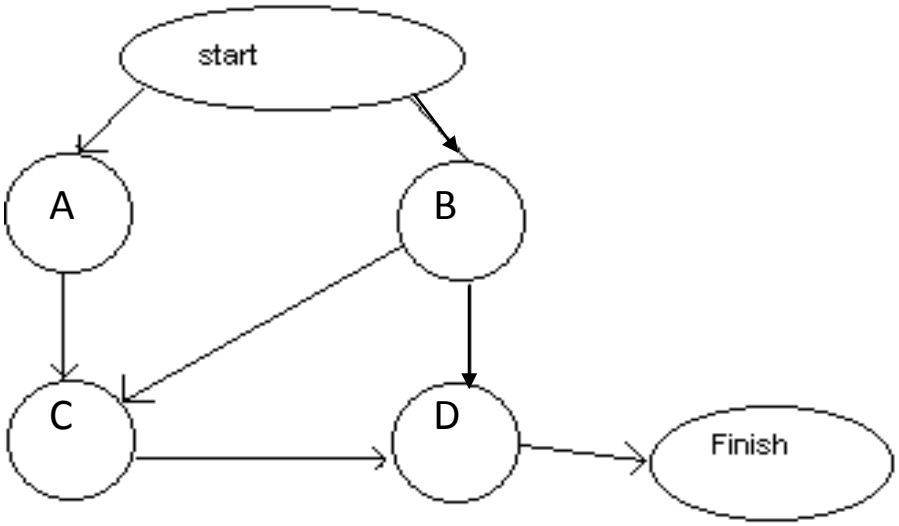
Scheduling, Tracking & Monitoring Tools

- There are various tools that can be used to schedule a project and track its progress.
- Some of them are
  - Activity graph
  - PERT/CPM diagrams
  - GANTT charts
  - Earned Value Analysis
  - Error tracking

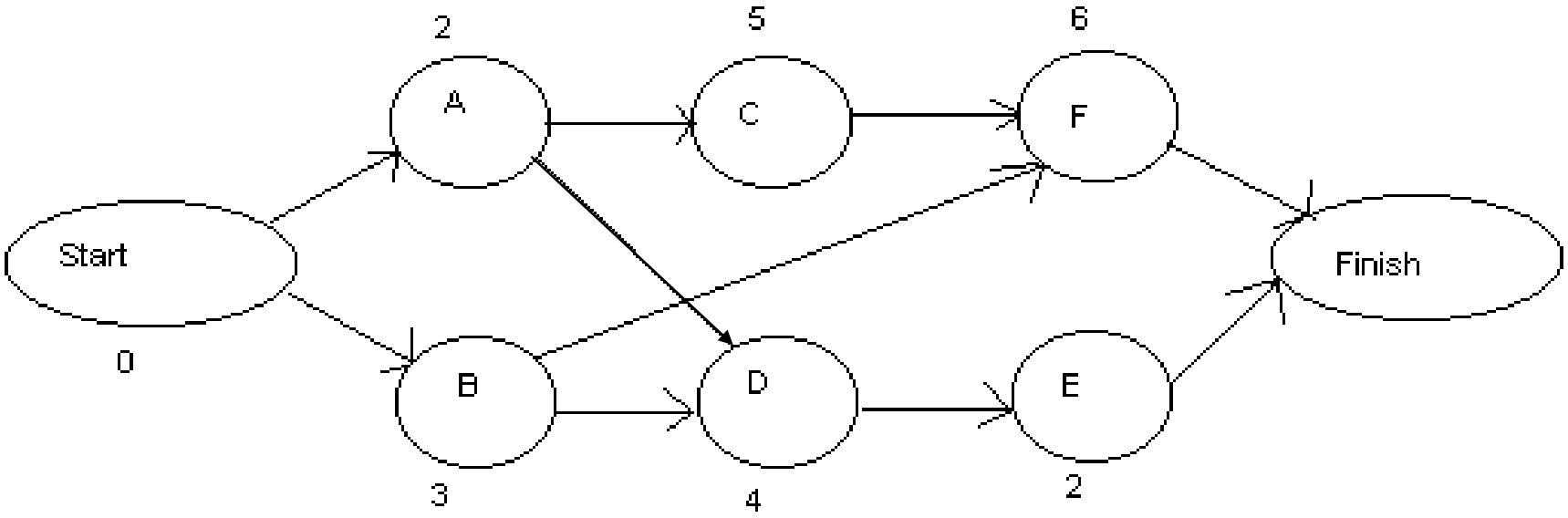
Activity Graph



Each activity has:

1. Precursor
2. Duration
3. Due date
4. End point (milestone or deliverable)

Activity Graph & Critical Path Method (CPM)



EST = earliest start time, EFT = earliest finish time.

LST = latest start time, LFT = latest finish time.

CPM Equations

EST(START) = 0 always, EFT(START) = 0 always.

LST(START) = 0 always, LFT(START) = 0 always.

EFT(i) = EST(i) + DUR(i).

EST(i) = max(EFT of all predecessors).

LST(i) = LFT(i) - DUR(i).

LFT(i) = min(LST of all successors).

LFT(FINISH) = LST(FINISH) = EST(FINISH) = EFT(FINISH)

Slack(i) = (LST(i) – EST(i)) or (LFT(i) – EFT(i)).

EST = earliest start time, EFT = earliest finish time.

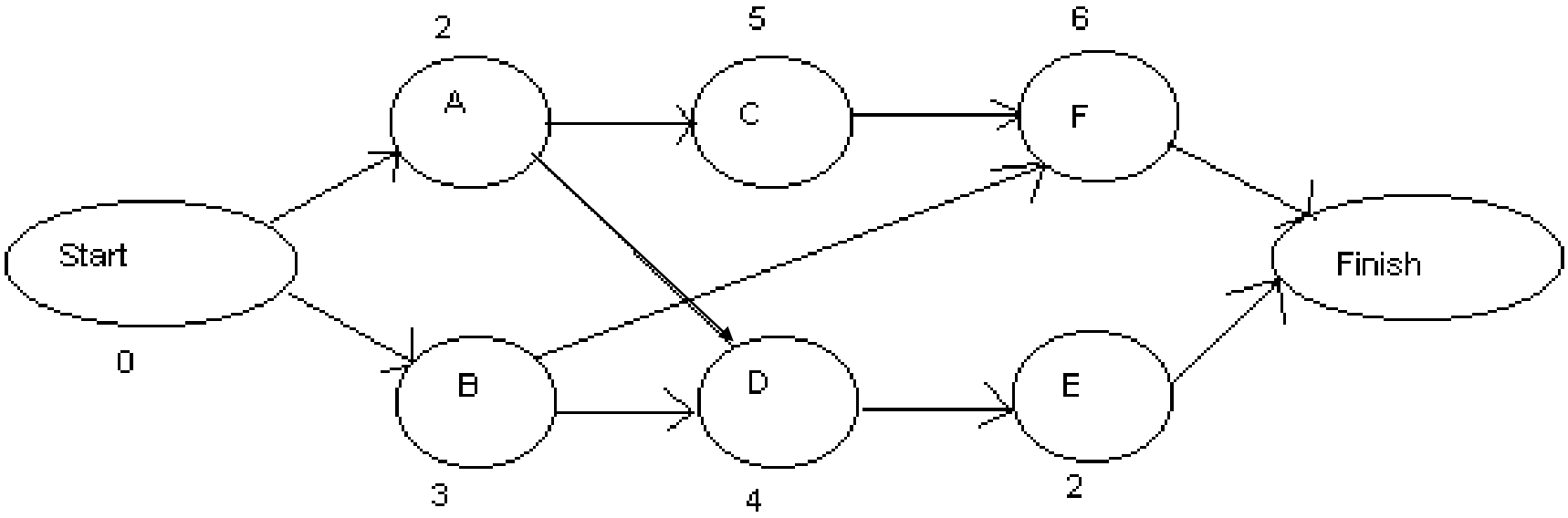
LST = latest start time, LFT = latest finish time.

Activity Graph - Exercise

Activity	Precursor	Duration
Start	-	0
A	Start	4
B	Start	7
C	A	6
D	A,B	3
E	D	5
F	B,C	4
FINISH	E,F	0

Draw the activity graph for the above set of activities.

Activity Graph - Solution



EST = earliest start time, EFT = earliest finish time.

LST = latest start time, LFT = latest finish time.

Activity Graph/CPM - Exercise

Activity	Precursor	Duration	EST	EFT	LST	LFT	Slack
Start	-	0					
A	Start	2					
B	Start	3					
C	A	5					
D	A,B	4					
E	D	2					
F	B,C	6					
FINISH	E,F	0					

Complete the above table

PERT Chart (1/8)

- PERT stands for *Program Evaluation Review Technique*, a methodology developed by the U.S. Navy in the 1950s to manage the Polaris submarine missile program.
- A PERT chart is a project management tool used to schedule, organize, and coordinate tasks within a project.
- For large projects, the dependencies among activities are important to determine which are the critical activities, whose completion should not be delayed.

PERT Chart (2/8)

NOTE: A similar methodology, the *Critical Path Method* (CPM), which was developed for project management in the private sector at about the same time, has become synonymous with PERT, so that the technique is known by any variation on the names: PERT, CPM or PERT/CPM

- A PERT chart is a graph-based chart and can be used to determine the activities that form the “critical path”.

PERT Chart (3/8)

- A PERT chart presents a graphic illustration of a project as a network diagram consisting of numbered *nodes* (either circles or rectangles) representing events, or milestones in the project linked by labeled *vectors* (directional lines) representing tasks in the project.
- The direction of the arrows on the lines indicates the sequence of tasks.
- Numbers on the opposite sides of the vectors indicate the time allotted for the task.

PERT Chart (4/8)

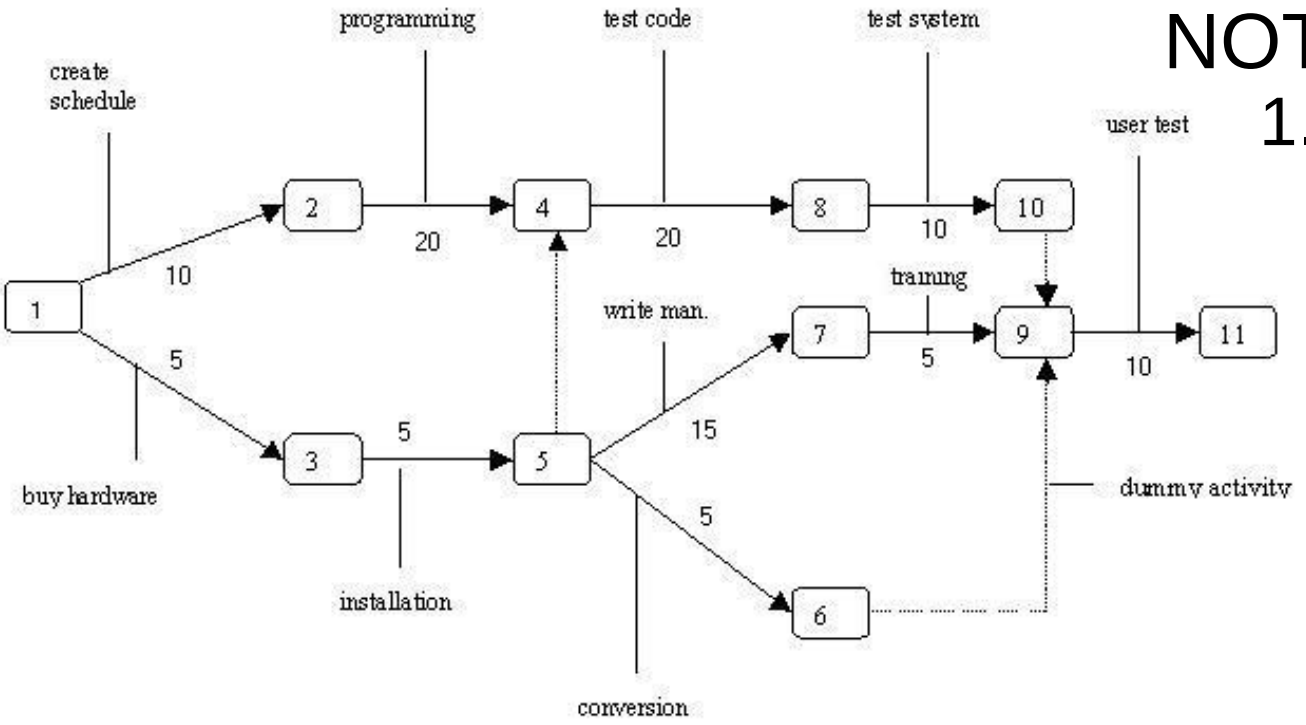


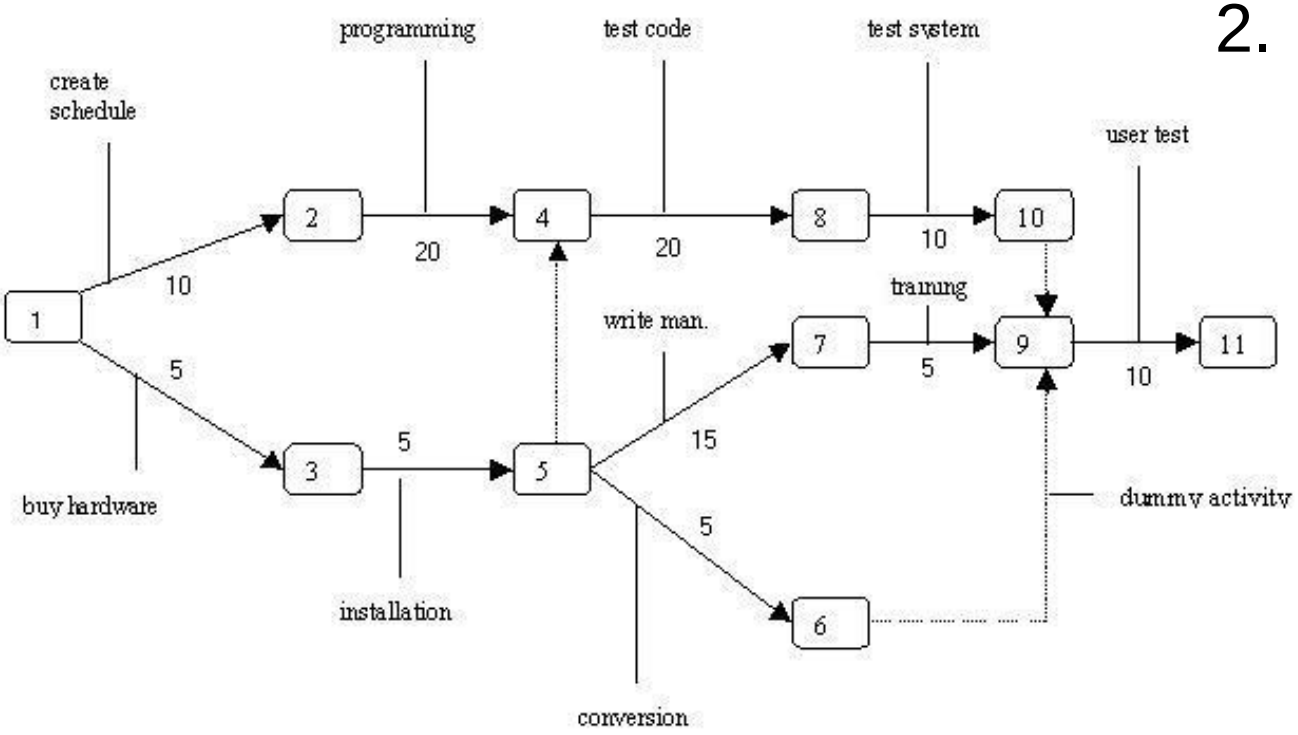
Fig. 1:
PERT Chart

* Numbered rectangles are nodes and represent events or milestones.
* Directional arrows represent dependent tasks that must be completed sequentially.
* Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks
* Dotted lines indicate dependent tasks that do not require resources.

NOTE:

1. The tasks between nodes 1, 2, 4, 8, and 10 must be completed in sequence. These are called *dependent* or *serial* tasks.

