
Software
Engineering

Unit 8: Software Engineering

- Introduction to Software Engineering: Characteristics, Emergence of Software Engineering, Software Metrics & Models, Process & Product Metrics. Software Life Cycle Models: Waterfall, Prototype and Spiral Models and their Comparison.
- Software Project Management: Size Estimation- LOC and FP Metrics, Cost Estimation-Delphi and Basic COCOMO, Introduction to Halstead's Software Science, Staffing Level Estimation- Putnam's Model. Software Requirements Specification: SRS Documents, their Characteristics and Organization.

Unit 8: Software Engineering

- Software Design: Classification, Software Design Approaches, Function Oriented Software Design, Structured Analysis- Data flow Diagrams and Structured Design, Introduction to Object Oriented Design.
- Coding and Testing of Software: Unit Testing, Block Box Testing, White Box Testing, Debugging, Program Analysis Tools, System Testing. Software Reliability and Quality Assurance: Reliability Metric- Musa's Basic Model.
- Software Quality Assurance: ISO 9000 and SEI CMM and their Comparison. Software Maintenance: Maintenance Process Models and Reverse Engineering, Estimation of Maintenance Costs.

- Software is a program or set of programs containing instructions that provide desired functionality.

- Engineering is the process of designing and building something that serves a particular purpose and finds a cost-effective solution to problems.

- What is Software Engineering?

- Software Engineering is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software.

- Software engineering includes a variety of techniques, tools, and methodologies, including requirements analysis, design, testing, and maintenance.

- It is a rapidly evolving field, and new tools and technologies are constantly being developed to improve the software development process.

- By following the principles of software engineering and using the appropriate tools and methodologies, software developers can create high-quality, reliable, and maintainable software that meets the needs of its users.

- Software Engineering is mainly used for large projects based on software systems rather than single programs or applications.

- The main goal of Software Engineering is to develop software applications for improving quality, budget, and time efficiency.

- Software Engineering ensures that the software that has to be built should be consistent, correct, also on budget, on time, and within the required requirements.

INTRODUCTION:

- Software is more than just a program code.
- A program is an executable code, which servers some computational purpose.
- Software is the collection of computer programs, procedures rules and associated documentation and data.
- Software is an information transformer- producing, managing, modifying, displaying or transforming information that can simple as a single bit or a complex as a multimedia application.

Software Products:

- Software products may be developed for a particular customer or may be developed for a general market.
- **Software products may be:**
  - Generic
  - Bespoke/Specific
- **What are the attributes of good software?**
  - Maintainability.
  - Dependability
  - Efficiency
  - Usability

Key Principles of Software Engineering

- Modularity: Breaking the software into smaller, reusable components that can be developed and tested independently.
- Abstraction: Hiding the implementation details of a component and exposing only the necessary functionality to other parts of the software.
- Encapsulation: Wrapping up the data and functions of an object into a single unit and protecting the internal state of an object from external modifications.
- Reusability: Creating components that can be used in multiple projects, which can save time and resources.

Key Principles of Software Engineering

- Maintenance: Regularly updating and improving the software to fix bugs, add new features, and address security vulnerabilities.
- Testing: Verifying that the software meets its requirements and is free of bugs.
- Design Patterns: Solving recurring problems in software design by providing templates for solving them.
- Agile methodologies: Using iterative and incremental development processes that focus on customer satisfaction, rapid delivery, and flexibility.
- Continuous Integration & Deployment: Continuously integrate and deploy code changes into the production environment.

Dual Role of Software

- There is a dual role of software in the industry. The first one is as a product and the other one is as a vehicle for delivering the product. We will discuss both of them.

- 1. As a Product
- It delivers computing potential across networks of Hardware.
- It enables the Hardware to deliver the expected functionality.
- It acts as an information transformer because it produces, manages, acquires, modifies, displays, or transmits information.

Dual Role of Software

2. As a Vehicle for Delivering a Product

- It provides system functionality (e.g., payroll system).
- It controls other software (e.g., an operating system).
- It helps build other software (e.g., software tools).

What is the difference between software engineering and computer science?

Computer Science	Software Engineering
is concerned with	
➢ theory ➢ fundamentals	➢ the practicalities of developing ➢ delivering useful software
Algorithms, date structures, complexity theory, numerical methods	SE deals with practical problems in complex software products

Computer science theories are currently insufficient to act as a complete underpinning for software engineering, BUT it is a foundation for practical aspects of software engineering

Software Engineering Paradigms:

Software Characteristics:

- Software is developed or engineered, it is not manufactured in the classical sence.
- Software doesn't "wear out".
- Although the industry is moving towards component based assembly, most software continues to be custom to built.

Software Applications Types:

- System Software.
- Real-time Software.
- Business Software.
- Engineering and Scientific Software.
- Embedded Software.
- Personal Computer Software.
- Web-based Software.
- Artifical Intelligence Software.

Different types of software

System Software

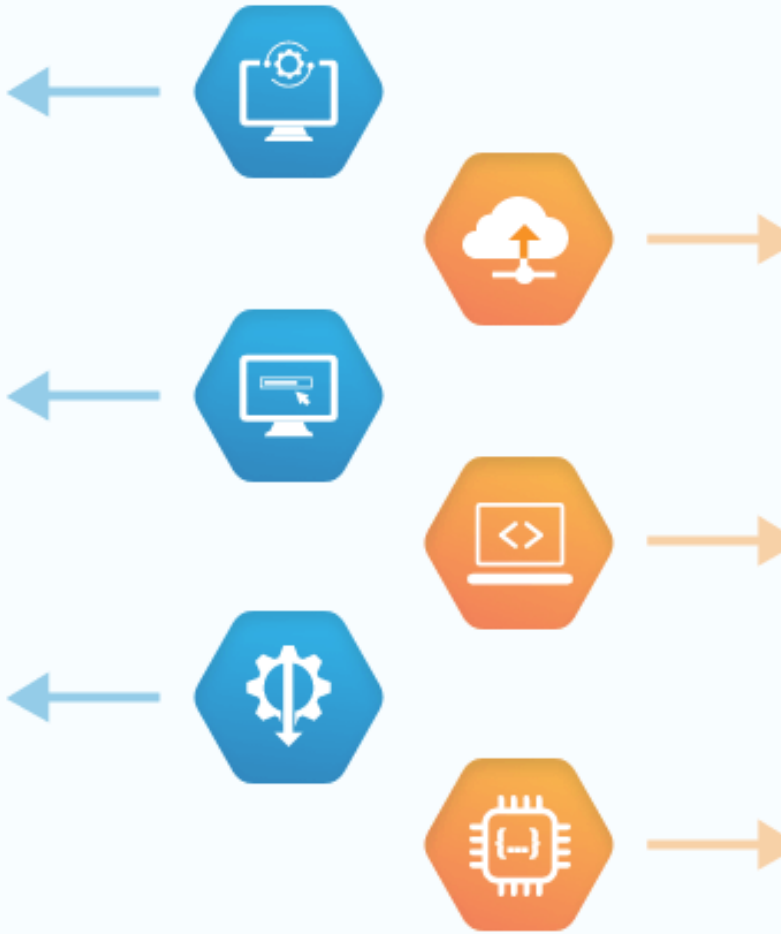
This software provides an environment to integrate hardware with software and thus create the

Application Software

This software provides users with functionalities to perform a task of interest.

Driver Software

This software drives all the devices that make a computer system.



Middleware

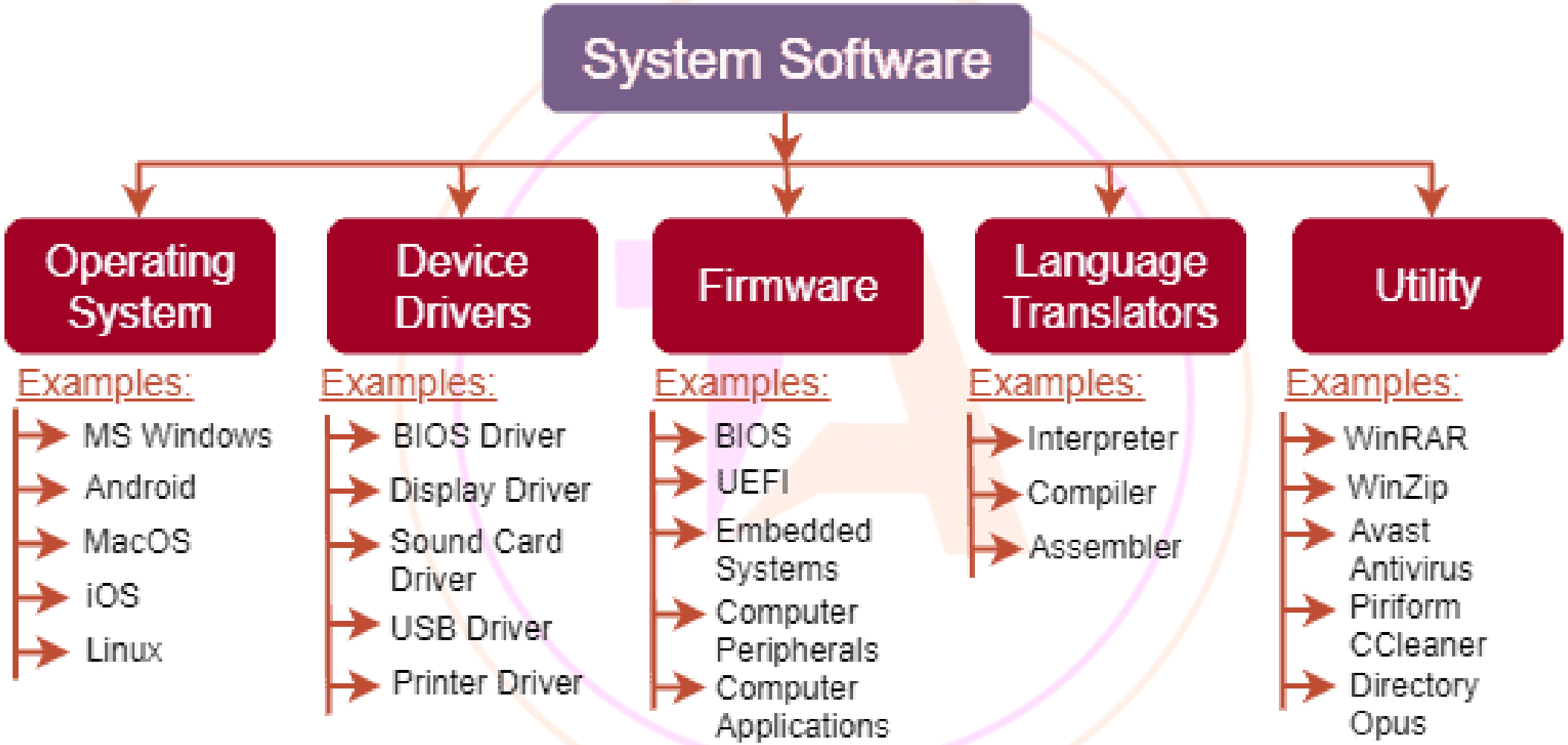
This software serves as the interface between distributed applications

Programming Software

This is commonly referred to as programming language and is used to develop programs, build software applications and analyze data.

