Addis Ababa University
College of Natural and Computational Sciences
Department of Computer Science

Introduction to Software Engineering (CoSc3311)

Ayalew Belay (PhD)

Chapter 4: 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 organizations 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 organization 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 coherent 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.

Factors influencing project management

- Company size
- Software customers
- Software size
- Software type
- Organizational culture
- Software development processes

These factors mean that project managers in different organizations may work in quite different ways.

Management Activities

- **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 managers are responsible for planning.
  - estimating and scheduling project development and assigning people to tasks.
- **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.

- **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.

Responsibility of Software Project Managers

- A software project manager is the most important person inside a team who takes the overall responsibilities to manage the software projects and plays an important role in the successful completion of the projects.
- A project manager has to face many difficult situations to accomplish these works. The job responsibilities of a project manager range from invisible activities like building up team morale to highly visible customer presentations.

Responsibility of Software Project Managers..

- Most of the managers take responsibility for writing the project proposal, project cost estimation, scheduling, project staffing, software process tailoring, project monitoring and control, software configuration management, risk management, managerial report writing, and presentation, and interfacing with clients.

Activities Performed by Project Manager

- **Project Estimation**
  - **Cost Estimation:** Total expenses to develop the software product is estimated.
  - **Time Estimation:** The total time required to complete the project.
  - **Effort Estimation:** The effort needed to complete the project is estimated.
- **Scheduling**
- **Staffing**
- **Risk Management**

Activities Performed..

- **Miscellaneous Plans**
  - quality assurance plans, configuration management plans, etc.
- **Lead the team:** The project manager must be a good leader who makes a team of different members of various skills and can complete their individual tasks.
- **Motivate the team-member:** One of the key roles of a software project manager is to encourage team members to work properly for the successful completion of the project.
- **Tracking the progress:** The project manager should keep an eye on the progress of the project. A project manager must track whether the project is going as per plan or not.
  - If any problem arises, then take the necessary action to solve the problem. Moreover, check whether the product is developed by maintaining correct coding standards or not.

Activities Performed…

- **Liaison:** The project manager is the link between the development team and the customer. Project manager analysis the customer requirements and convey it to the development team and keep telling the progress of the project to the customer. Moreover, the project manager checks whether the project is fulfilling the customer's requirements or not.
- **Monitoring and reviewing:** Project monitoring is a continuous process that lasts the whole time a product is being developed, during which the project manager compares actual progress and cost reports with anticipated reports as soon as possible. While most firms have a formal system in place to track progress, qualified project managers may still gain a good understanding of the project's development by simply talking with participants.
- **Documenting project report:** The project manager prepares the documentation of the project for future purposes. The reports contain detailed features of the product and various techniques. These reports help to maintain and enhance the quality of the project in the future.
- **Reporting:** Reporting project status to the customer and his or her organization is the responsibility of the project manager. Additionally, they could be required to prepare brief, well-organized pieces that summarize key details from in-depth studies.

Features of a Good Project Manager

1. Knowledge of project estimation techniques.
2. Good decision-making abilities at the right time.
3. Previous experience managing a similar type of projects.
4. Good communication skills to meet the customer satisfaction.
5. A project manager must encourage all the team members to successfully develop the product.
6. He must know the various type of risks that may occur and the solution to these problems.

Software Project planning

Adapted from Ian Sommerville,

Software Engineering, 9th edition.

Chapter 23

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

- Planning is done:
  - 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.

Proposal planning

- Planning may be necessary with only outline software requirements.
- The aim of planning at this stage is to provide information that will be used in setting a price for the system to customers.

Software pricing

- Estimates are made to discover the cost, to the developer, of producing a software system.
  - You take into account, hardware, software, travel, training and effort costs.
- There is not a simple relationship between the development cost and the price charged to the customer.
- Broader organizational, economic, political and business considerations influence the price charged.

Factors affecting software pricing

Factor	Description
Requirements volatility	If the requirements are likely to change, an organization may lower its price to win a contract. After the contract is awarded, high prices can be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a smaller than normal profit or break even than to go out of business. Cash flow is more important than profit in difficult economic times.

Plan-driven development

- Plan-driven or plan-based development is an approach to software engineering where the development process is planned in detail.
  - Plan-driven development is based on engineering project management techniques and is the 'traditional' way of managing large software development projects.
- A project plan is created that records the work to be done, who will do it, the development schedule, and the work products.
- Managers use the plan to support project decision making and as a way of measuring progress.

Plan-driven development – pros and cons

- The arguments in favor of a plan-driven approach are that early planning *allows organizational issues* (availability of staff, other projects, etc.) *to be closely taken into account, and that potential problems and dependencies are discovered before the project starts*, rather than once the project is underway.
- The principal argument against plan-driven development is that *many early decisions have to be revised* because of changes to the environment in which the software is to be developed and used.

Project plans

- In a plan-driven development project, a project plan sets out the resources available to the project, the work breakdown and a schedule for carrying out the work.
- Plan sections
  - Introduction: objectives and constraints
  - Project organization: team, people, roles
  - Risk analysis: risks, probabilities, strategies to address risks
  - Hardware and software resource requirements
  - Work breakdown: activities, milestones, deliverables
  - Project schedule: dependencies among activities, people and time allocated
  - Monitoring and reporting: mechanisms and reports

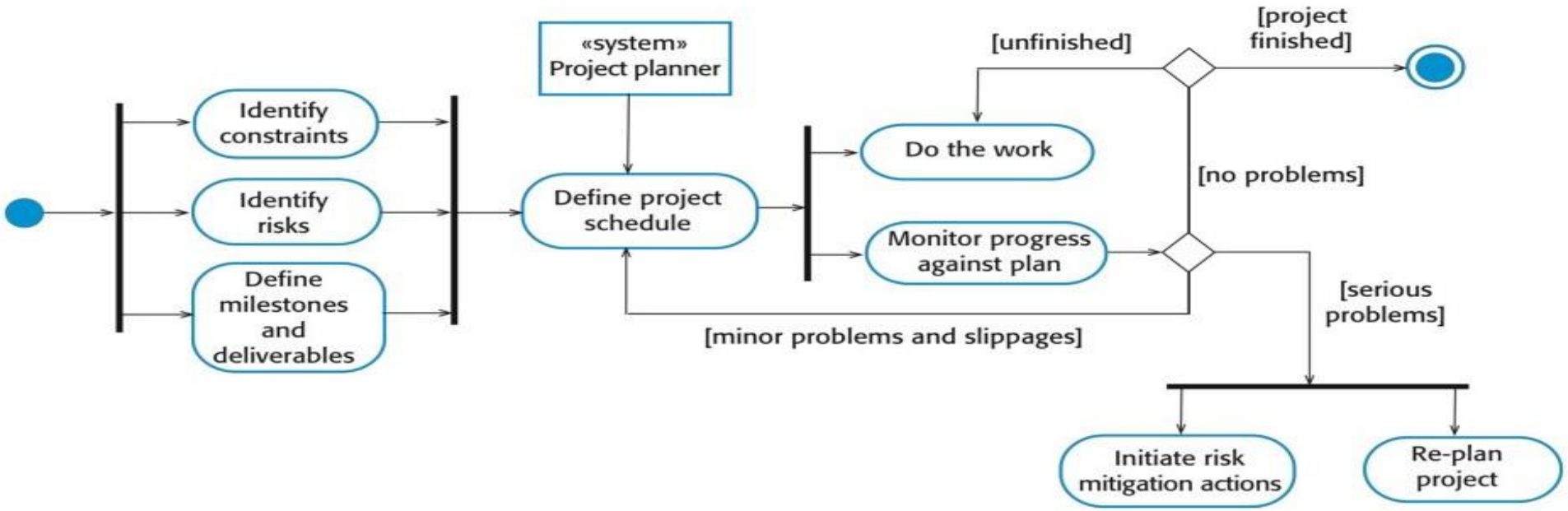
Project plan supplements

Plan	Description
Quality plan	Describes the quality procedures and standards that will be used in a project.
Validation plan	Describes the approach, resources, and schedule used for system validation.
Configuration management plan	Describes the configuration management procedures and structures to be used.
Maintenance plan	Predicts the maintenance requirements, costs, and effort.
Staff development plan	Describes how the skills and experience of the project team members will be developed.