PERT Chart (5/8)



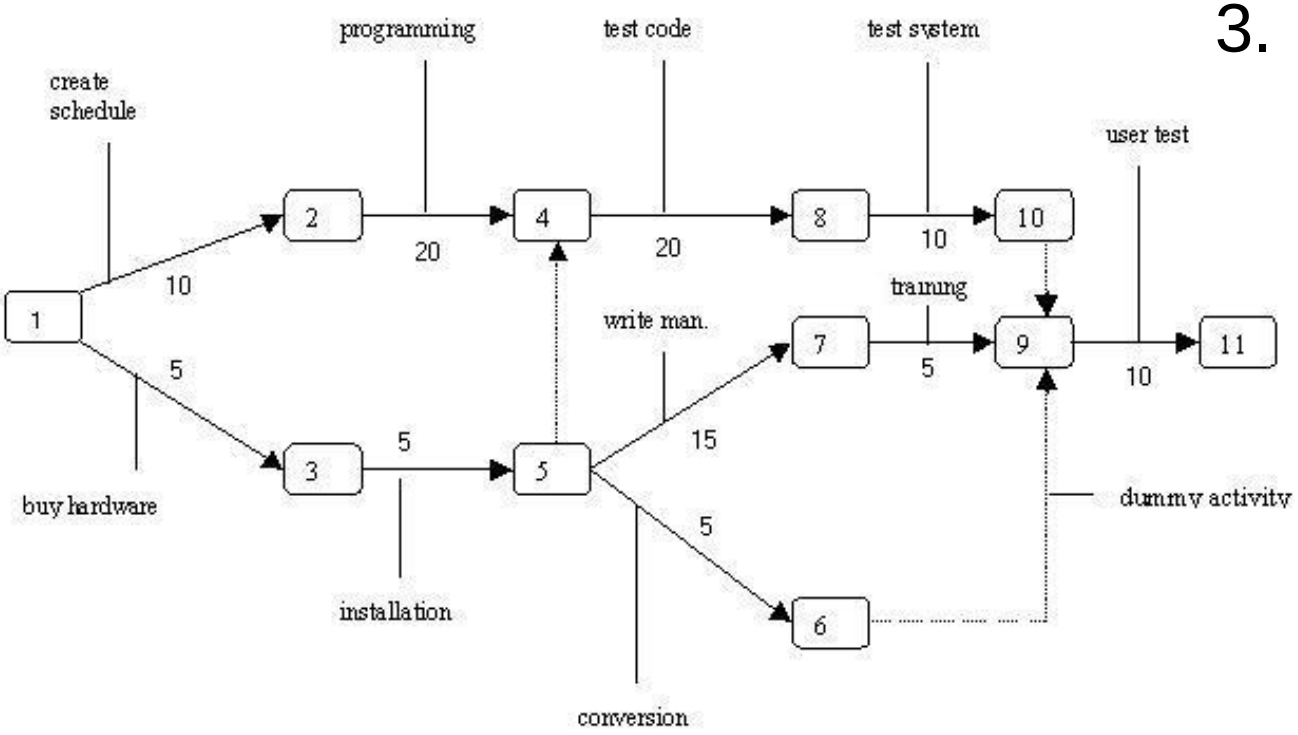
* Numbered rectangles are nodes and represent events or milestones.
* Directional arrows represent dependent tasks that must be completed sequentially.
* Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks
* Dotted lines indicate dependent tasks that do not require resources.

Fig. 1: PERT Chart

NOTE:

2. The tasks between nodes 1 and 2, and nodes 1 and 3 are not dependent on the completion of one to start the other and can be undertaken simultaneously. These tasks are called *parallel* or *concurrent* tasks.

PERT Chart (6/8)



* Numbered rectangles are nodes and represent events or milestones.
* Directional arrows represent dependent tasks that must be completed sequentially.
* Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks
* Dotted lines indicate dependent tasks that do not require resources.

Fig. 1: PERT Chart

NOTE:

3. Tasks that must be completed in sequence but that don't require resources or completion time are considered to have *event dependency*. These are represented by dotted lines with arrows and are called *dummy activities*.

PERT Chart (7/8)

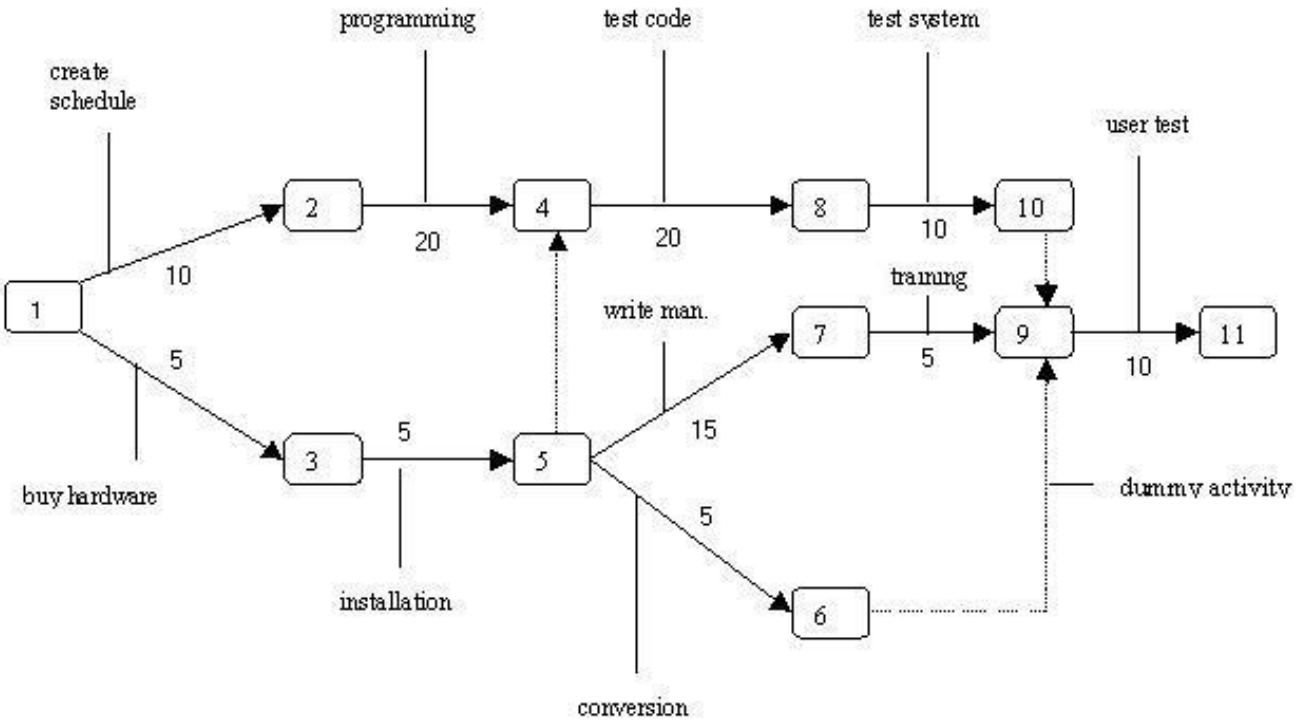
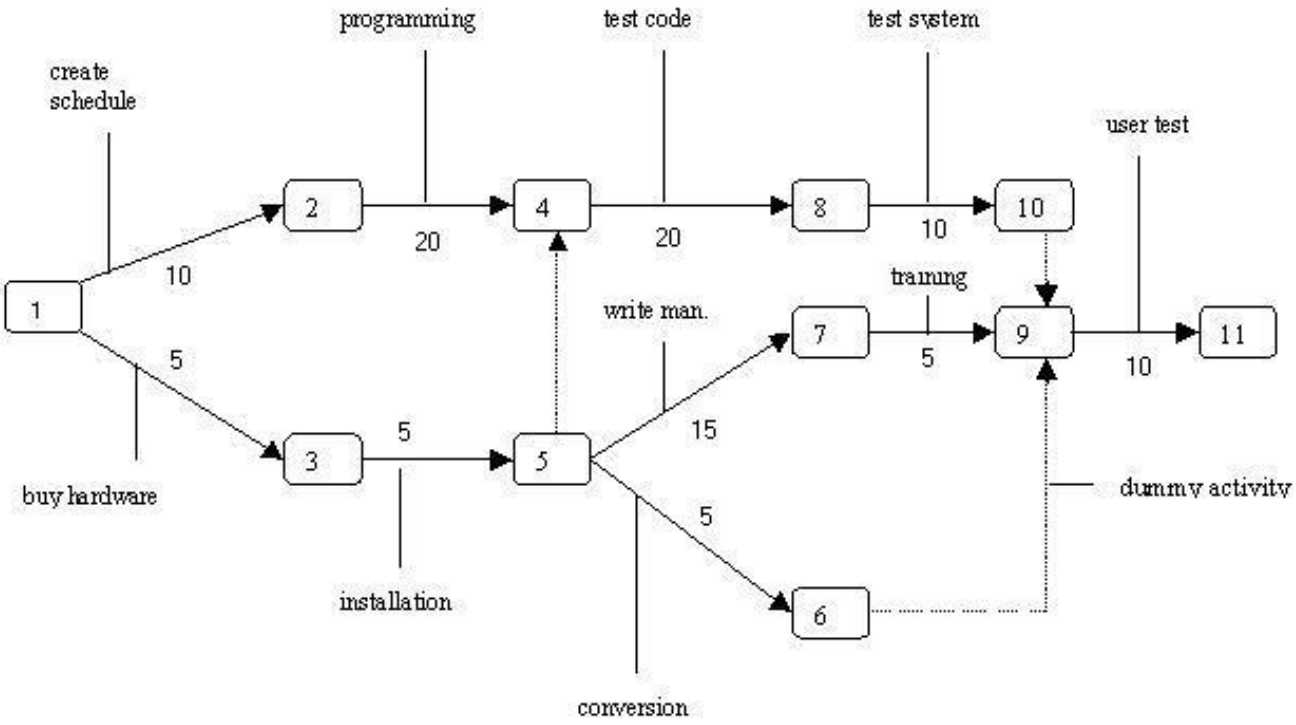


Fig. 1:
PERT Chart

* Numbered rectangles are nodes and represent events or milestones.
* Directional arrows represent dependent tasks that must be completed sequentially.
* Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks
* Dotted lines indicate dependent tasks that do not require resources.

For example, the dashed arrow linking nodes 6 and 9 indicates that the system files must be converted before the user test can take place, but that the resources and time required to prepare for the user test (writing the user manual and user training) are on another path.

PERT Chart (8/8)



Numbers on the opposite sides of the vectors indicate the time allotted for the task.

Fig. 1: PERT Chart

* Numbered rectangles are nodes and represent events or milestones.
* Directional arrows represent dependent tasks that must be completed sequentially.
* Diverging arrow directions (e.g. 1-2 & 1-3) indicate possibly concurrent tasks
* Dotted lines indicate dependent tasks that do not require resources.

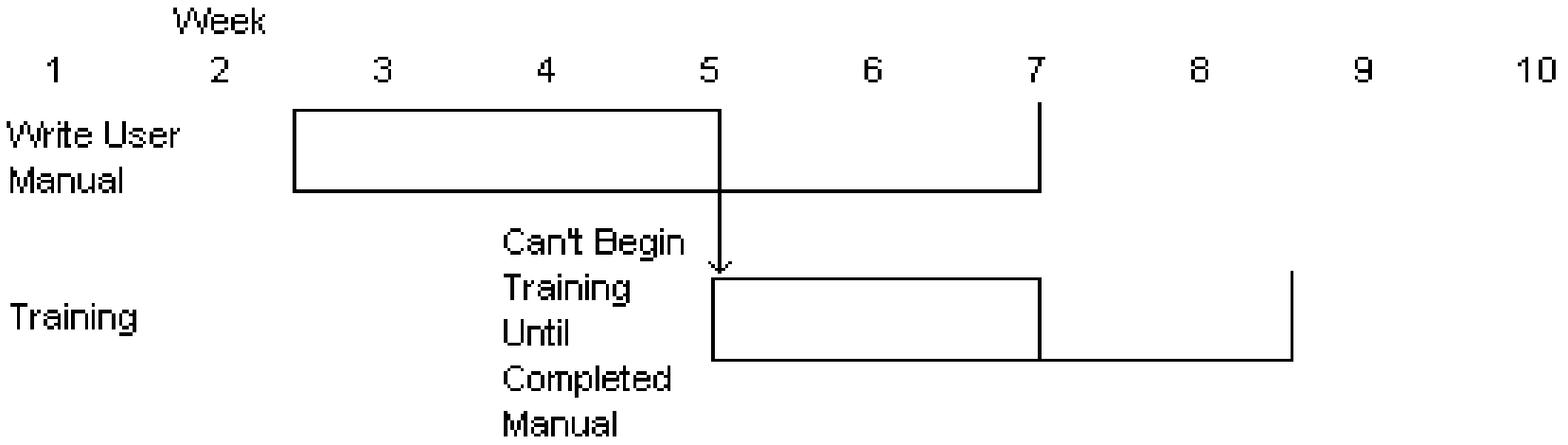
Gantt Chart (1/6)

- The Gantt chart is a graphic display of the duration of a set of activities.
- The activities are listed along the left side of the chart and a time line is given at the top or bottom of the chart.
- The duration of an activity is shown with a horizontal bar that extends over the applicable period of the time line, starting from the starting date of the activity and ending at the ending date for that activity.

Gantt Chart (2/6)

- The start and end date of each activity become milestones for the project.
- Progress can be represented easily in a Gantt chart, by ticking off each milestone when completed.
- Alternatively, for each activity another bar can be drawn specifying when the activity actually started and ended.
- The main drawback of the earlier versions of the Gantt chart was that it did not depict dependency relationships of different activities.

Gantt Chart (3/6)



Gantt Chart (4/6)