Embedded Software

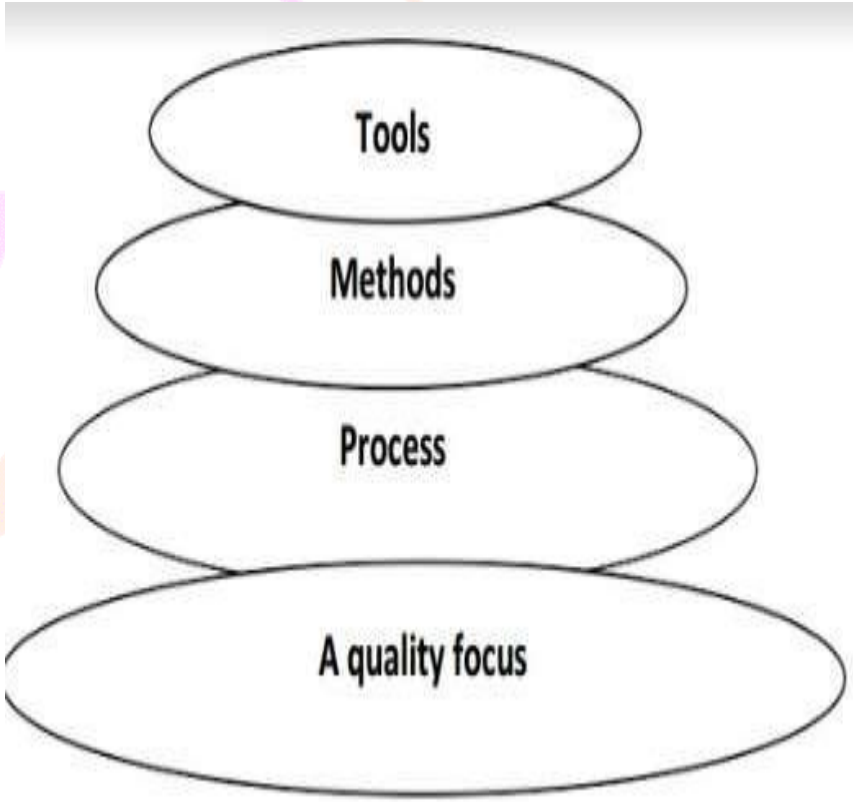
These software interfaces are embedded into hardware and dictate the hardware to act in a certain way.



Types of System Software

Software Engineering -A layered Technology:

- Application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software that is, the application of engineering software.



Software Engineering Layers

What are the five generic process framework activities?

- The following generic process framework is applicable to the majority of software projects.

- Communication.
- Planning.
- Modeling.
- Construction.
- Deployment.

Parameters Defining Software Project

The software should be produced at a reasonable cost, in a reasonable time, and should be of good quality. These three parameters often drive and define a software project.

Cost: As the main cost of producing software is the manpower employed, the cost of developing software is generally measured in terms of person-months of effort spent in development. The productivity in the software industry for writing fresh code mostly ranges from a few hundred to about 1000 + LOC per person per month..

Parameters Defining Software Project

Schedule: The schedule is another important factor in many projects. Business trends are dictating that the time to market a product should be reduced; that is, the cycle time from concept to delivery should be small. This means that software needs to be developed faster and within the specified time.

Quality: Quality is one of the main mantras, and business strategies are designed around it. Developing high-quality software is another fundamental goal of software engineering

What is open source software?

- Open source software is software with source code that anyone can inspect, modify, and enhance.
- "Source code" is the part of software that most computer users don't ever see; it's the code computer programmers can manipulate to change how a piece of software—a "program" or "application"—works.
- Programmers who have access to a computer program's source code can improve that program by adding features to it or fixing parts that don't always work correctly.

PROPRIETARY SOFTWARE

- Proprietary software is owned by an organization or an individual, as opposed to “public-domain software,” which is freely distributed.
- The explosion in the use of the Internet has expanded the reach of public-domain software since it is now much easier to transmit these programs.

PROPRIETARY SOFTWARE

- Only the original authors of proprietary software can legally copy, inspect, and alter that software.

- And in order to use proprietary software, computer users must agree (usually by signing a license displayed the first time they run this software) that they will not do anything with the software that the software's authors have not expressly permitted. Microsoft Office and Adobe Photoshop are examples of proprietary software.

WHAT IS FREEWARE?

- Freeware (not to be confused with free software) is a type of proprietary software that is released without charge to the public.

What's the difference between freeware and shareware?

- Shareware is a type of trial software that lets people use the program for a certain period or with certain features disabled. If users find the program useful, they can pay to unlock the full version.

Proprietary Software	Free Software
a. Proprietary software is a computer software where the source codes are not publicly available. Only the company which has created it can modify it.	a. Free source software is a computer software whose source code is available openly on the internet and programmers can modify it to add new features and capabilities without any cost.
b. These softwares are developed and tested by the individual or organization by which it is owned, not by the public.	b. These softwares are developed and tested through open collaboration.
c. Users need to have a valid and authenticated license to use this software.	c. Users do not need to have any authenticated license to use this software.
d. Users must have to pay to get the proprietary software.	d. Users can get free software for free of charge.
e. Examples are Windows, MacOS etc.	e. Examples are VLC Media Player, Android etc.

Freeware	Shareware
Freeware refers to software that anyone can download from the Internet and use for free.	Sharewares give users a chance to try the software before buying it.
All the features are free.	Most of the times, all features are not available, or have limited use. To use all the features of the software, user has to purchase the software.
Freeware programs can be distributed free of cost.	Shareware may or may not be distributed freely. In many cases, author's permission is needed, to distribute the shareware.
Adobe PDF, Google Talk, yahoo messenger, MSN messenger	Winzip, Internet Download Accelerator 3.1

Proprietary	Shareware	Freeware
Source code not available		
Need to purchase license	Free for only specific period	Free of cost
Features are available on purchase	Some features available only on purchase	All features available
Microsoft Office, iTunes	WinZIP, MySQL	Yahoo Messenger, Adobe PDF

License

- License is permission granted by the holder of a copyright to another to use an original work.

1. EULA: End User License Agreement.
2. GPL: General Public License.
3. LGPL: Lesser General Public License.
4. CC: Creative Common License.

LICENSE

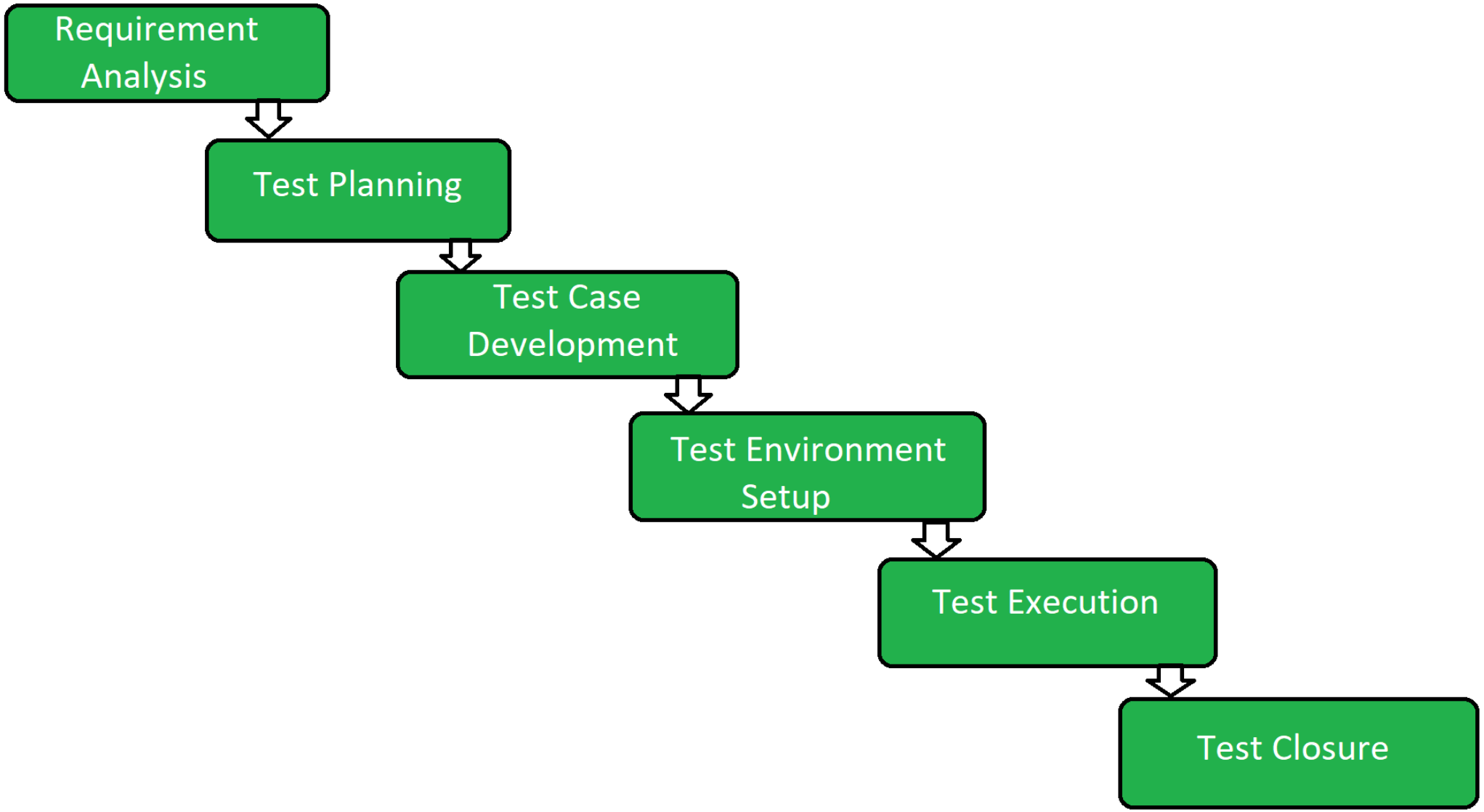
- **Workstation licenses** is where License is given for s use on single computer
- **Concurrent use licenses** is where you can install on multiple machines but no of uses should be less than licenses purchased.
- **Site licenses** is where you can install on any computer at a specified site
- **Perpetual licenses** come without an expiry date but Non-perpetual licenses “lease” the software for specific time.
- **License with Maintenance** offer “maintenance” along with usage.