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.

The project planning process



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.

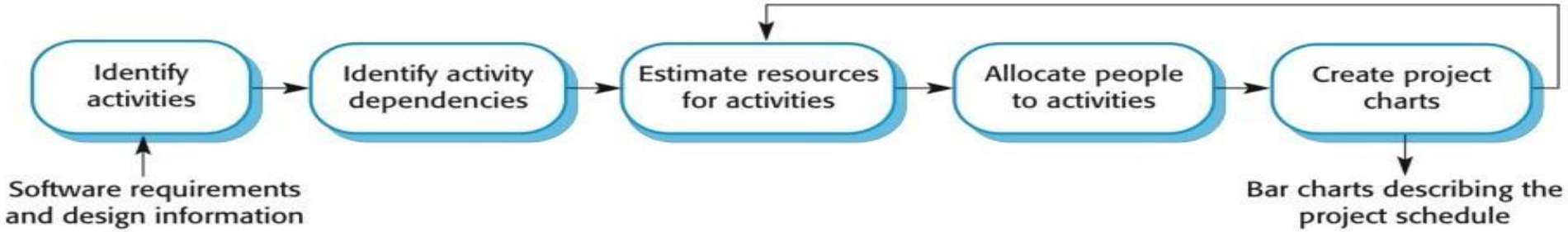
Project scheduling activities

- Split project into tasks and estimate time and resources required to complete each task.
- Organize tasks concurrently to make optimal use of workforce.
- Minimize task dependencies to avoid delays caused by one task waiting for another to complete.
- Dependent on project managers intuition and experience.

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.

The project scheduling process



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.

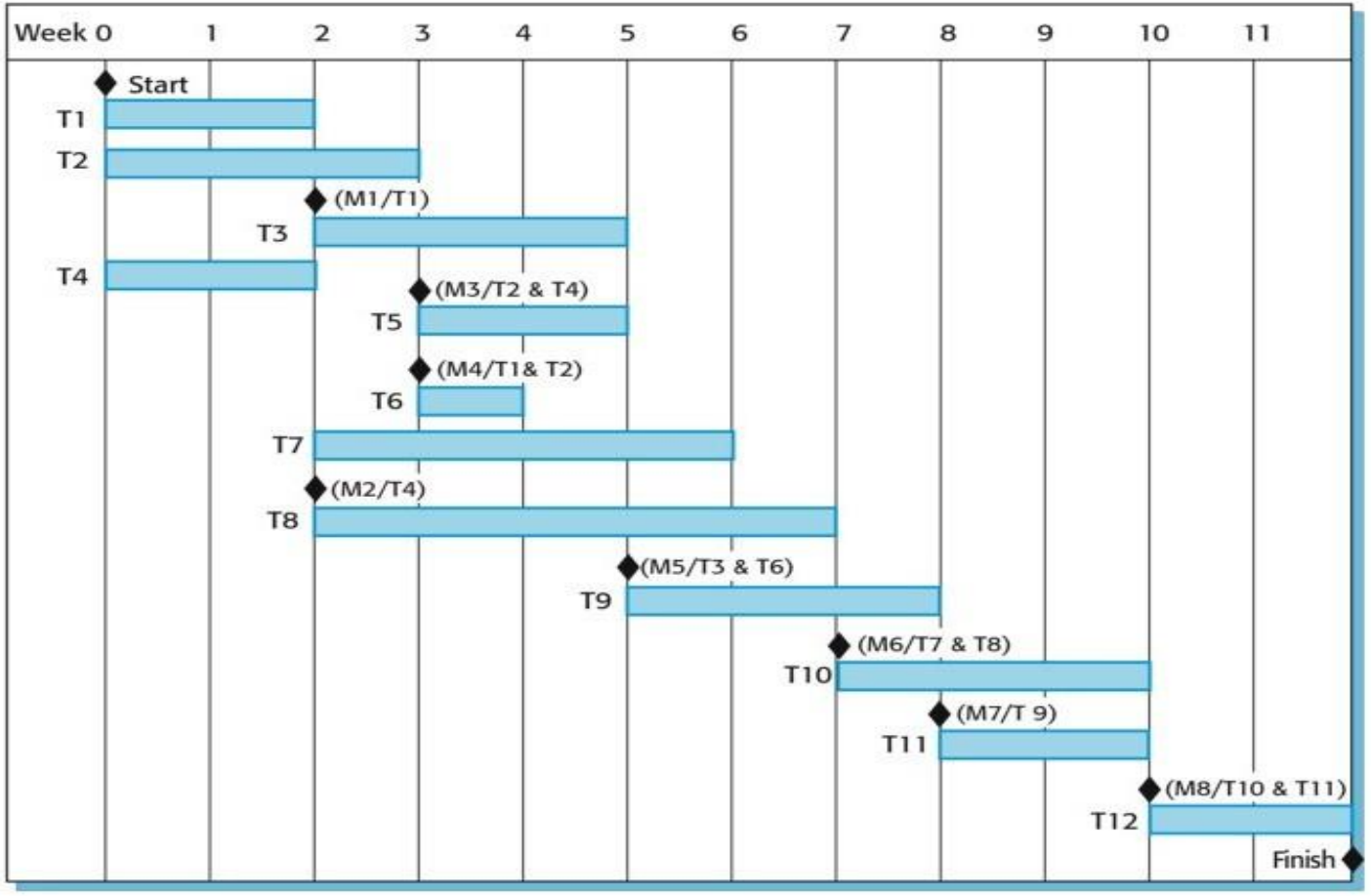
Schedule representation

- 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.