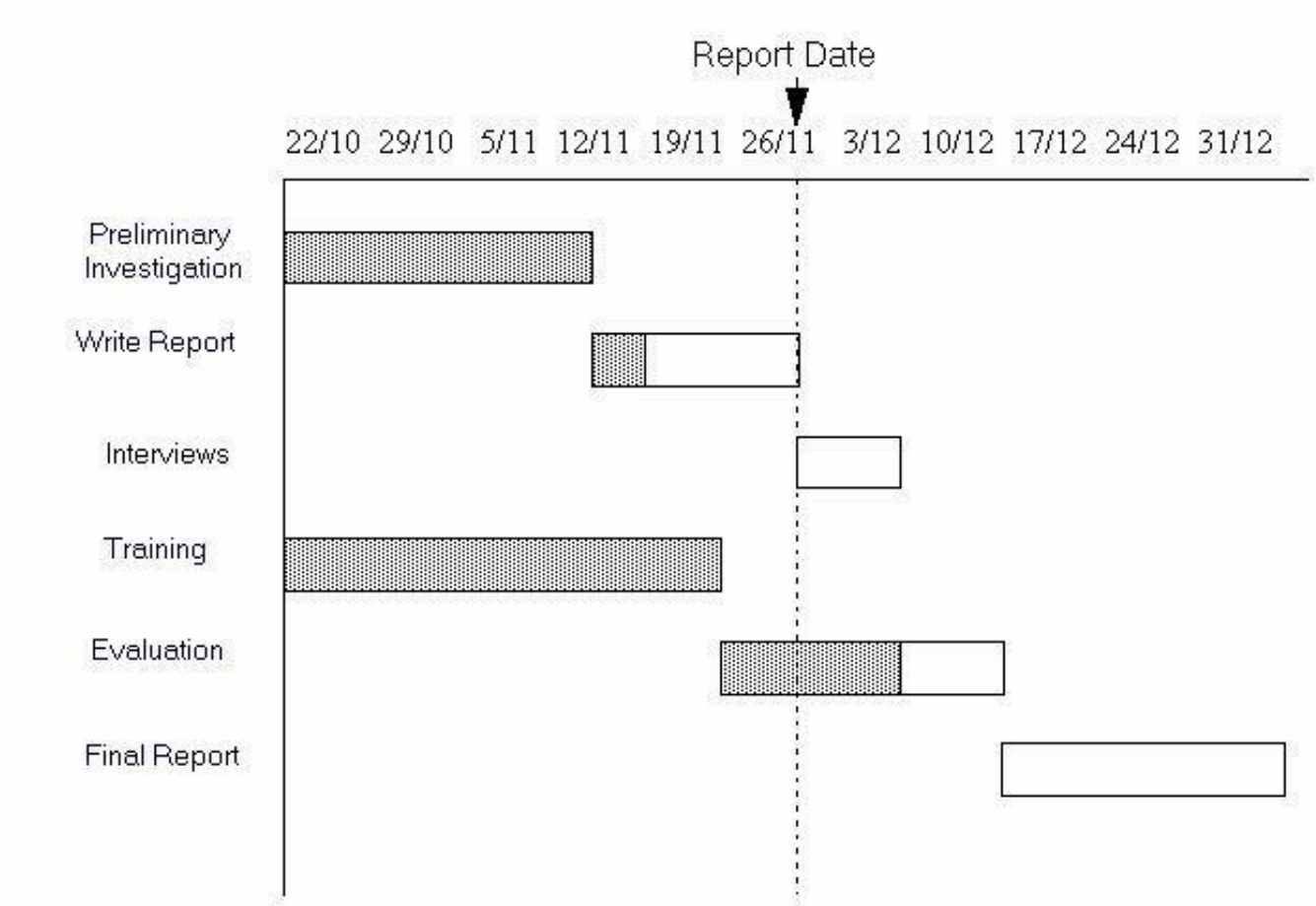
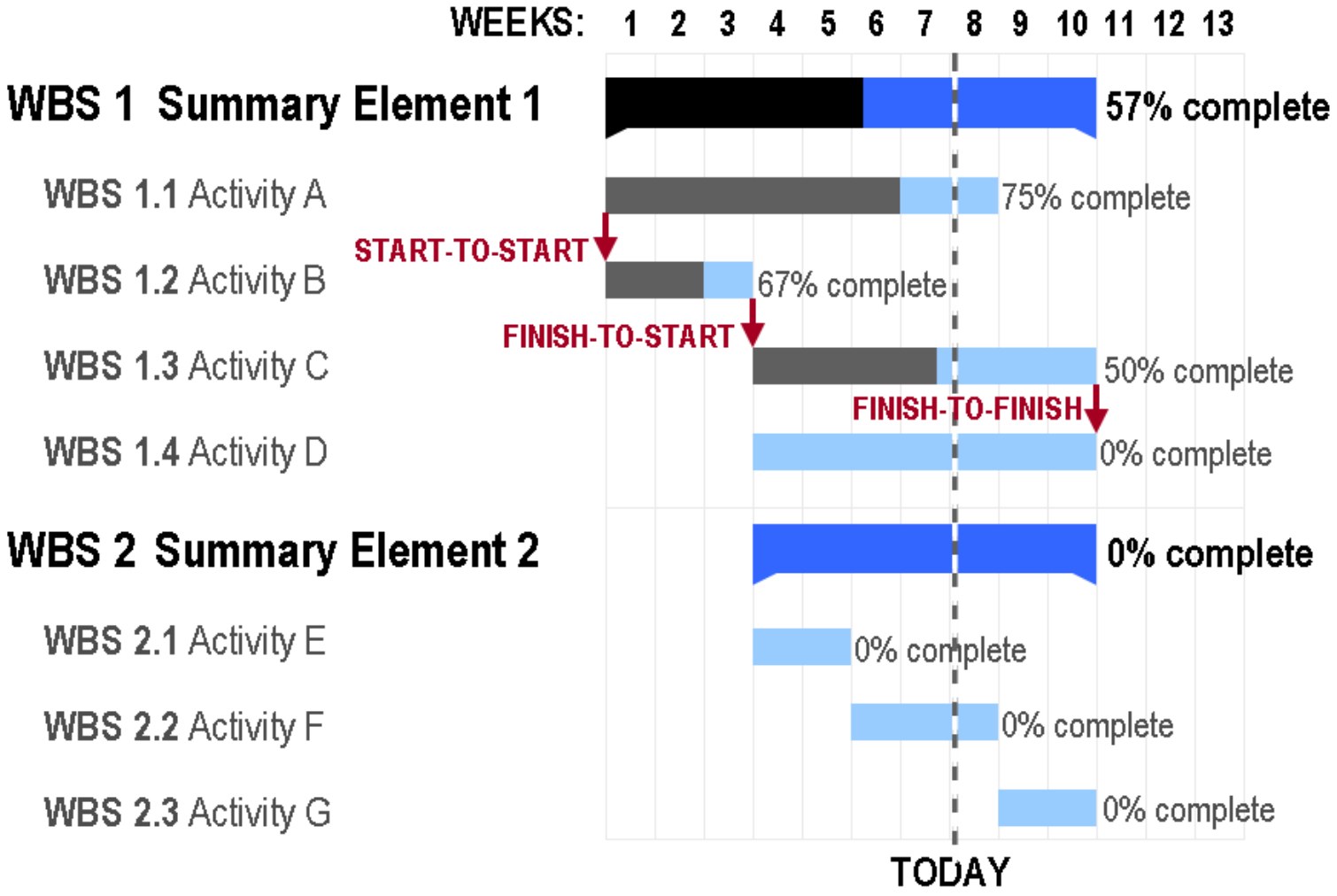
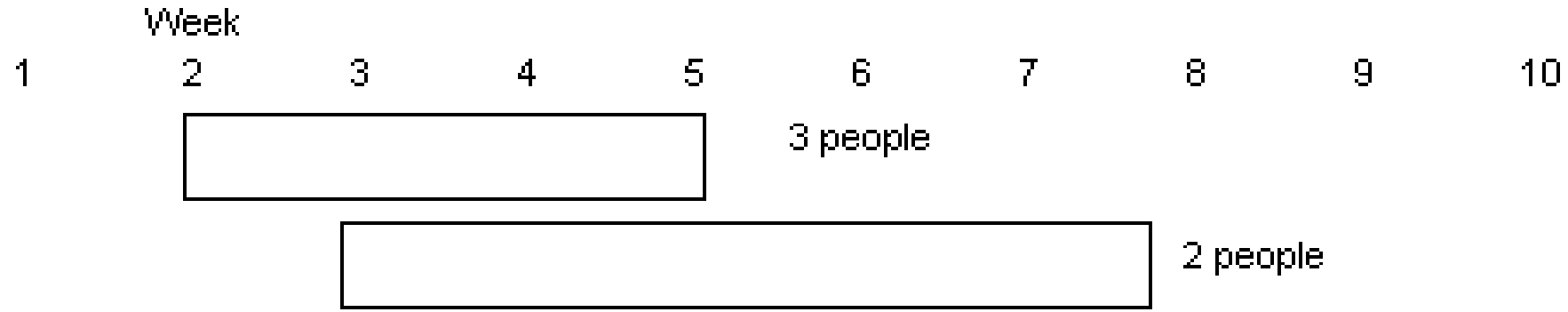


Figure 1: Gantt Chart

Gantt Chart (5/6)



Gantt Chart (6/6)



Overlay Resources on Gantt Chart

Earned Value Analysis

- Earned value is a quantitative measure of percent of project completed so far.
- It is a good technique for measuring progress of a project in terms of time, the cost and the performance.
- The total hours to complete the entire project are estimated and each task is given an earned value based on its estimated percentage contribution to the total.
- Earned value can also be looked upon as the total budgeted cost of the project multiplied by the percent of work completed, i.e. earned value is the amount that should have been spent, according to the budget (plan), on the work completed so far.

Project Tracking Using EVA

- Comparing the actual cost incurred with the budgeted cost, enables one to find out if the project is on the right track.

  NOTE: The difference between the two is called the spending variance and reveals how well the cost targets have been met.

- To find if the work is proceeding as per the schedule, schedule variance (difference between the earned value of the works completed till now and the budgeted costs of the works that should have been completed by now) is calculated.

  NOTE: A negative schedule variance indicates the project is lagging.

  NOTE: The sum of the two variances is called the total variance, gives the overall view of the progress of the project.

Error Tracking

- Allows comparison of current work to past projects and provides a quantitative indication of the quality of the work being conducted.
- With a more quantitative the approach to project tracking and control, enables the more likely problems to be anticipated and dealt with in a proactive manner.

Staffing & Personnel Planning (1/2)

- Once the project schedule is determined and the effort and schedule of different phases and tasks are known, staff requirements can be obtained.
- The average staff size can be determined by dividing the total effort (in person-months) by the overall project duration (in months).
- Average staff size is not detailed enough for personnel planning, especially if the variation in the actual staff requirement at different phases is large.

Staffing & Personnel Planning (2/2)

- Typically the staff requirement for a project is small during requirement and design, maximum during coding and testing and drops again during the final phases of integration and testing.
- Using COCOMO, average staff requirement for different phases can be determined as the effort and schedule for each phase are known.

  NOTE: This presents staffing as a function of time.

Personnel Plan (1/3)

- Once the schedule and average staff level for each activity is known, the overall personnel allocation for the project can be planned.
- This plan will specify how many people will be needed for the different activities at different times for the duration of the project.
- A method of producing the personnel plan is to make it a calendar-based representation, containing all the months in the duration of the project.

Personnel Plan (2/3)

- For each of the different tasks identified and for which cost and schedule estimates are known, the number of people required in each month is listed.
- The total effort for each month and the total effort for each activity can easily be computed from this plan.

  NOTE: The process may be iterative, to ensure that the effort requirement for the different phases and activities is consistent with the estimates obtained earlier.

Personnel Plan (3/3)

- It is usually not desirable to state staff requirement in a unit less than 0.5 person to make the plan consistent with the estimates.

  NOTE: Some difference between the estimates and the totals in the personnel plan is acceptable.

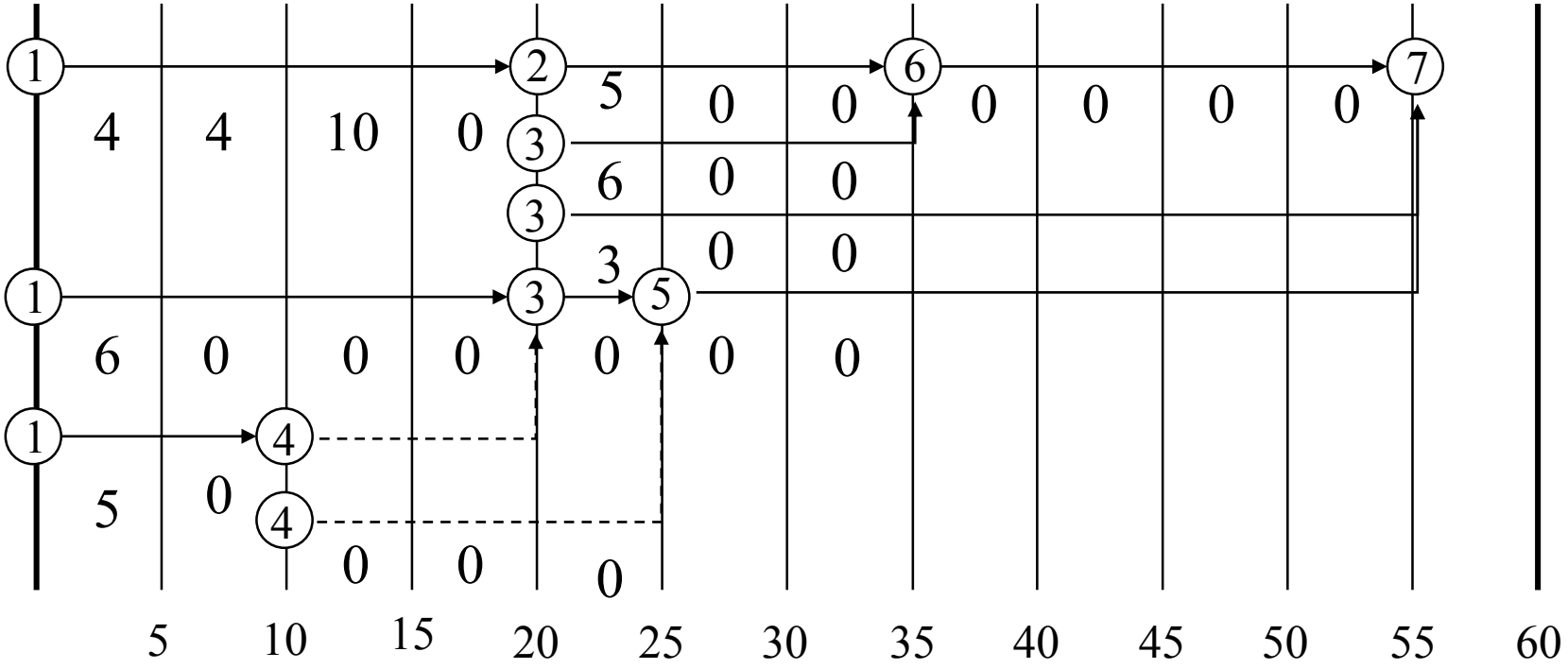
Detailed Personnel Plan

- The plan so obtained has the overall staff requirement, but does not distinguish between different types of people.
- A more detailed plan will list the requirements of people by their specialty i.e. how many programmers, analysts, quality assurance people etc. are needed at different times.

Resource Allocation

- The major issue in scheduling is resource allocation.
- Resources include machinery (machine time), funds, labor, etc.
- There are two issues with resource allocation
  - Finding the peak level requirements and arranging resources so that the project is completed on time (resource loading)
  - Adjusting the activities within the available slack times so that the additional resources to be brought in are minimized (resource leveling)

Gantt Chart – Resource Overlaid



Resource required in various time periods summarized in the next table

Resource Loading

Time Periods	Resource Requirement
0 - 5	15
5 – 10	4
10 – 15	10
15 – 20	0
20 - 25	14

Resource Leveling

Time Periods	Resource Requirement
0 - 5	15
5 – 10	4
10 – 15	10
15 – 20	0
20 – 25	5
25 – 30	6
30 - 35	3

Resource Leveling exploits the slack times to appropriately schedule activities

Team (1/2)

- A *team* is a small number of people with complementary skills who are committed to:
  - a common purpose,
  - a set of performance goals,
  - an approach for which they hold themselves to be mutually accountable.
- Team members interact with each other on a regular basis.

Team(2/2)

- Teams share performance goals and individuals on a team are mutually responsible for end results.
- The team environment produces *synergy*.
- This allows individuals to blend complementary skills and talents to produce a product that is more valuable than the sum of the individual contributions.

Work Group

- Members of a *work group* are held accountable for their individual work.
- They are not responsible for the output of the entire group.
- A work group is more likely to have a strong, directive leader who seeks input from group members and then delegates work to various individuals to complete.

Not All Groups Are Teams (1/2)

Characteristic	Working Group	Team
Leadership	Strong, clearly focused leader	Shared leadership roles
Accountability	Individual	Individual and mutual
Purpose	Same as the broader organization mission	Team purpose that the team itself delivers
Work Products	Individual	Collective

Not All Groups Are Teams (2/2)

Characteristic	Working Group	Team
Meeting Style	Efficient	Open-ended discussion, active problem-solving
Performance Measurement	Indirectly, by its influence on others	Directly, by collective work products
Decision-making Process	Discusses, decides, and delegates	Discusses, decides, and does real work together

The Benefits of Teams

Costs and Productivity

Quality Improvements

Speed

Innovation

Team Structure (1/2)

- Often a team of people are assigned to a project and for the team to work as a cohesive group and contribute the most to the project, the group has to be organized in some manner.

  NOTE: The structure of the team has a direct impact on the product quality and project productivity.
- Two basic types of team structures are possible
  - Egoless teams
  - Chief programmer teams.

Team Structure (2/2)

NOTE: Large projects may incorporate hybrid team structures called *controlled decentralized team*.

Egoless Teams (1/2)

- The team strength has to be ten or less and the goals of the group are set by consensus.
- The structure allows input from all members, which can lead to better decision in difficult problems.

  NOTE: Better suited for long term research type projects that do not have time constraints.
- Group leadership rotates among the group members.

Egoless Teams (2/2)

NOTE: Such teams are also called democratic teams or decentralised teams.

- The structure results in many communication paths between people.

  NOTE: More the communication paths between people, more the overhead in systematizing the process of development.

Chief Programmer Teams (1/2)

- This team has a hierarchy and consists of a chief programmer, a backup programmer, a program librarian and some programmers.
- The **chief programmer** is responsible for
  - all the major technical decisions of the project
  - does most of the design and
  - assigns coding of the different parts of the design to the programmers.

Chief Programmer Teams (2/2)

- The **backup programmer** helps the chief programmer in the decision making process and takes over in the absence of the chief programmer.
- The **program librarian** is responsible for maintaining the documentation and other communication related work.

  NOTE: This structure considerably reduces interpersonal communication.
- This structure is well suited for projects with simple solutions and strict deadlines.