Software Testing Life Cycle (STLC)

The Software Testing Life Cycle (STLC) is a systematic approach to testing a software application to ensure that it meets the requirements and is free of defects. It is a process that follows a series of steps or phases, and each phase has specific objectives and deliverables. The STLC is used to ensure that the software is of high quality, reliable, and meets the needs of the end-users.

Software Testing Life Cycle (STLC)

The main goal of the STLC is to identify and document any defects or issues in the software application as early as possible in the development process. This allows for issues to be addressed and resolved before the software is released to the public.



Types of Software Testing

- Testing is the process of executing a program to find errors. To make our software perform well it should be error-free. If testing is done successfully it will remove all the errors from the software. In this article, we will discuss first the principles of testing and then we will discuss, the different types of testing.

- Principles of Testing

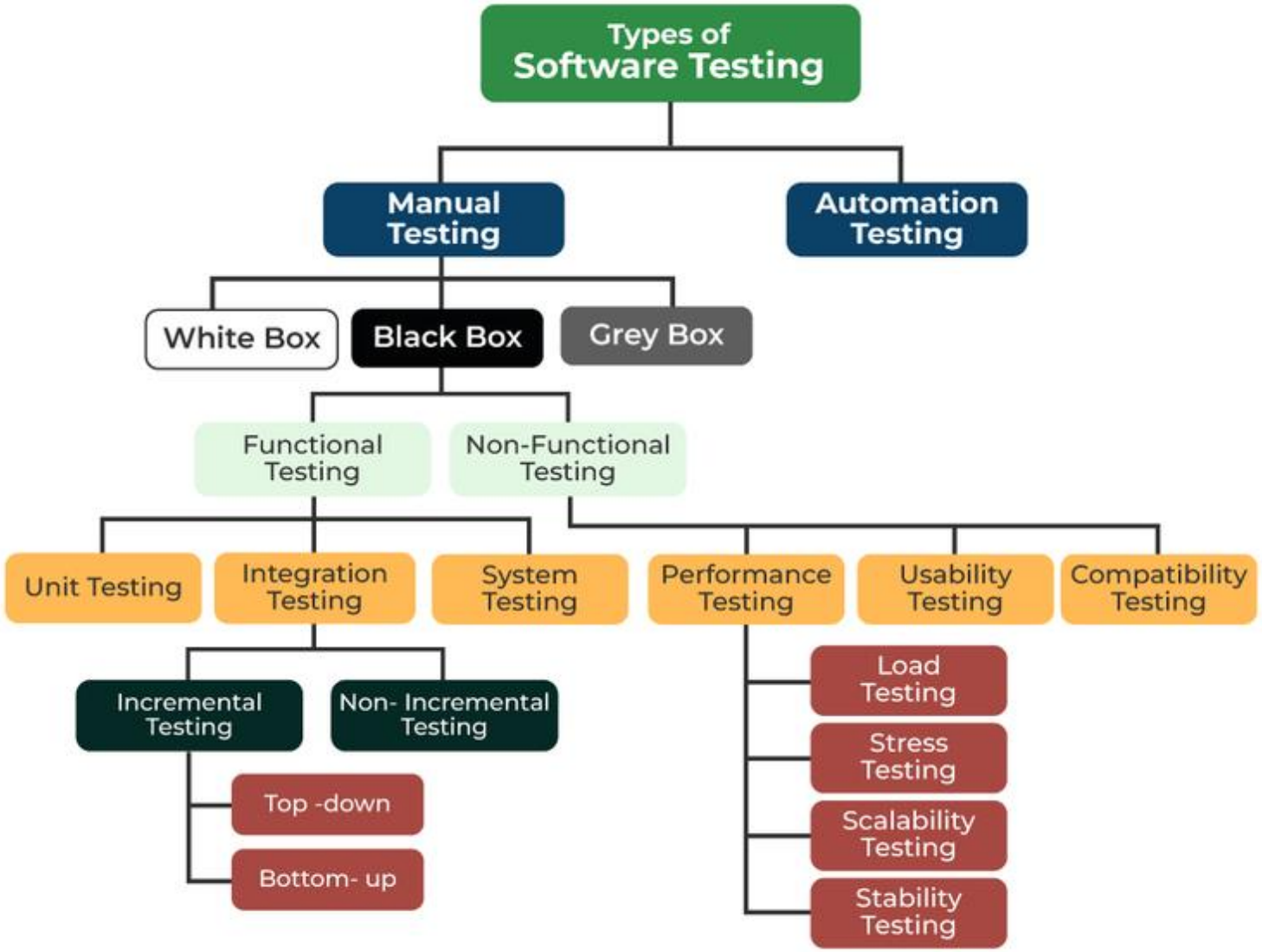
- All the tests should meet the customer’s requirements.
- To make our software testing should be performed by a third party.
- Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
- All the tests to be conducted should be planned before implementing it

- Principles of Testing

- It follows the Pareto rule(80/20 rule) which states that 80% of errors come from 20% of program components.
- Start testing with small parts and extend it to large parts.

- There are basically 10 types of Testing.

- Unit Testing
- Integration Testing
- System Testing
- Functional Testing
- Acceptance Testing
- Smoke Testing
- Regression Testing
- Performance Testing
- Security Testing
- User Acceptance Testing



- Unit Testing

- Unit testing is a method of testing individual units or components of a software application.
- It is typically done by developers and is used to ensure that the individual units of the software are working as intended. Unit tests are usually automated and are designed to test specific parts of the code, such as a particular function or method. Unit testing is done at the lowest level of the software development process, where individual units of code are tested in isolation.

- Integration Testing

- Integration testing is a method of testing how different units or components of a software application interact with each other. It is used to identify and resolve any issues that may arise when different units of the software are combined. Integration testing is typically done after unit testing and before functional testing and is used to verify that the different units of the software work together as intended.

- Regression Testing

- Regression testing is a method of testing that is used to ensure that changes made to the software do not introduce new bugs or cause existing functionality to break. It is typically done after changes have been made to the code, such as bug fixes or new features, and is used to verify that the software still works as intended.

- Smoke Testing
- Smoke Testing is done to make sure that the software under testing is ready or stable for further testing
- It is called a smoke test as the testing of an initial pass is done to check if it did not catch fire or smoke in the initial switch-on.

- Example:

- If the project has 2 modules so before going to the module make sure that module 1 works properly.

- Alpha Testing
- Alpha testing is a type of validation testing. It is a type of acceptance testing that is done before the product is released to customers. It is typically done by QA people.

- Example:

- When software testing is performed internally within the organisation.

- Beta Testing
- The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for a limited number of users for testing in a real-time environment.

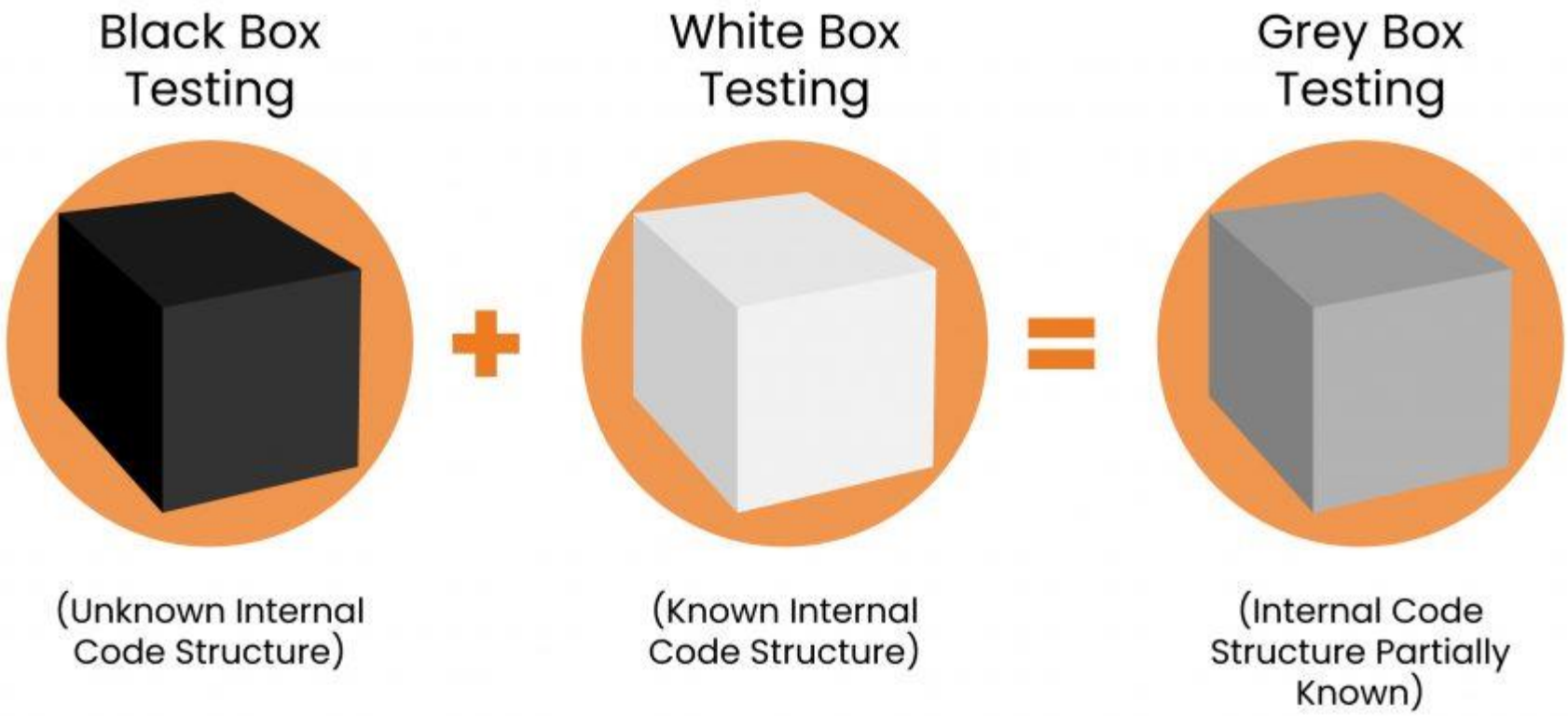
- Example:

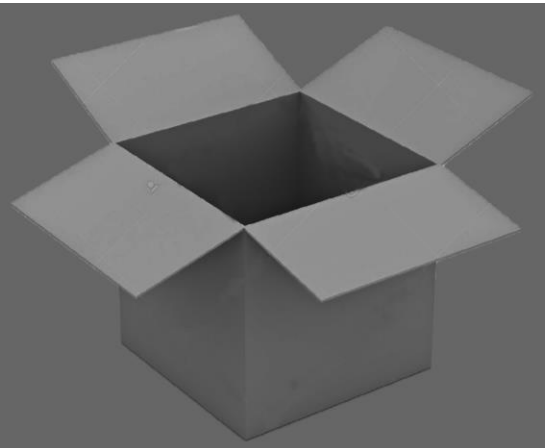
- When software testing is performed for the limited number of people.