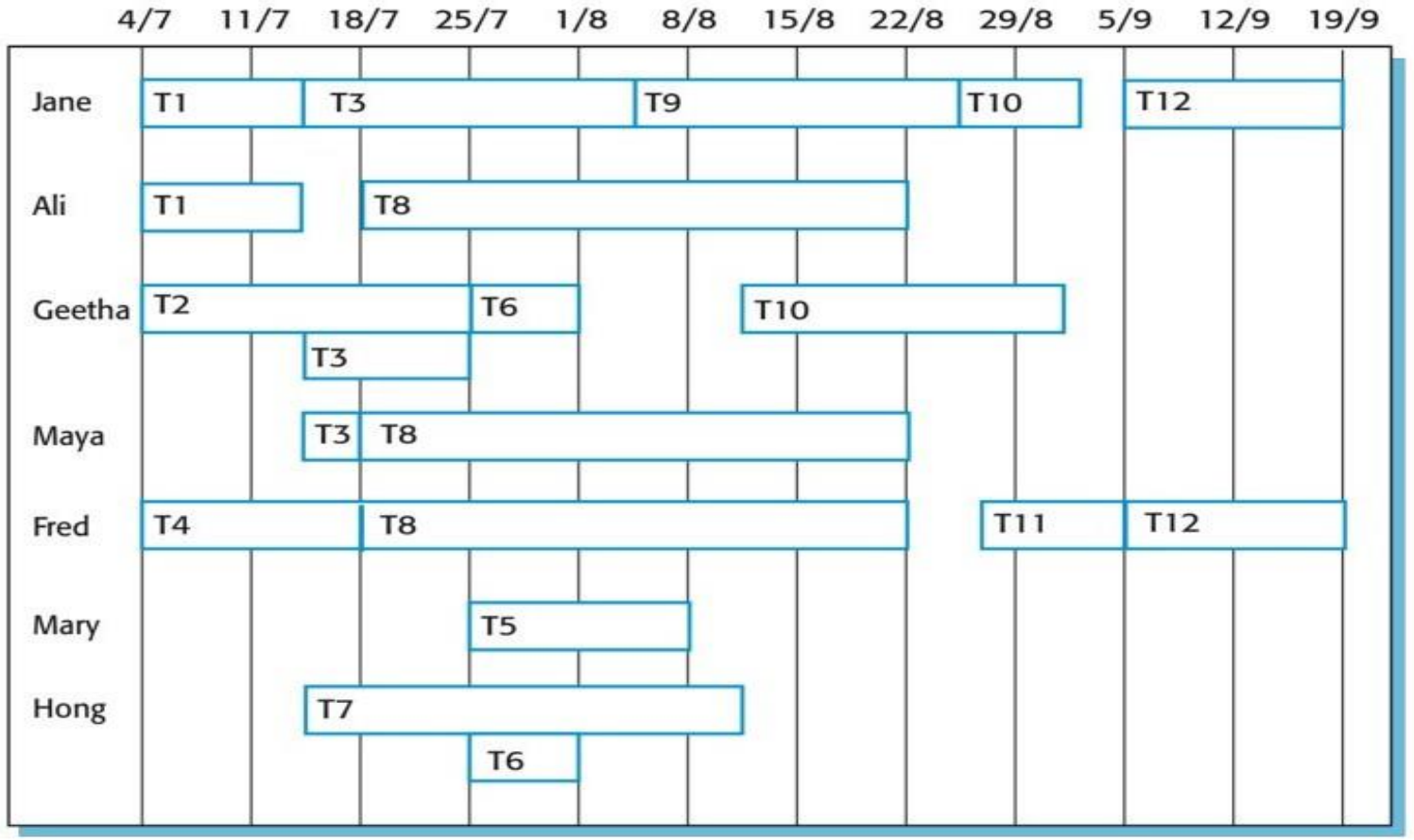
Tasks, durations, and dependencies

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



Agile planning

- Agile methods of software development are iterative approaches where the software is developed and delivered to customers in increments.
- Unlike plan-driven approaches, the functionality of these increments is not planned in advance but is decided during the development.
  - The decision on what to include in an increment depends on progress and on the customer's priorities.
- The customer's priorities and requirements change so it makes sense to have a flexible plan that can accommodate these changes.

Agile planning stages

- There are two stages in agile planning:
  - Release planning, which looks ahead for several months and decides on the features that should be included in a release of a system.
  - Iteration planning, which has a shorter term outlook, and focuses on planning the next increment of a system. This is typically 2-4 weeks of work for the team.

Planning in XP

Story-based planning

- The system specification in XP is based on user stories that reflect the features that should be included in the system.
- The project team read and discuss the stories and rank them in order of the amount of time they think it will take to implement the story.
- Release planning involves selecting and refining the stories that will reflect the features to be implemented in a release of a system and the order in which the stories should be implemented.
- Stories to be implemented in each iteration are chosen, with the number of stories reflecting the time to deliver an iteration (usually 2 or 3 weeks).

Estimation techniques

- Organizations need to make software effort and cost estimates. There are two types of technique that can be used to do this:
  - *Experience-based techniques* The estimate of future effort requirements is based on the manager's experience of past projects and the application domain. Essentially, the manager makes an informed judgment of what the effort requirements are likely to be.
  - *Algorithmic cost modeling* In this approach, a formulaic approach is used to compute the project effort based on estimates of product attributes, such as size, and process characteristics, such as experience of staff involved.

Experience-based approaches

- Experience-based techniques rely on judgments based on experience of past projects and the effort expended in these projects on software development activities.
- Typically, you identify the deliverables to be produced in a project and the different software components or systems that are to be developed.
- You document these in a spreadsheet, estimate them individually and compute the total effort required.
- It usually helps to get a group of people involved in the effort estimation and to ask each member of the group to explain their estimate.

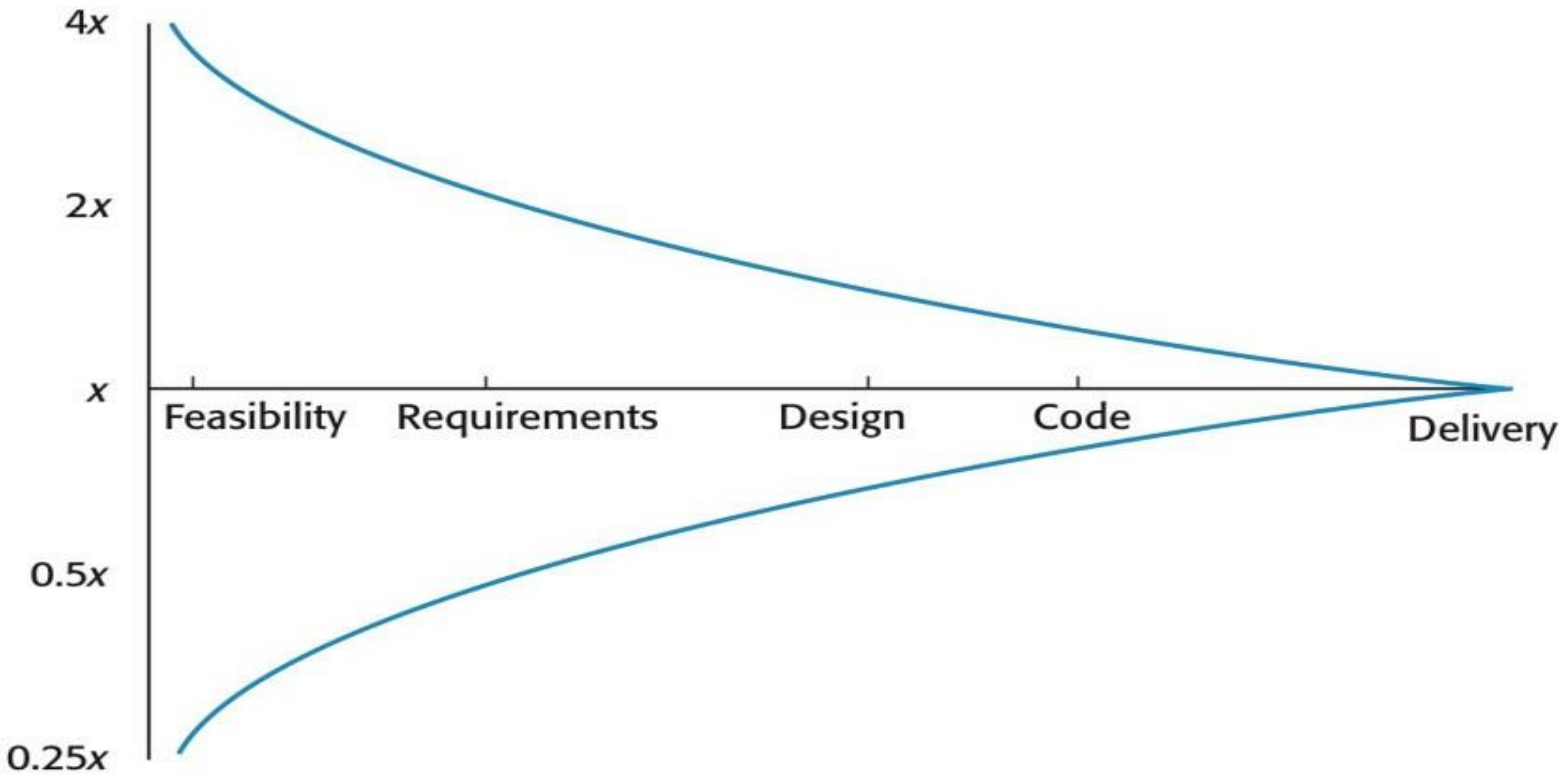
Algorithmic cost modelling

- Cost is estimated as a mathematical function of product, project and process attributes whose values are estimated by project managers:
  - Effort = A ✹ Size^{B} ✹ M
  - A is an organization-dependent constant, B reflects the disproportionate effort for large projects and M is a multiplier reflecting product, process, and people attributes.
- The most commonly used product attribute for cost estimation is code size.
- Most models are similar but they use different values for A, B and M.

Estimation accuracy

- The size of a software system can only be known accurately when it is finished.
- Several factors influence the final size
  - Use of COTS and components;
  - Programming language;
  - Distribution of system.
- As the development process progresses then the size estimate becomes more accurate.
- The estimates of the factors contributing to B and M are subjective and vary according to the judgment of the estimator.