Other Types of Teams

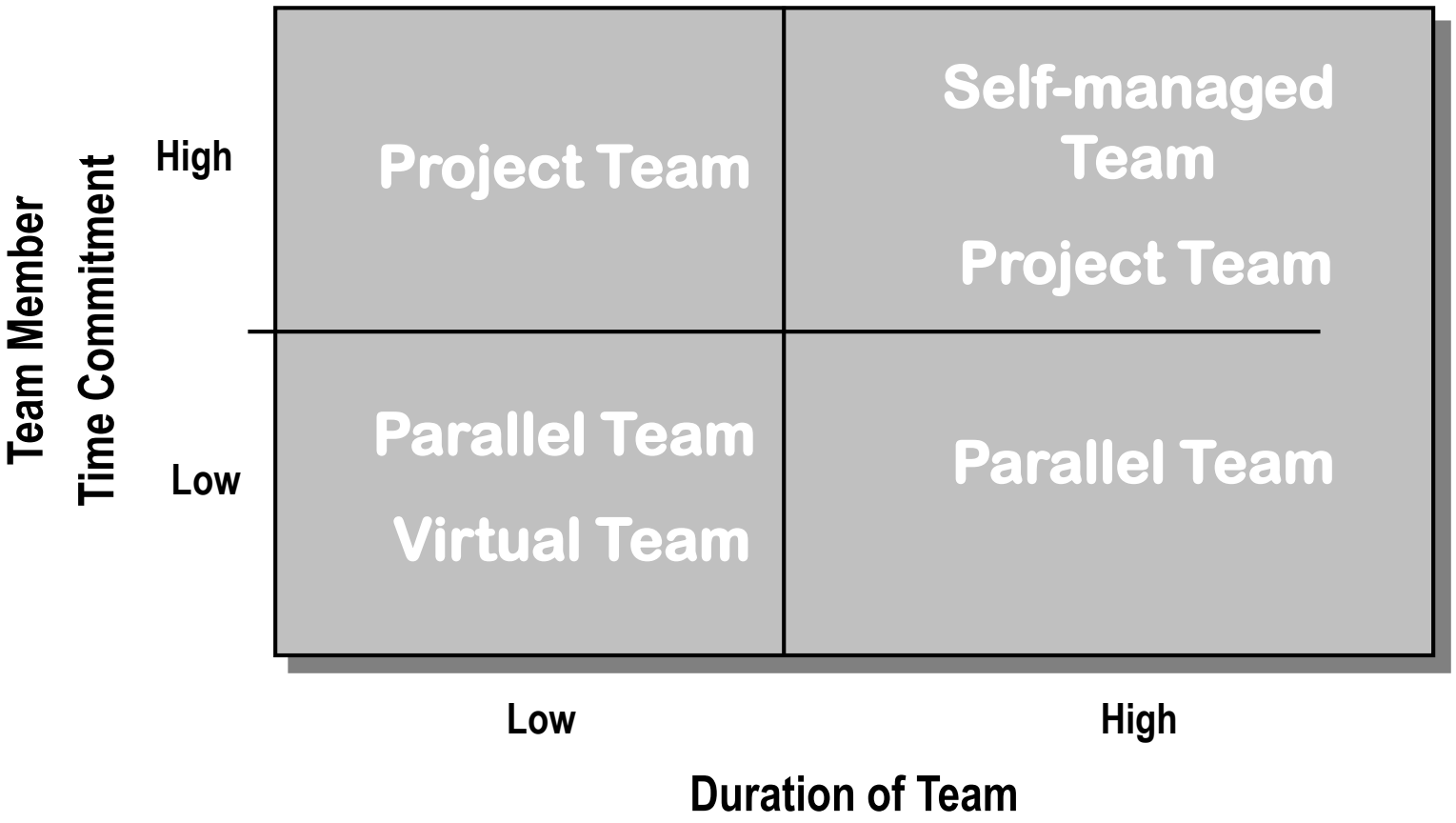
Self-Managed Teams

Project Teams

Parallel Teams

Virtual Teams

Team Characteristics



Self-Managed Teams (SMT) (1/2)

- Responsible for producing an entire product, component, or service.
- Formalized as part of the organization structure.
- Employees are assigned to it on a full-time basis, and its duration is long.
- Utilize employees whose jobs are similar but who may have different levels of skill.

Self-Managed Teams (SMT) (2/2)

- Team members combine their skills to produce an important organizational outcome.
- Have authority to make many decisions that traditionally have been made by supervisors or managers.
- Members need a variety of skills:
  - Technical skills
  - Management skills
  - Interpersonal Skills

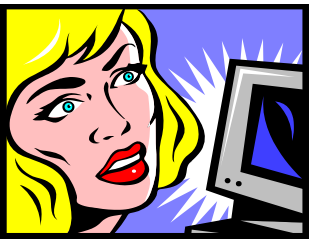
Project Teams

- Work on a specific project that has a beginning and an end.
- Team members work full-time until the project is completed.
- Composed of members from different functions or different technical disciplines.
- Key criterion for judging team performance is meeting or exceeding milestone deadlines.

Parallel Teams

- Sometimes called *problem-solving teams* or *special-purpose teams*.
- Focus on a problem or issue that requires only part-time commitment from team members.
- Employee spends a few hours per week with the parallel team, and the remainder of the time on his/her regular job.
- When the problem is solved the team is disbanded.
- Can be of short or long duration.

Virtual Teams



- Take advantage of interactive computer technologies to enable distant people to work together.
- Require only a part-time commitment.
- Make it possible for companies to cross organizational boundaries:

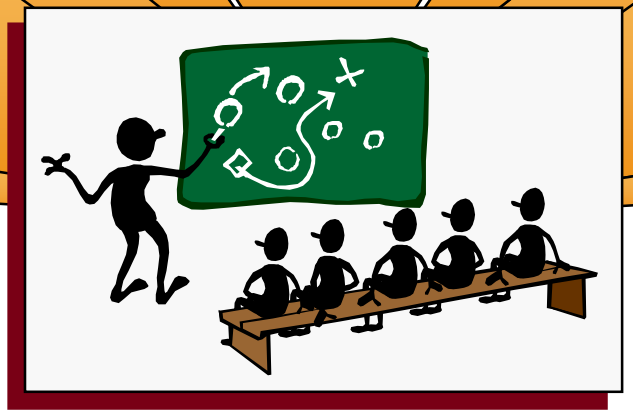
  - Linking customers, suppliers, and business partners to improve the quality and increase the speed with which a new product or service is brought to the market.

Managing Team Performance

- Team performance requires vigilant management.
- Factors that need to be taken into account in managing effective team performance are:
  - The stages of team development.
  - The roles of team members and leaders.
  - Team member behaviors.

Stages of Team Development

1. Forming
2. Storming
3. Norming
4. Performing
5. Adjourning



Roles of Team Members

Task-Facilitating Role

- Direction giving
- Information seeking
- Information giving
- Coordinating
- Summarizing

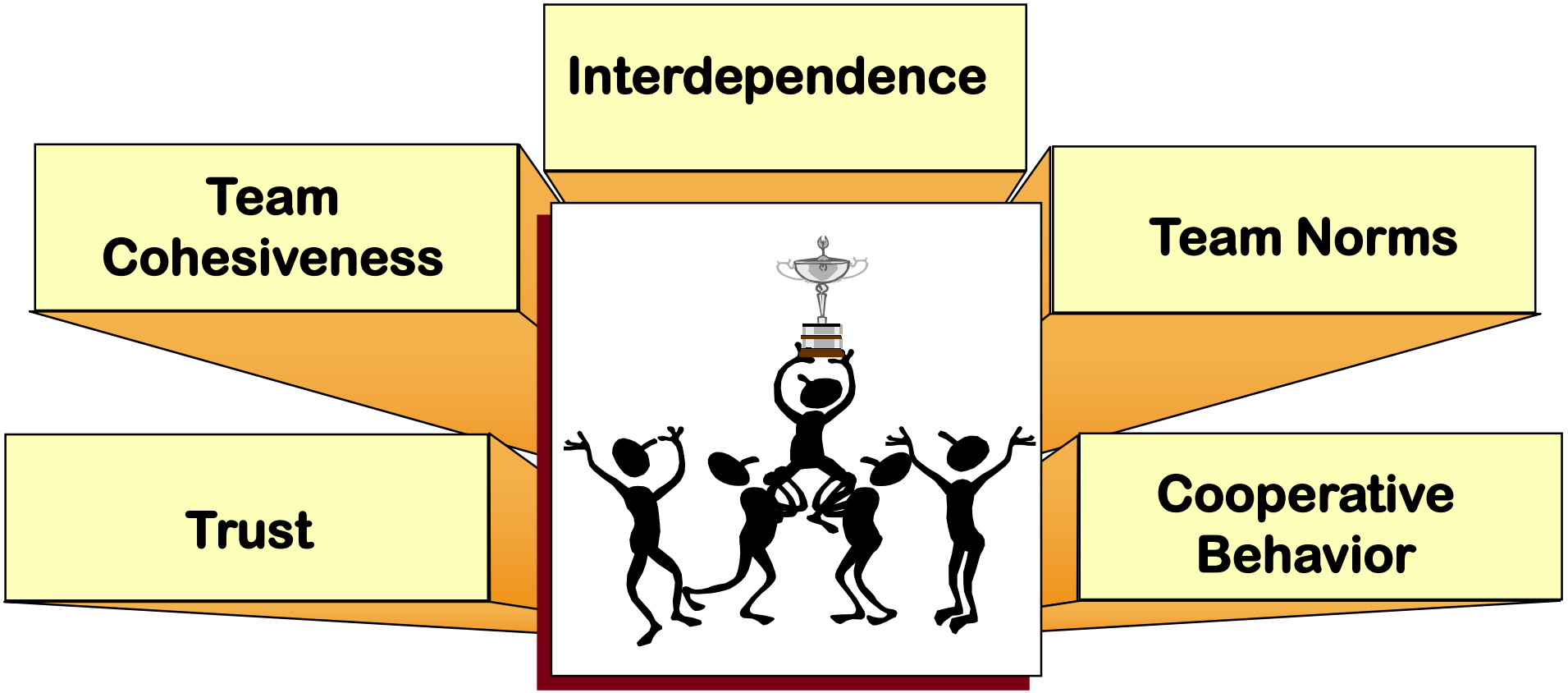
Relationship-Building Role

- Supporting
- Harmonizing
- Tension relieving
- Energizing
- Facilitating

Effective Ways to Enact the Role of Team Leader

- How can a team leader positively influence team processes and outcomes?
  - Take care of team members
  - Communicate with team members
  - Share power with the team
  - Learn to relax and admit your ignorance

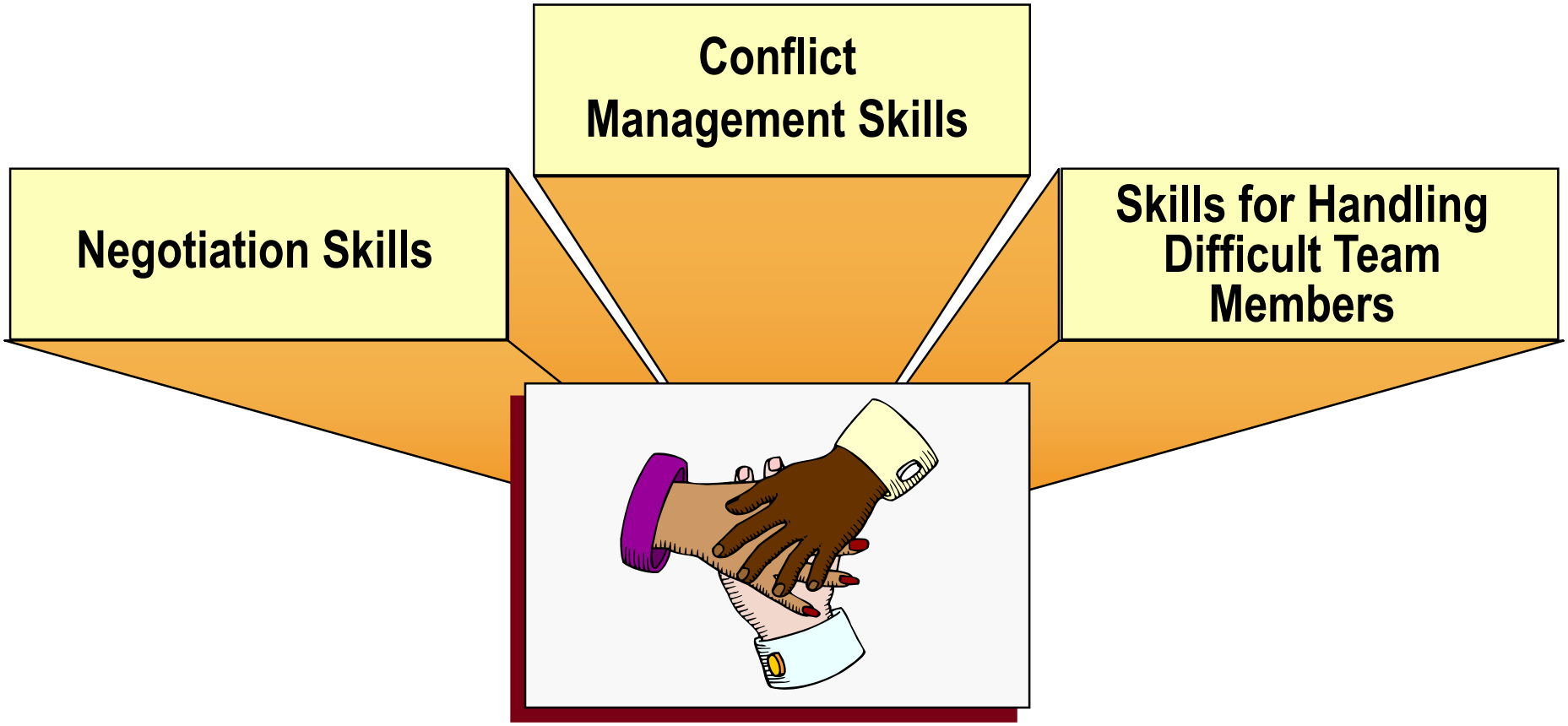
Behavioral Dimensions of Effective Teams



Team Performance Problems

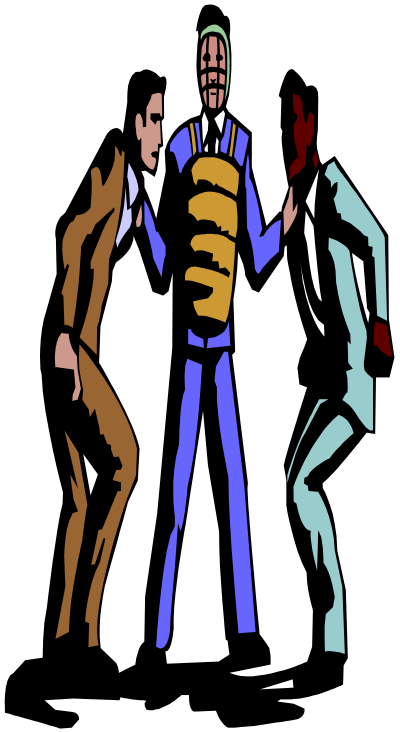
- Free Riders
- Nonconforming High Performers
- Lack of Teamwork Rewards

Skills for Managing Teams



Team Management Skills

- Conflict Management Skills
  - Functional conflict - conflict that stimulates team and organizational performance
  - Dysfunctional conflict - conflict that has a negative effect on team and organizational performance



Applying the Problem-Solving Style of Conflict Management

- The willingness of both parties is necessary
- Convene meetings at the right time and place
- Give both parties ample time to cool down
- Resume discussion until a workable solution is achieved

Negotiation Skills (1/2)

- Win-win Style, or integrative bargaining
  - determine a personal bottom line
  - understand the other party's real needs and objectives
  - emphasize common ground, de-emphasize differences
  - search for mutually agreeable solutions
  - focus on building a relationship rather than a one-time deal

Negotiation Skills (2/2)

- Win-lose Style, or distributive bargaining
  - one party will receive the most beneficial distribution of a fixed amount of goods.
- Three Common Mistakes of Negotiation
  - Do not assume that a negotiation must always result in a settlement.
  - Avoid becoming fixated on one particular issue in the negotiation.
  - Do not assume that the other party has all the power due to greater levels of experience.

Applications of Management Perspectives—*For the Manager*

- When determining how many people should be on a team, the best rule of thumb is to keep the team small.
- When assigning people to work on a team, it is important to select the right mix of people with complementary skills.
  - Technical skills
  - Administrative skills
  - Interpersonal skills

Applications of Management Perspectives—*For Managing Teams*

- It takes time for a team to earn the right to manage itself.
- Not all groups of people are capable of functioning as a team.

Applications of Management Perspectives—*For Individuals*

- It is important to know how to cope with disruptive team members.
- The general rule is to avoid embarrassing or intimidating participants regardless of their disruptive behavior.
- It is best to use supportive communication and collaborative conflict management techniques.