- System Testing
- System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. The software is tested such that it works fine for the different operating systems. It is covered under the black box testing technique. In this, we just focus on the required input and output without focusing on internal work. In this, we have security testing, recovery testing, stress testing, and performance testing.

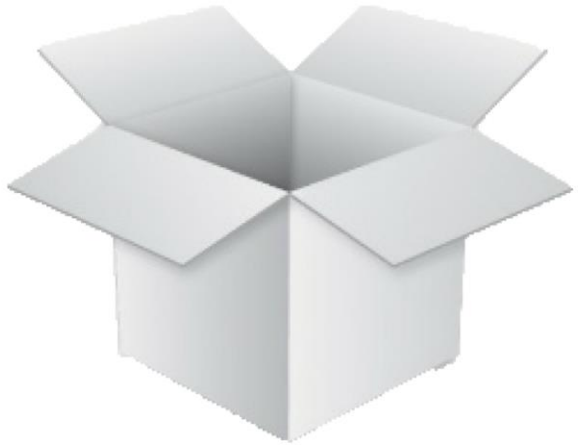
- System Testing
- Example:
- This includes functional as well as nonfunctional testing.

Types Of Testing Methods

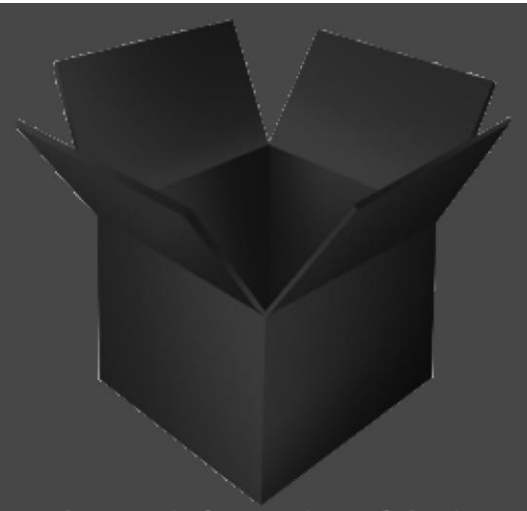




- Both black box testing and white box testing are utilized (Mainly for database testing)
- In gray box testing techniques inner programming is partially known.
- Somewhat knowledge of internal working of application is known.
- Gray box testing non intrusive also known as translucent testing.
- Performed by end clients and furthermore by testers and developers.
- Gray box testing done on the premise of abnormal state database outlines and information stream chart.
- Incompletely tedious and exhaustive.
- Not suited to calculation testing.



- Testers have full knowledge of inner programming rationale of the IT product under test.
- Execution of automated white box testing is the selective domain of the testing and improvement group.
- Outlining of test cases takes quite a more time.
- Viewed as ideal for calculation testing.
- White box testing in software engineering is the most tedious type of testing.
- Not utilized for testing product strength against viral attacks.
- WBT also called clear box testing, open box testing, auxiliary testing and logic-driven testing.



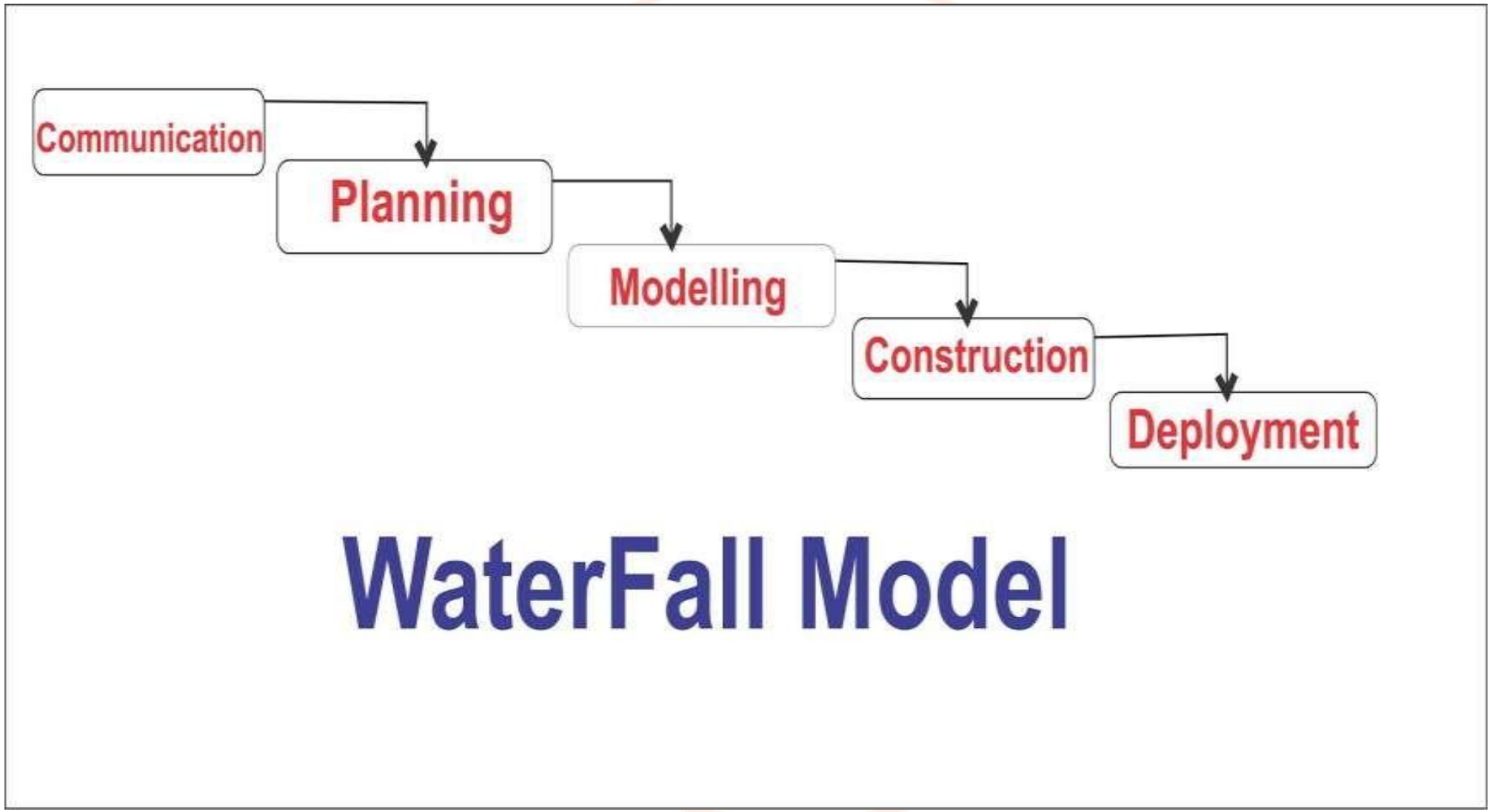
- Tester has no information of the inner workings of the IT product under test.
- Black box testing techniques can be performed by developers, user groups and testers.
- The sample space for test inputs is entirely enormous and the biggest among all.
- A fast outlining of test cases is conceivable.
- Automated black box testing is not appropriate for calculation testing.
- Black box testing methodologies is the slightest time depleting type of testing.
- Black box testing in software engineering also called as opaque testing and specifications based testing.

Process Models:

- Every software engineering organization should describe a unique set of framework activities for the software process it adopts.
    - Waterfall Life Cycle Model.
    - Iterative Waterfall Life Cycle Model.
    - Prototyping Model.
    - Incremental Model.
    - Sprial Model.
    - RAD Model.
    - Sprial Model.

Waterfall Life Cycle Model.

- It is called classic life cycle or Linear model.
- Requirements are well defined and stable.
- It suggests a systematic, sequential approach to software development.
- It begins with customer specification of requirements and progresses.
    - Planning.
    - Modeling.
    - Construction and
    - Deployment.



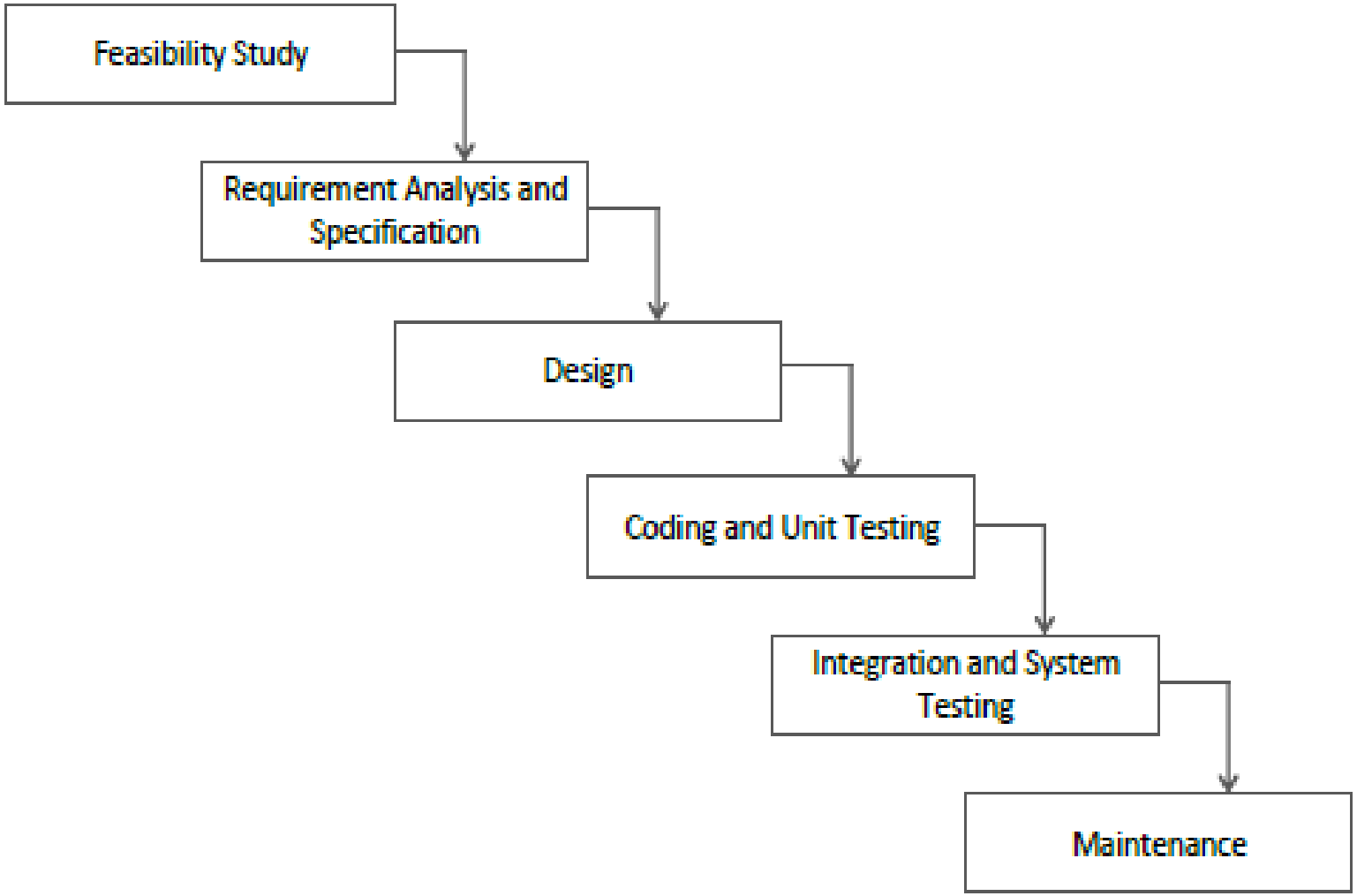
Advantages:

- Easy to understand.
- Each phase has well-defined input and output.
- Helps project manager in proper planning of the project.
- Provides templates into which methods of analysis, design, code and support can be placed.