Estimate uncertainty



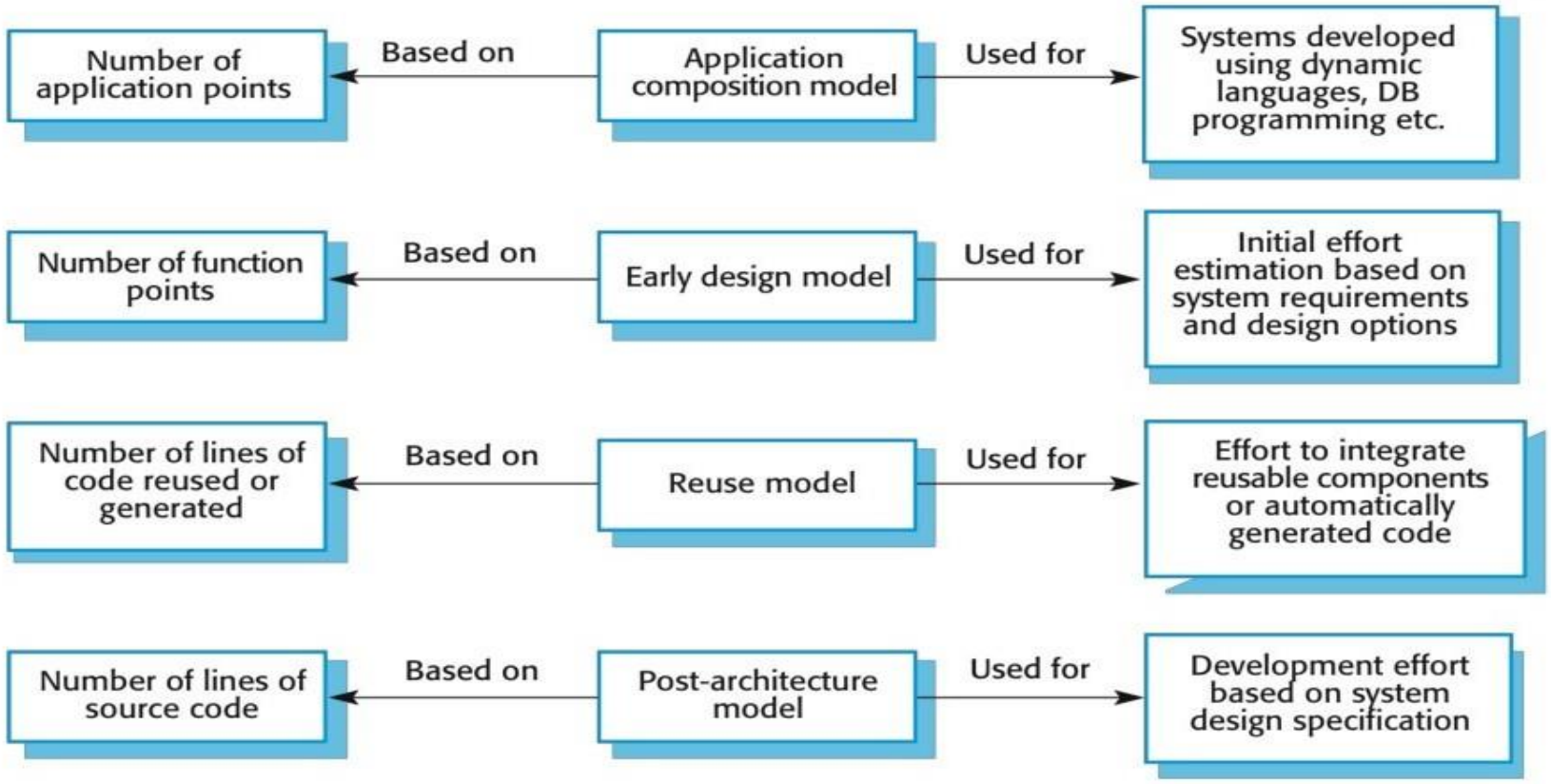
The COCOMO 2 model

- An empirical model based on project experience.
- Well-documented, 'independent' model which is not tied to a specific software vendor.
- Long history from initial version published in 1981 (COCOMO-81) through various instantiations to COCOMO 2.
- COCOMO 2 takes into account different approaches to software development, reuse, etc.

COCOMO 2 models

- COCOMO 2 incorporates a range of sub-models that produce increasingly detailed software estimates.
- The sub-models in COCOMO 2 are:
  - Application composition model. Used when software is composed from existing parts.
  - Early design model. Used when requirements are available but design has not yet started.
  - Reuse model. Used to compute the effort of integrating reusable components.
  - Post-architecture model. Used once the system architecture has been designed and more information about the system is available.

COCOMO estimation models



Key points

- Estimation techniques for software may be experience-based, where managers judge the effort required, or algorithmic, where the effort required is computed from other estimated project parameters.
- The COCOMO II costing model is an algorithmic cost model that uses project, product, hardware and personnel attributes as well as product size and complexity attributes to derive a cost estimate.

The organization of SPMP document

- **Introduction:**
  - Objectives
  - Major Functions
  - Performance Issues
  - Management and Technical Constraints

The organization of SPMP document…

- **Project Estimates:**
  - Historical Data Used
  - Estimation Techniques Used
  - Effort, Resource, Cost, and Project Duration Estimates
- **Schedule:**
  - Work Breakdown Structure
  - Task Network Representation
  - Gantt Chart Representation
  - PERT Chart Representation

The organization of SPMP document…

- **Project Resources:**
- People
- Hardware and Software
- Special Resources
- **Staff Organization:**
- Team Structure
- Management Reporting

The organization of SPMP document…

- **Risk Management Plan:**
  - Risk Analysis
  - Risk Identification
  - Risk Estimation
  - Risk Abatement Procedures
- **Project Tracking and Control Plan:**
- **Miscellaneous Plans:**
  - Process Tailoring
  - Quality Assurance Plan
  - Configuration Management Plan
  - Validation and Verification
  - System Testing Plan
  - Delivery, Installation, and Maintenance Plan

Project Size Estimation Metrics

- Accurate estimation of the problem size is fundamental to satisfactory estimation of effort, time duration and cost of a software project
- In order to be able to accurately estimate the project size, some important metrics should be defined in terms of which the project size can be expressed.
- The project size is a measure of the problem complexity in terms of the effort and time required to develop the product

Project Size Estimation Metrics…

- Currently two metrics are popularly being used widely to estimate size:
  - Lines of code (LOC) and function point (FP)

Lines of Code (LOC)

- LOC is the simplest among all metrics available to estimate project size.
- This metric is very popular because it is the simplest to use.
- Using this metric, the project size is estimated by counting the number of source instructions in the developed program.

Lines of Code (LOC)…

- Determining the LOC count at the end of a project is a very simple job.
- However, accurate estimation of the LOC count at the beginning of a project is very difficult.
- In order to estimate the LOC count at the beginning of a project, project managers usually divide the problem into modules, and each module into sub modules and so on, until the sizes of the different leaf-level modules can be approximately predicted.
- To be able to do this, past experience in developing similar products is helpful. By using the estimation of the lowest level modules, project managers arrive at the total size estimation.

Cons and pros of LOC

Advantages

- Universally accepted and is used in many models like COCOMO.
- Estimation is closer to the developer's perspective.
- Both people throughout the world utilize and accept it.
- At project completion, LOC is easily quantified.
- It has a specific connection to the result.
- Simple to use.

Disadvantages:

- Different programming languages contain a different number of lines.
- No proper industry standard exists for this technique.
- It is difficult to estimate the size using this technique in the early stages of the project.
- When platforms and languages are different, LOC cannot be used to normalize.

Function point (FP)

- Function point metric was proposed by Albrecht [1983].
- This metric overcomes many of the shortcomings of the LOC metric.
- One of the important advantages of using the function point metric is that it can be used to easily estimate the size of a software product directly from the problem specification.
- This is in contrast to the LOC metric, where the size can be accurately determined only after the product has fully been developed.
- The conceptual idea behind the function point metric is that the size of a software product is directly dependent on the number of different functions or features it supports.
- Computation of the number of input and the output data values to a system gives some indication of the number of functions supported by the system.