The Requirements Problem

- The goal of software development is to develop quality software – on time and on budget – that meets the customer's real needs
- The project success depends on good requirements gathering and management.
- Requirements errors are the most common type of system development error and the most costly to fix.
- A few key skills can significantly reduce requirement errors and thus improve software quality.

Customer – Types (1/2)

- The customers can be of different types viz.
  - The customer can be an external entity, purchase order in hand, whom we must convince to disregard our competitor's claims and buy our product because it is easier to use, has more functionality and in the final analysis, is just better
  - The customer could be sitting down the hall or across the country, waiting anxiously for that new application to do their tasks more efficiently, so that the company we work for will ultimately be more profitable and our jobs more rewarding and just more fun

Customer – Types (2/2)

- The customers … different types viz. (continued)
  - The customer can be a company that has hired us to develop the software, based on expectations that the software developed will be of the highest quality achievable at today's state of the art and will transform the company into a more competitive, more profitable organization in their marketplace

Project Failures – Causes (1/2)

- The three most common reasons that caused projects to exceed their original estimates were
  - Lack of user inputs – 13% of all projects
  - Incomplete requirements and specification – 12% of all projects
  - Changing requirements and specifications – 12% of all projects

  NOTE: Other reasons could be unrealistic schedule or time frame in the first place (4%); inadequate staffing and resources (6%); inadequate technology skills (7%)

Project Failures – Causes (2/2)

- Thus, one-third of the development projects run into trouble for reasons that are directly related to requirements gathering, documenting and management.

Project Success - Causes

- The three most common reasons that caused projects to succeed are
  - User involvement – 16% of all successful projects
  - Executive management support – 14% of all successful projects
  - Clear statement of requirements – 12% of all successful projects
- Thus the role of requirements gathering, analysis and specification cannot be over emphasized.

Software Requirements - Definition

- A condition of capability needed by a user to solve a problem or achieve an objective
- A condition or a capability that must be met or possessed by a system to satisfy ac contract, standard, specification or other formally imposed document

Requirements Analysis & Specification – Why?

- Earlier little importance was attached to this area because of the tacit assumption that *developers understood the problem clearly when it was explained to them informally*.
- A fundamental problem of software engineering is the problem of scale
  - Complexity and size of applications and software systems are continuously increasing
- With large and complex systems, goals of the entire system is difficult to comprehend.

Requirements Analysis & Specification – Challenges (1/3)

- It is believed that the software engineering discipline is the weakest in this critical area.
- The scope of this phase involves capturing the requirements and needs that are in the minds of the various people in the client organization
- This information is not formally stated or organized, hence inputs is inherently informal and imprecise and likely to be incomplete.
- Inputs from multiple people are likely to be inconsistent.

Requirements Analysis & Specification – Challenges (2/3)

- Often the requirements are not known to the client themselves – they have to be visualized and created.
- The software requirements specified should satisfy all parties concerned
- The process of specifying requirements cannot be totally formal
  - Formal approach requires precise and unambiguous input

Requirements Analysis & Specification – Challenges (3/3)

- Software requirements activity cannot be fully automated
- Any method for identifying requirements can be at best a *set of guidelines*
- Changing requirements is a continuous irritant for software developers and leads to bitterness between the client and developers.
- Many change requests have their origin in incorrect interpretation or specification of the requirements.

Requirements Analysis & Specification – Issues

- Requirements must be precisely defined before starting implementation

  NOTE: Many projects have suffered because they did not adhere to this practice
- Goals of this phase are:
  1. Clearly understand the customer requirements (Requirements Analysis)
  2. Organize the requirements into a document (Requirements Specification)

Requirements Analysis (1/2)

- Requirements analysis involves:
  - obtaining *clear* and *thorough understanding* of the product to be developed
    - with a view of *removing* all *ambiguities* and *inconsistencies*
    - from the *customer's perception* of problem.

Requirements Analysis (2/2)

NOTE : Understand and find answers to these basic questions before commencing analysis

- What is the problem?
- Why is it important to solve it?
- What are the possible solutions to the problem?
- What are the inputs & outputs from the system?
- What are the complexities in solving the problem?

Requirements Analysis – Activities

- Requirements gathering:
  - It involves heavy customer interaction (including end users)
  - All possible data about the system needs to be collected

    NOTE: This task is more challenging, if a working model of the system does not exist
- Analyze the gathered requirements i.e. clearly understand requirements)
  - Resolve anomalies
  - Resolve conflicts
  - Resolve inconsistencies

    NOTE: Many of these problems get resolved automatically if a formal method is used for understanding and specifying the system

Requirements Specification – Activities

- Involves a very systematic documentation of the user requirements gathered into a SRS document.

  NOTE: The process of requirements analysis and specification may require one to iterate, with verifying and validating with the customer in between iterations.
- Usually done by SYSTEMS ANALYSTS

The SRS Document (1/2)

- One of the outputs from this phase is the Software Requirements Specification (SRS) document
- The SRS document:
  - Forms a basis for agreement between customer and developer
  - After approval by customer, it is used as a reference document for all further development
  - An SRS provides a reference for validation of the final product
  - A high quality SRS is a prerequisite to high quality software
  - A high quality SRS reduces the development cost

The SRS Document (2/2)

- SRS document should clearly specify:
  - Functional requirements of the system
    - Specify input, transformation, and output requirements of each function
  - Non-functional requirements of the system e.g. Usability, Maintainability, Portability, Reliability, Accuracy, Human-computer interface issues, Performance.
  - Constraint under which system must work e.g. co-existence with other systems
- SRS document is reviewed (first internally and then by the customer) for unambiguousness, completeness, correctness, etc.

Good SRS - Characteristics (1/3)

- It must be *correct* i.e. every requirement included in the SRS represents something required in the final system.
- It must be *complete* i.e. everything the software is supposed to do and the responses of the software to all types of input data should be specified in the SRS.
- It must be *unambiguous* i.e. every requirement stated has one and only one interpretation.

Good SRS - Characteristics (2/3)

- It must be *verifiable* i.e. every requirement stated should have a cost effective process to check whether the final software meets that requirement.
- It must be *consistent* i.e. one requirement should not conflict with another requirement
- It must be *modifiable* i.e. its structure and style should allow necessary changes easily while preserving completeness and consistency.

  NOTE: Customer requirement may change

Good SRS - Characteristics (3/3)

- It must *rank* the requirements i.e. the requirements should be categorized into critical, important, desirable.
- It must be *traceable* i.e. origin of each requirements is clear and facilitates the referencing of each requirement for future development.
- It must posses *conceptual integrity* i.e. easy to understand
- It must address maintainability, portability, and modifiability related issues
- A good SRS must satisfy all parties and involves tradeoffs and persuasion

Note: SRS formats are usually available

SRS Document Contents

- The SRS must specify:
  - WHAT the system must do (and **not** HOW the system should do it)
    - Specify only the externally visible behavior
    - View the system as a black box
  - All goals and constraints
    - Goals guide in trade-off amongst design decisions e.g. efficiency v/s maintainability
  - Response to unexpected events

Structure of SRS Document (1/2)

- Introduction
  - Purpose
  - Scope
  - Definitions, Acronyms and Abbreviations
  - References
  - Overview
- Overall description
  - Product Perspective
  - Product functions
  - User Characteristics
  - General constraints
  - Assumptions and Dependencies

Structure of SRS Document (2/2)

- Specific Requirements
  - External Interface Requirements
  - Functional Requirements
  - Performance Requirements
  - Design Constraints
  - Attributes
  - Other Requirements

NOTE: Many sample templates are available on the internet…choose one for your project carefully!!!

Software Changes – Problems (1/2)

- We know that:
  - Software project produces large number of object e.g. source code, documents, test cases
  - During development, these objects are referred to and modified by several developers
- Inconsistency problem:
  - May occur when the objects are replicated
  - Different changes may get made to different copies of the object i.e. leads to inconsistency
  - Changes may not get communicated to all concerned
  - Difficulties may surface during integration

Software Changes – Problems (2/2)

- Concurrent access problem:
  - May occur when several developers are authorized to modify single copy of an object e.g. change may not get communicated to all concerned
  - Overwriting may occur if several developers attempt make modification simultaneously

Stable Development Environment (1/2)

- If a strict discipline is not enforced during updation and storage of software objects:
  - Several problems may appear
  - Chaos may prevail
- Developers need stable environment to undertake development
  - Before any developer starts using a software product (developed by another developer), its configuration must be frozen.

Stable Development Environment (2/2)

- Freezing of an object involves “archiving everything needed to rebuild the product”
- Freezing of various objects results in establishing a *baseline* for others to use.

  NOTE:

  1. Old baseline are never changed
  2. Only new baselines are declared

Software Variants – Problems

- Variant:
  - New version of an object that co-exists with the object from which it was derived
- For business reasons, several variants of an object may co-exist at the same time
- Problems with having several variants of the same object:
  - What to do if a bug is found in one of the several variants of an object?
    i.e. it needs to be fixed in each of the variants
  - How to prevent simultaneous access to the same object by different developers?

Configuration Management (CM)

- What is Configuration?
  - State of the software objects
- What is Configuration Management?
  - Set of activities by which a large number of objects are managed and maintained
- Main CM activities:
  - Configuration identification
  - Configuration control

Configuration Identification (1/2)

- Classification of development objects:
  - Controlled
    - Objects which are under configuration control
  - Pre-controlled
    - Objects which are not yet but will be under configuration control
  - Uncontrolled
    - Objects which are not and will not be under configuration control
- CM plan lists all controllable objects e.g.
  - Requirements document, design document
  - Tools used e.g. compilers, linkers

Configuration Identification (2/2)

- CM plan … controllable objects e.g. (Continued)
  - Source code
  - Test cases
  - Problem logs
- CM plan must strike a balance between controlling:
  - Too many objects i.e. more overheads
  - Too few objects i.e. more confusion

Configuration Control (1/2)

- Configuration control involves management of changes to controlled objects

  Procedure for making changes:

  – Change Control Board (CCB) approves change request based on:
    - Need for change
    - Change impact analysis
    - Review of change impact
  – Developer gets a private copy of the object (by reserve operation)

    NOTE: if an item is issued to one person, then it will not be issued to anyone else

Configuration Control (2/2)

Procedure for making changes (Continued):

- Developer makes changes to private copy
- CCB approves the changes
- Developer submits back modified private copy (by restore operation)

  NOTE: For small projects, CCB may consist of one member

Source Code Control System (SCCS) (1/2)

- Role of CM Tools:
  - Help in tracking various objects
  - Enable determining of “the current states of various products” in an unambiguous manner
  - Facilitate changes to various components in controlled manner
- SCCS - CM tool available on UNIX
  - Useful for controlling and managing versions of text files
  - Provides efficient way of storing different versions by minimizes disk space occupied i.e. stores only “changes” for subsequent versions

Source Code Control System (SCCS) (2/2)

SCCS - CM tool available on UNIX (Continued)

- Access control facilities include:
  - Restrictions on creation of new versions
  - Checking in and checking out of objects
- Revisions are denoted by numbers in ascending order e.g. 1.1, 1.2, 1.3 etc.
- *Fork* can be used to create variants

Software Quality

- Everyone agrees that software quality is important, but in order to achieve the same one has to
  - explicitly define what is meant by software quality
  - create a set of activities that will help ensure that every work product exhibits high quality
  - perform quality assurance activities on every software project
  - use metrics to develop strategies for improving the software process and as a consequence the quality of the end product

Software Quality Assurance (1/2)

- Quality assurance is the application of planned and systematic activities implemented within the quality system to provide confidence that the project will satisfy the relevant quality standards.
- It should be performed throughout the tenure of the project to ensure that the project will employ all processes needed to meet requirements.
- The assurance may be provided to the project management team and to the management of the performing/developing organisation (internal quality assurance) or to the customer and others not actively involved in the work of the project (external quality assurance).

Software Quality Assurance (2/2)

- The tasks include:
  - Periodic Audits
  - Verification and Validation
  - Inspection and Reviews

NOTE:

1. Each task should be defined with an entry and exit criterion.
2. The documents that govern the development, verification, validation, use and maintenance of the software should be identified.
3. How the documents are to be checked for adequacy also should be specified in the QAP.

Software Quality Control

- Software Quality Control (SQC) involves monitoring specific project results to determine whether they comply with the relevant quality standards and identifying ways to eliminate causes of unsatisfactory results.
- It is performed throughout the project by either the SQC department of the organisation.
- SQC can include taking actions to eliminate causes of unsatisfactory project performance.

Some Quality Related Terms

- **Prevention** enables keeping errors out of the process
- **Inspection** enables keeping errors from reaching the customer
- **Attribute sampling** tells us if the result conforms or does not conform
- **Variable sampling** where the result is rated on a scale that measures the degree of conformity.
- **Special causes** are unusual events while **common causes** are normal process variations (a.k.a. **random causes**).
- **Tolerances** specify the limits of acceptable variations.
- The process is in control if the result falls within the **control limits**

Whose Responsibility?

- Everyone involved in the software engineering process is responsible for quality

NOTE :

1. The project team has a Quality Team Leader (QTL) who is responsible for ensuring adherence to the quality process by the project team members.
2. For the organization, it is the Quality Coordinator (QC) who is responsible for ensuring adherence to the quality process by the entire organization.

SQ Assurance - Importance

- One can do it right or one can do it over again.
- Emphasizing on quality in all software engineering activities reduces the amount of rework.
- This in turn results in lower costs and improved time-to-market

  NOTE : This is in spite of the fact that there are costs associated with ensuring quality.

Steps In Quality Assurance

1. Define “software quality”
2. Identify the set of activities that will ensure the achievement of the defined software quality

   NOTE : The activities would filter out errors of work products before they are passed on

SQA Work Products

- The software quality assurance plan (SQAP) is created to define the project's SQA strategy
- During analysis, design and coding the primary SQA work product is the formal technical review summary report
- During testing, test plans and procedures and test reports are produced
- Other work products associated with process improvement may also be generated

Quality Assurance Plans (QAP) (1/2)

- To ensure that the final product is of high quality, some quality control activities must be performed throughout the development.
- The purpose of the software QAP is to specify
  - all the work products that need to be produced during the project
  - Activities that need to be performed for checking the quality of each of the work products
  - The tools and methods that may be used for the software QA activities.

Quality Assurance Plans (QAP) (2/2)

NOTE: Software QAP is interested in the quality of not only the final product, but also of the intermediate work products.

- The documents that should be produced to enhance the software quality should be specified in the QAP.

SQA Activities

- The tasks include:
  - Audits
  - Verification and Validation
  - Inspection and Reviews

NOTE:

1. Each task should be defined with an entry and exit criterion.
2. The documents that govern the development, verification, validation, use and maintenance of the software should be identified.
3. How the documents are to be checked for adequacy also should be specified in the QAP.

