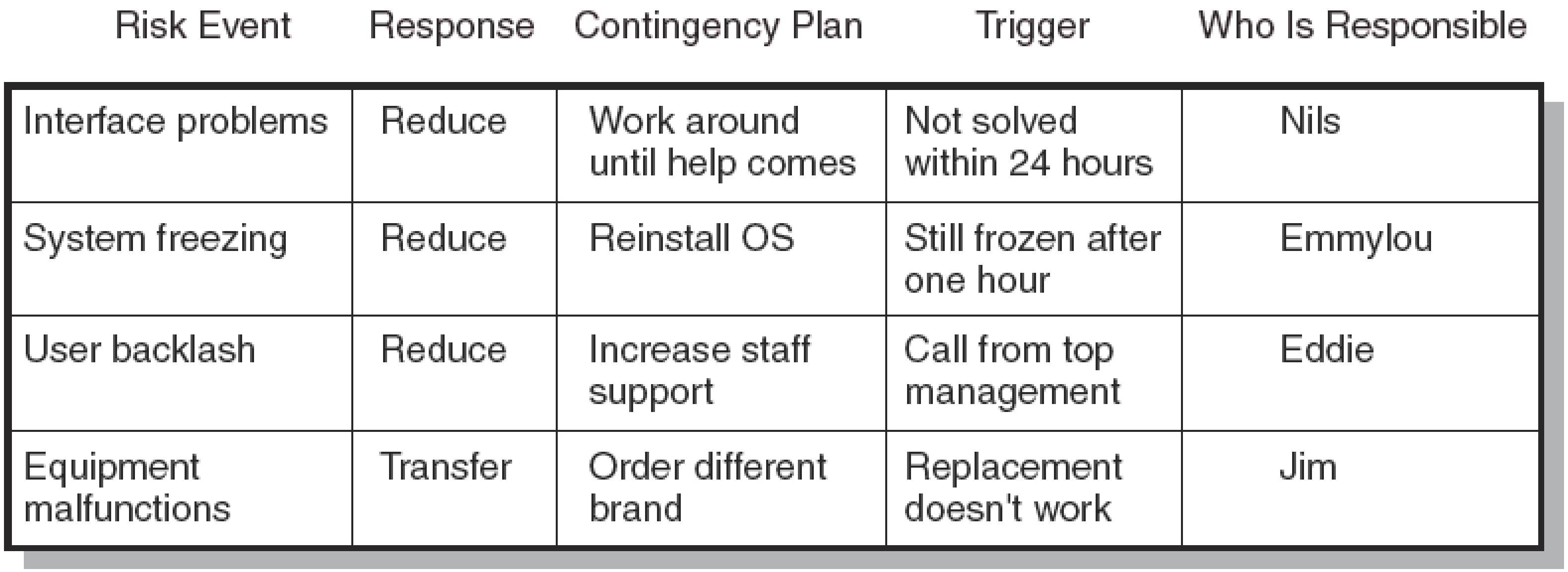
🟩 Green zone (minor risk)

- Step 3: Risk Response Development
  - Mitigating or Reducing Risk
    - Reducing the likelihood an adverse event will occur
    - Reducing impact of adverse event
  - Transferring Risk
    - Paying a premium to pass the risk to another party
  - Avoiding Risk
    - Changing the project plan to eliminate the risk or condition
  - Sharing Risk
    - Allocating risk to different parties
  - Retaining Risk
    - Making a conscious decision to accept the risk

Contingency Planning

- Contingency Plan
  - An alternative plan that will be used if a possible foreseen risk event actually occurs
  - A plan of actions that will reduce or mitigate the negative impact (consequences) of a risk event
- Risks of Not Having a Contingency Plan
  - Having no plan may slow managerial response
  - Decisions made under pressure can be potentially dangerous and costly

Risk Response Matrix



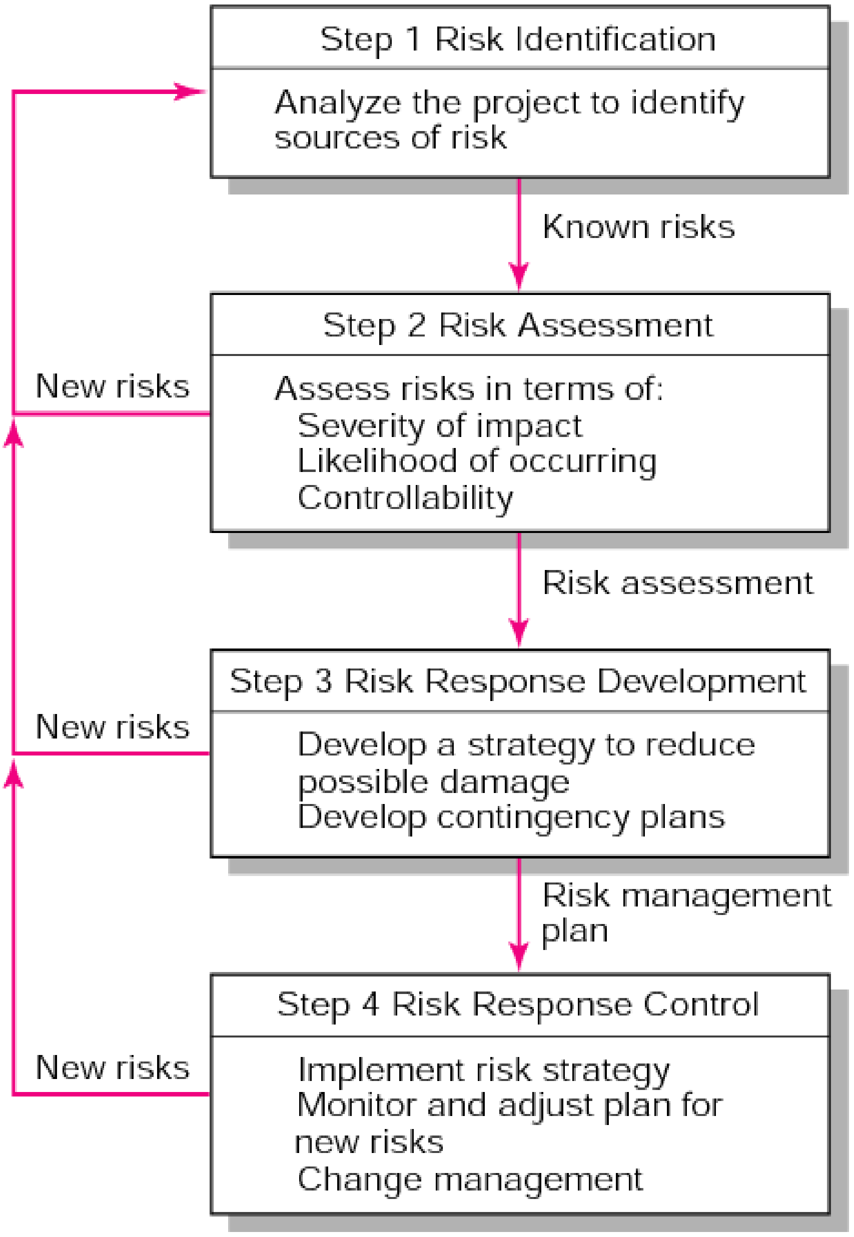
Risk Event	Response	Contingency Plan	Trigger	Who Is Responsible
Interface problems	Reduce	Work around until help comes	Not solved within 24 hours	Nils
System freezing	Reduce	Reinstall OS	Still frozen after one hour	Emmylou
User backlash	Reduce	Increase staff support	Call from top management	Eddie
Equipment malfunctions	Transfer	Order different brand	Replacement doesn't work	Jim

Risk and Contingency Planning

- Technical Risks
  - □ Backup strategies if chosen technology fails
  - □ Assessing whether technical uncertainties can be resolved
- Schedule Risks
  - □ Use of slack increases the risk of a late project finish
  - □ Imposed duration dates (absolute project finish date)
  - □ Compression of project schedules due to a shortened project duration date

- Step 4: Risk Response Control
  - □ Risk control
    - Execution of the risk response strategy
    - Monitoring of triggering events
    - Initiating contingency plans
    - Watching for new risks
  - □ Establishing a Change Management System
    - Monitoring, tracking, and reporting risk
    - Fostering an open organization environment
    - Repeating risk identification/assessment exercises
    - Assigning and documenting responsibility for managing risk

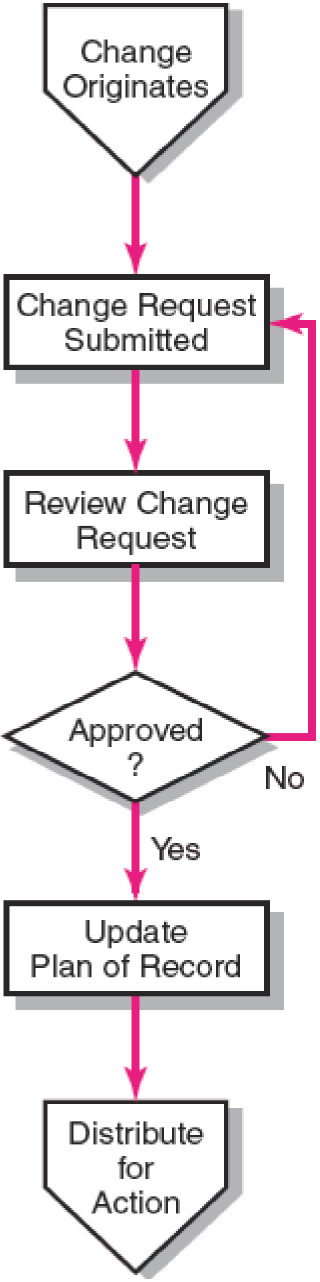
The Risk Management Process



Step 1 Risk Identification
Analyze the project to identify sources of risk
Known risks
Step 2 Risk Assessment
Assess risks in terms of:
Severity of impact
Likelihood of occurring
Controllability
Risk assessment
Step 3 Risk Response Development
Develop a strategy to reduce possible damage
Develop contingency plans
Risk management plan
Step 4 Risk Response Control
Implement risk strategy
Monitor and adjust plan for new risks
Change management
New risks
New risks
New risks

Change Management Control

- **The Change Control Process**
  - Identify proposed changes.
  - List expected effects of proposed changes on schedule and budget.
  - Review, evaluate, and approve or disapprove of changes formally.
  - Negotiate and resolve conflicts of change, condition, and cost.
  - Communicate changes to parties affected.
  - Assign responsibility for implementing change.
  - Adjust master schedule and budget.
  - Track all changes that are to be implemented.

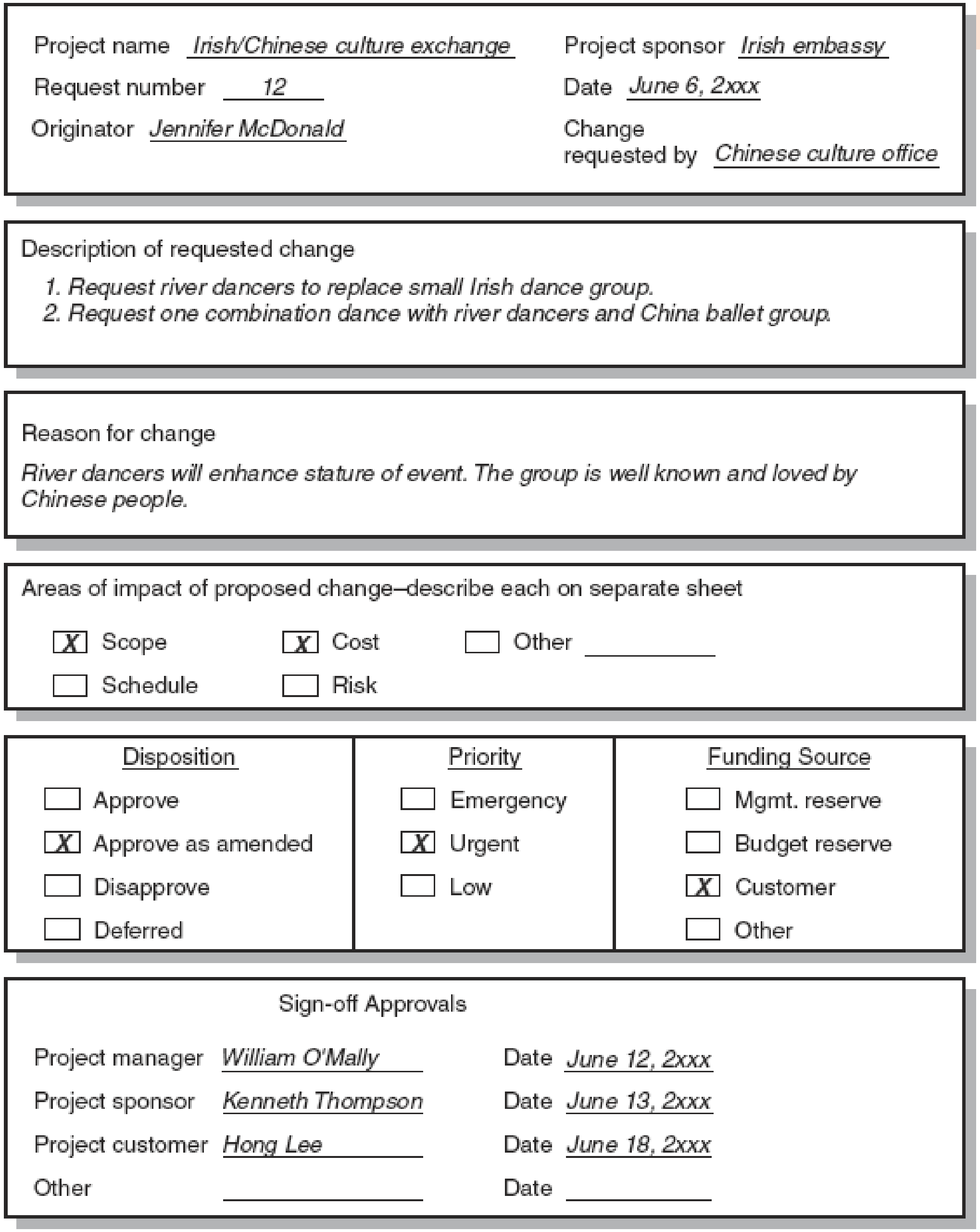


The Change Control Process

Benefits of a Change Control System

1. Inconsequential changes are discouraged by the formal process.
2. Costs of changes are maintained in a log.
3. Integrity of the WBS and performance measures is maintained.
4. Allocation and use of budget and management reserve funds are tracked.
5. Responsibility for implementation is clarified.
6. Effect of changes is visible to all parties involved.
7. Implementation of change is monitored.
8. Scope changes will be quickly reflected in baseline and performance measures.

Change Request Form



Project name *Irish/Chinese culture exchange*

Project sponsor *Irish embassy*

Request number *12*

Date *June 6, 2xxx*

Originator *Jennifer McDonald*

Change requested by *Chinese culture office*

Description of requested change

1. *Request river dancers to replace small Irish dance group.*
2. *Request one combination dance with river dancers and China ballet group.*

Reason for change

River dancers will enhance stature of event. The group is well known and loved by Chinese people.

Areas of impact of proposed change–describe each on separate sheet

☒ Scope ☒ Cost ☐ Other ______

☐ Schedule ☐ Risk

Disposition	Priority	Funding Source
☐ Approve	☐ Emergency	☐ Mgmt. reserve
☒ Approve as amended	☒ Urgent	☐ Budget reserve
☐ Disapprove	☐ Low	☒ Customer
☐ Deferred		☐ Other

Sign-off Approvals

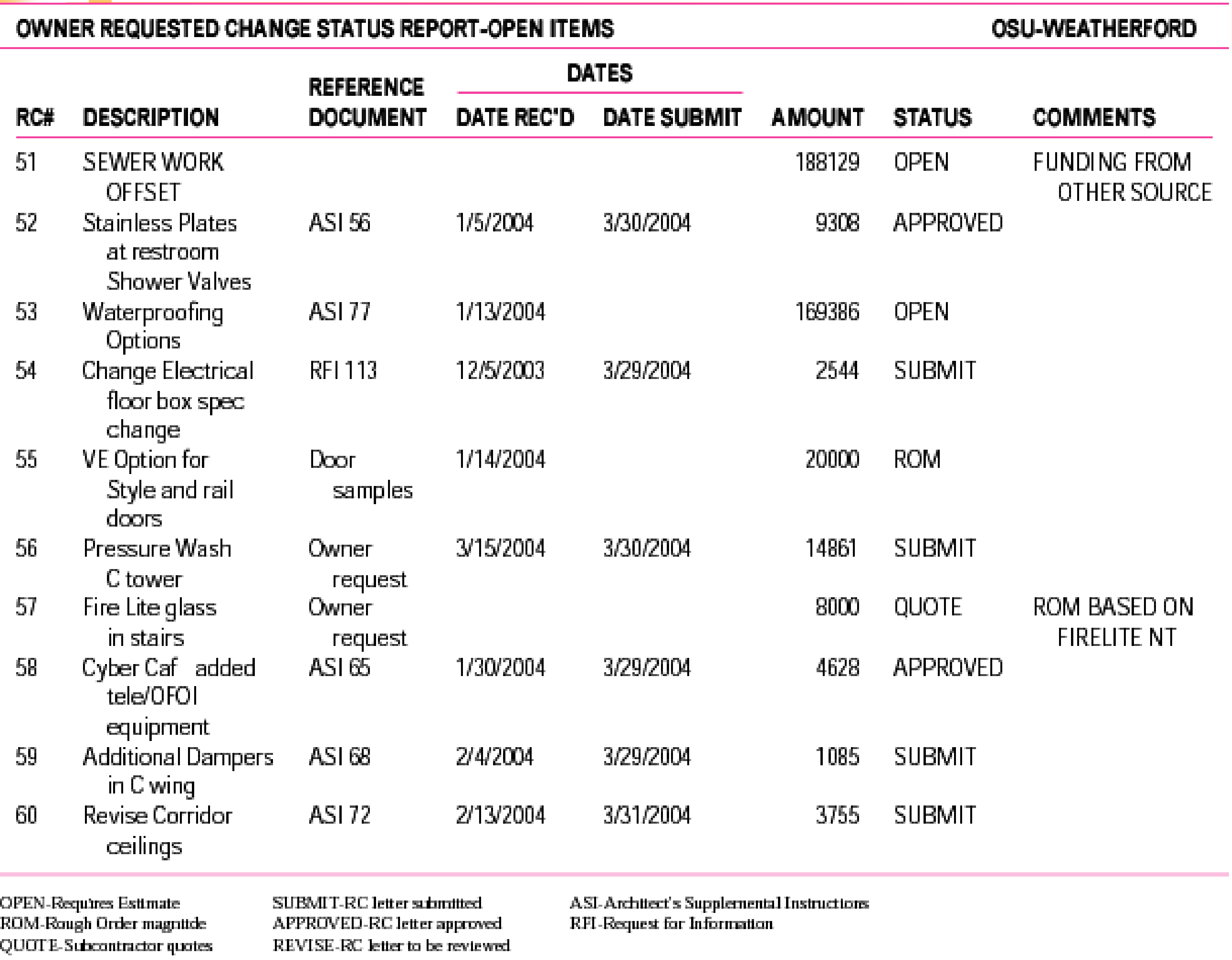
Project manager *William O'Mally* Date *June 12, 2xxx*