Disadvantages:

- One-way street.
- It lacks overlapping and interactions among phases.

Phases of the Classical Waterfall Model:



Feasibility Study:

- It involves analysis of the problem and collection of allrelevant information relating to the product.
- The collected data are analysed.
  - Requriments of the Customer.
  - Formulations of the different strategies for solving the problem.
  - Evaluation of different solution strategies.

Requriments Analysis and Specification:

- It is understand the exact requriments of the customer and to document them properly.
  - Requirements gathering and analysis.
  - Requirements specification.

Design:

- The design phase is to transform the requirements specified in the document into a structure that is suitable for implementation in some programming language.
  - Traditional Design Approach.
  - Object-Oriented Design Approach.

Coding and Unit Testing:

- The purpose of the coding and unit testing phase of software development is to translate the software design into source code.

Integration and System Testing:

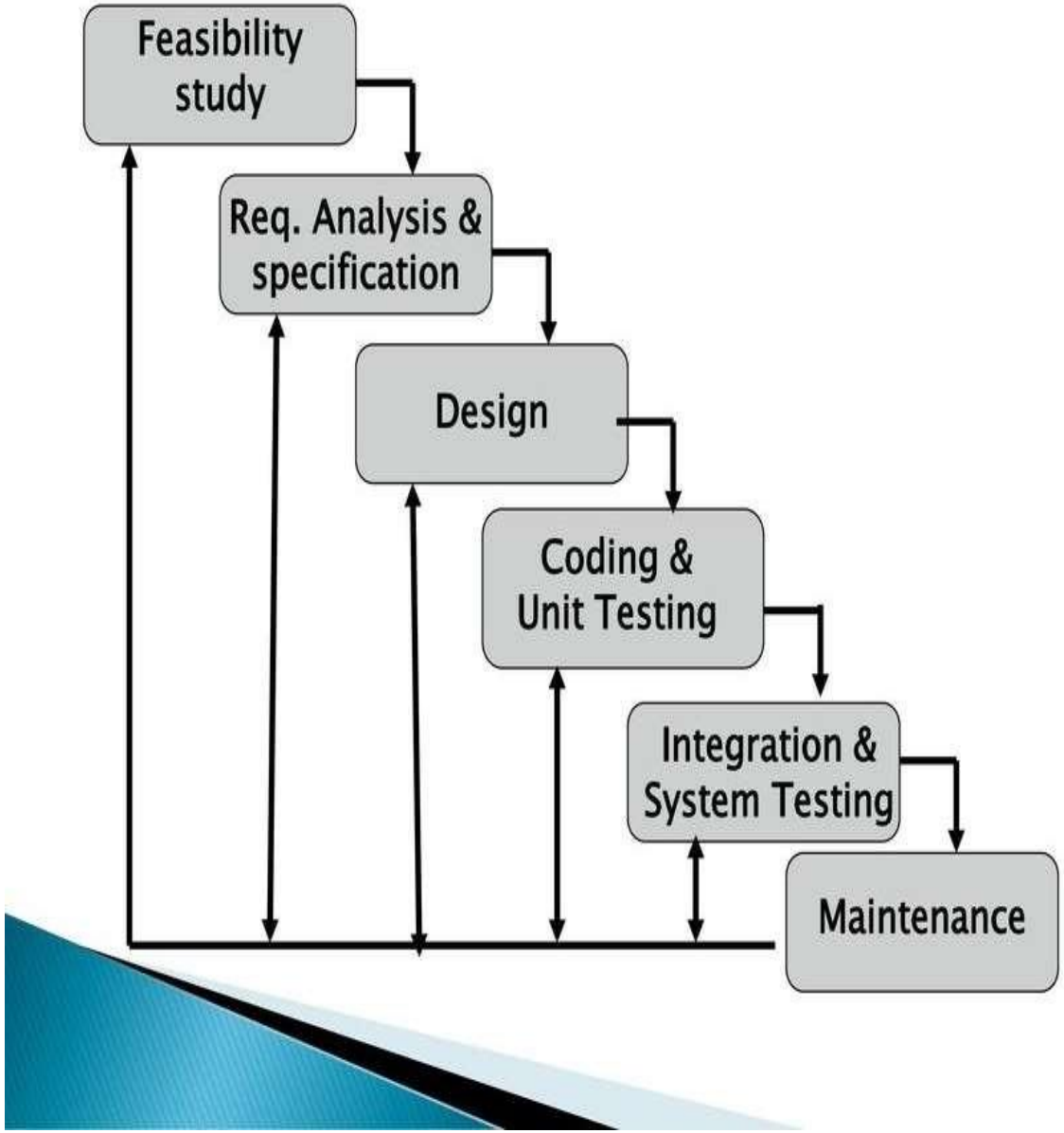
- ‘Integration of different modules is coded and unit tested.
  - $\alpha - Testing$
  - $\beta - Testing$
  - Accsptance Testing.

Maintenance:

- Maintenance of a typical software products requires much more than the effort necessary to develope the product itself.

Iterative Waterfall life cycle model:

- The main changes is done by providing feedback paths from every phase to its preceding phase.



Prototype Model:

- Prototyping Model is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved.

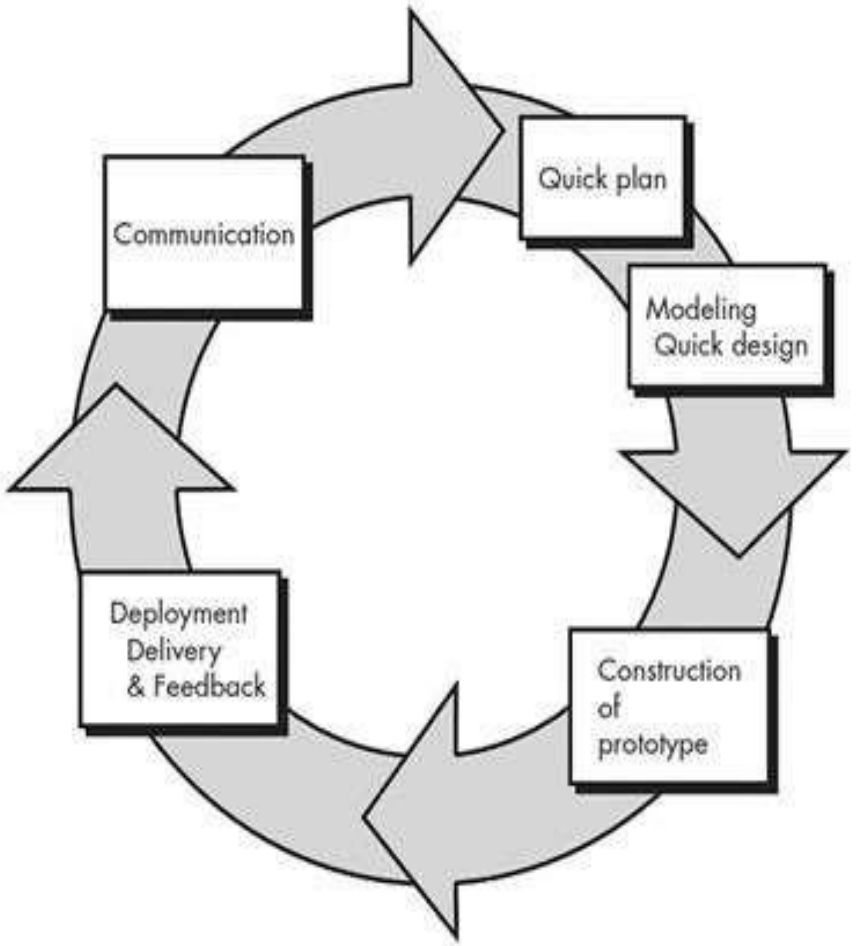


Figure: Prototype Model

Advantages:

- Clarity.
- Risk Identification.
- Good Environment.
- Take less time to complete.

Disadvantages:

- High cost.
- Slow process.
- Too many changes.

RAD Model:

- Rapid Application Development(RAD) is an incremental software model that a short development cycle.
- The RAD model is a “high-speed” of the waterfall model.
- The RAD process enables a development team to create a fully functional system within a very short time period.