Cost Of Quality (1/2)

- They include
  - Prevention costs
    - Quality planning
    - Formal technical reviews
    - Test equipment
    - Training
  - Appraisal costs
    - In-process and inter-process inspection
    - Equipment calibration and maintenance
    - Testing

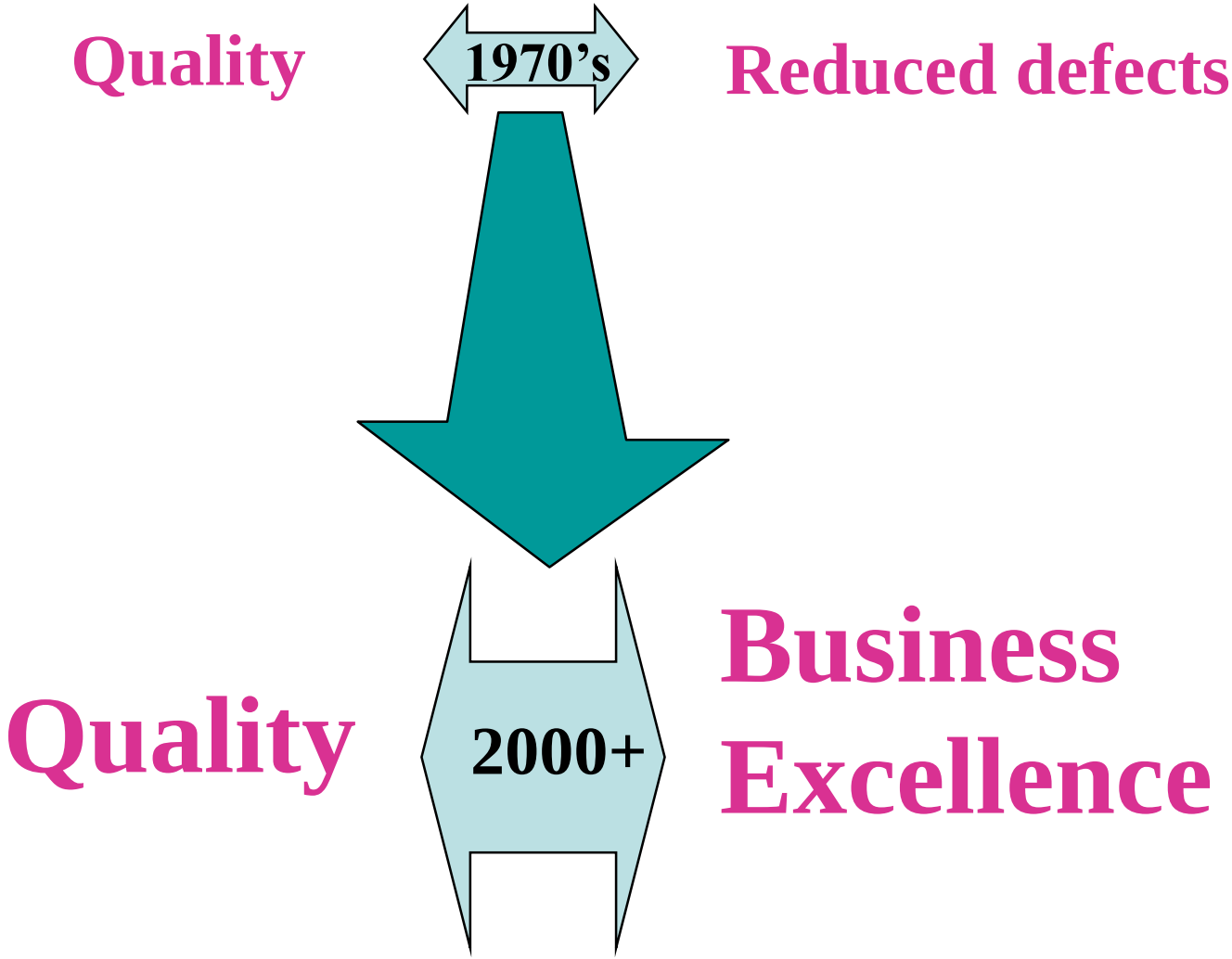
Cost Of Quality (2/2)

- They include (Continued)
  - Failure costs
    - Internal failure
      - **Rework**
      - **Repair**
      - **Failure mode analysis**
    - External failure
      - **Complaint resolution**
      - **Product return and replacement**
      - **Support (help line)**
      - **Warranty work**

How To Do It Right?

- Guidelines for a good QAP are:
  - Find errors before they become defects
  - Improve the defect removal efficiency
  - Thus reducing the amount of rework

What Is Quality?

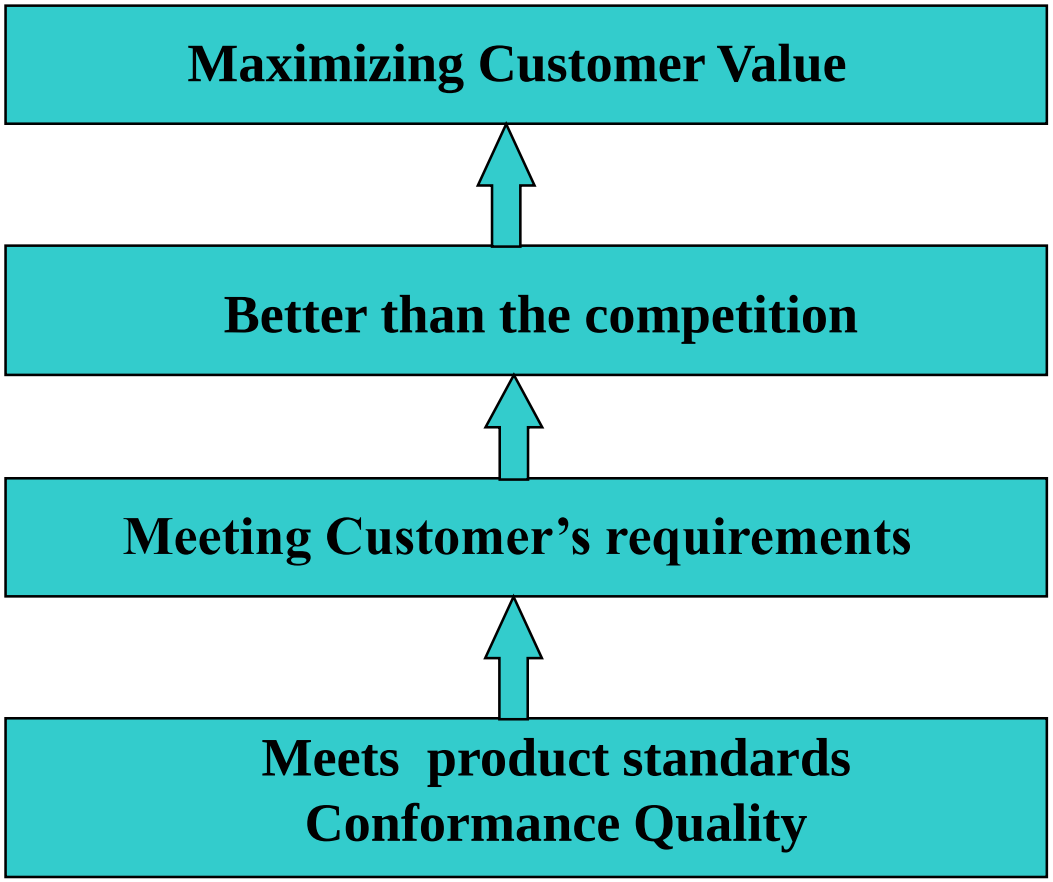


Quality Definition - Evolution

- 1924 - Statistical process control charts
- 1930 - Tables for acceptance sampling
- 1940’s - Statistical sampling techniques
- 1950’s - Quality assurance/TQC
- 1960’s - Zero defects
- 1970’s - Quality assurance in services

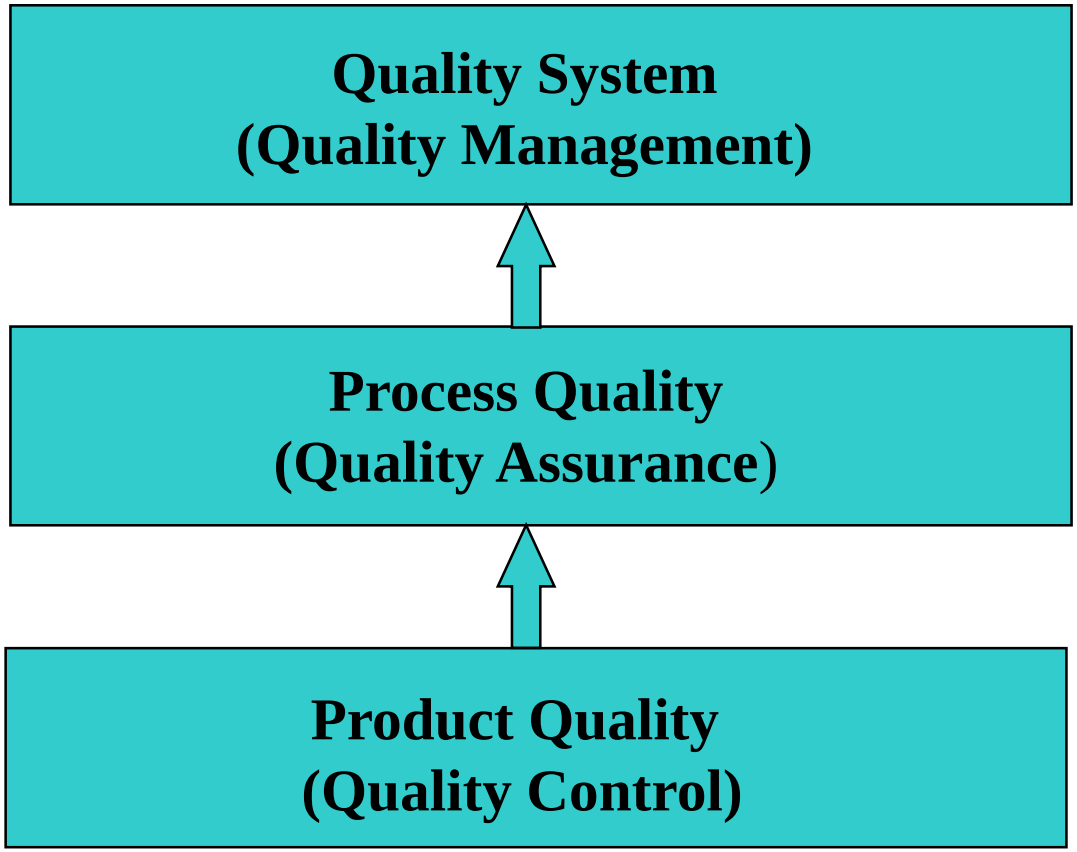
  …
- 2000+ - Business excellence

The Quality Evolution (1/2)



Definition of Quality

The Quality Evolution (2/2)



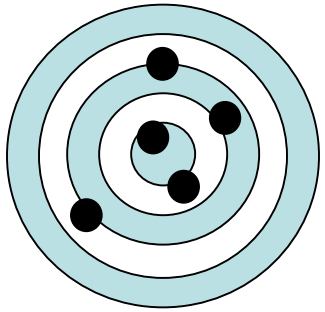
Approach to Quality

Grade v/s Quality

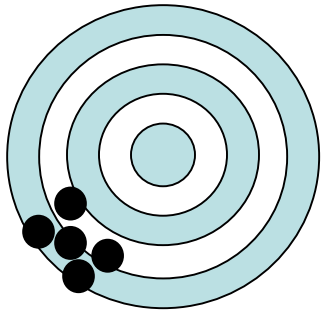
- Grade is the category assigned to products or services having the same functional use but different technical characteristics.
- *Low grade* may imply limited number of features while *low quality* implies many defects, poorly organised user documentations, etc.
- *Low quality* is always a problem while a *low grade* may not be a problem.

Have you ever…

- Shot a rifle?
- Played darts?



Player 1



Player 2

Who is the better shot?

Precision v/s Accuracy

- Precision is the *consistency* that the value of *repeated* measurements are clustered and have *little* scatter or variations.
- Accuracy is the *correctness* of that measured value i.e. how close it is to the true value.

NOTE:

1. Precise values are not necessarily accurate.
2. Accurate measurements are not necessarily precise.

ISO 9000 - Overview

- What is ISO 9000 ? Why ISO 9000 ?
- ISO 9000 family of standards
- ISO 9001 Requirements (20 clauses)
- ISO 9000: 3 (1997) Requirements
- Certification procedure
- Some facts about ISO 9000
- ISO 9000 - evolution to TQM
- ISO 9000 : 2000

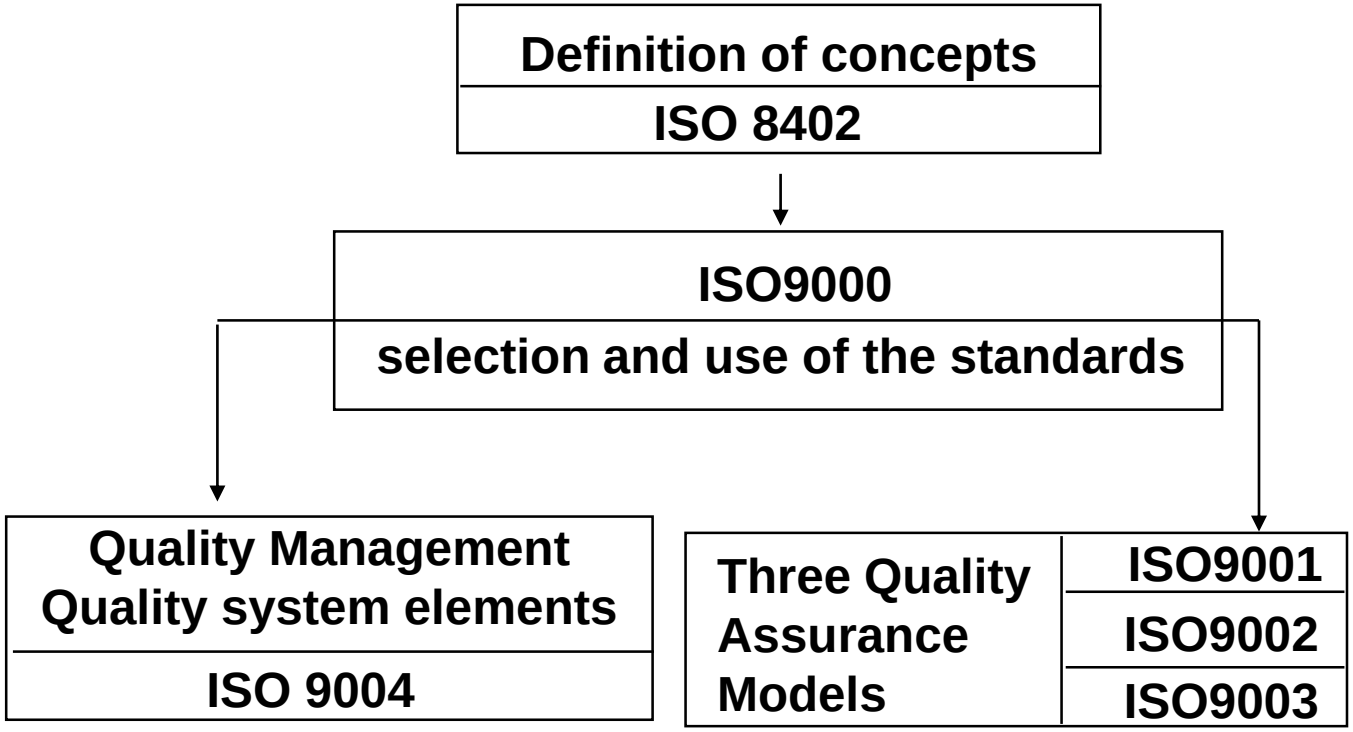
What is ISO 9000 ?

- Checks a Quality System standard based on quality elements
- Assesses the quality mangement system (QMS) for
  - intent
  - implementation
  - effectiveness
- Initial assessment followed by periodic assessment
- Internationally recognized (more than 90 countries recognize ISO 9000)
- Applicable to any industry

Why ISO 9000 ?