Project sponsor *Kenneth Thompson* Date *June 13, 2xxx*

Project customer *Hong Lee* Date *June 18, 2xxx*

Other ______ Date ______

Change Request Log



OWNER REQUESTED CHANGE STATUS REPORT-OPEN ITEMS — OSU-WEATHERFORD

			DATES				
RC#	DESCRIPTION	REFERENCE DOCUMENT	DATE REC'D	DATE SUBMIT	AMOUNT	STATUS	COMMENTS
51	SEWER WORK OFFSET				188129	OPEN	FUNDING FROM OTHER SOURCE
52	Stainless Plates at restroom Shower Valves	ASI 56	1/5/2004	3/30/2004	9308	APPROVED	
53	Waterproofing Options	ASI 77	1/13/2004		169386	OPEN	
54	Change Electrical floor box spec change	RFI 113	12/5/2003	3/29/2004	2544	SUBMIT	
55	VE Option for Style and rail doors	Door samples	1/14/2004		20000	ROM	
56	Pressure Wash C tower	Owner request	3/15/2004	3/30/2004	14861	SUBMIT	
57	Fire Lite glass in stairs	Owner request			8000	QUOTE	ROM BASED ON FIRELITE NT
58	Cyber Caf added tele/OFOI equipment	ASI 65	1/30/2004	3/29/2004	4628	APPROVED	
59	Additional Dampers in C wing	ASI 68	2/4/2004	3/29/2004	1085	SUBMIT	
60	Revise Corridor ceilings	ASI 72	2/13/2004	3/31/2004	3755	SUBMIT	

OPEN-Requires Estimate
ROM-Rough Order magnitude
QUOTE-Subcontractor quotes

SUBMIT-RC letter submitted
APPROVED-RC letter approved
REVISE-RC letter to be reviewed

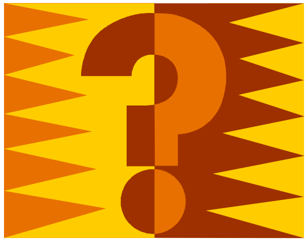
ASI-Architect's Supplemental Instructions
RFI-Request for Information

Thinking About Risks

- What kinds of things could go wrong with your class project in this class?
- What will you do if one of them does?
- How likely are they to happen?

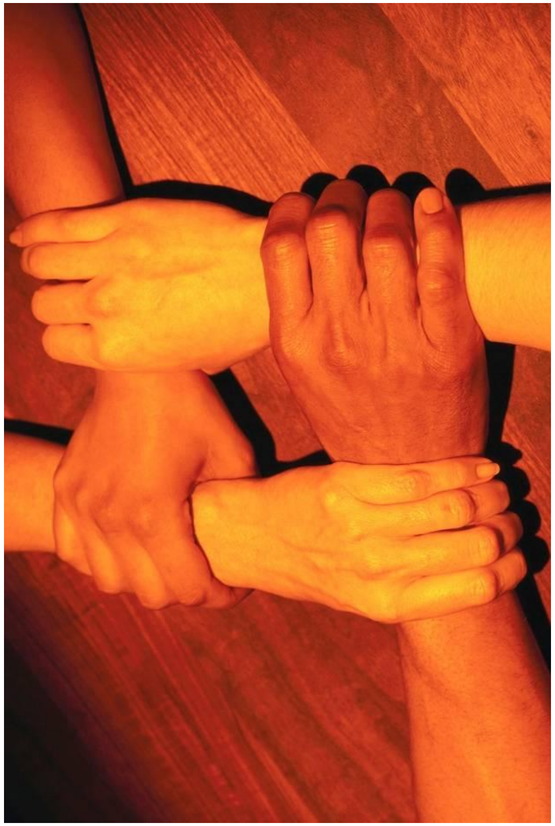
Project Status???

- Project is anticipated to take 10 weeks at $400,000 per week
- After week 5
  - Actual cost incurred $2,400,000
  - Is the project going to be overbudget?
- After week 8
  - Actual cost incurred $3,000,000
  - Is the project going to be underbudget?



In Class Project

- Think of a manager/leader from your past
- What were the good qualities?
- What were the bad qualities?
- You'll be asked to share one of each that has not already been given



Managing versus Leading a Project

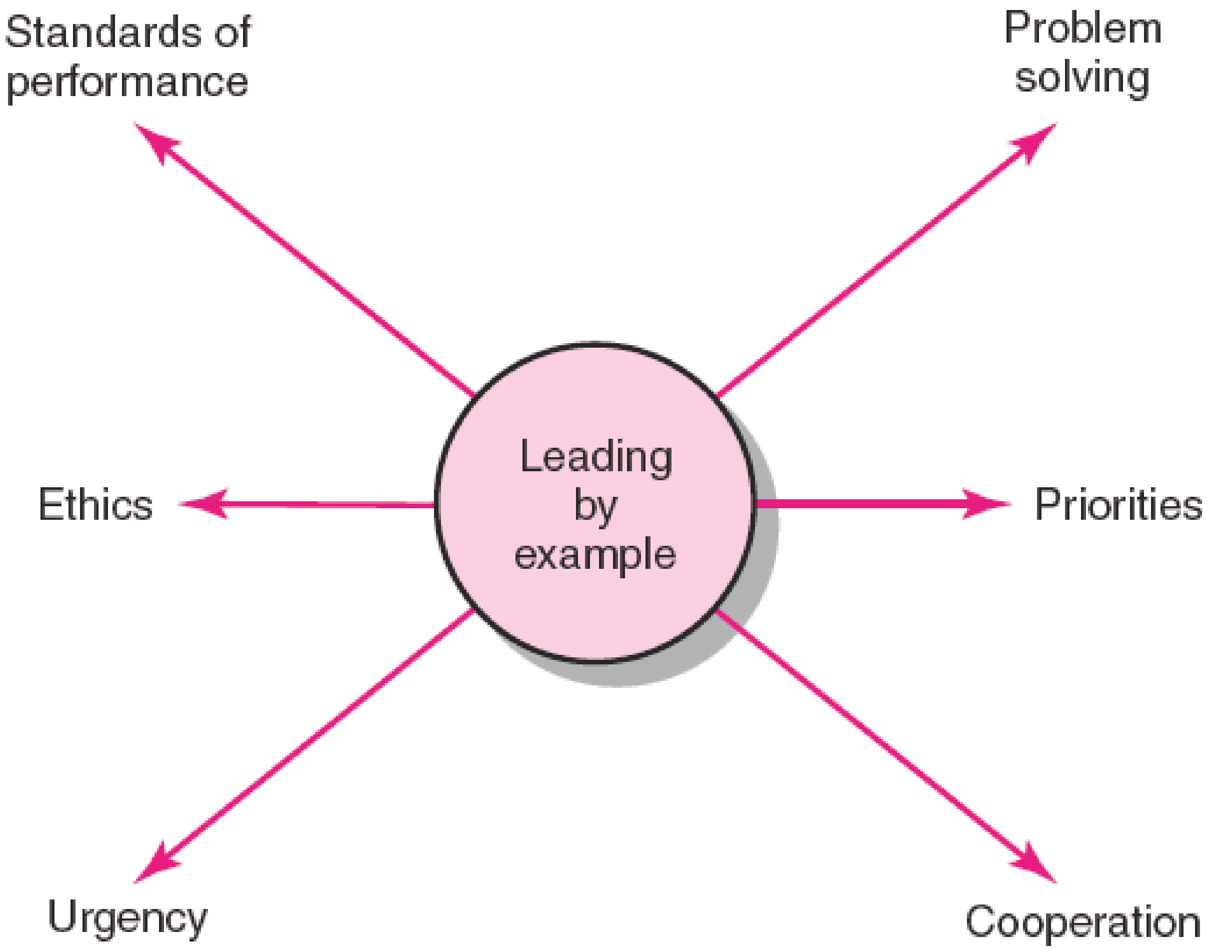
- **Managing: Coping with Complexity**
  - Formulate plans and objectives
  - Monitor results
  - Take corrective action
  - Expedite activities
  - Solve technical problems
  - Serve as peacemaker
  - Make tradeoffs among time, costs, and project scope

- **Leading: Coping with Change**
  - Recognize the need to change to keep the project on track
  - Initiate change
  - Provide direction and motivation
  - Innovate and adapt as necessary
  - Integrate assigned resources

Project Management Maxims

- You can‘t do it all and get it all done.
  - Projects usually involve a vast web of relationships.
- Hands-on work is not the same as leading.
  - More pressure and more involvement can reduce your effectiveness as a leader.
- What‘s important to you likely isn‘t as important to someone else.
  - Different groups have different stakes (responsibilities, agendas, and priorities) in the outcome of a project.
- The challenge is relationships
  - Build them before you need them!

Leading by Example



Contradictions of Leading/Managing

- Innovate and maintain stability
- See the big picture while getting your hands dirty
- Encourage individuals but stress the team
- Hands-off/hands-on
- Flexible but firm
- Team versus organizational loyalties

Qualities of an Effective Project Manager

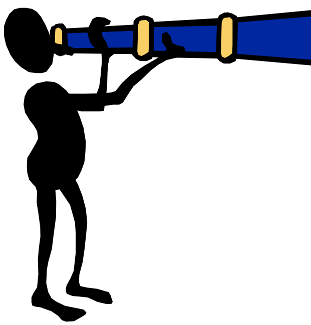
- Systems thinker
- Personal integrity
- Proactive
- High emotional intelligence
- General business perspective
- Effective time management
- Skillful politician
- Optimist

Suggestions for Effective Leadership

- Build relationships before you need them.
- Trust is sustained through frequent face-to-face contact.
- People have different temperaments

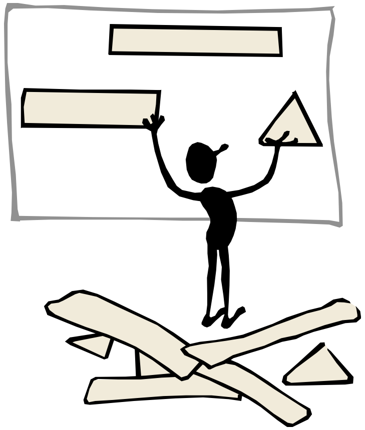
The 4 Dimensions of Personality Style

Introversion/Extraversion—
What energizes you?



Sensing/Intuiting—
What is the focus of
your attention?

Thinking/Feeling—
How do you make
decisions?



Judging/Perceiving—
How do you structure
your behavior?

What energizes you? Introversion-Extraversion

Introversion

- quiet concentration
- like details & dislike generalizations
- not remember names and faces
- work one project for long periods
- interest in idea behind task
- think before acting
- work well alone
- less communicative

Extraversion

- variety & action
- like fast, uncomplicated procedures
- good at greeting people
- impatient with long slow tasks
- results oriented
- don’t mind interruptions
- act first, think later
- like people around
- communicate freely

What is the focus of your attention?
Sensing-Intuiting

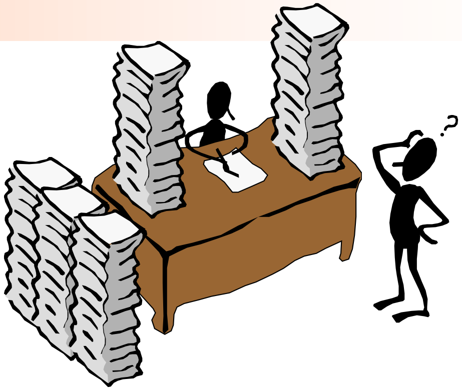
Sensing

- Dislike new problems
- Use established methods
- Like using old skills more
- Work steady and paced
- Step by step conclusion
- Patient with routine details
- Don’t trust inspiration
- Rare errors of fact
- Good at precise work

Intuiting

- Like new problems
- Dislike repetition
- Enjoy learning new skills
- Bursts of energy
- Reach conclusions quickly
- Impatient with routine details
- Patient with complexity
- Follow inspirations
- Errors of fact
- Dislike time for precision

How do you make decisions?
Thinking-Feeling



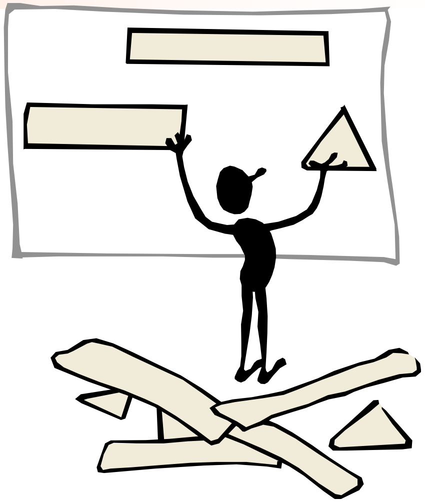
Thinking

- Not show or uncomfortable with emotions
- Hurt feelings without knowing
- Analysis & logical order
- Conflict is OK
- Decide impersonally
- Fairness & justice important
- Can reprimand & discipline
- Responds to ideas
- Firm-minded

Feeling

- Aware of people & feelings
- Pleasing people
- Like harmony; dislike conflict
- Decisions influence by likes & wishes
- Need occasional praise
- Dislike discipline and control
- Respond to values & feelings
- Sympathetic

How do you structure your behavior? Judging (structure)-Perceiving (change)

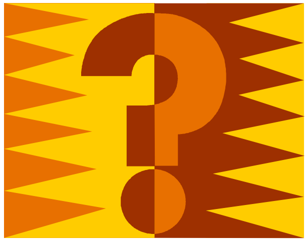


Judging

- Make plans and follow them
- Things settled and finished
- Decide too quickly
- Dislike changing priorities
- Not notice new things
- Just the essentials
- Satisfied with decision

Perceiving

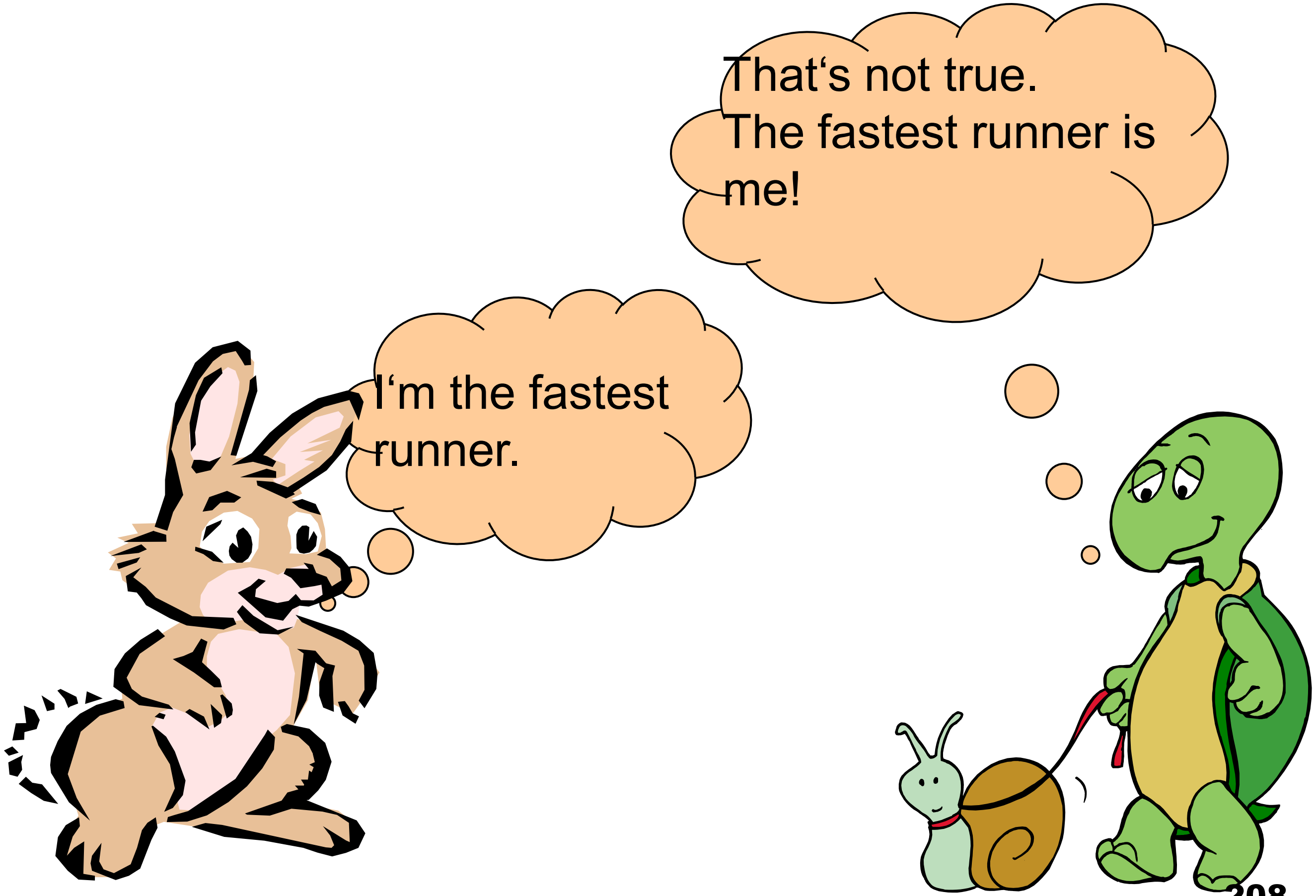
- Adapt to changing situations
- Leave things open
- Open-ended decisions
- Too many unfinished projects
- Postpone unpleasant jobs
- Want to know everything
- Curious and open to ideas