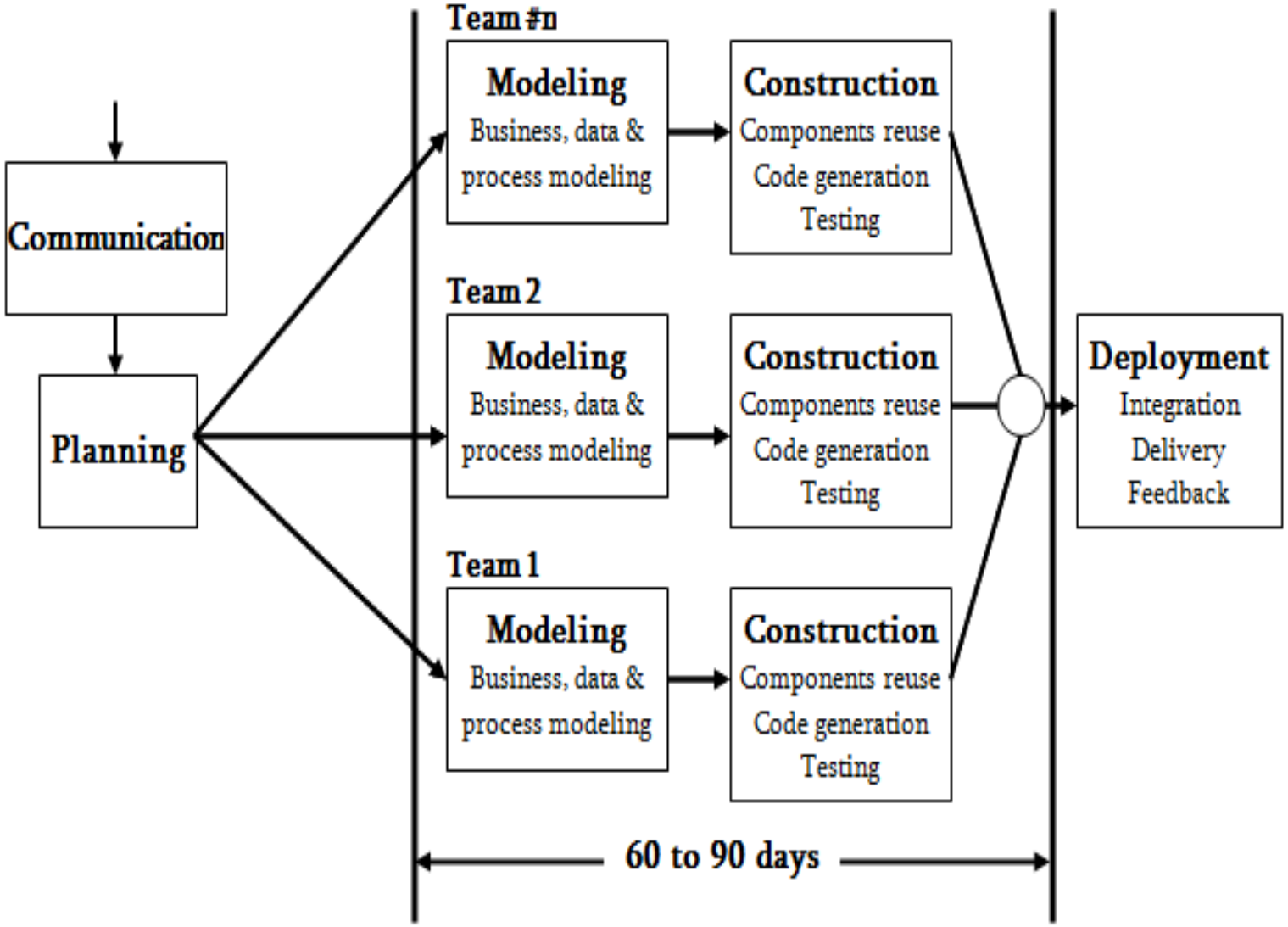


Figure : Flowchart of RAD model

- Graphical user development environment.
- Reusable Components.
- Code generator.
- Programming Language.

Advantages:

- Fast products.
- Efficient Documentation.
- Interaction with user.

Disadvantages:

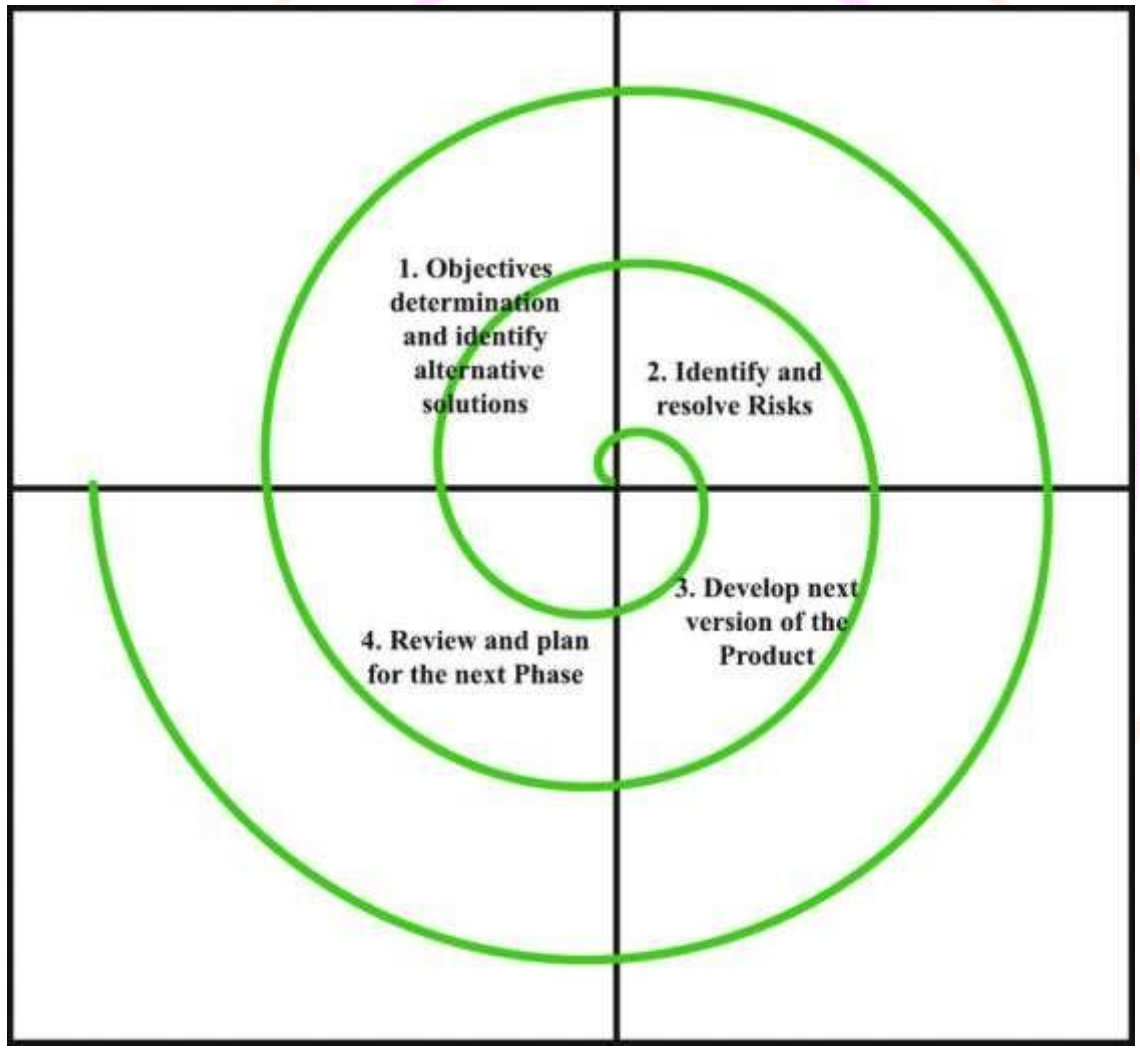
- User may not like fast activities.
- Not suitable for technical risks.

Contents of RAD Pakages:

Sprial Model :

- This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis.
- The spiral model has four phases: Planning, Design, Construct and Evaluation.

Quadrants in sprial model :



Advantages :

- Risk Identification at early stage.
- Suitable for high rk projects.
- Flexibility for adding functionaility.

Disadvantages:

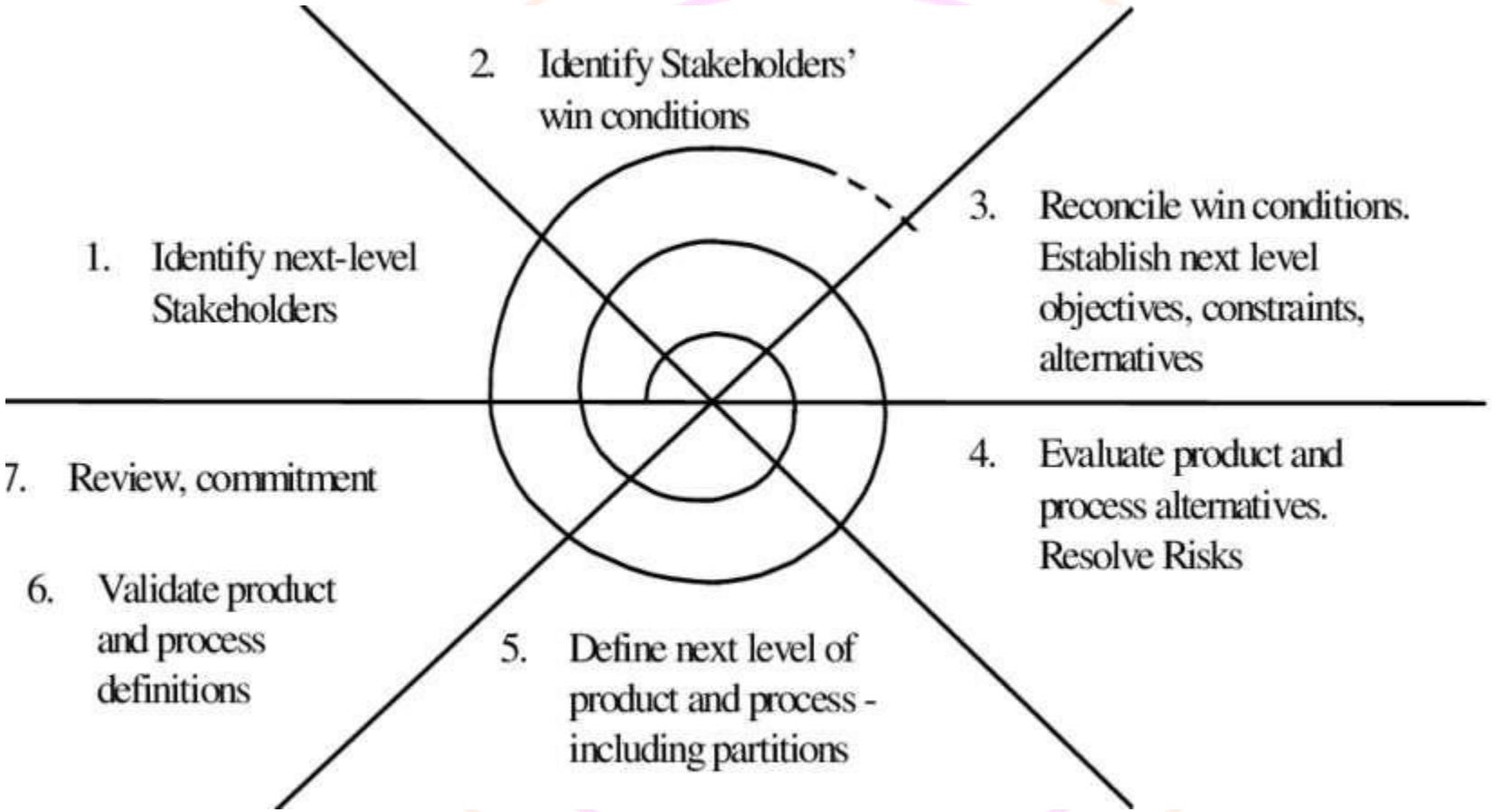
- Costly.
- Risk dependent.
- Not suitable for smaller projects.
- Difficult to meeting budget.