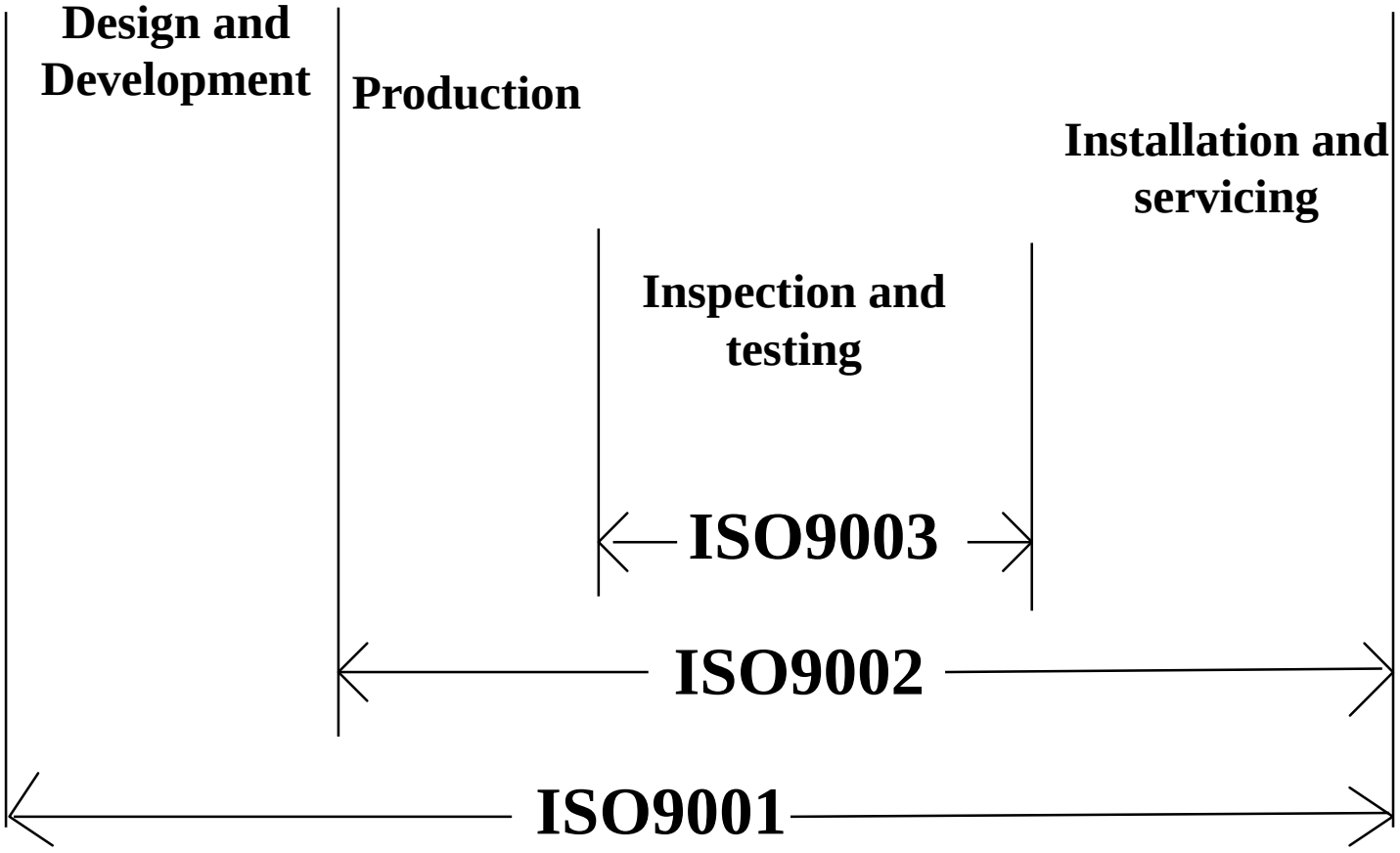
- To ensure consistency of services offered
- Impact of quality improvement on business and bottom-line
- Gain competitive edge
- Project a better image
- Mandatory requirement for some exports
- Government incentives

Structure of the ISO 9000 Quality Standards



ISO 9000 (1984)

ISO 9001, 9002, 9003 Coverage



ISO 9000 (1984)

ISO 9001 Requirements - The “20 clauses”

(1/2)

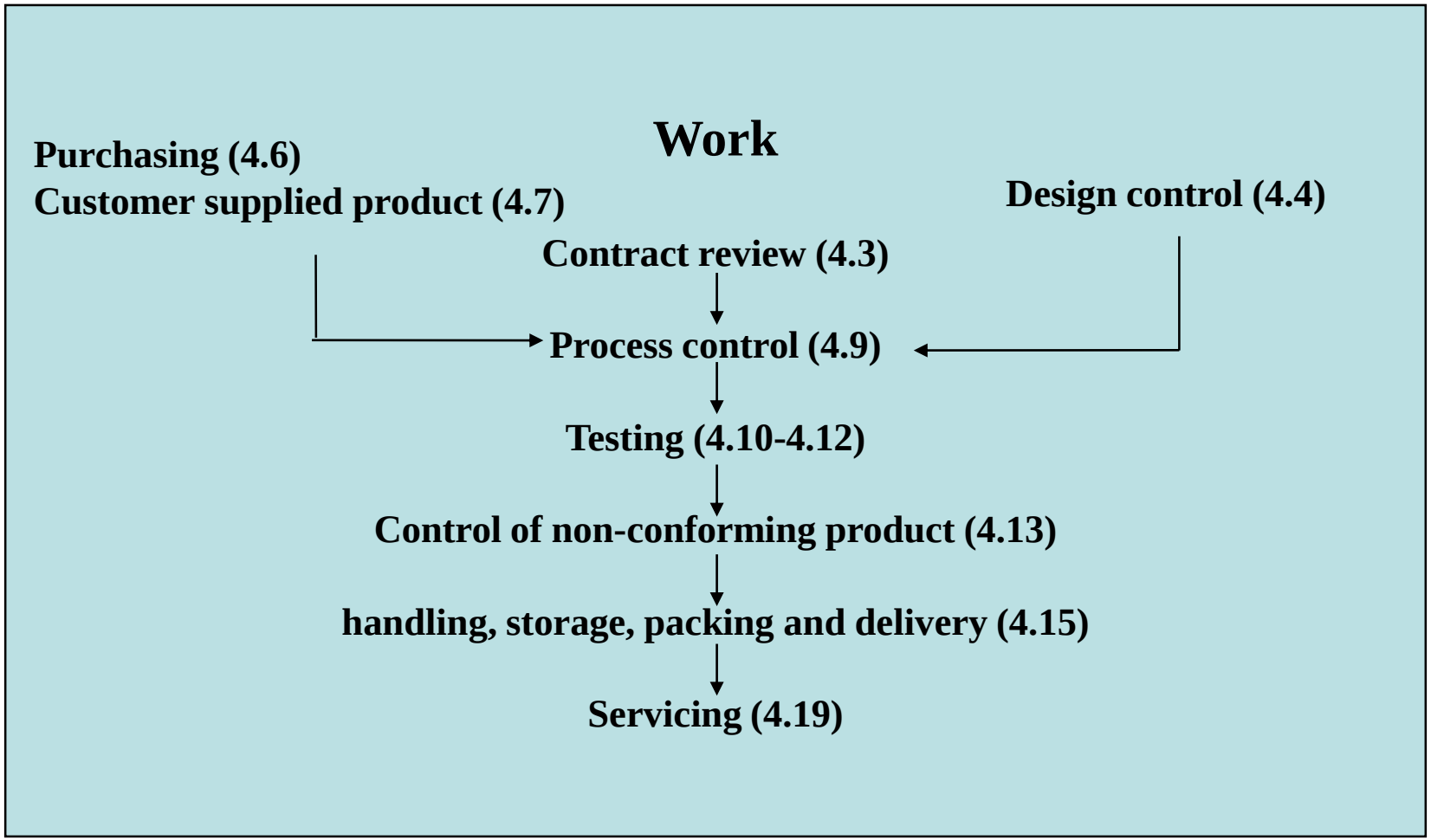
Clause No.	Relating to
4.1	Management Responsibility
4.2	Quality System
4.3	Contract Review
4.4	Design Control
4.5	Document and Data Control
4.6	Purchasing
4.7	Control of Customer Supplied Product
4.8	Product Identification and Traceability
4.9	Process Control
4.10	Inspection and Testing
4.11	Control of Inspection, Measuring & Test Equipment

ISO 9001 Requirements - The “20 clauses”

(2/2)

Clause No.	Relating to
4.12	Inspection and Test Status
4.13	Control of Non - Conforming Product
4.14	Corrective and Preventive Action
4.15	Handling, Storage, Packaging, Presentation & Delivery
4.16	Control of Quality Records
4.17	Internal Quality Audits
4.18	Training
4.19	Servicing
4.20	Statistical Techniques

ISO9001 model (1/2)



ISO9001 model (2/2)

People	System	Information
Management responsibility (4.1) **Training (4.18)**	**Quality system (4.2)** **Document control (4.5)** **Corrective action (4.14)** **Internal quality audit (4.17)**	**Product identification (4.8)** **Records (4.16)** **Statistical techniques (4.20)**

ISO 9000-3 : 1997 Revision

- Guideline for the application of ISO 9001 : 1994 to the development, supply, installation and maintenance of computer software.
- Published in December 1998 by Bureau of Indian standards.
- All the clauses from 4.1 to 4.20 explained as applicable to the software industry. e.g. clause 4.8 Product identification and traceabilty

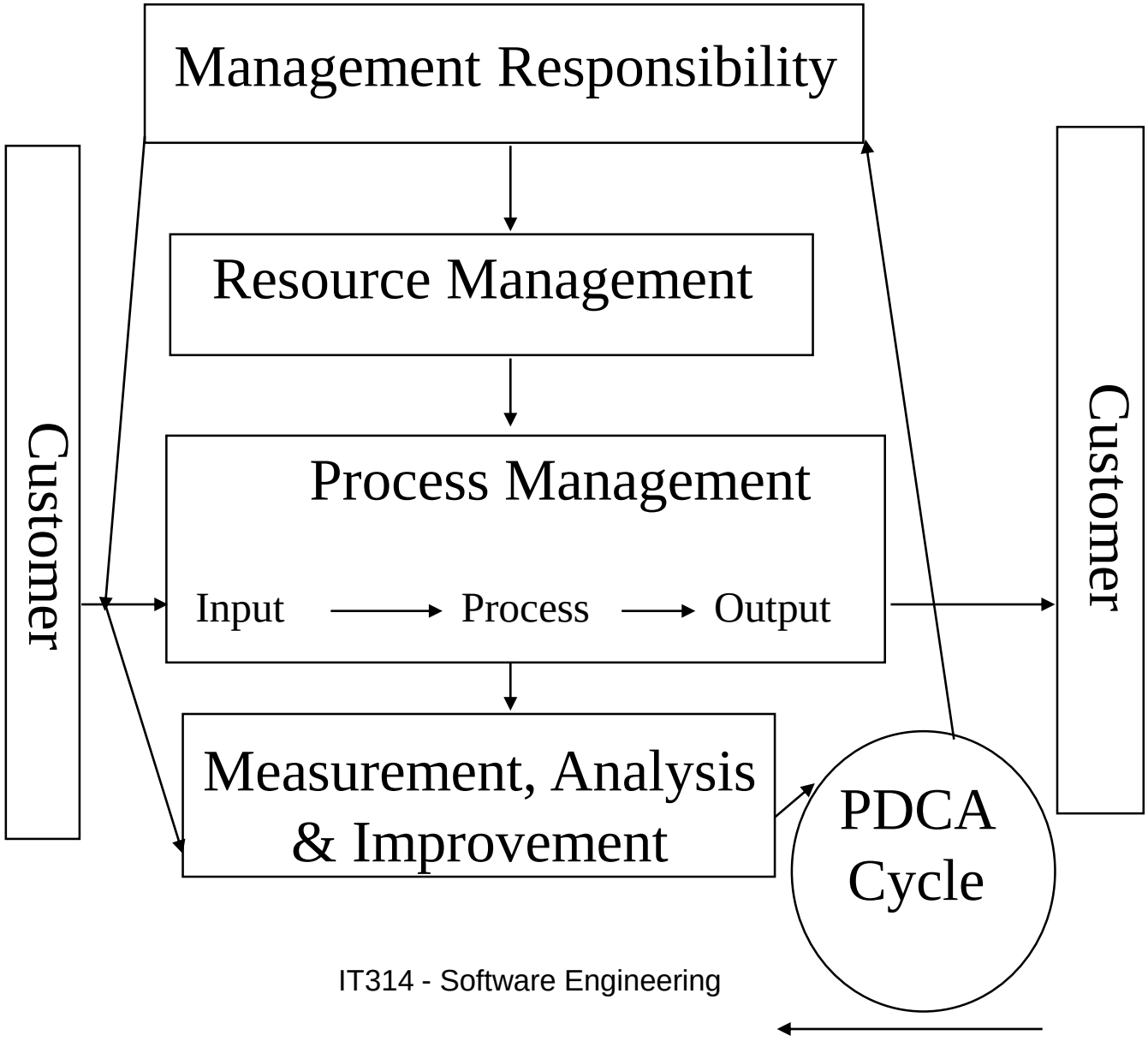
4.8 Product Identification and Traceability

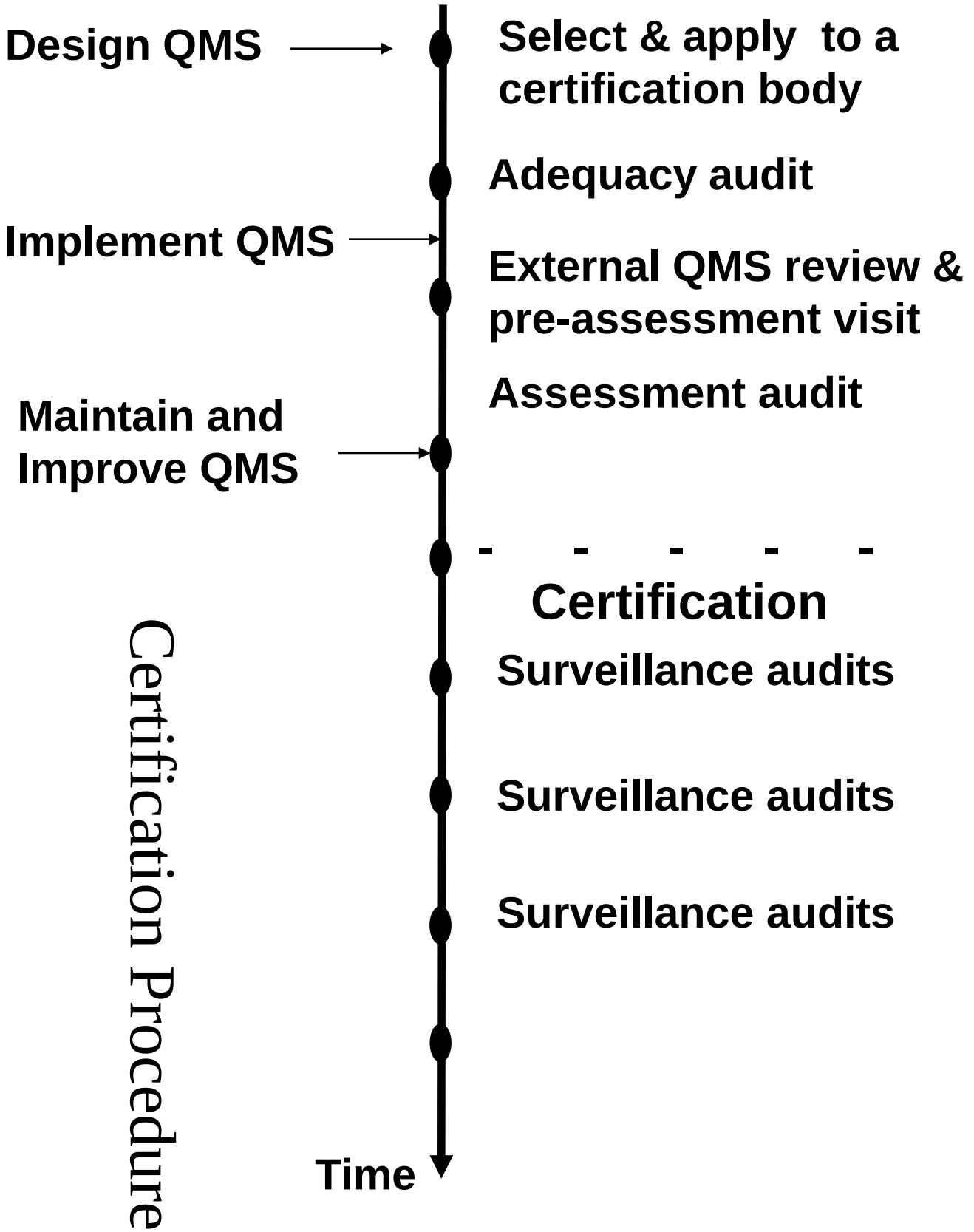
- In software the means by which identification and traceability is done is Configuration Management.
- Must identify software items during all phases of product life cycle.
- Configuration Management may provide the capability to :
  - Identify uniquely the versions of each software item.
  - Control simultaneous updating of a given software item by two people working independently.
  - Identify and track all actions & changes resulting from a change request from initiation through to release.