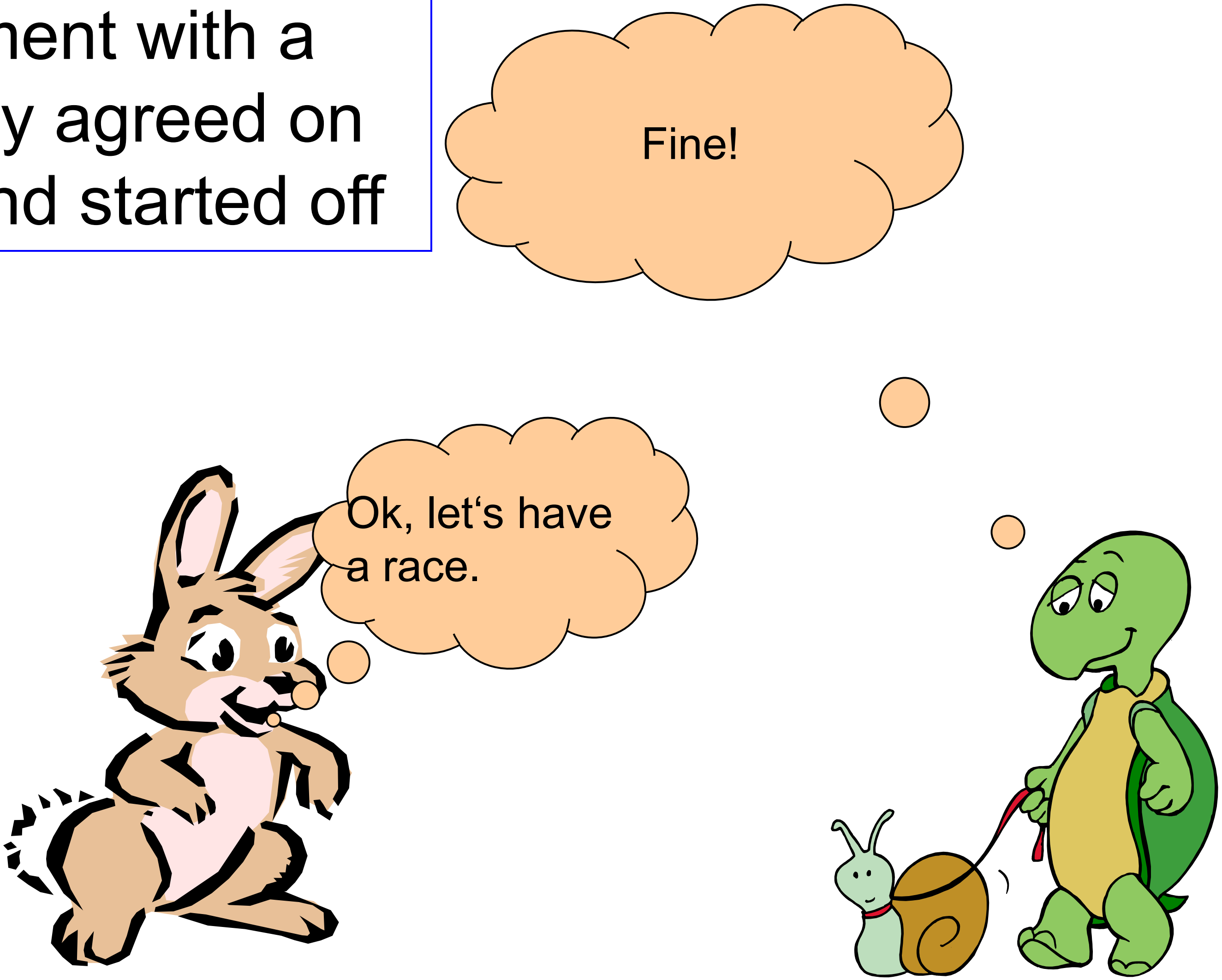
Good old lessons in *teamwork* from an age-old fable

The Tortoise
And
The Hare

Once upon a time a tortoise and a hare had an argument about who was faster.

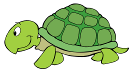


They decided to settle the argument with a race. They agreed on a route and started off the race.



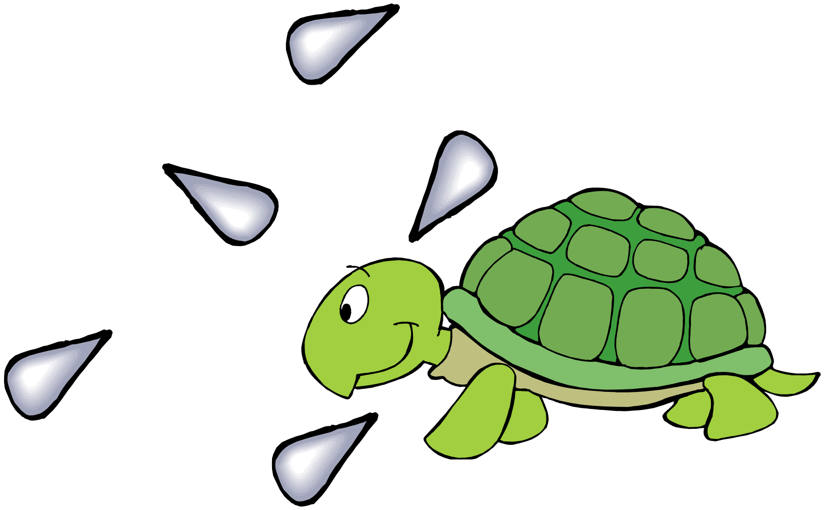
The hare shot ahead and ran briskly for some time. Then seeing that he was far ahead of the tortoise, he thought he'd sit under a tree for some time and relax before continuing the race.

Poor guy! Even if I take a nap, he could not catch up with me.

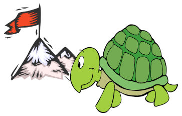


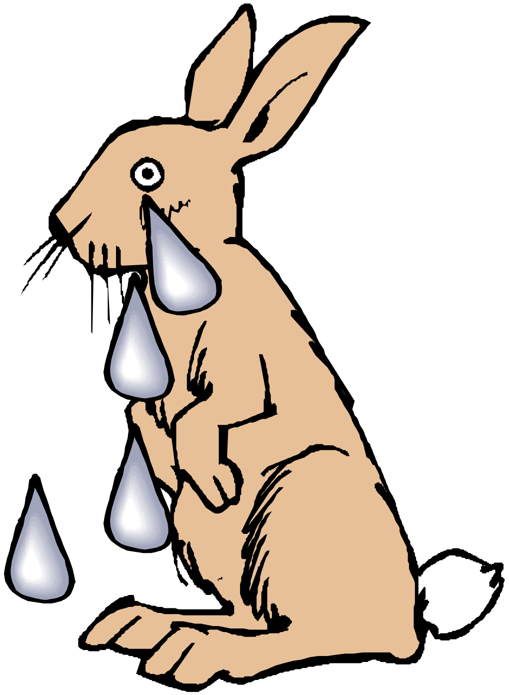
He sat under the tree and soon fell asleep.

The tortoise plodding on overtook him and soon finished the race, emerging as the undisputed champ.



The hare woke up and realized that he'd lost the race.



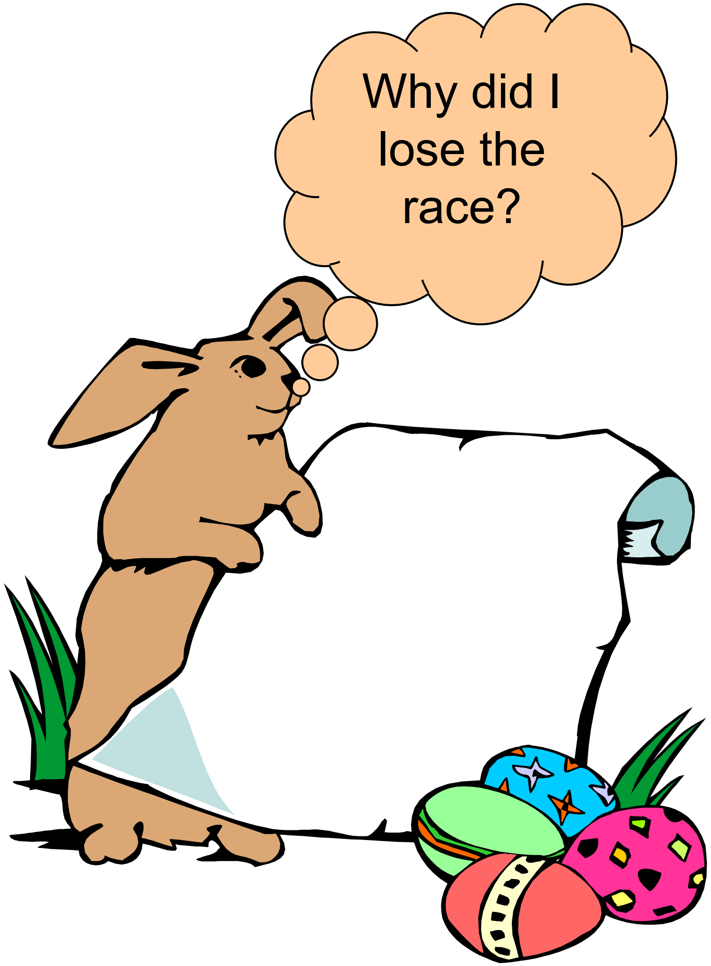


The moral of the story is that *slow and steady wins the race*.

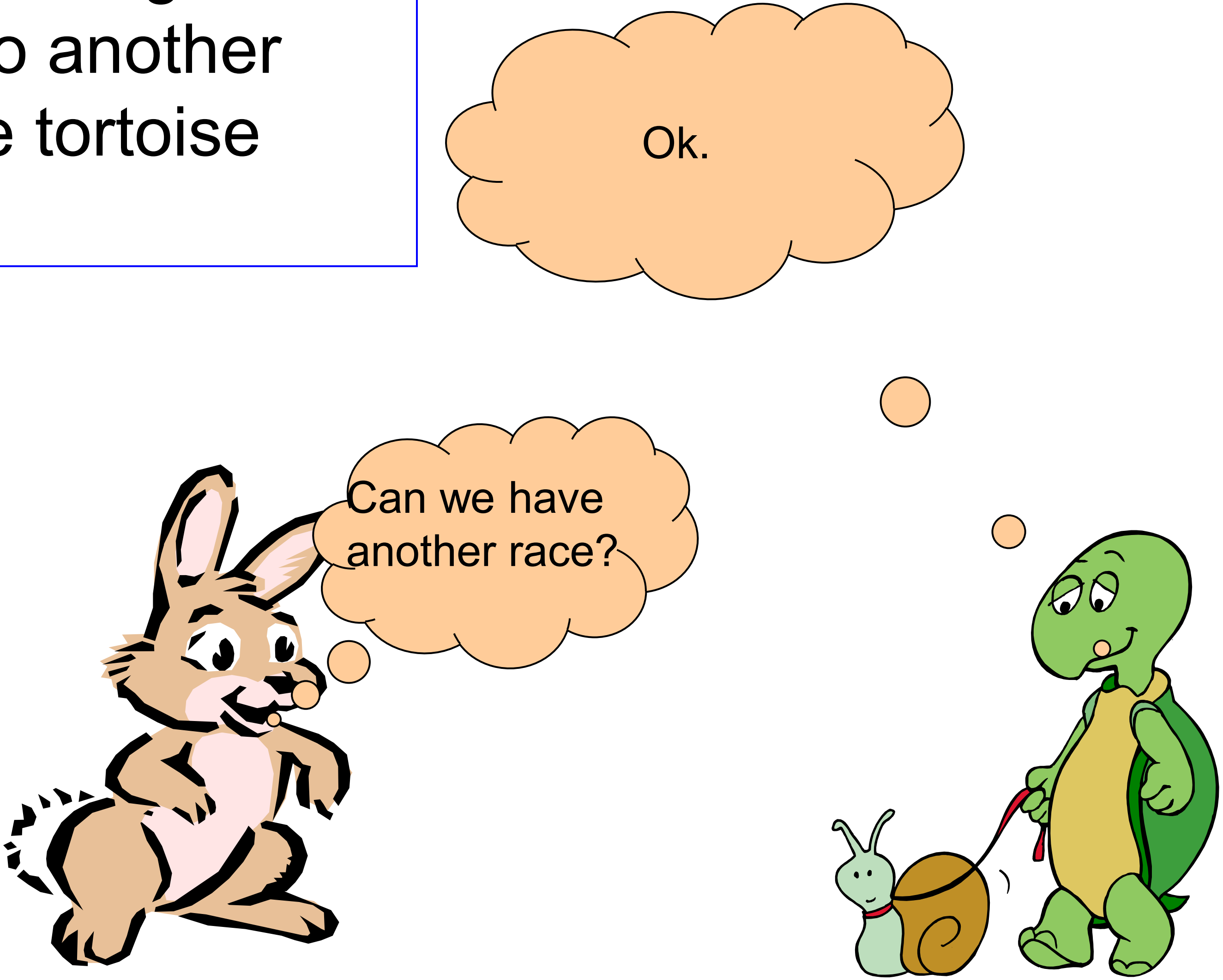
This is the version of the story that we've all grown up with.

The story continues …

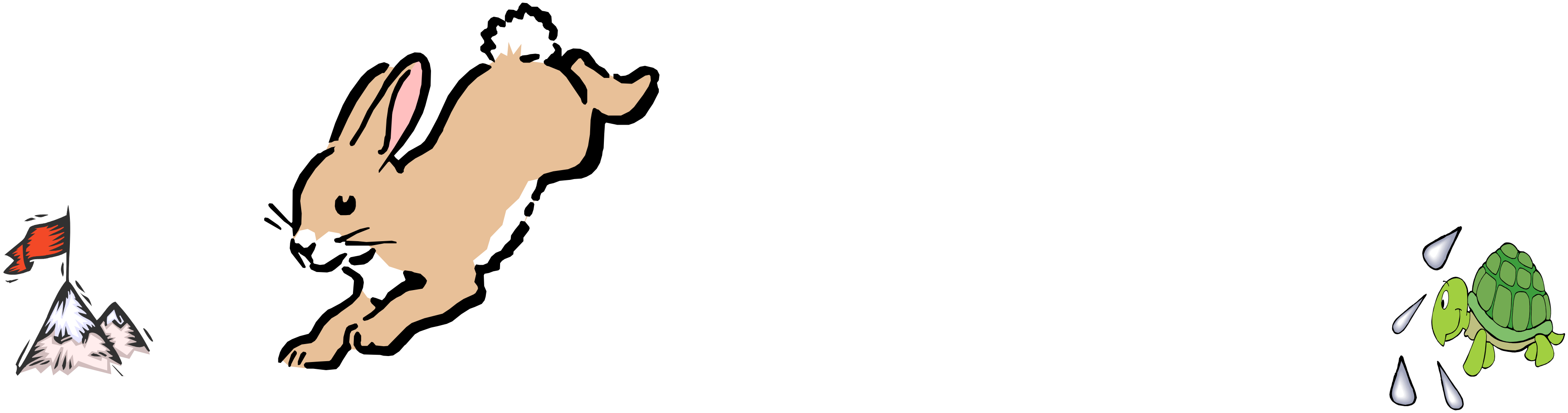
The hare was disappointed at losing the race and he did some soul-searching. He realized that he'd lost the race only because he had been overconfident, careless and lax. If he had not taken things for granted, there's no way the tortoise could have beaten him.



So he challenged the tortoise to another race. The tortoise agreed.



This time, the hare went all out and ran without stopping from start to finish. He won by several miles.