Win-Win Sprial Model:

- The customer wins by getting the system satisfying most of thier requirements and developers winsby working on achievable budgets and deadlines.
- **Advantages:**
- Lieghtweight methods suit small-medium size project.
- Produces good team.
- Test based approach to requirements and quality assurance

Diadvantages:

- Programming pairs is costly.
- Difficult to scale up to large projects where documentation.



Fourth Generation Techniques:

- ## Introduction:

- The tools in automatically generate ource code based on the developers specification.

Software development environment that supports the 4GT paradigm includes some or all of the following tools:

- 1) Non-procedural languages for database query
- 2) Report generation
- 3) Data manipulation
- 4) Screen interaction and definition
- 5) Code generation and High-level graphics capability
- 6) Spreadsheet capability
- 7) Automated generation of HTML and similar languages used for Web-site creation using advanced software tools.

Advantages:

- Reduction in software development.
- Improved productivity of software engineers.
- 4GT helped by CASE tools and code generators.

Disadvantages:

- Some 4GT are not at all easier than programming languages.
- Generated source code are sometimes inefficient.
- Time is reduced for only small and medium projects.

Planning:

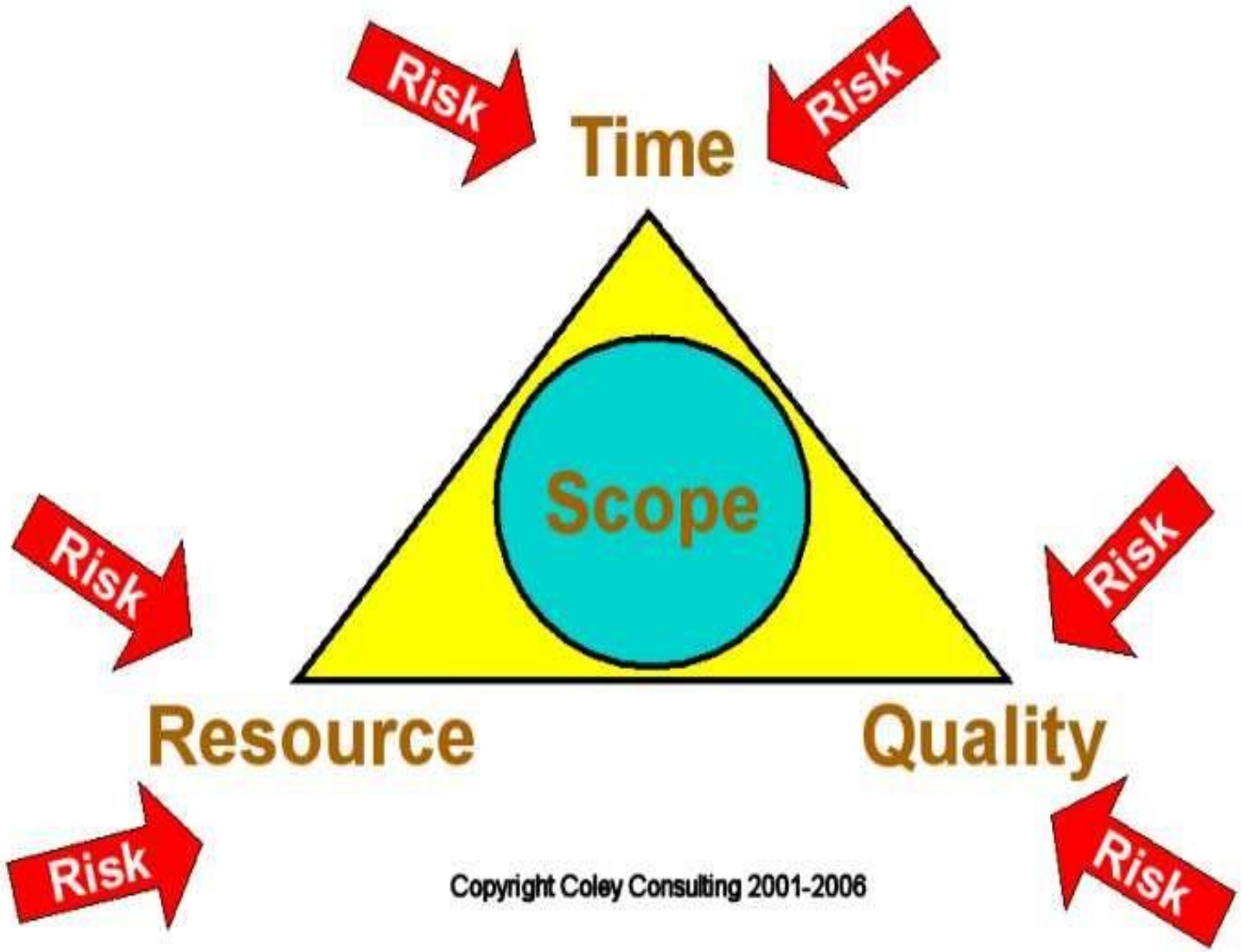
- Software planning process include steps to estimate the size of the software work products and the resources needed produces a schedule identify and access software risks.
- **During planning a project is split into several activities :**
- How much efforts is required to complete each activities?
- How much calender time is needed?
- How much will the completed activity cost?

Planning Objectives:

- Understand the scope of the problem.
- Make use of past historical data.
- Estimate effort or function or size.
- Define a project schedule.

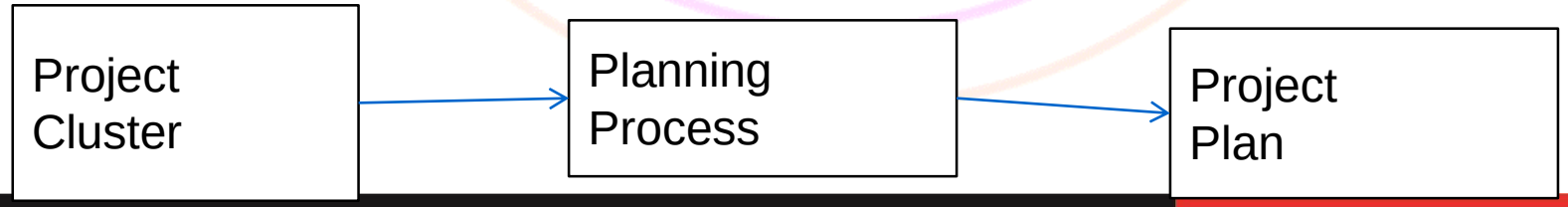
Characteristics of software project planning:

- Scope.
- Resource.
- Time.
- Quality.
- Risk.



Project Plan:

- The biggest single problem that afflicts software developing is that of underestimating resources required for a project.
- According to the project management body of knowledge.
- According to PRINCE(PRojects IN Controlled Environments).



Types of Project Plan:

- Software development plan.
- Quality Assurance Plan.
- Validation Plan.
- Configuration Management Plan.
- Maintenance Plan.
- Staff development plan.

Structure of a software project management plan:

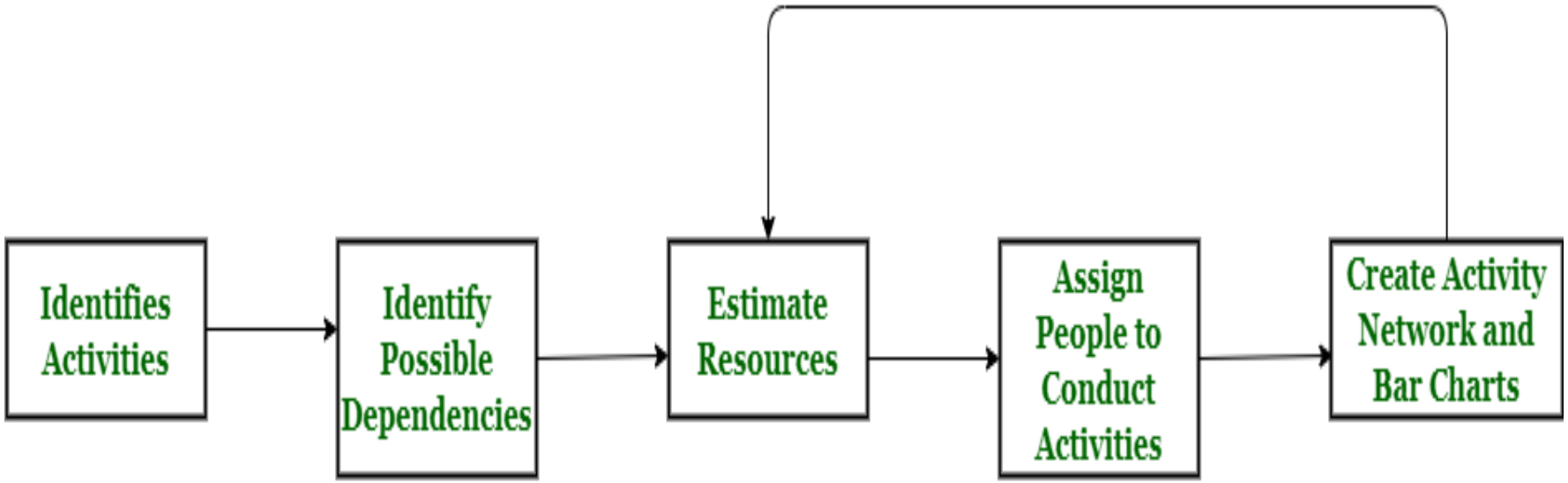
Project summary.
Project planning.

Major issues in planning a software project:

- Software requiremments are frequently incorrect and incomplete.
- Planning schedule and cost are not updated and are based on marketing needs, not system requirements.
- Cost and schedule are not re-estimated when requirements or development environment change.

Software Project Scheduling:

- **Introduction:**
- Software project scheduling is an distributes estimated effort across the planned project.
- Project schedulinginvolves seperating the total work involved in a project in seperate activities and judging the time required to complete the activities.



Project Scheduling Process

Basic principles of software project scheduling:

- Compartmentalization.
- Interdependency.
- Time Allocation.
- Effort Validation.
- Defined Responsibilities.
- Defined outcomes.
- Defined Milestones.

Relationship between people and effort:

- The PNR curve provides an indication of the relationship between effort applied and delivery time for a software project.