Function point…

- In addition to the number of basic functions that software performs, the size is also dependent on the number of files and the number of interfaces.
- The size of a product in function points (FP) can be expressed as the weighted sum of these five problem characteristics.
- Function point is computed in two steps.
  - The first step is to compute the **unadjusted function point** (UFP).
- **UFP** = (Number of inputs)*4 + (Number of outputs)*5 + (Number of inquiries)*4 + (Number of files)*10 + (Number of interfaces)*10
-

Function point…

External Inputs (EI): It consists of each user input that provides application data and control information.

External Outputs (EO): It includes user output that provides application oriented information to the user such as Reports.

External Inquiries (EQ): An Inquiry is defined as online input those results into the generation of immediate response in the form of online output.

Internal Logical Files (ILF's): It is the logical grouping of data that resides within application boundaries.

External Interface Files (EIF's): It is logical grouping of data that resides out of application.

UFP = (Number of inputs)*4 + (Number of outputs)*5 + (Number of inquiries)*4 + (Number of files)*10 + (Number of interfaces)*10

Type of components	Complexity of Component			
	Low	***Average***	***High***	***Total***
External Inputs (EI)	__×3=__	__× 4=__	__× 6=__	
External Outputs (EO)	__× 4=__	__× 5=__	__×7=__	
External Inquiries(EQ)	__×3=__	__× 4=__	__×6=__	
Internal logical Files(IFL)	__×7=__	_×10=__	__×15=__	
External Interface Files(EIF)	__×5=__	__×7=__	__×10=__	
Total Number of Unadjusted Function Points				

Source: Chemuturi, K., Murali," *Software Estimation Best Practices, Tools, and Techniques: A Complete Guide for Software Project Estimators*", 2009.

Function point…

- The second step is to compute the **technical complexity factor** (TCF)
- Each of these 14 factors is assigned from 0 (not present or no influence) to 5 (strong influence). The resulting numbers are summed, yielding the **total degree of influence (DI)**.

Factors	Components	Description
F1	Data communications	How many communication facilities are there to transfer or exchange information with the application or system?
F2	Distributed data processing	How are distributed data and processing functions Handled?
F3	Performance	Did the user require response time or throughput?
F4	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
F5	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
F6	On-Line data entry	What percentage of the information is entered on-line?
F7	End-user efficiency	Was the application designed for end-user efficiency?
F8	On-Line update	How many ILF's are updated by On-Line Transaction?
F9	Complex processing	Does the application have extensive logical or mathematical processing?
F10	Reusability	Was the application developed to meet one or many users' needs?
F11	Installation ease	How difficult is conversion and installation?
F12	Operational ease	How effective and/or automated are start-up, backup, and recovery procedures?
F13	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
F14	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Source: Chemuturi, K., Murali," *Software Estimation Best Practices, Tools, and Techniques: A Complete Guide for Software Project Estimators*", 2009.

Rate each factor on a scale of 0 to 5.

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

TCF is computed as (0.65+0.01*DI).
Where **DI= $\sum(f_i)$**
Finally, FP=UFP*TCF.

Cons and pros of FP

Advantages:

- It can be easily used in the early stages of project planning.
- It is independent of the programming language.
- It can be used to compare different projects even if they use different technologies(database, language, etc).

Disadvantages:

- It is not good for real-time systems and embedded systems.
- Many cost estimation models like COCOMO use LOC and hence FPC must be converted to LOC.

Comparison between Function point (FP) and Lines Of Code (LOC)

FP	LOC
1. FP is specification based.	1. LOC is an analogy based.
2. FP is language independent.	2. LOC is language dependent.
3. FP is user-oriented.	3. LOC is design-oriented.
4. It is extendible to LOC.	4. It is convertible to FP (backfiring)

$E = a(KLOC)^b$

$Time = c(Effort)^d$

$Person\ required = Effort/\ time$

The above formula is used for the cost estimation of the basic COCOMO model and also is used in the subsequent models. The constant values a, b, c, and d for the Basic Model for the different categories of the system:

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

No	Programming language	LOC/FP
1	C	128
2	COBOL	105
3	FORTAN	105
4	PASCAL	90
5	ADA	70
6	C++	53
7	Object Oriented Languages	30
8	4. Generation Languages	20
9	Code Generators	15
10	SQL	13

Example: Compute the function point, productivity, documentation, cost per function for the following data:

1. Number of user inputs = 24
2. Number of user outputs = 46
3. Number of inquiries = 8
4. Number of files = 4
5. Number of external interfaces = 2
6. Effort = 36.9 p-m
7. Technical documents = 265 pages
8. User documents = 122 pages
9. Cost = $7744/ month

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Solution:

Measurement Parameter	Count		Weighing factor
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total →	2	*	5 = 10 378

So sum of all f_i (i ← 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

FP = Count-total * [0.65 + 0.01 *∑(f_i)]

= 378 * [0.65 + 0.01 * 43]

= 378 * [0.65 + 0.43]

= 378 * 1.08 = 408

$$\text{Productivity} = \frac{\text{FP}}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

Total pages of documentation = technical document + user document

= 265 + 122 = 387pages

Documentation = Pages of documentation/FP

= 387/408 = 0.94

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$

Consider a software project with the following information domain characteristic for the calculation of function point metric.

Number of external inputs (I) = 30

Number of external output (O) = 60

Number of external inquiries (E) = 23

Number of files (F) = 08

Number of external interfaces (N) = 02

It is given that the complexity weighting factors for I, O, E, F, and N are 4, 5, 4, 10, and 7, respectively. It is also given that, out of fourteen value adjustment factors that influence the development effort, four factors are not applicable, each of the other four factors has value 3, and each of the remaining factors has value 4. The computed value of the function point metric is _____

Project Estimation Techniques

- **The popular project size estimation techniques used in software engineering are:**
  - **Expert Judgment:** a group of experts in the relevant field estimates the project size based on their experience and expertise. This technique is often used when there is limited information available about the project.
  - **Analogous Estimation:**This technique involves estimating the project size based on the similarities between the current project and previously completed projects. This technique is useful when historical data is available for similar projects.
  - **Bottom-up Estimation:** the project is divided into smaller modules or tasks, and each task is estimated separately. The estimates are then aggregated to arrive at the overall project estimate.

Project Estimation Techniques…

- **Three-point Estimation:** This technique involves estimating the project size using three values: optimistic, pessimistic, and most likely. These values are then used to calculate the expected project size using a formula such as the PERT formula
- **Function Points:** his technique involves estimating the project size based on the functionality provided by the software. Function points consider factors such as inputs, outputs, inquiries, and files to arrive at the project size estimate.
- **Use Case Points:** This technique involves estimating the project size based on the number of use cases that the software must support. Use case points consider factors such as the complexity of each use case, the number of actors involved, and the number of use cases.

Project Estimation Techniques…

- **Parametric Estimation:** For precise size estimation, mathematical models founded on project parameters and historical data are used.
- **COCOMO (Constructive Cost Model):** It is an algorithmic model that estimates effort, time, and cost in software development projects by taking into account a number of different elements.
- **Wideband Delphi:** Consensus-based estimating method for balanced size estimations that combines expert estimates from anonymous experts with cooperative conversations.
- **Monte Carlo simulation:** This technique, which works especially well for complicated and unpredictable projects, estimates project size and analyses hazards using statistical methods and random sampling.

Why Project Size Estimation?

- **Resource Allocation:** Appropriate distribution of financial and human resources is ensured by accurate estimation.
- **Risk management:** Early risk assessment helps with mitigation techniques by taking into account the complexity of the project.
- **Time management:** Facilitates the creation of realistic schedules and milestones for efficient time management.
- **Cost control and budgeting:** Both the terms are closely related, which lowers the possibility of cost overruns.
- **Resource Allocation:** Enables efficient task delegation and work allocation optimization.
- **Scope Definition:** Defines the scope of a project, keeps project boundaries intact and guards against scope creep.

Scheduling, Organization and Team Structures

Project Scheduling

- Project-task scheduling is a significant project planning activity. It comprises deciding which functions would be taken up when. To schedule the project plan, a software project manager wants to do the following:

1. Identify all the functions required to complete the project.
2. Break down large functions into small activities.
3. Determine the dependency among various activities.
4. Establish the most likely size for the time duration required to complete the activities.
5. Allocate resources to activities.
6. Plan the beginning and ending dates for different activities.
7. Determine the critical path. A critical way is the group of activities that decide the duration of the project.