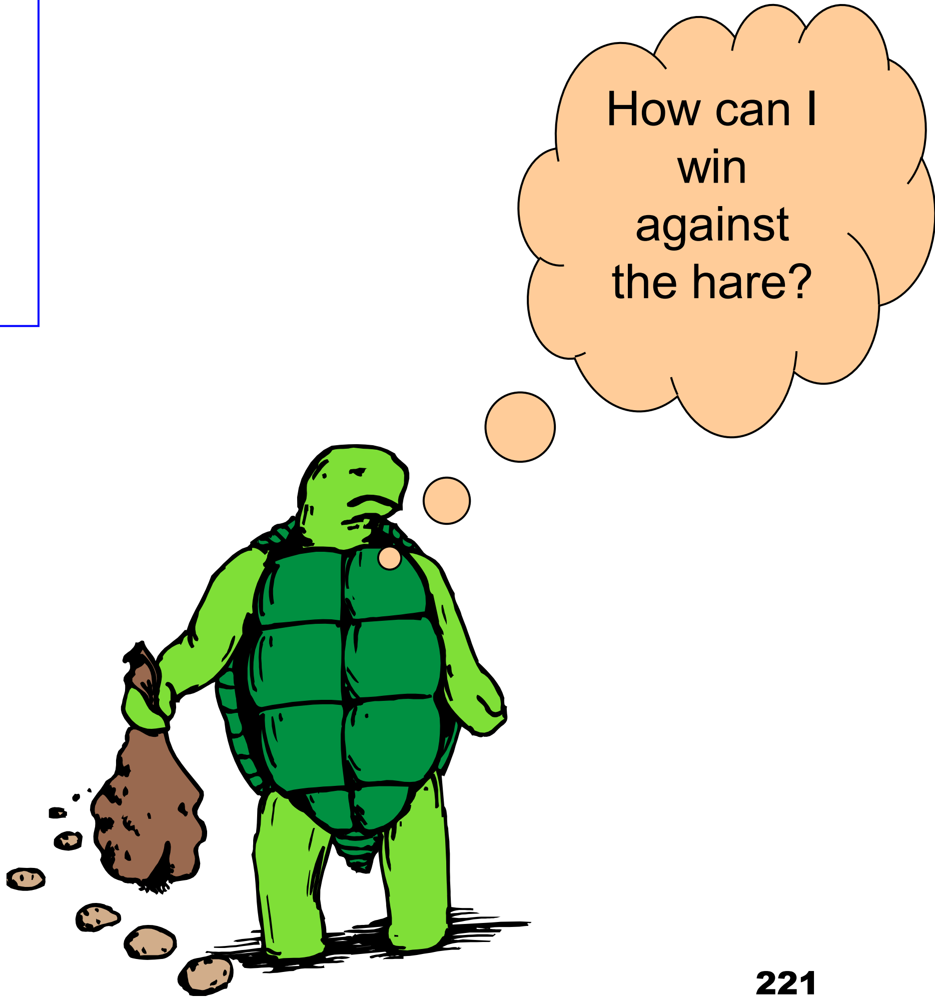
The moral of the story?

Fast and consistent will always beat the slow and steady. If you have two people in your organization, one slow, methodical and reliable, and the other fast and still reliable at what he does, the fast and reliable chap will consistently climb the organizational ladder faster than the slow, methodical chap.

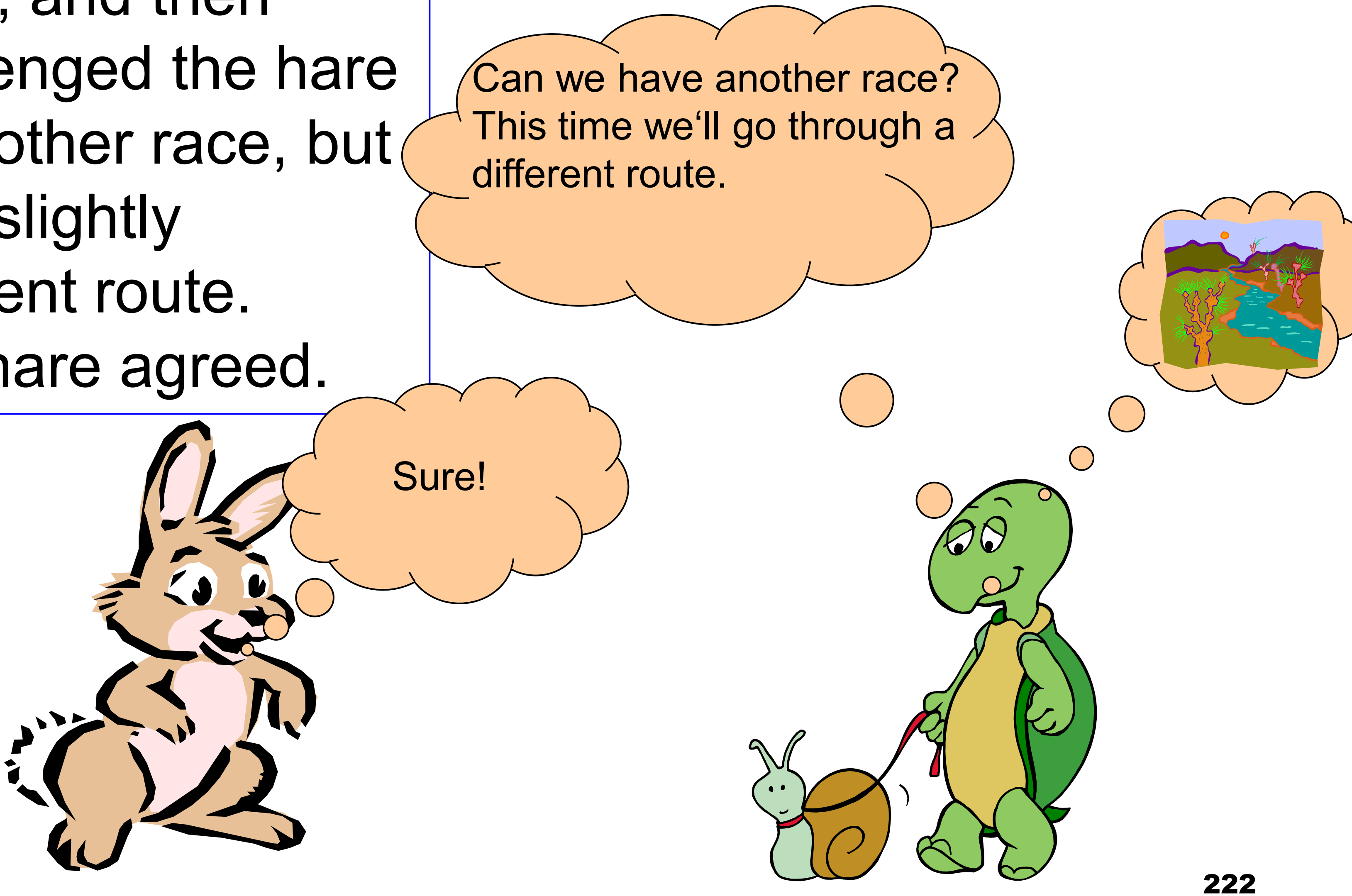
It's good to be slow and steady; but it's better to be fast and reliable.

But the story doesn't end here …

The tortoise did some thinking this time, and realized that there's no way he can beat the hare in a race the way it was currently formatted.

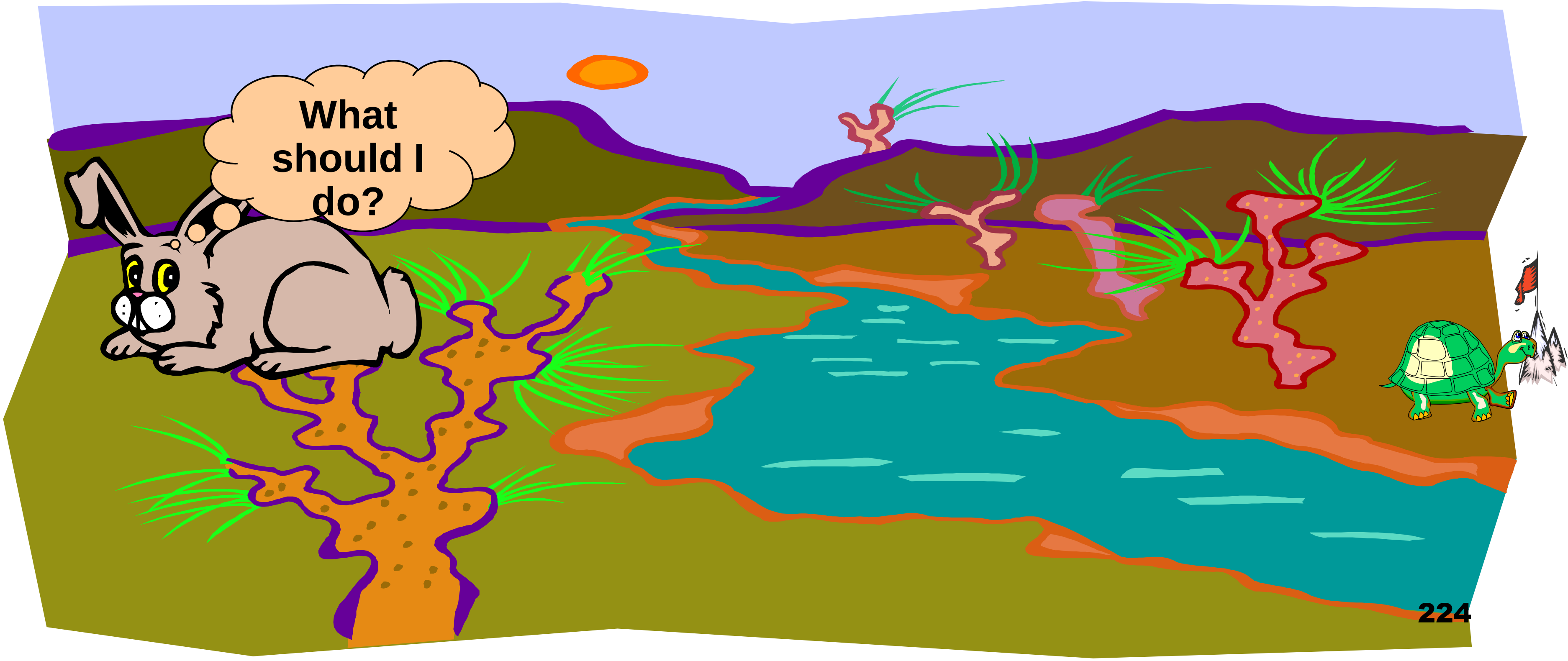


He thought for a while, and then challenged the hare to another race, but on a slightly different route. The hare agreed.


Can we have another race? This time we‘ll go through a different route.
Sure!

They started off. In keeping with his self-made commitment to be consistently fast, the hare took off and ran at top speed until he came to a broad river. The finishing line was a mile on the other side of the river.

The hare sat there wondering what to do. In the meantime the tortoise trundled along, got into the river, swam to the opposite bank, continued walking and finished the race.



The moral of the story?

First identify your core competency and then change the playing field to suit your core competency.

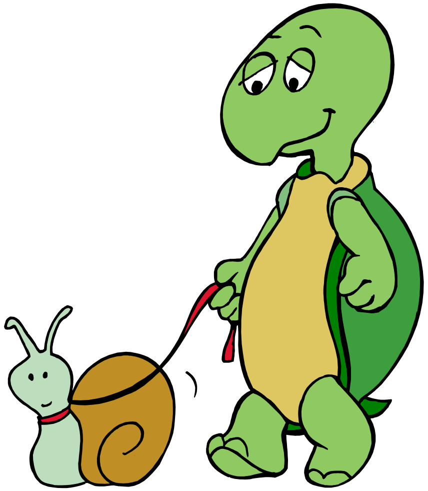
In an organization, if you are a good speaker, make sure you create opportunities to give presentations that enable the senior management to notice you.

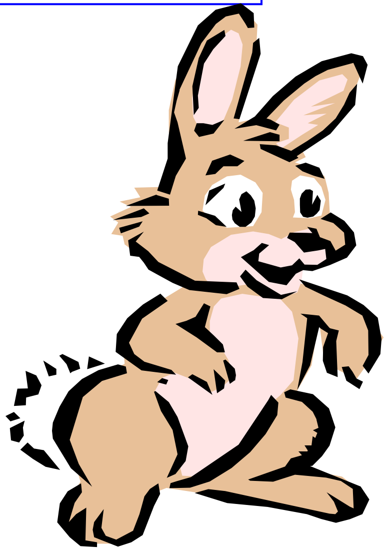
If your strength is analysis, make sure you do some sort of research, make a report and send it upstairs.

Working to your strengths will not only get you noticed, but will also create opportunities for growth and advancement.

The story still hasn't ended …

The hare and the tortoise, by this time, had become pretty good friends and they did some thinking together. Both realized that the last race could have been run much better.

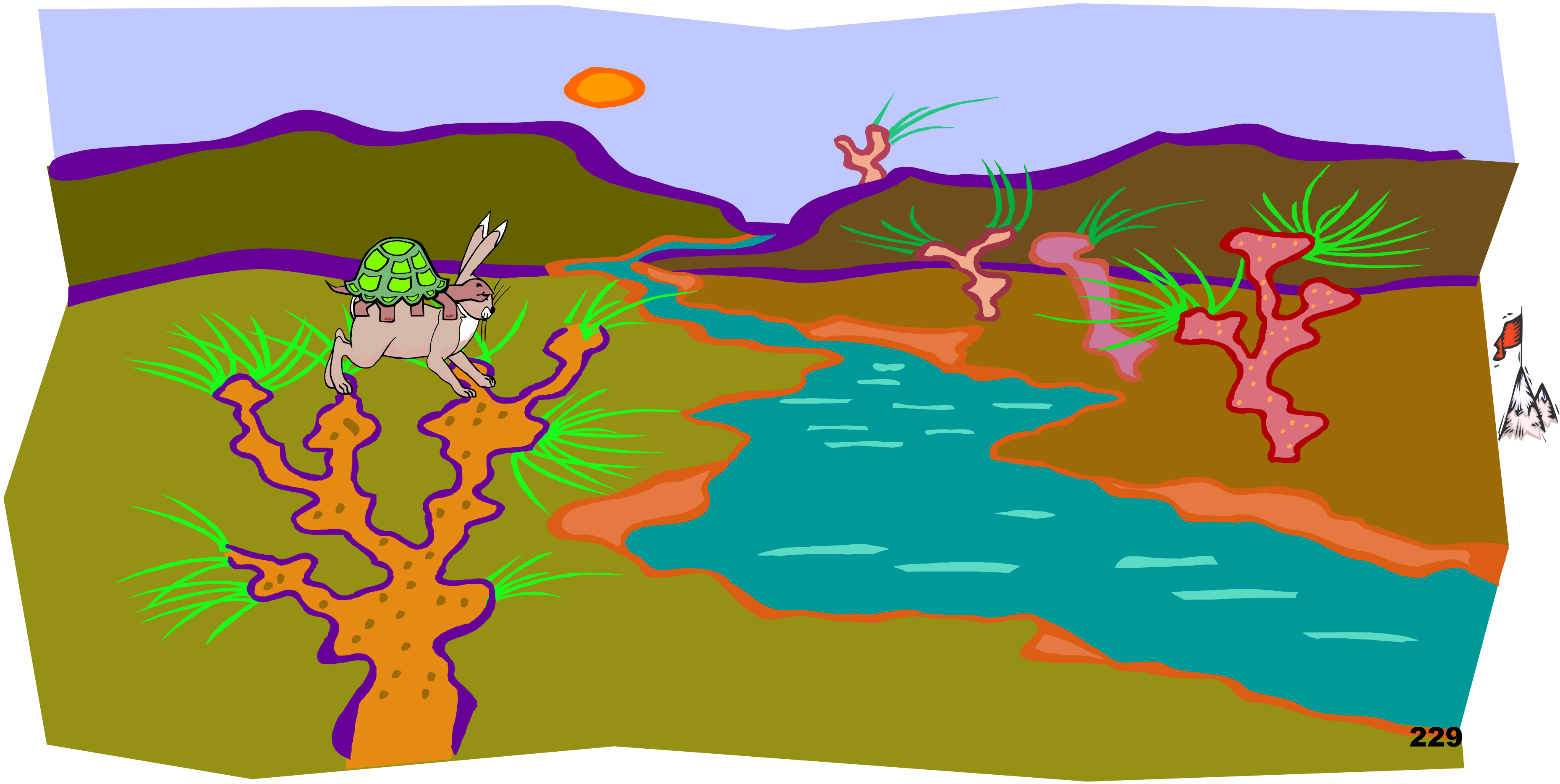




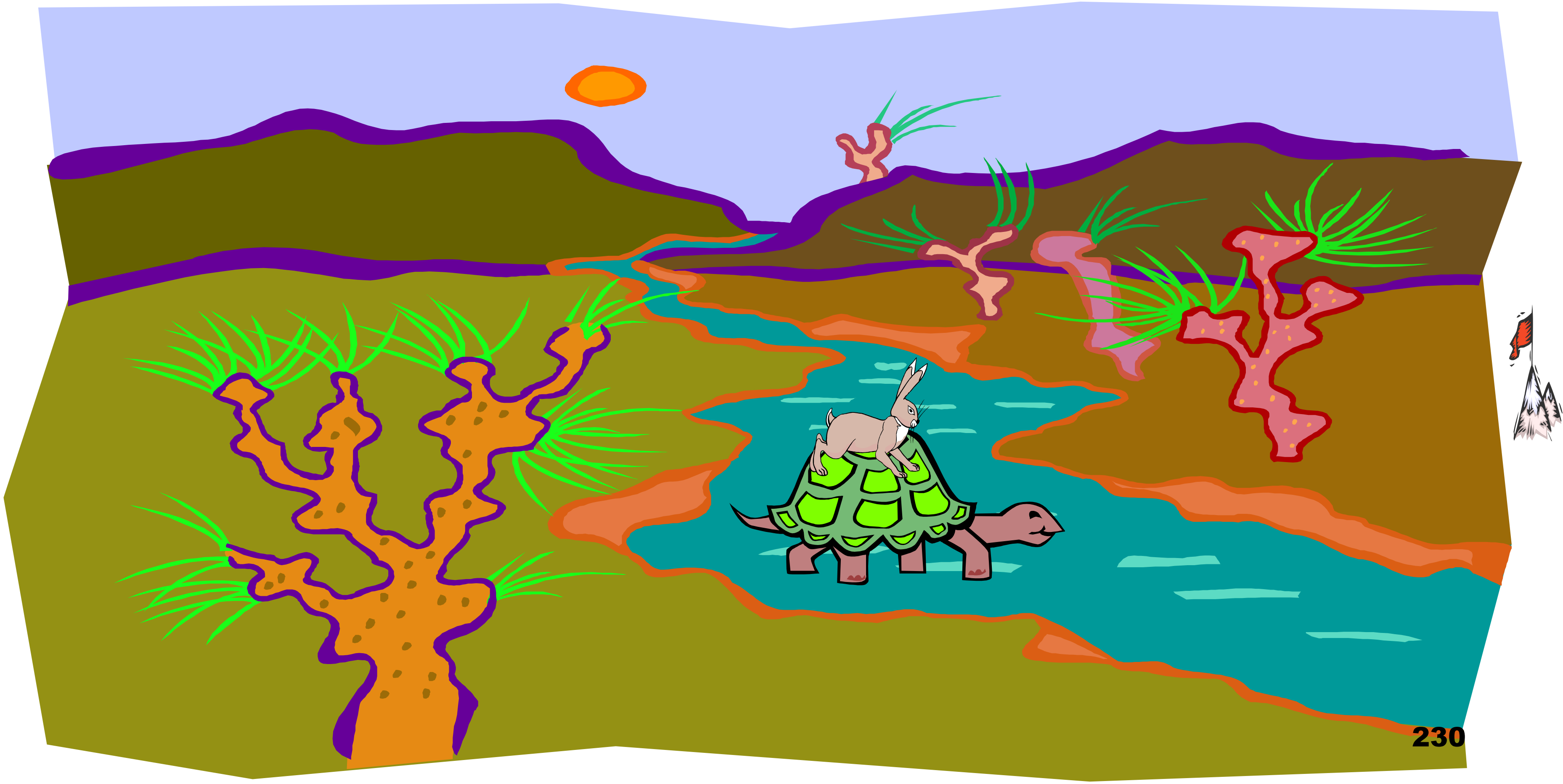
So they decided to do the last race again, but to run as a team this time.