ISO 9000 : 2000

- Development of consistent pair of Quality Management & Quality Assurance standards
- Use of following 8 management principles
  - Customer driven organization
  - Leadership
  - Involvement of people
  - Process approach
  - System approach to management
  - Continual improvement
  - Factual approach to decision making
  - Mutually beneficial supplier relationships
- Introduction of Business Process model

Business Process Model




Design QMS
Implement QMS
Maintain and Improve QMS
Select & apply to a certification body
Adequacy audit
External QMS review & pre-assessment visit
Assessment audit
Certification
Surveillance audits
Surveillance audits
Surveillance audits
Time
Certification Procedure

Certification Agencies

- SQ Scheme - STQC (Standardization, Testing and Quality Certification) Directorate (a wing of DOE)
- BSI (British Standards Institute)
- BSI - TickIT scheme
- BVQI (Bureau Veritas Quality International)
- TUV Germany
- KPMG (Swiss based)
- DNV (Norwegian) (Det Norske Veritas)

Certification - Parties Involved

- ISO
  - Provides standards
- Applicant
  - Applies for certificate
- Certification body
  - Conducts assessment of quality system against the standard
- Accreditation body
  - Certifies the certification body

Certification - Benefits

- Objective assessment
  - independent third party
- Uniform standard
- Worldwide acceptance
  - Accreditation
  - Mutual recognition
  - Facilitates trade
- Assurance to management
- Assurance to customers
  - Less customer audits
- Prerequisite for entry in several markets

Accreditation

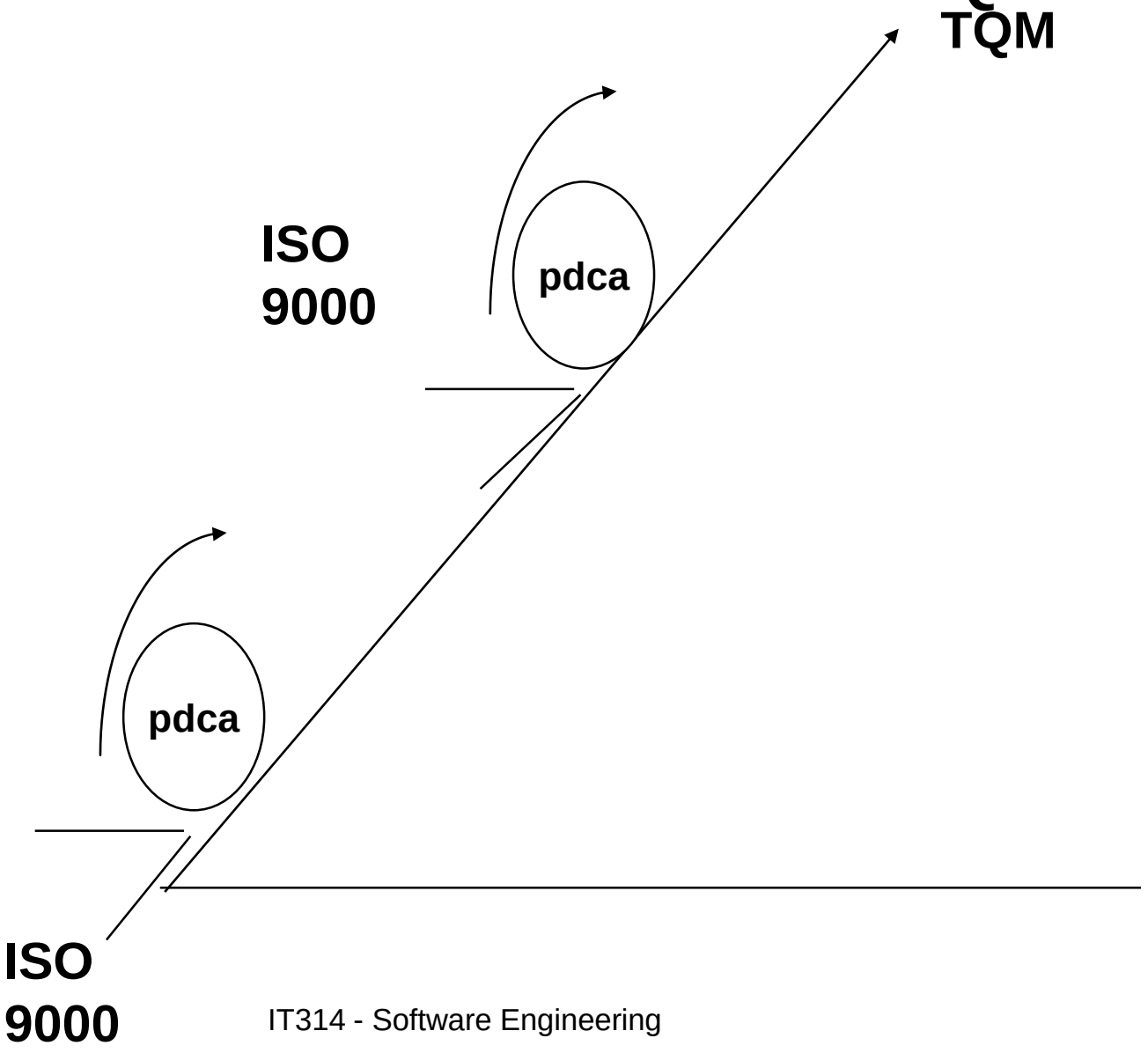
- National schemes by accreditation bodies
  - UK - UKAS (TickIT for IT) (United Kingdom Acredition Service)
  - Netherlands - RvA (Raad voor Accreditatie)
  - US - ANSI/RAB (American National Standards Institute / Registrar Accreditation Board)
  - India - NQC (National Quality Council) (Chaired by Mr. R N Tata)

    NOTE: NQC has been set up by government by act of Parliament

Some Facts : ISO Standards

- The ISO 9000 series is not a European standard. The ANSI is an active participant in ISO working groups.
- The U.S. Department of Defense (DoD) has decided to adopt ISO standards. DoD will eventually replace its own standards with ISO 9001.
- ISO 9001 is not applicable only to the Software Industry.
- ISO 9000-3 is a guideline for applying ISO 9001 to software industry

ISO 9000 - Evolution To TQM



ISO 9000 - Survey

- Top three benefits of ISO 9000 are
  - To meet international consumer demand
  - To improve internal efficiency
  - To remove barriers to export trade
- Sectors with most ISO 9000 registrations - Top 2
  - Manufacturing (35%)
  - Construction (12%)

  NOTE: The software industry is fast catching up, and no software industry of reasonable repute can do without ISO certification.

ISO 9000 - Positioning

- Strengths
  - International acceptance
  - Normally expected of a company
  - External mandatory assessment
  - Low assessment cost
  - Periodic surveillance prevents regression
- Weaknesses
  - Does not address all areas of an organization
  - Not adequate in itself to project a positive image
  - Does not give a clear road map for improvement
  - Minimum requirements
  - Slow updates
  - Less emphasis on overall results

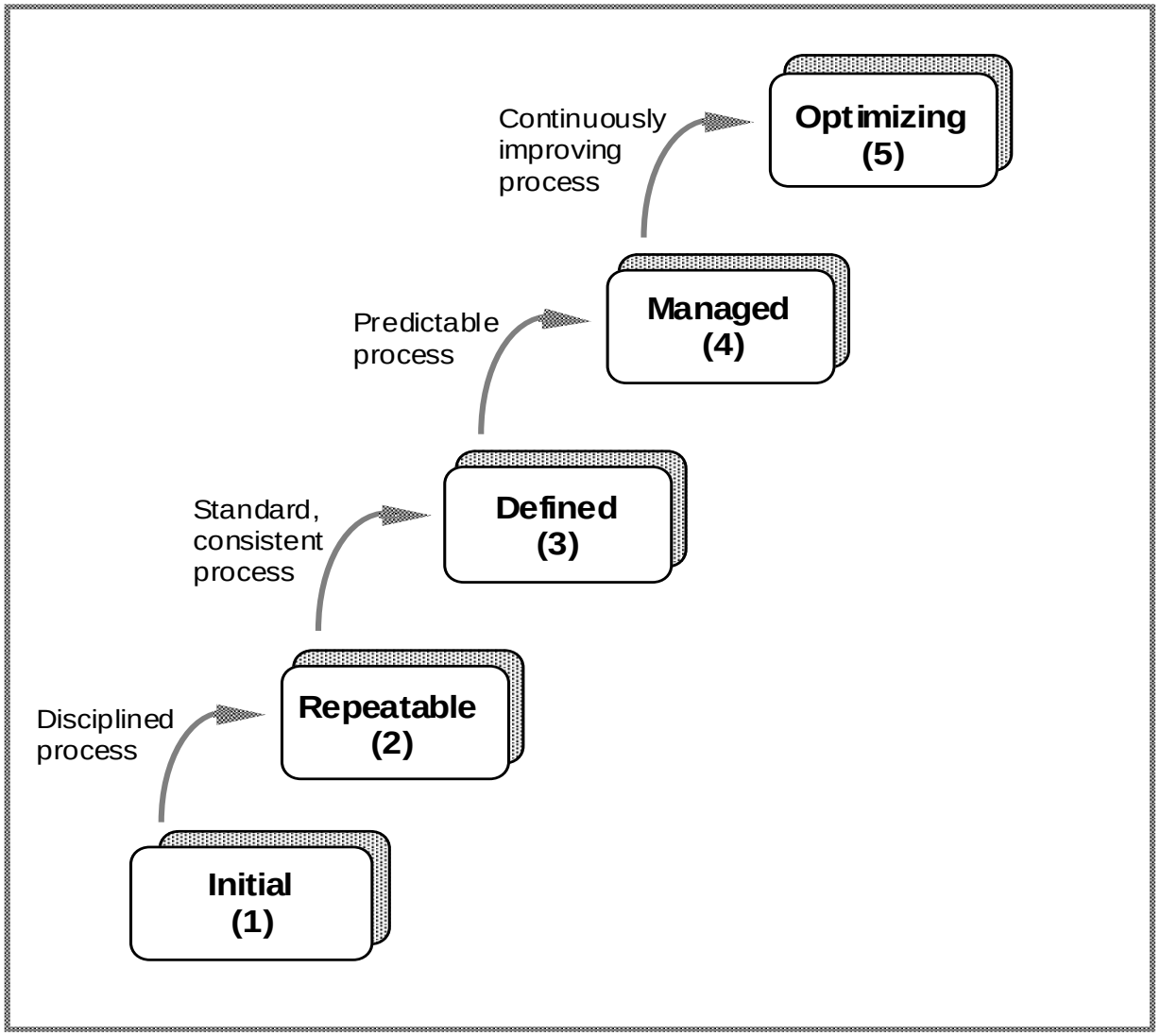
SEI CMM – Introduction (1/2)

- SEI-CMM stands for Software Engineering Institute - Capability Maturity Model
- A model that positions a software engineering (SE) organization in one of 5 maturity levels
- Can be used as an improvement tool by an SE organization
- Can be used as an evaluation tool in a subcontracting situation
- A maturity level is a well-defined evolutionary plateau toward achieving a mature software process.

SEI CMM – Introduction (2/2)

- Each maturity level provides a layer in the foundation for continuous process improvement.
- It uses the concept of key process areas (KPAs)
- Each level comprises a set of process goals that, when satisfied, stabilize an important component of the software process.
- Achieving each level of the maturity framework establishes a different component in the software process, resulting in an increase in the process capability of the organization.

SEI-CMM - Five Levels



The Five Levels - Characteristics (1/3)

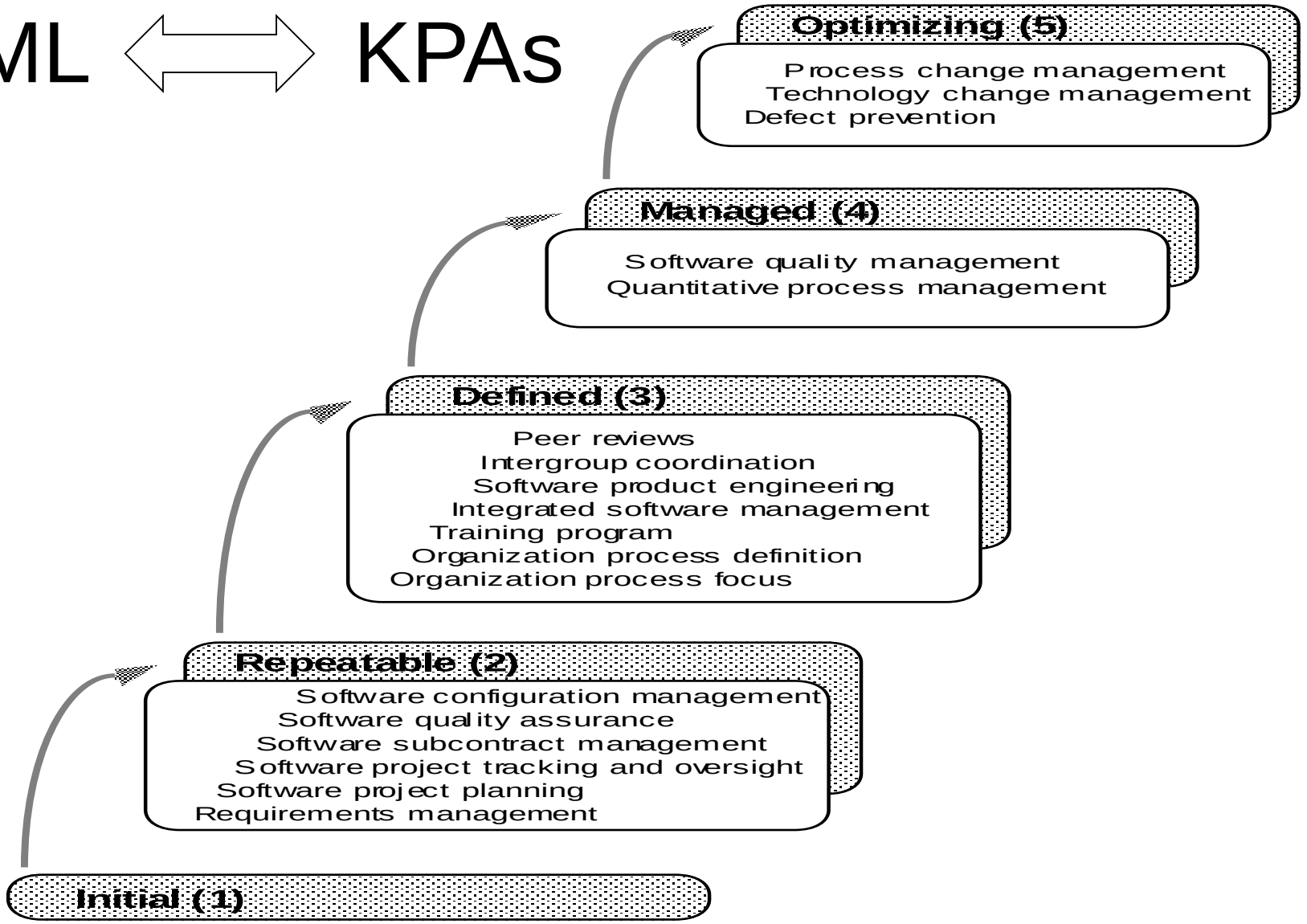
1. **Initial** - Ad hoc and occasionally even chaotic. Few processes are defined. Success depends on individual effort
2. **Repeatable** - Basic project management processes established to track cost, schedule and functionality. Necessary process discipline in place to repeat earlier successes

The Five Levels - Characteristics (2/3)

3. **Defined** - Processes for both management and engineering activities are documented, standardized and integrated into a standard software process. All projects use an approved, tailored version of the organization's standard software process
4. **Managed** - Detailed measures of the software process and product quality are collected and both are quantitatively understood and controlled.

The Five Levels - Characteristics (3/3)

5. **Optimizing** - Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.


ML
KPAs
Optimizing (5)
Process change management
Technology change management
Defect prevention
Managed (4)
Software quality management
Quantitative process management
Defined (3)
Peer reviews
Intergroup coordination
Software product engineering
Integrated software management
Training program
Organization process definition
Organization process focus
Repeatable (2)
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management
Initial (1)

SEI CMM – Details (1/2)

- Each of the Maturity levels will indicate the process capability and contain *Key process areas.*
- Each key process area is organized into five sections called *common features.* The common features specify the key practices that, when collectively addressed, accomplish the *goals* of the key process area.

SEI CMM – Details (2/2)

- The 5 common features are:
  - Commitment to perform
  - Ability to perform
  - Activities performed
  - Measurement and analysis
  - Verify and implementation

Requirement Management KPA

Requirements Management: To establish a common understanding between customer and software project of customer's requirements that will be addressed by the software project.

This agreement with the customer is basis for planning and managing software project. Control of relationship with customer depends on following an effective change control process.

Software Project Planning KPA

Software Project Planning: to establish reasonable plans for performing the software engineering and for managing the software project

These plans are necessary foundation for managing the software project. Without realistic plans, effective project management cannot be implemented

SEI CMM - Assessment

- Assessments are typically conducted every 1-1/2 to 3 years.
- Assessments look at all software processes used in the organization, but may do this by sampling process areas and projects.
- An example of a method to assess an organization's software process capability is the SEI Software Process Assessment method.

SEI CMM - Positioning

- Strengths
  - Specific to software engineering
  - Much more process focus than ISO
  - Road map for improvement
- Weaknesses
  - Specific to software engineering (critical processes like HR, Strategic planning not addressed)
  - Initially, not as widely recognized as ISO
  - Slow updates (current version - 1993)
  - Less emphasis on (overall) results