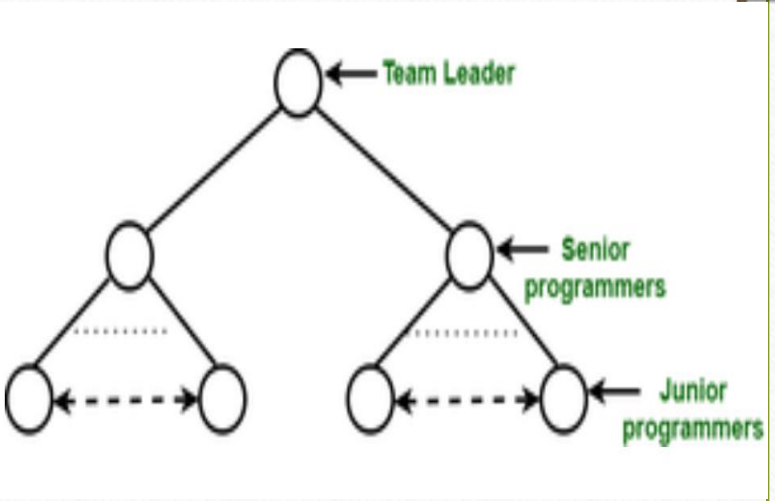
Software project Team Organization and Structures

Software Project Team Organization

- There are many ways to organize the project team

Hierarchical team organization :

- the people of organization at different levels following a tree structure. People at bottom level generally possess most detailed knowledge about the system. People at higher levels have broader appreciation of the whole project.



Benefits of hierarchical team organization :

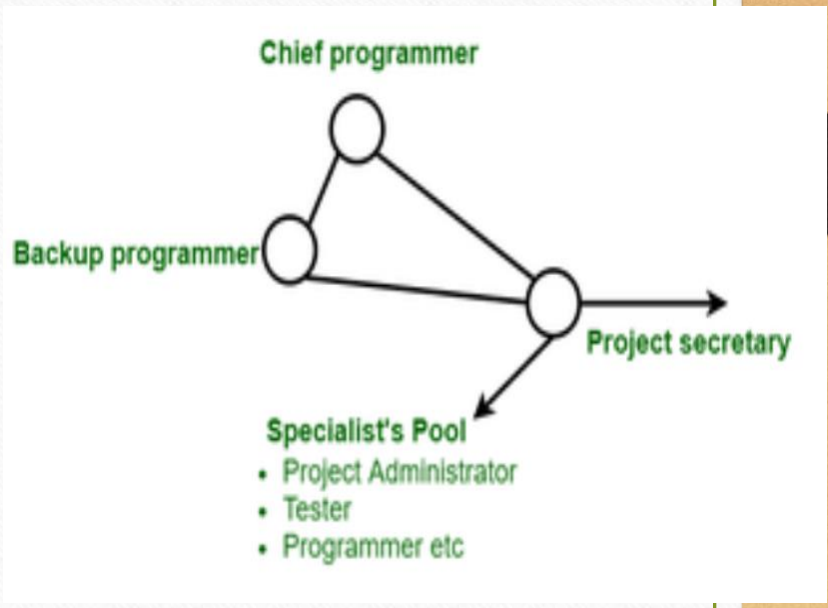
- It limits the number of communication paths and stills allows for the needed communication.
- It can be expanded over multiple levels.
- It is well suited for the development of the hierarchical software products.
- Large software projects may have several levels.

Limitations of hierarchical team organization

- As information has to be travel up the levels, it may get distorted.
- Levels in the hierarchy often judges people socially and financially.
- Most technical competent programmers tend to be promoted to the management positions which may result in loss of good programmer and also bad manager.

Chief-programmer team organization

- This team organization is composed of a small team consisting the following team members
- **The Chief programmer :** It is the person who is actively involved in the planning, specification and design process and ideally in the implementation process as well.
- **The project assistant :** It is the closest technical co-worker of the chief programmer.
- **The project secretary :** It relieves the chief programmer and all other programmers of administration tools.
- **Specialists :** These people select the implementation language, implement individual system components and employ software tools and carry out tasks.



Advantages of Chief-programmer team organization

- Centralized decision-making
- Reduced communication paths
- Small teams are more productive than large teams
- The chief programmer is directly involved in system development and can exercise the better control function.

Disadvantages of Chief-programmer team organization

- Project survival depends on one person only.
- Can cause the psychological problems as the "chief programmer" is like the "king" who takes all the credit and other members are resentful.
- Team organization is limited to only small team and small team cannot handle every project.
- Effectiveness of team is very sensitive to Chief programmer's technical and managerial activities.

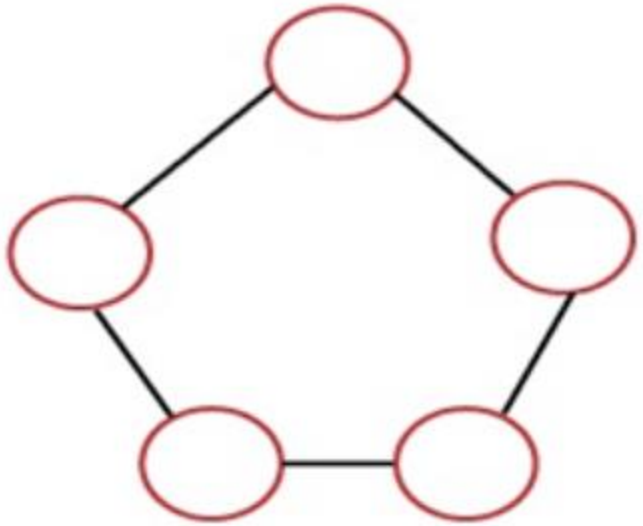
Matrix Team Organization

- In matrix team organization, people are divided into specialist groups. Each group has a manager. Example of Metric team organization is as follows :

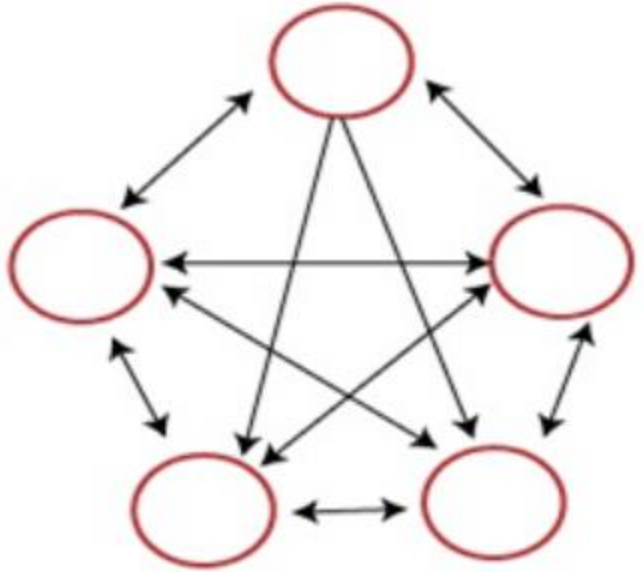
Egoless Team Organization

Egoless programming is a state of mind in which programmer are supposed to separate themselves from their product. In this team organization goals are set and decisions are made by group consensus. Here group, 'leadership' rotates based on tasks to be performed and differing abilities of members.

Ego-Less Programming Team structure and communication paths



(a) Structure



(b) Communication path

Democratic Team Organization

- It is quite similar to the egoless team organization, but one member is the team leader with some responsibilities :

- Coordination
- Final decisions, when consensus cannot be reached.

Democratic…

- **Advantages of Democratic Team Organization :**

- Each member can contribute to decisions.
- Members can learn from each other.
- Improved job satisfaction.
- **Disadvantages of Democratic Team Organization :**

- Communication overhead increased.
- Need for compatibility of members.
- Less individual responsibility and authority.

Staffing

- Staffing is the art of acquiring, developing, and maintaining a satisfactory and satisfied workforce. Staffing is that function by which a manager builds an organization through the recruitment, selection, and development of the individual, which also includes a series of activities.
- It ensures that the organization has the right number of people at the right places, at the right time, and performing the right thing.