Great! I think we could do it much better, if we two help each other.
Hi, buddy. How about doing our last race again?

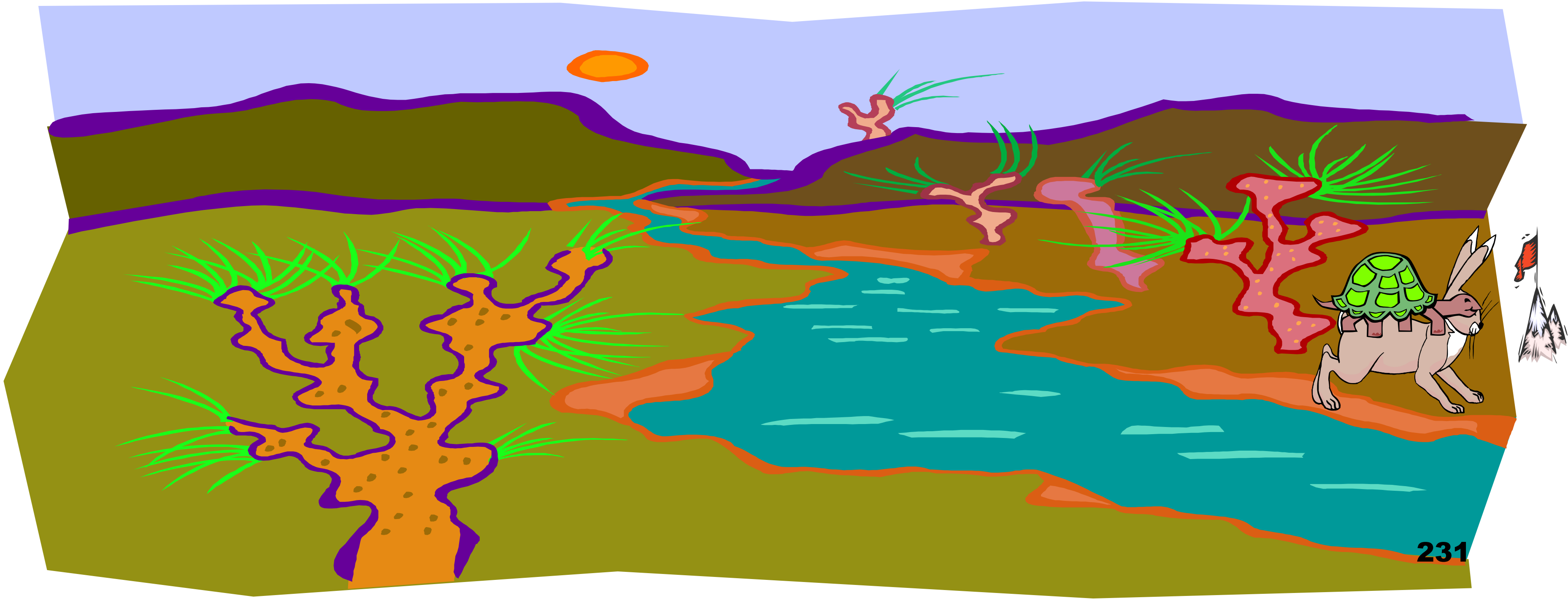
They started off, and this time the hare carried the tortoise till the riverbank.



There, the tortoise took over and swam across with the hare on his back.



On the opposite bank, the hare again carried the tortoise and they reached the finishing line together. They both felt a greater sense of satisfaction than they'd felt earlier.



The moral of the story?

It's good to be individually brilliant and to have strong core competencies; but unless you're able to work in a team and harness each other's core competencies, you'll always perform below par because there will always be situations at which you'll do poorly and someone else does well.

Teamwork is mainly about situational leadership, letting the person with the relevant core competency for a situation take leadership.

There are more lessons to be learned from this story.

Note that neither the hare nor the tortoise gave up after failures. The hare decided to work harder and put in more effort after his failure. The tortoise changed his strategy because he was already working as hard as he could.

In life, when faced with failure, sometimes it is appropriate to work harder and put in more effort. Sometimes it is appropriate to change strategy and try something different. And sometimes it is appropriate to do both.

The hare and the tortoise also learned another vital lesson. *When we stop competing against a rival and instead start competing against the situation, we perform far better.*

To sum up, the story of the hare and tortoise teaches us many things:

✓ *Never give up when faced with failure*

✓*Fast and consistent will always beat slow and steady*

✓ *Work to your competencies*

✓*Compete against the situation, not against a rival.*

✓ *Pooling resources and working as a team will always beat individual performers*

What is a Team?

- Two or more individuals with a high degree of interdependence geared toward the achievement of a goal or the completion of a task.

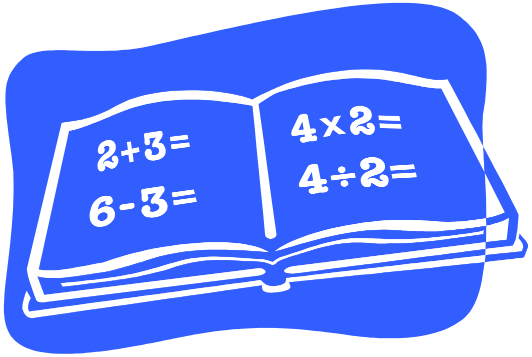
- Teams make decisions, solve problems, provide support, accomplish missions, and plan their work.

There are Many Types of Teams

- Examples of Teams:
  - □ **Athletic Team** – people working together to win a game
  - □ **Natural Work Group** – people working together every day in same office with similar processes and equipment
  - □ **Business Team** – cross-functional team overseeing a specific product line or customer segment
  - □ **Improvement Team** – ad hoc team with responsibility for improving an existing process
  - □ **Student Team** – students working together on a course project

High-Performing Teams

- Synergy
  - □ 1 + 1 + 1 = 10
    - (positive synergy)
  - □ 1 + 1 + 1 = 2
    - (negative synergy)



Attitudes for Effective Teamwork

- Appreciation for value of team decisions
- Respect for team members
- Mutual trust
- Openness to feedback
- Reflection on group process and interest in improving
- Shared vision

What are Characteristics of Effective or High Performing Teams?

- Members have a clear goal
- The focus is on achieving results
- There is a plan for achieving the goal
- Members have clear roles
- Members are committed to the goal
- Members are competent
- They achieve decisions through consensus
- There is diversity among team members
- Members have effective interpersonal skills
- They know each other well and have good relationships

More Characteristics

- Each member feels empowered to act, speak up, offer ideas
- Each member has a high standard of excellence
- An informal climate and easiness exists among members
- The team has the support of management
- The team is open to new ideas
- There is periodic self-assessment
- There is shared leadership of the team
- The team is a relatively small size
- There is recognition of team member accomplishments
- There are sufficient resources to support the team work

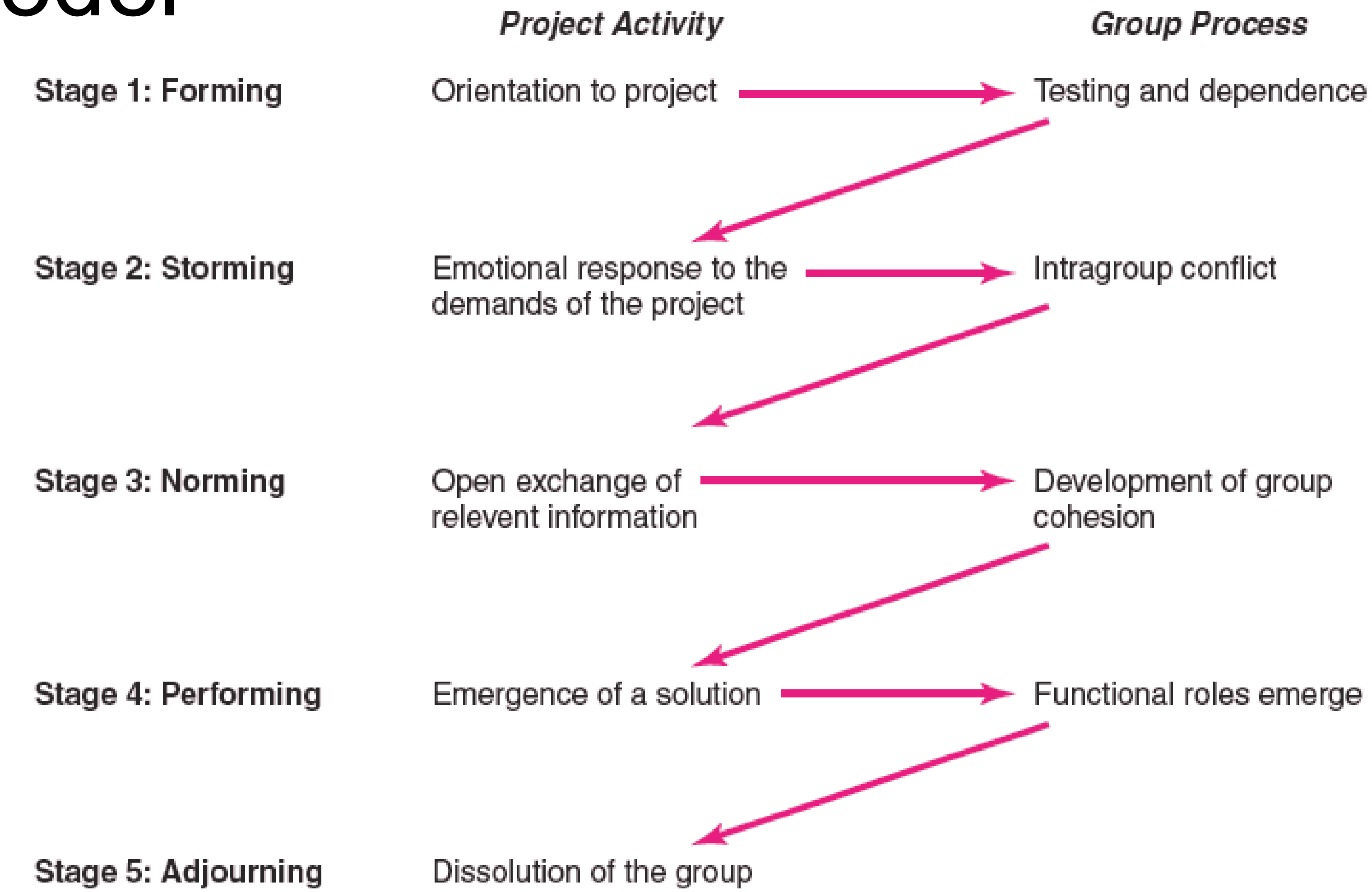
Communication is Important!!!

- Team Communication Is Important

Effective Team-Building Takes Time

- There must be frequent and prolonged contact
- Team members come together around a specific goal or project
- Effective teams go through four stages of team development

The Five-Stage Team Development Model



	Project Activity	*Group Process*
Stage 1: Forming	Orientation to project	Testing and dependence
Stage 2: Storming	Emotional response to the demands of the project	Intragroup conflict
Stage 3: Norming	Open exchange of relevent information	Development of group cohesion
Stage 4: Performing	Emergence of a solution	Functional roles emerge
Stage 5: Adjourning	Dissolution of the group	

Establishing a Team Identity

Effective Use of Meetings

Co-location of team members

Creation of project team name

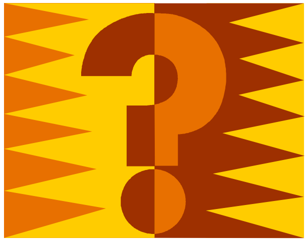
Team rituals

Individual Assignment

- Fill out the column labeled Individual Rank of Lost At Sea Worksheet
- No discussion allowed
- Put your name at the top

Team Assignment

- As a group, discuss and decide on a common group decision for Lost At Sea
- Fill In The Column labeled Group Rank



Teamwork: Simply stated, it is less me and more we.

TEAM = Together Everyone Achieves More

Coming together,
sharing together,
working together,
succeeding together.

A group becomes a team when each member is sure enough of himself and his contribution to praise the skills of the others.

Thomas Edison, when asked why he had a team of twenty-one assistants, **"If I could solve all the problems myself, I would."**

Major Tasks of Project Closure

1. Evaluate if the project delivered the expected benefits to all stakeholders.
   - Was the project managed well?
   - Was the customer satisfied?
2. Assess what was done wrong and what contributed to successes.
3. Identify changes to improve the delivery of future projects.

Project Closure and Review Deliverables

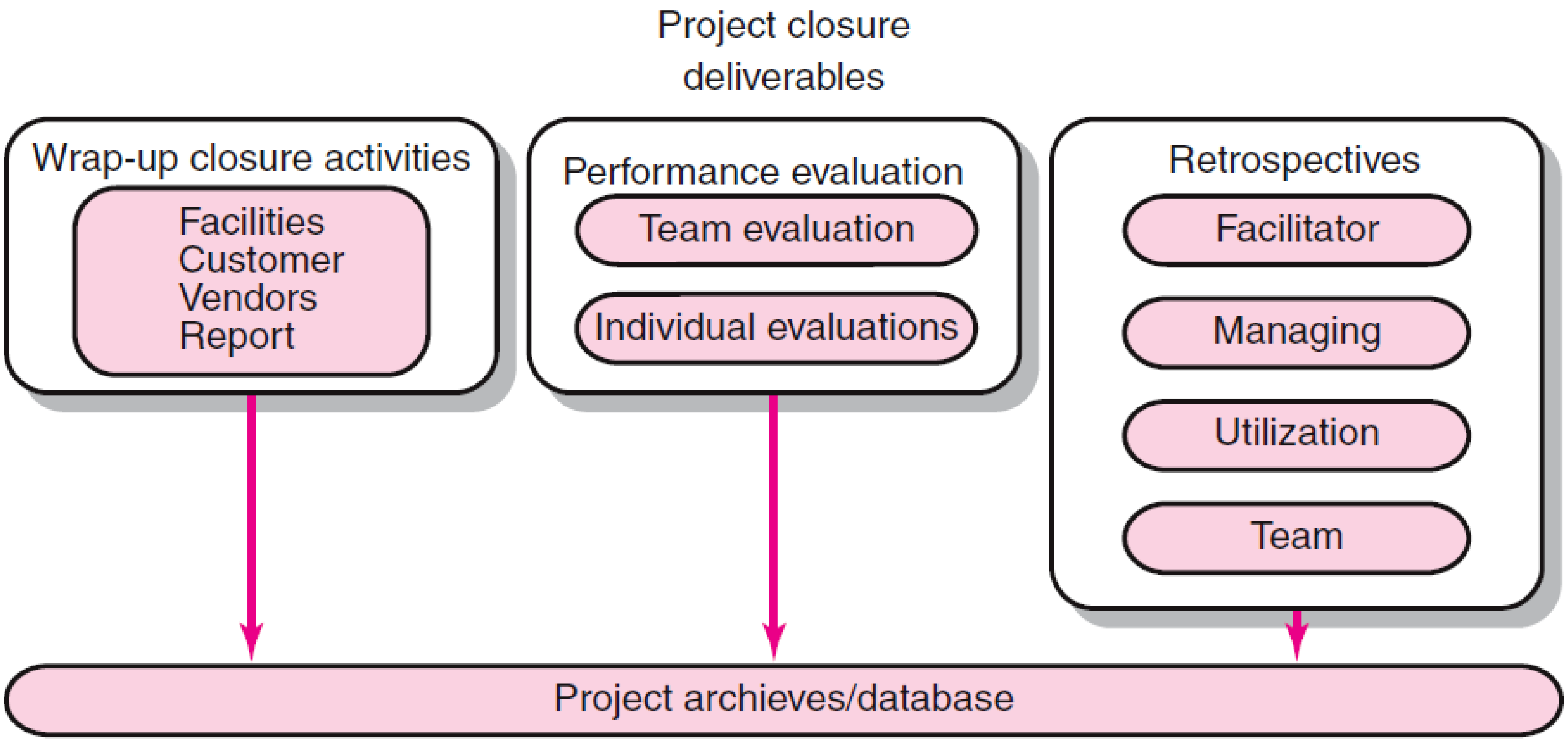


FIGURE 14.1

Project Closure

- Types of Project Closure
  - □ Normal
  - □ Premature
  - □ Perpetual
  - □ Failed Project
  - □ Changed Priority

- Close-out Plan: Questions to be Asked
  - □ What tasks are required to close the project?
  - □ Who will be responsible for these tasks?
  - □ When will closure begin and end?
  - □ How will the project be delivered?