Staffing ….


Manpower Requirements
Recruitment
Selection
Placement and Orientation
Training and Development
Performance Appraisal
Promotion / Career Planning
Compensation
Staffing Process

Risk Management

- R isk is a problem that could cause some loss or threaten the progress of the project, but which has not happened yet
- Risk Management is the system of identifying addressing and eliminating these problems before they can damage the project.
- There are three main classifications of risks which can affect a software project:

Project risks

- Project risks concern differ forms of budgetary, schedule, personnel, resource, and customer-related problems.
- A vital project risk is schedule slippage. Since the software is intangible, it is very tough to monitor and control a software project. It is very tough to control something which cannot be identified.

Technical risks

- Technical risks concern potential method, implementation, interfacing, testing, and maintenance issue. It also consists of an ambiguous specification, incomplete specification, changing specification, technical uncertainty, and technical obsolescence. Most technical risks appear due to the development team's insufficient knowledge about the project.

Business risks

- This type of risks contain risks of building an excellent product that no one need, losing budgetary or personnel commitments, etc.

Other risk categories

1. Known risks: Those risks that can be uncovered after careful assessment of the project program, the business and technical environment in which the plan is being developed, and more reliable data sources (e.g., unrealistic delivery date)

2. Predictable risks: Those risks that are hypothesized from previous project experience (e.g., past turnover)

3. Unpredictable risks: Those risks that can and do occur, but are extremely tough to identify in advance.

Risk Management Activities

Risk Identification

1. **Technology risks:** Risks that assume from the software or hardware technologies that are used to develop the system.
2. **People risks:** Risks that are connected with the person in the development team.
3. **Organizational risks:** Risks that assume from the organizational environment where the software is being developed.
4. **Tools risks:** Risks that assume from the software tools and other support software used to create the system.
5. **Requirement risks:** Risks that assume from the changes to the customer requirement and the process of managing the requirements change.
6. **Estimation risks:** Risks that assume from the management estimates of the resources required to build the system

Risk Control

- **There are three main methods to plan for risk management:**

1. **Avoid the risk:** This may take several ways such as discussing with the client to change the requirements to decrease the scope of the work, giving incentives to the engineers to avoid the risk of human resources turnover, etc.
2. **Transfer the risk:** This method involves getting the risky element developed by a third party, buying insurance cover, etc.
3. **Risk reduction:** This means planning method to include the loss due to risk. For instance, if there is a risk that some key personnel might leave, new recruitment can be planned.

Quality Asurance

- **Software Quality Assurance (SQA)** is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.
- Software Quality Assurance is a process which works parallel to Software Development.
- It focuses on improving the process of development of software so that problems can be prevented before they become a major issue.
- Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

Elements of Software Quality Assurance (SQA)

- Standards
- Reviews and audits
- Testing
- Error/defect collection and analysis
- Change management
- Training/Education

Project Monitoring Plans

- **Project monitoring** refers to surveillance and tracking of the project to ensure that all the tasks are completed on time.
- It includes steps to figure out the hurdles or gaps and resolve them to increase the efficiency of the project.
- It is also known as project monitoring and control.
- It starts as soon as a project starts

Importance of Project Monitoring

1. Ensure that the allotted budget is spent correctly and can be altered if needed.
2. To make sure that the selected task and deadlines are met.
3. To encourage accountability regarding the task assigned by the members of the team.
4. To shift the workforce to a particular task if it requires so.
5. To boost communication between the team members to increase quality and reduce time.

Types of Project Monitoring

- Cost Monitoring
- Schedule Monitoring
- Process monitoring

Project Monitoring Process

- Initiation
- Planning
- Executing
- Closing
- Control and Monitoring

How to Create a Project Monitoring Plan

- Identify the goals of the project
- Define the indicators
- Define data collection methods and timing
- Identify roles and responsibilities during monitoring
- Create an analysis plan and report templates
- Plan data disclosure

Quality Assurance

- Coming soon

Chapter 7: Software Quality Assurance

An overview of Software testing

- Software testing is a process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.
- It is method to assess the functionality of the software program.
- The process checks whether the actual software matches the expected requirements and ensures the software is bug-free.
- The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements.
- It mainly aims at measuring the specification, functionality, and performance of a software program or application.

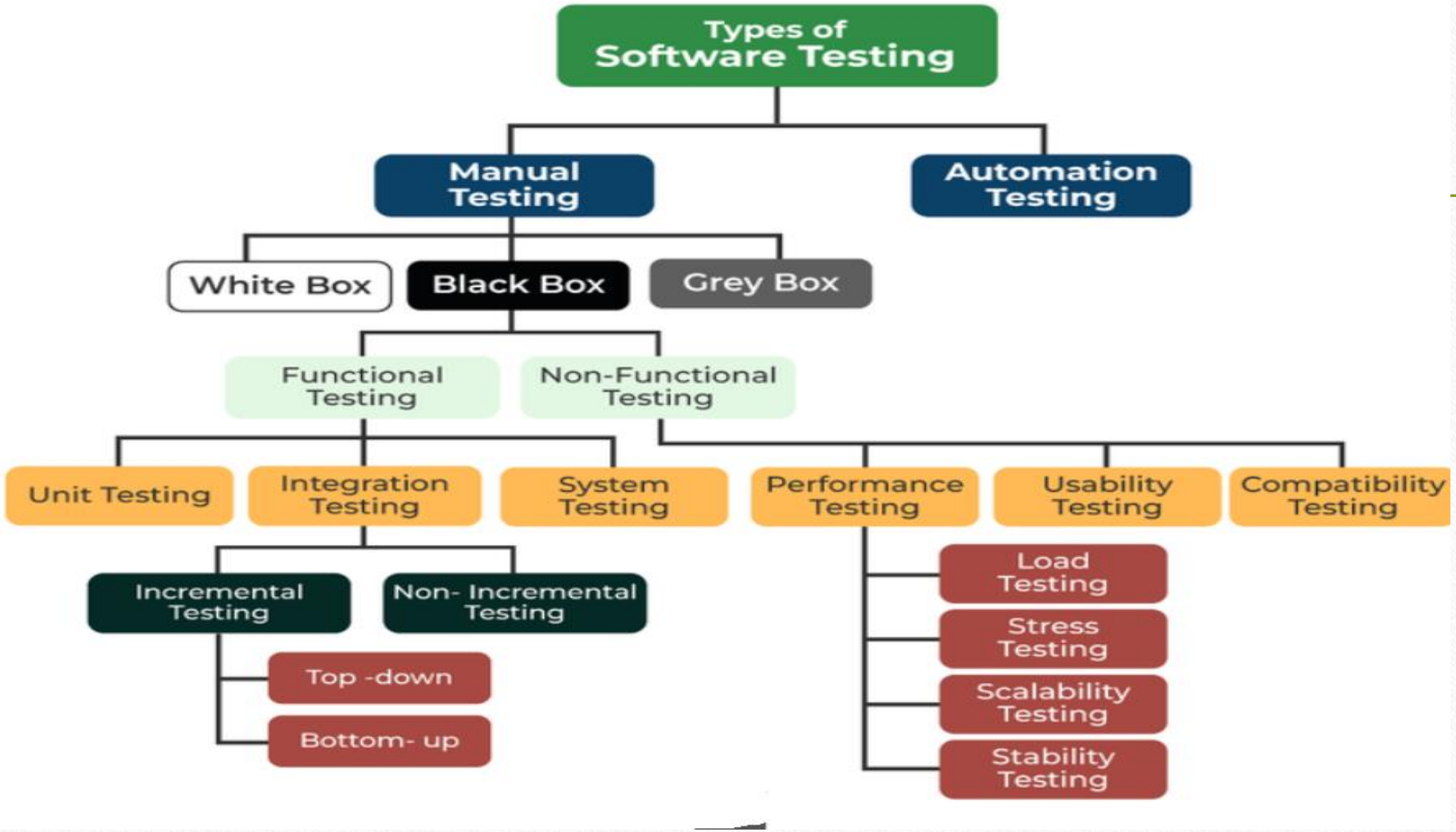
Software testing can be divided into two steps

- **Verification:** It refers to the set of tasks that ensure that the software correctly implements a specific function. It means "Are we building the product right?"
- **Validation:** It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means "Are we building the right product?"