Creating the Final Report

- Executive Summary
  - Project goals met/unmet
  - Stakeholder satisfaction with project
  - User reactions to quality of deliverables
- Analysis
  - Project mission and objective
  - Procedures and systems used
  - Organization resources used
- Recommendations
  - Technical improvements
  - Corrective actions
- Lessons Learned
  - Reminders
  - Retrospectives
- Appendix
  - Backup data
  - Critical information

Retrospectives

- Lessons Learned
  - □ An analysis carried out during and shortly after the project life cycle to capture positive and negative project learning——what worked and what didn‘t?”
- Goals of Retrospectives
  - □ To reuse learned solutions
  - □ To stop repetitive mistakes

Project Process Review Questionnaire

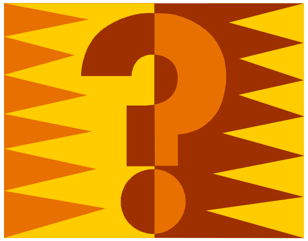
1. Were the project objectives and strategic intent of the project clearly and explicitly communicated?
2. Were the objectives and strategy in alignment?
3. Were the stakeholders identified and included in the planning?
4. Were project resources adequate for this project?
5. Were people with the right skill sets assigned to this project?
6. Were time estimates reasonable and achievable?
7. Were the risks for the project appropriately identified and assessed before the project started?
8. Were the processes and practices appropriate for this type of project? Should projects of similar size and type use these systems? Why/why not?
9. Did outside contractors perform as expected? Explain.
10. Were communication methods appropriate and adequate among all stakeholders? Explain.
11. Is the customer satisfied with the project product?
12. Are the customers using the project deliverables as intended? Are they satisfied?
13. Were the project objectives met?
14. Are the stakeholders satisfied their strategic intents have been met?
15. Has the customer or sponsor accepted a formal statement that the terms of the project charter and scope have been met?
16. Were schedule, budget, and scope standards met?
17. Is there any one important area that needs to be reviewed and improved upon? Can you identify the cause?

TABLE 14.3

Organizational Culture Review Questionnaire

1. Was the organizational culture supportive for this type of project?
2. Was senior management support adequate?
3. Were people with the right skills assigned to this project?
4. Did the project office help or hinder management of the project? Explain.
5. Did the team have access to organizational resources (people, funds, equipment)?
6. Was training for this project adequate? Explain.
7. Were lessons learned from earlier projects useful? Why? Where?
8. Did the project have a clear link to organizational objectives? Explain.
9. Was project staff properly reassigned?
10. Was the Human Resources Office helpful in finding new assignments? Comment.

TABLE 14.4



Metrics

Express in Numbers

Measurement provides a mechanism for *objective* evaluation

Software *Crisis*

- According to American Programmer, 31.1% of computer software projects get canceled before they are completed,
- 52.7% will overrun their initial cost estimates by 189%.
- 94% of project start-ups are restarts of previously failed projects.

- ***Solution?***
  systematic approach to ***software development and measurement***

Software Metrics

- **It refers to a broad range of quantitative measurements for computer software that enable to**
  - **improve the software process continuously**
  - **assist in quality control and productivity**
  - **assess the quality of technical products**
  - **assist in tactical decision-making**

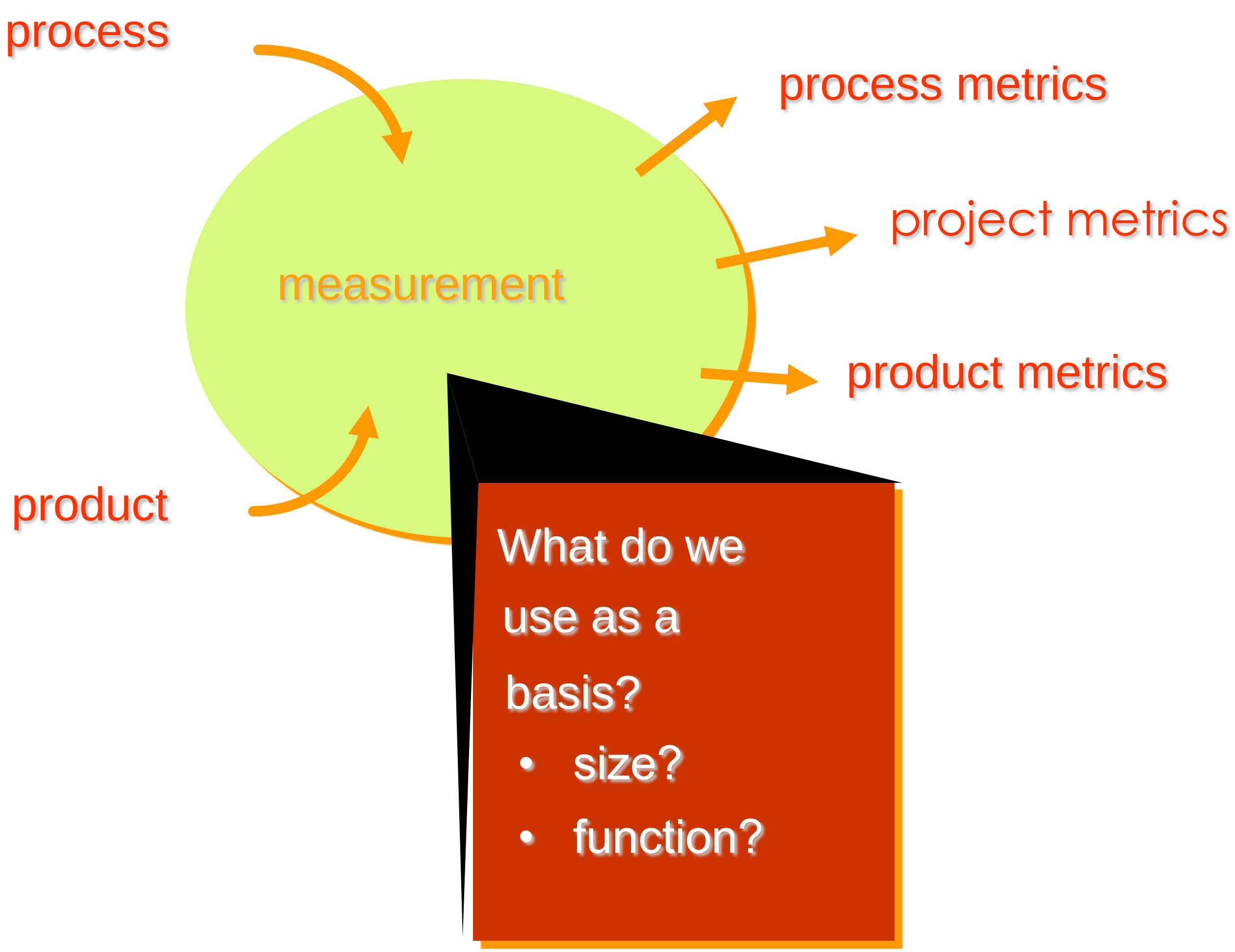
Measure, Metrics, Indicators

- **Measure.**
  - provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attributes of a product or process.
- **Metrics.**
  - relates the individual measures in some way.
- **Indicator.**
  - a combination of metrics that provide insight into the software process or project or product itself.

Motivation for Metrics

- Estimate the cost & schedule of future projects
- Evaluate the productivity impacts of new tools and techniques
- Establish productivity trends over time
- Improve software quality
- Forecast future staffing needs
- Anticipate and reduce future maintenance needs

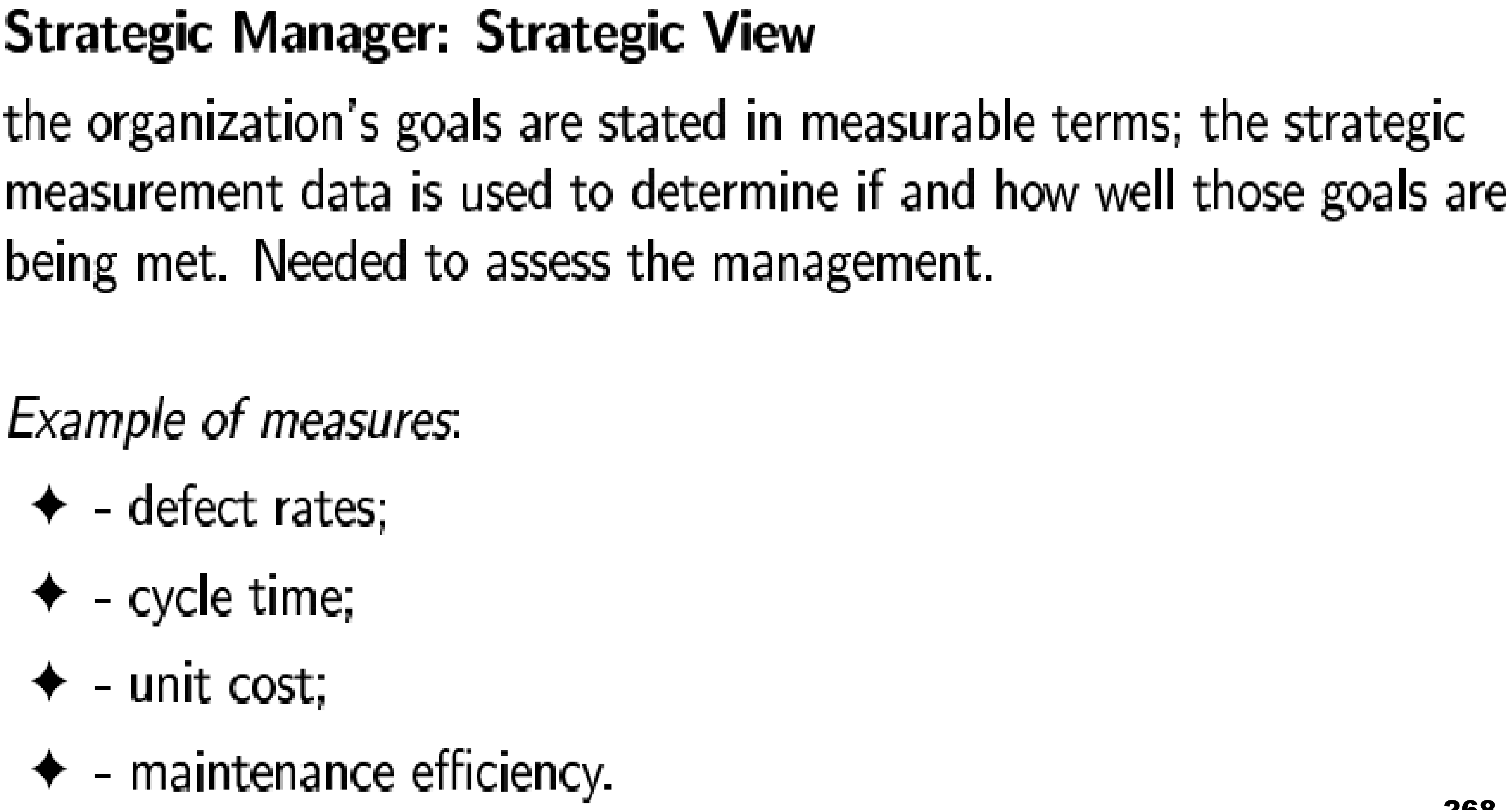
What Should Be Measured?



What do we use as a basis?

- size?
- function?

Views on SE Measurement



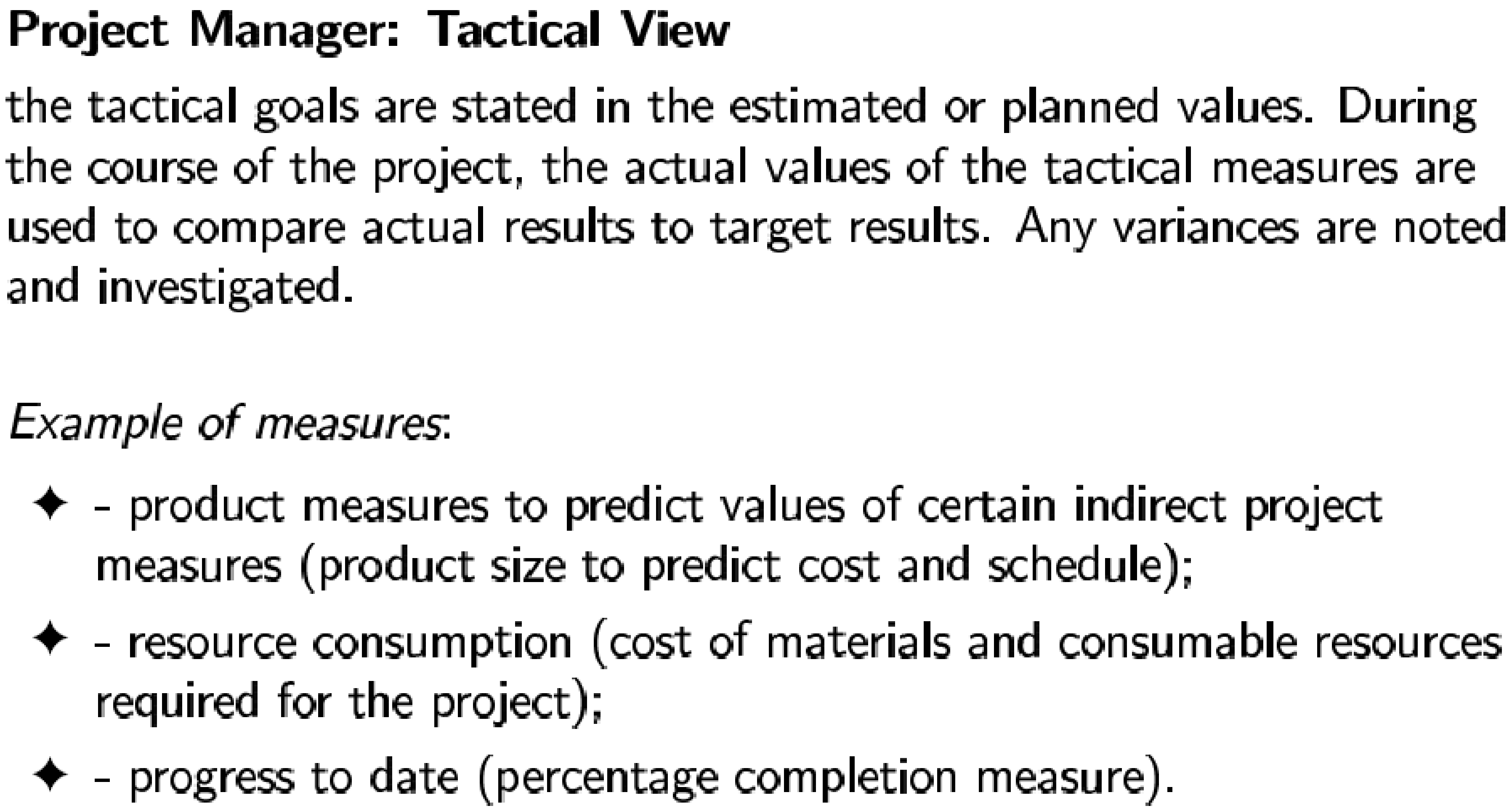
Strategic Manager: Strategic View

the organization's goals are stated in measurable terms; the strategic measurement data is used to determine if and how well those goals are being met. Needed to assess the management.

Example of measures:

- ✦ - defect rates;
- ✦ - cycle time;
- ✦ - unit cost;
- ✦ - maintenance efficiency.

Views on SE Measurement



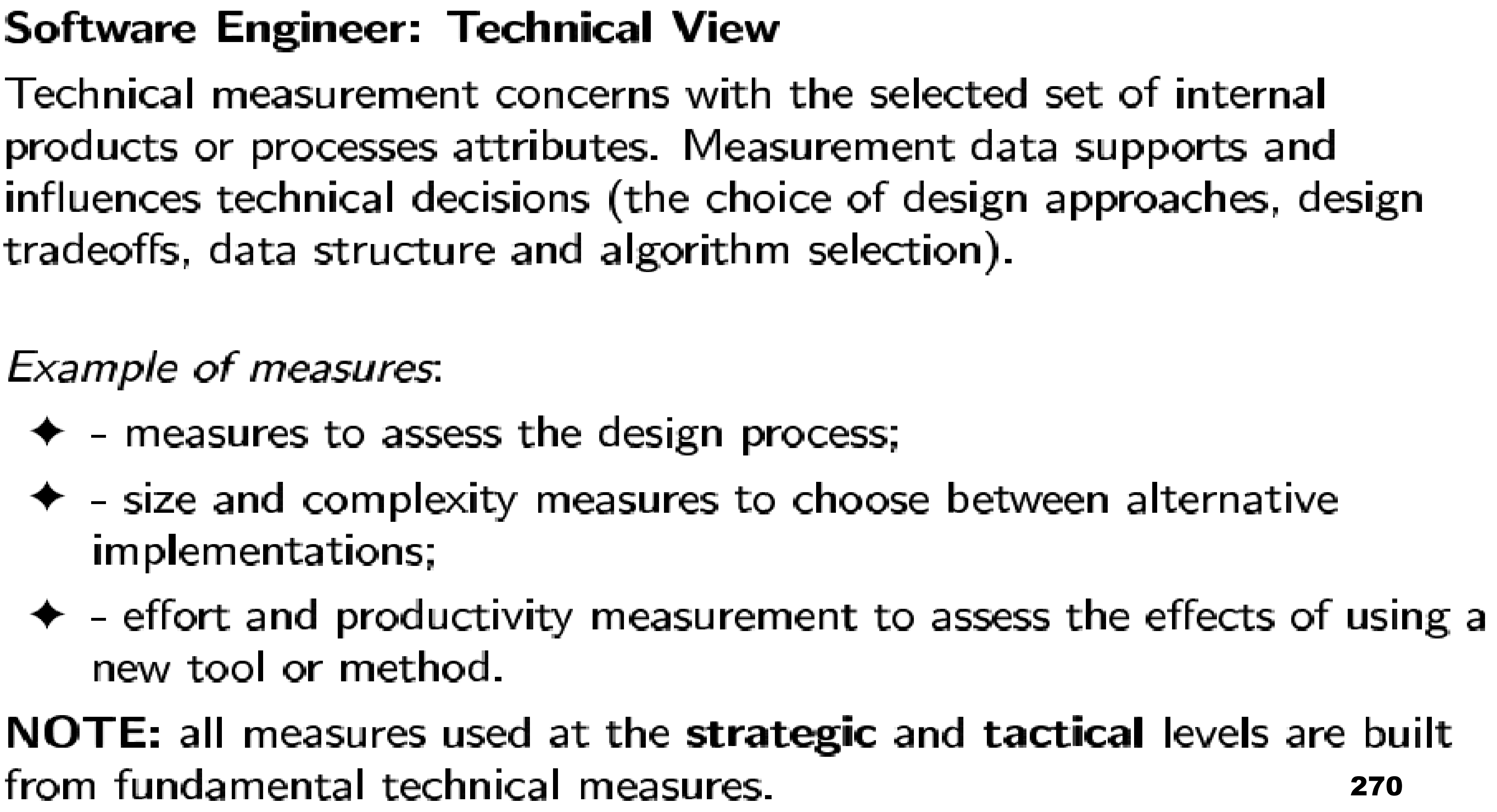
Project Manager: Tactical View

the tactical goals are stated in the estimated or planned values. During the course of the project, the actual values of the tactical measures are used to compare actual results to target results. Any variances are noted and investigated.