Importance of Software Testing:

- **Defects can be identified early:** Software testing is important because if there are any bugs they can be identified early and can be fixed before the delivery of the software.
- **Improves quality of software:** Software Testing uncovers the defects in the software, and fixing them improves the quality of the software.
- **Increased customer satisfaction:** Software testing ensures reliability, security, and high performance which results in saving time, costs, and customer satisfaction.
- **Helps with scalability:** Software testing type non-functional testing helps to identify the scalability issues and the point where an application might stop working.
- **Saves time and money:** After the application is launched it will be very difficult to trace and resolve the issues, as performing this activity will incur more costs and time. Thus, it is better to conduct software testing at regular intervals during software development.

Different Types Of Software Testing



Types of Software Testing

1. **Black box Testing**: Testing in which the tester doesn't have access to the source code of the software and is conducted at the software interface without any concern with the internal logical structure of the software known as black-box testing.
2. **White box Testing**: Testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing.
3. **Grey Box Testing**: Testing in which the testers should have knowledge of implementation, however, they need not be experts.

Levels of Software Testing

- **Unit testing:** It a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed
- **Integration testing:** It is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
- **System testing:** It is a level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements
- **Acceptance testing:** It is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery

Test Activities

1. Test planning
2. Test monitoring and control
3. Test analysis
4. Test Design
5. Test Implementation
6. Test Execution
7. Test Completion

Test Plan

- It is incumbent upon planners to consider the following issues before initiating a specific test plan:
  - What to test?
  - Which sources to use for test cases?
  - Who is to perform the tests?
  - Where to perform the tests?
  - When to terminate the tests?

Test plan ..

- Changes may be required in the testing plan as a result of:

■ Unavailability of resources.

■ Time requirements are too long and will cause the project to go beyond its completion schedule.

■ Disagreements may arise about the evaluations of the expected damage and risk severity levels and/or about estimates of time and resources required for the testing activities.

- The final testing plan will be completed only after these issues are resolved. Of course, the plan may need to be updated as the project proceeds, to reflect changes in conditions, including delays in project implementation.

The software test plan (STP) – template

1 Scope of the tests

1.1 The software package to be tested (name, version and revision)
1.2 The documents that provide the basis for the planned tests (name and version for each document)

2 Testing environment

2.1 Testing sites
2.2 Required hardware and firmware configuration
2.3 Participating organizations
2.4 Manpower requirements
2.5 Preparation and training required of the test team

3 Test details (for each test)

3.1 Test identification
3.2 Test objective
3.3 Cross-reference to the relevant design document and the requirement document
3.4 Test class
3.5 Test level (unit, integration or system tests)
3.6 Test case requirements
3.7 Special requirements (e.g., measurements of response times, security requirements)
3.8 Data to be recorded

4 Test schedule (for each test or test group) including time estimates for the following:

4.1 Preparation
4.2 Testing
4.3 Error correction
4.4 Regression tests

Test Design

The products of the test design stages are:

- Detailed design and procedures for each test

- Test case database/file.

The test design is carried out on the basis of the software test plan as documented by STP.

The test procedures and the test case database/file may be documented in a “software test procedure” document and “test case file” document or in a single document called the “software test description” (STD).

Test case design

A test case is a documented set of the data inputs and operating conditions required to run a test item together with the expected results of the run.

The tester is expected to run the program for the test item according to the test case documentation, and then compare the actual results with the expected results noted in the documents.

Application of the test case will produce one or more of the following types of expected results:

-Numerical

-Alphabetic (name, address, etc.)

-Error message. Standard output informing user about missing data, erroneous data, unmet conditions, etc.

Test case sources

There are two basic sources for test cases:

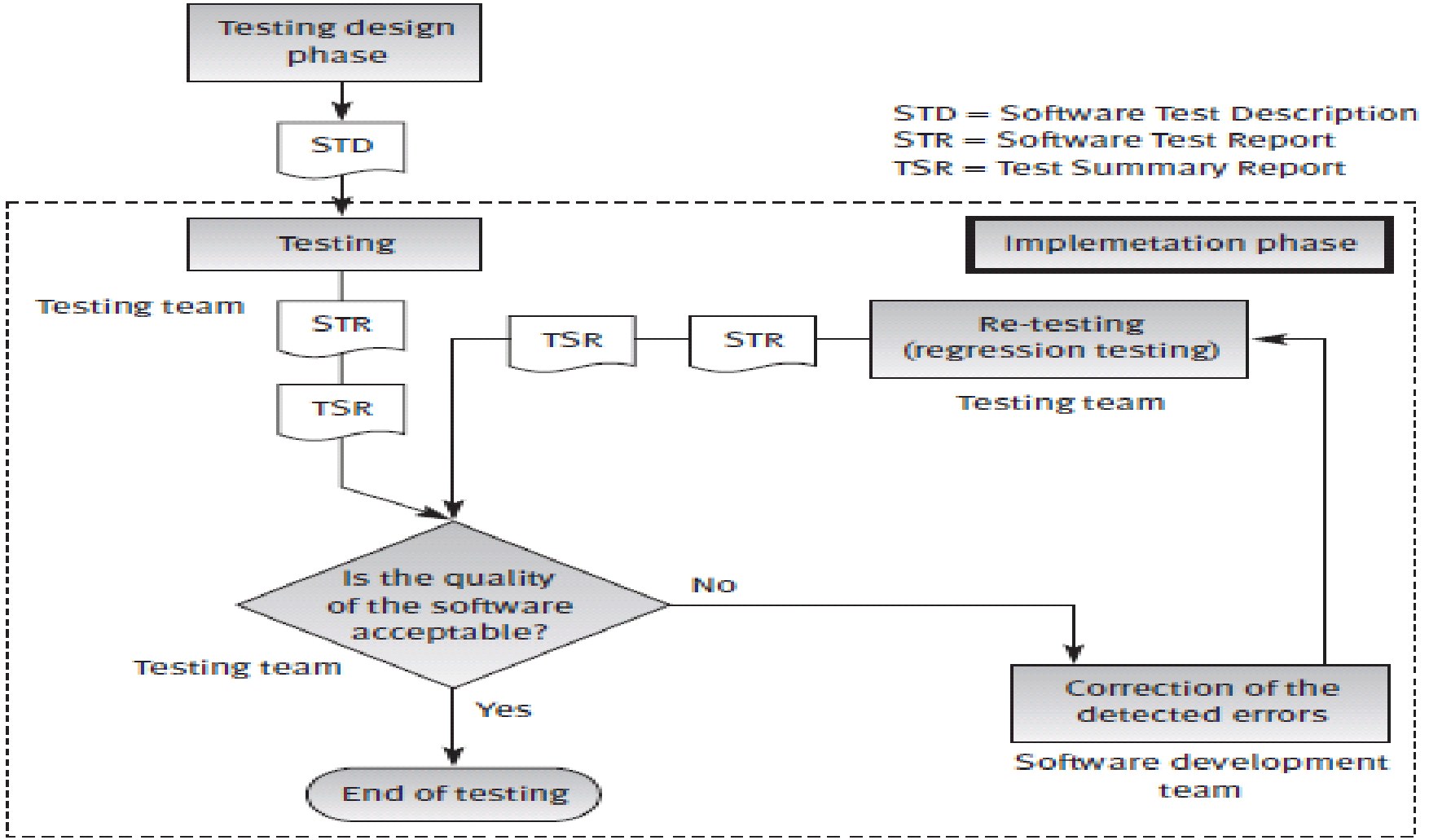
-Random samples of real life cases

-Synthetic test cases (also called “simulated test cases”) prepared by test designers.

In most cases, the test case file preferred should combine sample cases with synthetic cases so as to overcome the disadvantages of a single source of test cases and to increase the efficiency of the testing process.

In the case of combined test case files, test plans are often carried out in two stages: in the first stage, synthetic test cases are used. After correction of the detected errors, a random sample of test cases is used in the second stage.

Test implementation



Thank You