Example of measures:

- ✦ - product measures to predict values of certain indirect project measures (product size to predict cost and schedule);
- ✦ - resource consumption (cost of materials and consumable resources required for the project);
- ✦ - progress to date (percentage completion measure).

Views on SE Measurement



Software Engineer: Technical View

Technical measurement concerns with the selected set of internal products or processes attributes. Measurement data supports and influences technical decisions (the choice of design approaches, design tradeoffs, data structure and algorithm selection).

Example of measures:

✦ - measures to assess the design process;

✦ - size and complexity measures to choose between alternative implementations;

✦ - effort and productivity measurement to assess the effects of using a new tool or method.

NOTE: all measures used at the **strategic** and **tactical** levels are built from fundamental technical measures.

Metrics of Project Management

- Budget
- Schedule/ReResource Management
- Risk Management
- Project goals met or exceeded
- Customer satisfaction

Metrics of the Software Product

- Focus on Deliverable Quality
- Analysis Products
- Design Product Complexity – algorithmic, architectural, data flow
- Code Products
- Production System

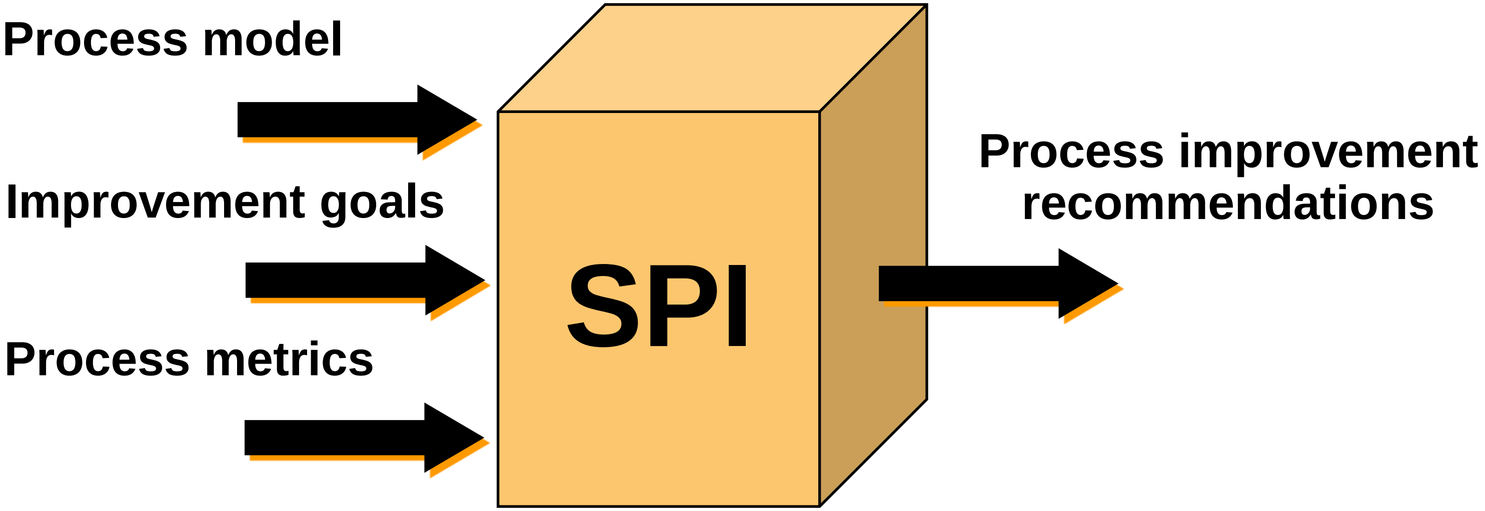
Process Measurement

- We measure the efficacy of a software process indirectly.
  - That is, we derive a set of metrics based on the outcomes that can be derived from the process.
  - Outcomes include
    - measures of errors uncovered before release of the software
    - defects delivered to and reported by end-users
    - work products delivered (productivity)
    - human effort expended
    - calendar time expended
    - schedule conformance
    - other measures.
- We also derive process metrics by measuring the characteristics of specific software engineering tasks.

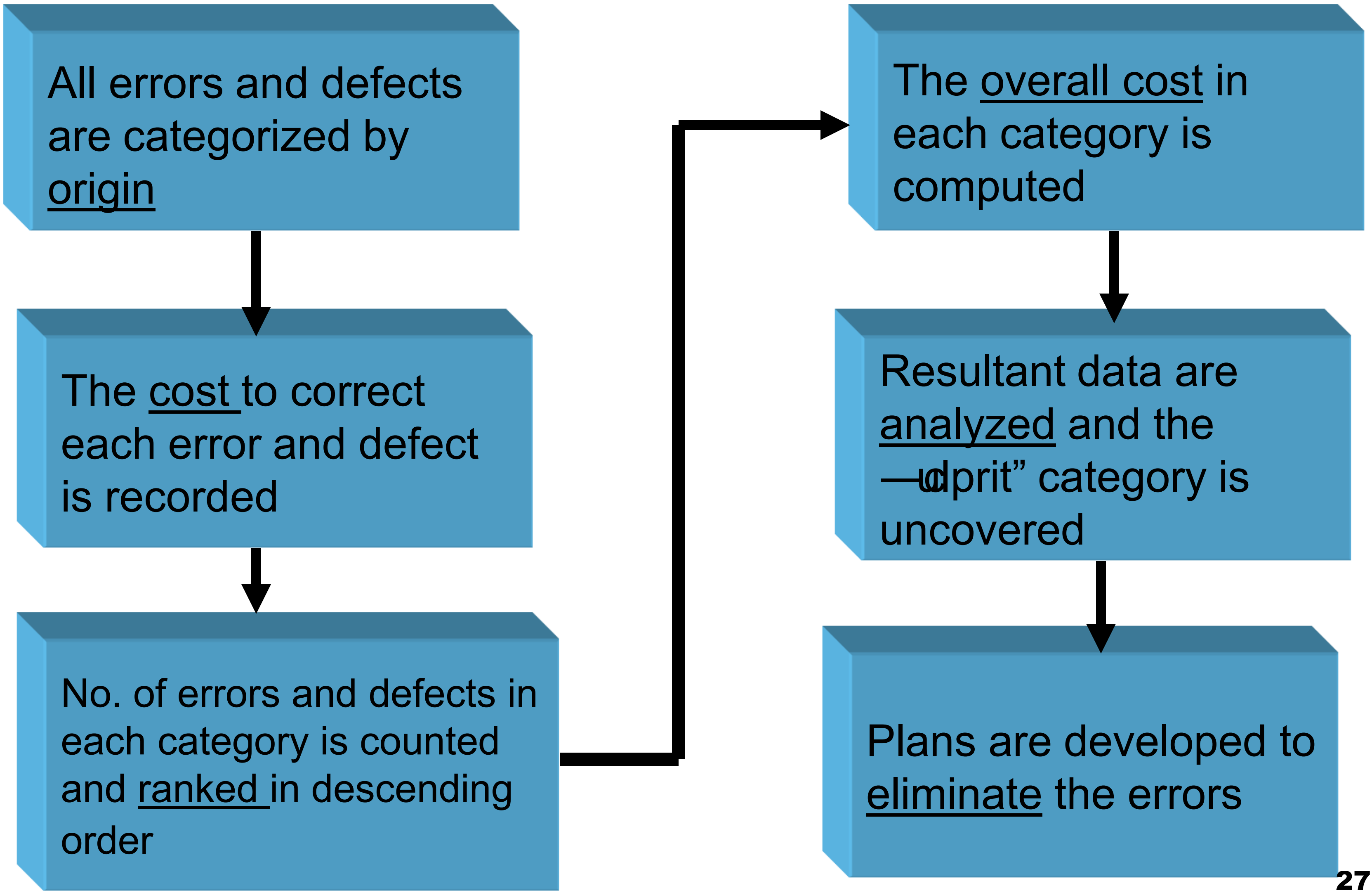
Process Metrics Guidelines

- 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 appraise 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 area should not be considered —negative.” These data are merely an indicator for process improvement.
- Don‘t obsess on a single metric to the exclusion of other important metrics.

Software Process Improvement



Statistical Software Process Improvement



All errors and defects are categorized by origin

The cost to correct each error and defect is recorded

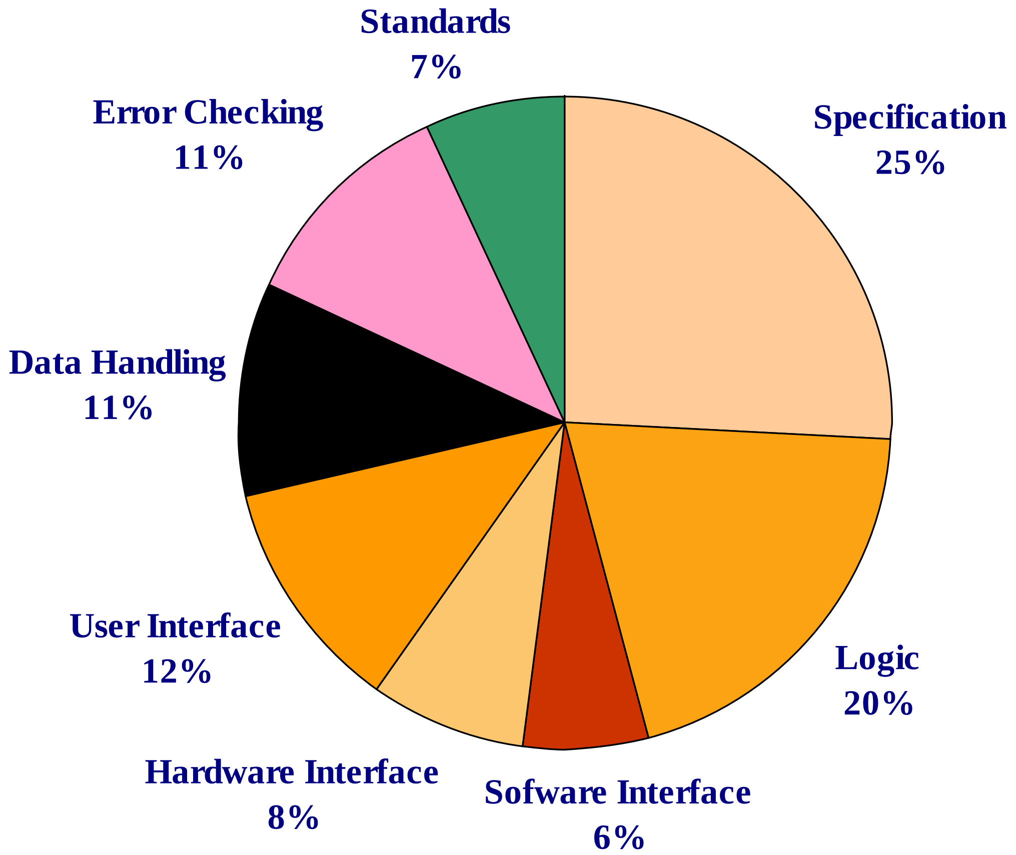
No. of errors and defects in each category is counted and ranked in descending order

The overall cost in each category is computed

Resultant data are analyzed and the —culprit” category is uncovered

Plans are developed to eliminate the errors

Causes and Origin of Defects



Defect Removal Efficiency

DRE = *E* /(*E* + *D*)

where:

E is the number of errors found before delivery of the software to the end-user

D is the number of defects found after delivery.

Measuring Quality

- Correctness — the degree to which a program operates according to specification
- Maintainability—the degree to which a program is amenable to change
- Integrity—the degree to which a program is impervious to outside attack
- Usability—the degree to which a program is easy to use

Source Lines of Code (SLOC)

- Measures the number of physical lines of active code

- In general the higher the SLOC in a module the less understandable and maintainable the module is

Function Oriented Metric - Function Points

- Function Points are a measure of —how big” is the program, independently from the actual physical size of it
- It is a weighted count of several features of the program
- Dislikers claim FP make no sense wrt the representational theory of measurement
- There are firms and institutions taking them very seriously

			weighting factor				
measurement parameter	count		simple	avg.	complex		
number of user inputs	□	X	3	4	6	=	□
number of user outputs	□	X	4	5	7	=	□
number of user inquiries	□	X	3	4	6	=	□
number of files	□	X	7	10	15	=	□
number of ext.interfaces	□	X	5	7	10	=	□
count-total							□
complexity multiplier							□
function points							□

$$\sum_{Inputs} Wi + \sum_{Output} Wo + \sum_{Inquiry} Win + \sum_{InternalFiles} Wif + \sum_{ExternalInterfaces} Wei$$

Factors are rated on a scale of 0 (not important) to 5 (very important):

data communications
distributed functions
heavily used configuration
transaction rate
on-line data entry
end user efficiency
on-line update
complex processing
installation ease
operational ease
multiple sites
facilitate change

Formula:

$$CM = \sum_{ComplexityMultiplier} F_{ComplexityMultiplier}$$

Typical Function-Oriented Metrics

- errors per FP (thousand lines of code)
- defects per FP
- $ per FP
- pages of documentation per FP
- FP per person-month

LOC vs. FP

- Relationship between lines of code and function points depends upon the programming language that is used to implement the software and the quality of the design

- Empirical studies show an approximate relationship between LOC and FP

Comparing LOC and FP

Programming	LOC per Function point			
Language	avg.	median	low	high
Ada	154	-	104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
Java	63	53	77	-
JavaScript	58	63	42	75
Perl	60	-	-	-
PL/1	78	67	22	263
Powerbuilder	32	31	11	105
SAS	40	41	33	49
Smalltalk	26	19	10	55
SQL	40	37	7	110
Visual Basic	47	42	16	158

LOC/FP (average)

Assembly language	320
C	128
COBOL, FORTRAN	106
C++	64
Visual Basic	32
Smalltalk	22
SQL	12
Graphical languages (icons)	4

Typical Size-Oriented Metrics

- errors per KLOC (thousand lines of code)
- defects per KLOC
- $ per LOC
- pages of documentation per KLOC
- errors per person-month
- errors per review hour
- LOC per person-month
- $ per page of documentation

Typical Function-Oriented Metrics

- errors per FP (thousand lines of code)
- defects per FP
- $ per FP
- pages of documentation per FP
- FP per person-month

12 Steps to Useful Software Metrics

Step 1 - Identify Metrics Customers

Step 2 - Target Goals

Step 3 - Ask Questions

Step 4 - Select Metrics

Step 5 - Standardize Definitions

Step 6 - Choose a Model

Step 7 - Establish Counting Criteria

Step 8 - Decide On Decision Criteria

Step 9 - Define Reporting Mechanisms

Step 10 - Determine Additional Qualifiers

Step 11 - Collect Data

Step 12 - Consider Human Factors

Step 1 - Identify Metrics Customers

Who needs the information?

Who‘s going to use the metrics?

If the metric does not have a customer -- do not use it.

Step 2 - Target Goals

Organizational goals

- □ **Be the low cost provider**
- □ **Meet projected revenue targets**

Project goals

- □ **Deliver the product by June 1st**
- □ **Finish the project within budget**

Task goals (entry & exit criteria)

- □ **Effectively inspect software module ABC**
- □ **Obtain 100% statement coverage during testing**

Step 3 - Ask Questions

Goal: Maintain a high level of customer satisfaction

- What is our current level of customer satisfaction?
- What attributes of our products and services are most important to our customers?
- How do we compare with our competition?

Step 4 - Select Metrics

Select metrics that provide information to help answer the questions

- Be practical, realistic, pragmatic
- Consider current engineering environment
- Start with the possible

Metrics don‘t solve problems
-- people solve problems

Metrics provide information so people can make better decisions

Step 5 - Standardize Definitions

Developer

User

Step 6 - Choose a Measurement

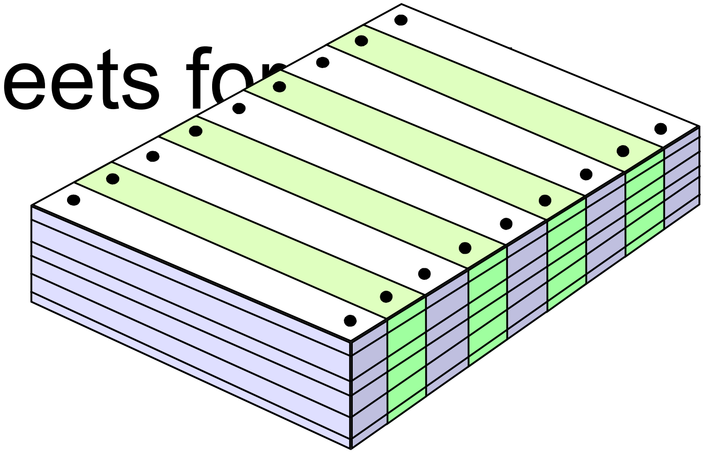
Models for code inspection metrics

- Primitive Measurements:
  - **Lines of Code Inspected = loc**
  - **Hours Spent Preparing = prep_hrs**
  - **Hours Spent Inspecting = in_hrs**
  - **Discovered Defects = defects**
- Other Measurements**:**
  - **Preparation Rate = loc / prep_hrs**
  - **Inspection Rate = loc / in_hrs**
  - **Defect Detection Rate = defects / (prep_hrs + in_hrs)**

Step 7 - Establish Counting Criteria

Lines of Code

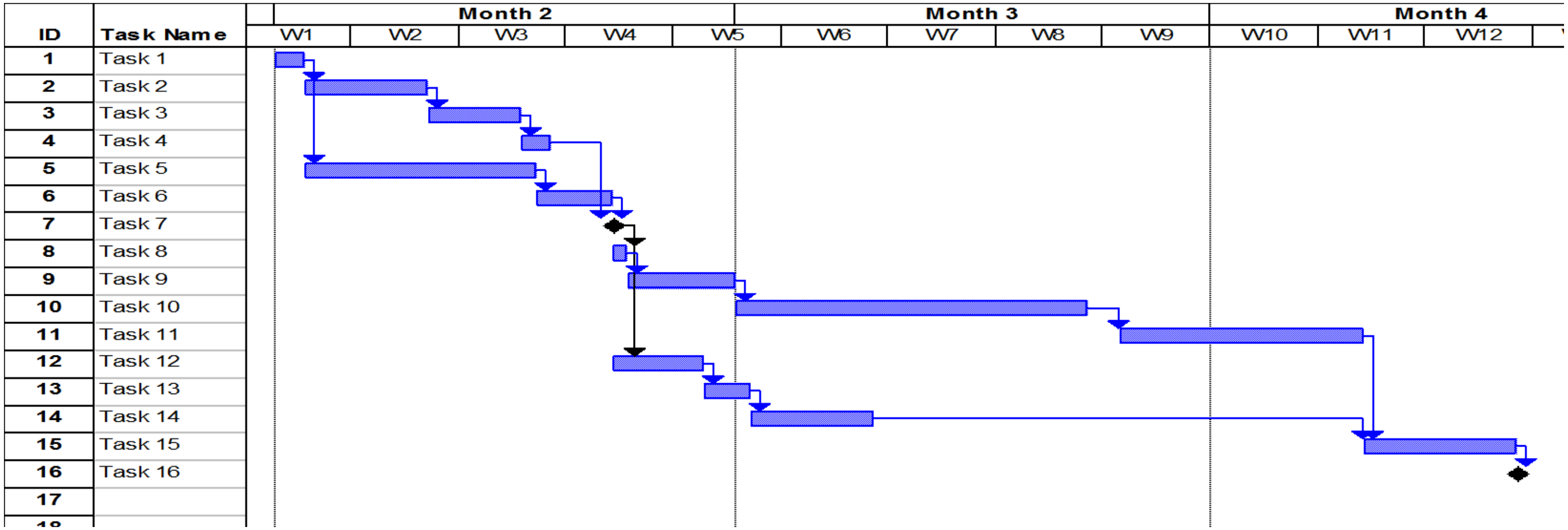
- Variations in counting
- No industry accepted standard
- SEI guideline - check sheets for
- Advice: use a tool



Counting Criteria - Effort

What is a Software Project?

- When does it start / stop?
- What activities does it include?



ID	Task Name
1	Task 1
2	Task 2
3	Task 3
4	Task 4
5	Task 5
6	Task 6
7	Task 7
8	Task 8
9	Task 9
10	Task 10
11	Task 11
12	Task 12
13	Task 13
14	Task 14
15	Task 15
16	Task 16
17	

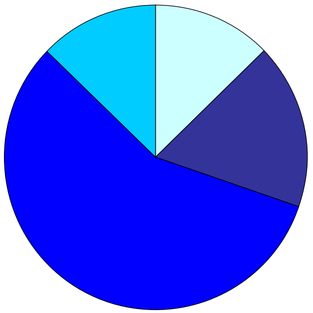
Step 8 - Decide On Decision Criteria

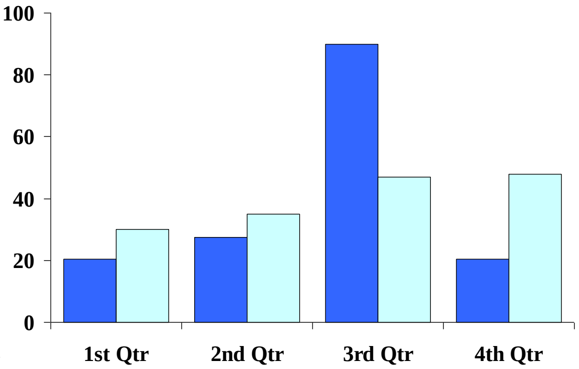
Establish Baselines

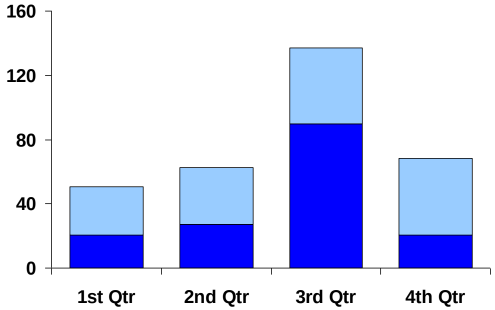
- Current value
  - Problem report backlog
  - Defect prone modules
- Statistical analysis (mean & distribution)
  - Defect density
  - Fix response time
  - Cycle time
  - Variance from budget (e.g., cost, schedule)

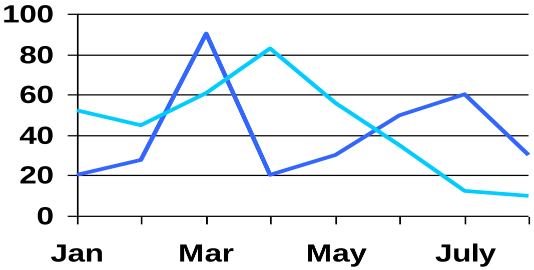
Step 9 - Define Reporting Mechanisms

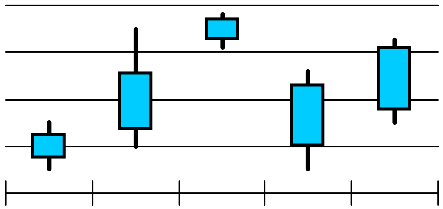
	Open	Fixed	Resolved
Jan-97	23	13	3
Feb-97	27	24	11
Mar-97	18	26	15
Apr-97	12	18	27

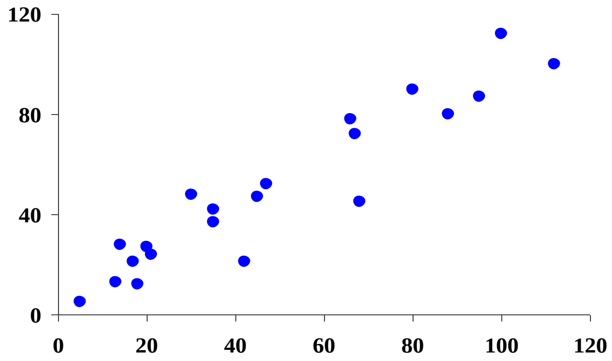


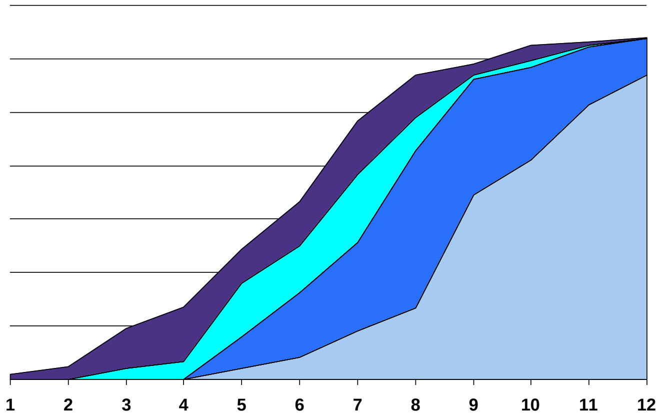












Step 10 –
Determine Additional Qualifiers

A good metric is a generic metric

Additional qualifiers:

- Provide demographic information
- Allow detailed analysis at multiple levels
- Define additional data requirements

Additional Qualifier Example

Metric: *software defect arrival rate*

- Release / product / product line
- Module / program / subsystem
- Reporting customer / customer group
- Root cause
- Phase found / phase introduced
- Severity

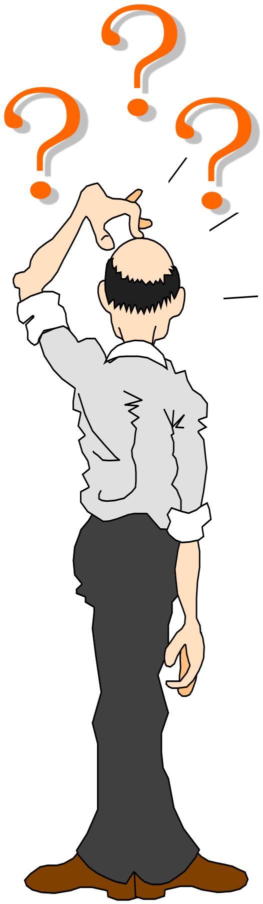
Step 11 – Collect Data

What data to collect?

- Metric primitives
- Additional qualifiers

Who should collect the data?

- The data owner
  - Direct access to source of data
  - Responsible for generating data
  - Owners more likely to detect anomalies
  - Eliminates double data entry



Examples of Data Ownership

Owner	Examples of Data Owned
• Management	• Schedule • Budget
• Engineers	• Time spent per task • Inspection data including defects found • Root cause of defects
• Testers	• Test Cases planned / executed / passed • Problems • Test coverage
• Configuration management specialists	• Lines of code • Modules changed
• Users	• Problems • Operation hours

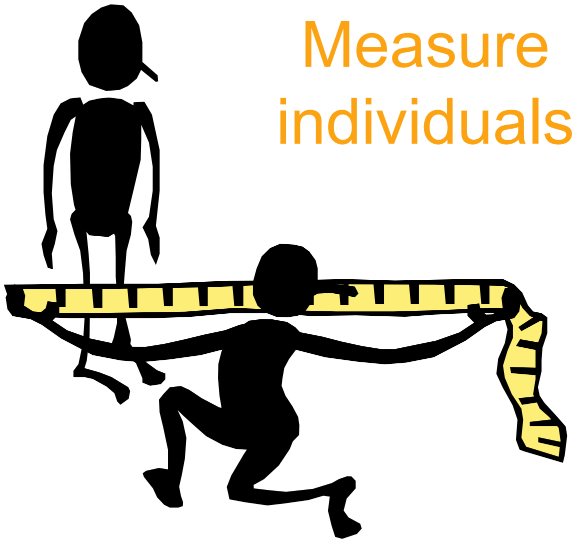
Step 12 – Consider Human Factors

The People Side of the Metrics Equation

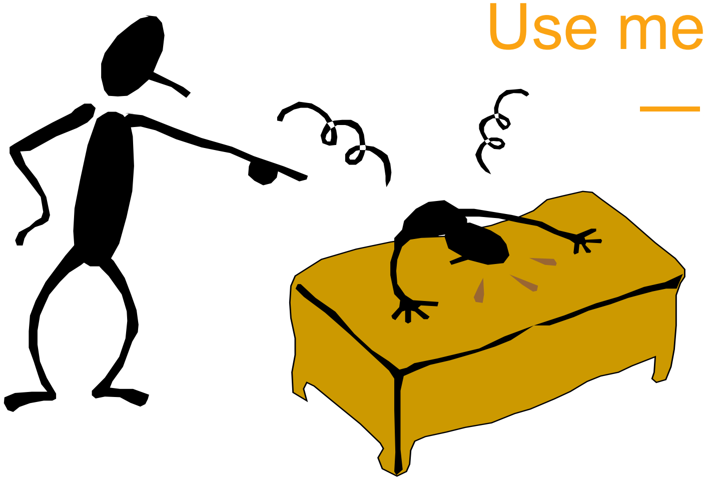
- How measures affect people
- How people affect measures

—Don‘t underestimate the intelligence of your engineers. For any one metric you can come up with, they will find at least two ways to beat it.” [unknown]

Don‘t

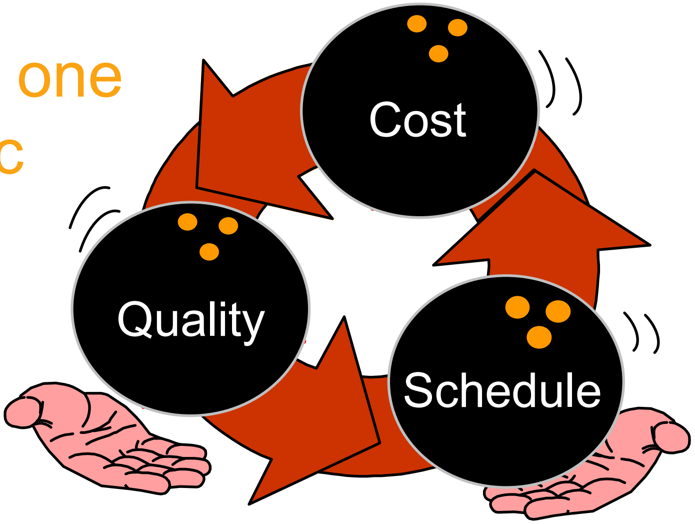


Measure individuals



Use metrics as a —stick”

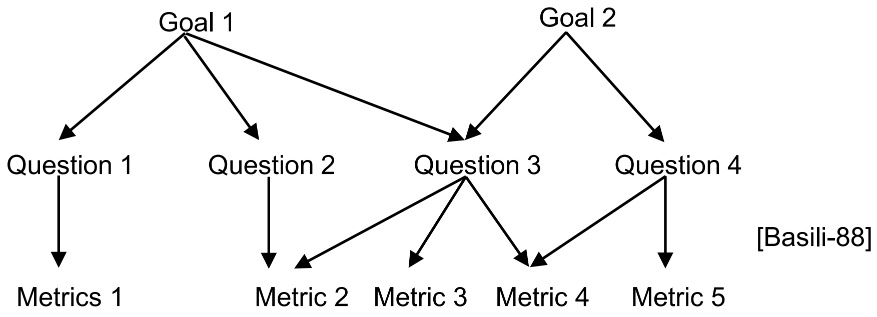
Use only one metric



Ignore the data

Do

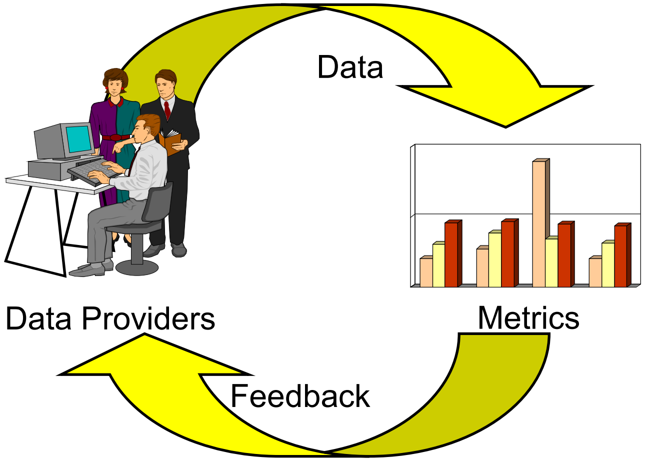
Select metrics based on goals



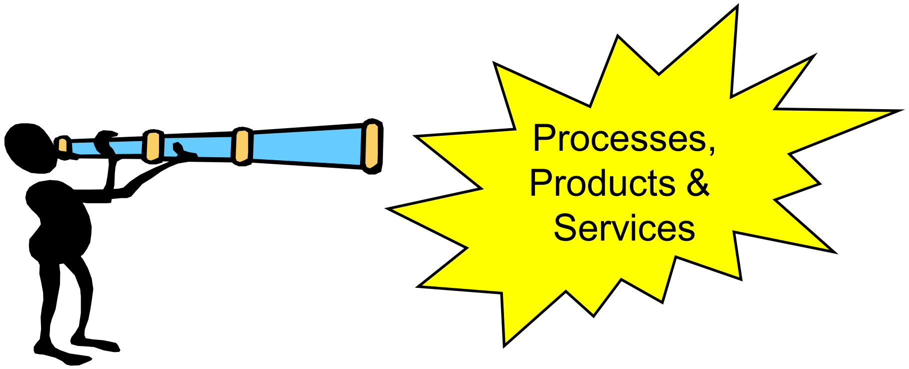
Goal 1
Goal 2
Question 1
Question 2
Question 3
Question 4
Metrics 1
Metric 2
Metric 3
Metric 4
Metric 5

[Basili-88]

Provide feedback



Data
Data Providers
Metrics
Feedback



Processes, Products & Services

Focus on processes, products & services

Obtain —buy-in”