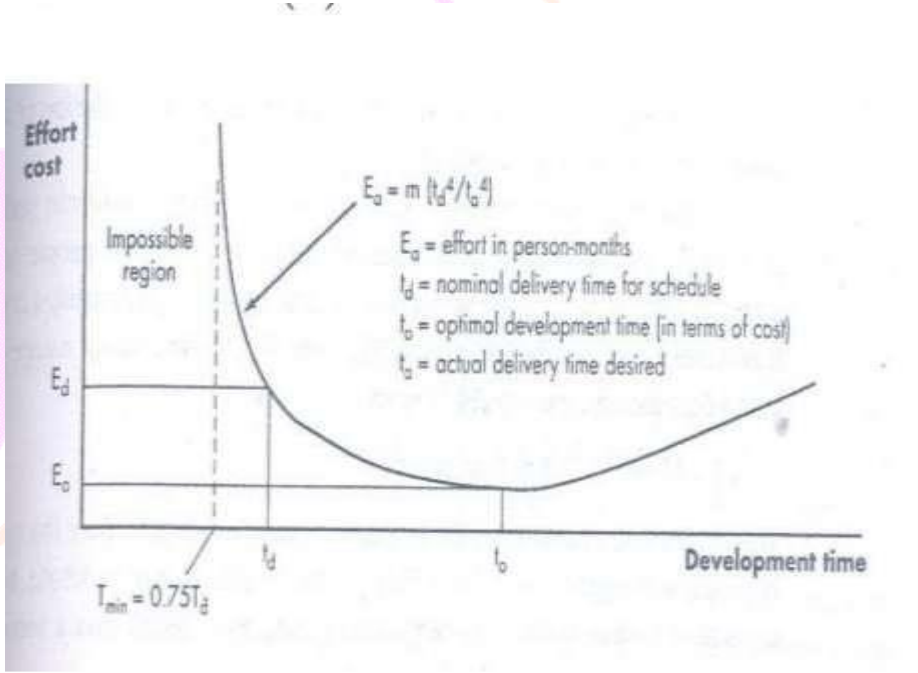


Figure 24.1: Putnam-Norden-Rayleigh (PNR) Curve

Effort Distribution:

- A recommended distribution of effort the software process is often referred to as the 40- 20-40 rule.

Defining a task set for the software project:

- A task set is a collection of software engineering work tasks, milestones, and work products that must be acomplished to complete a particular project.
  - Concept Development projects.
  - New application development projects.
  - Application enhancement projects.
  - Application maintenance projects.
  - Re-Engineering projects.

Example of a task set:

- **Concept Scoping:** It determines the overall scope of the project.
- **Preliminary concept planning:** It establishes the organization ability to undertake the work implied by the project scope.
- **Technology Risk Assessment:** It evaluates the risk associated with the technology to be implemented as part of project scope.
- **Concept Implementation:** It implement the concept representation in a manner that can be reviewed by a customer and is used marketing purposes.

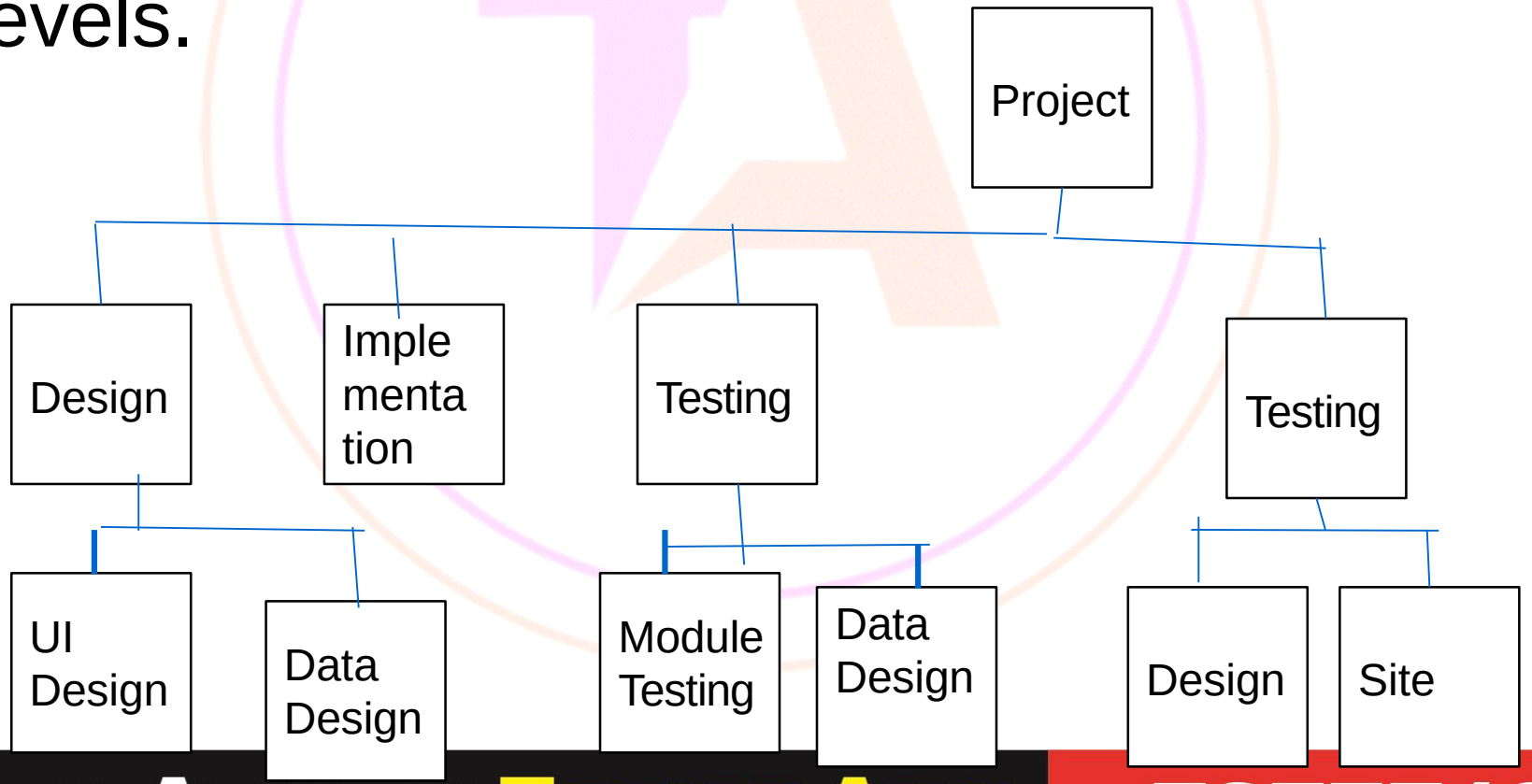
- **Customer Reaction:** Customer reaction to the concept feedback on a new technology concept and target specific customer applications.

Scheduling Techniques:

- Scheduling of a software project does not differ greatly from scheduling of any multitask engineering effort.
    - Work Breakdown Structure(WBS).
    - Activity Charts.
    - Project Evaluation Review Techniques(PERT).
    - Grant Charts.
    - Critical Path Method(CPM).

Work Breakdown Structure(WBS):

- A Work Breakdown Structure is a hierarcical decomposition or breakdown of a project or major activity into successive levels.

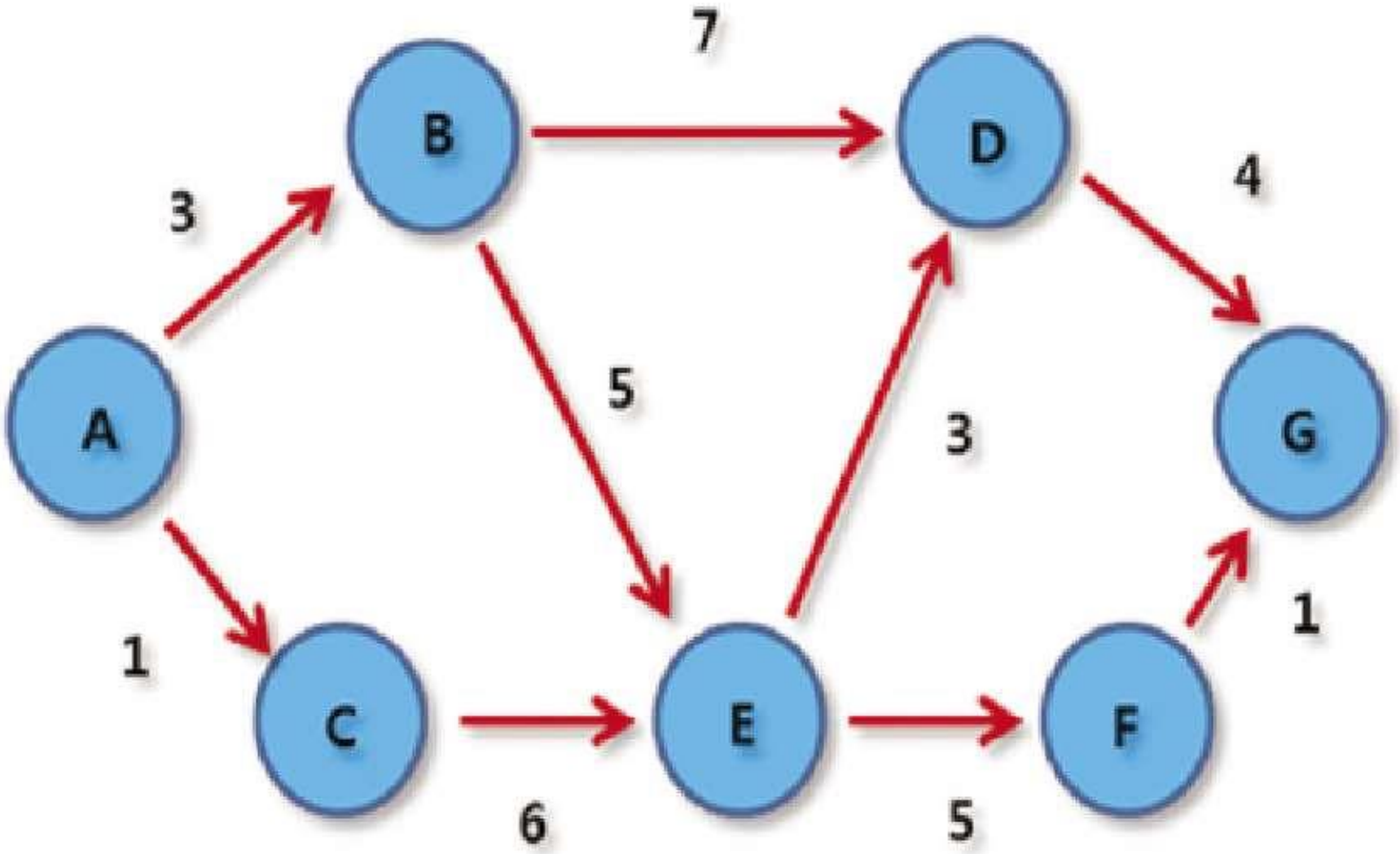


Features of WBS:

- Structure.
- Description.
- Coding.
- Depth.
- Level of Detail.

Activity Charts : Representation of WBS:

- Network of boxes and arrows.
- Shows different tasks making up a project.
- Represents the ordering among the tasks.

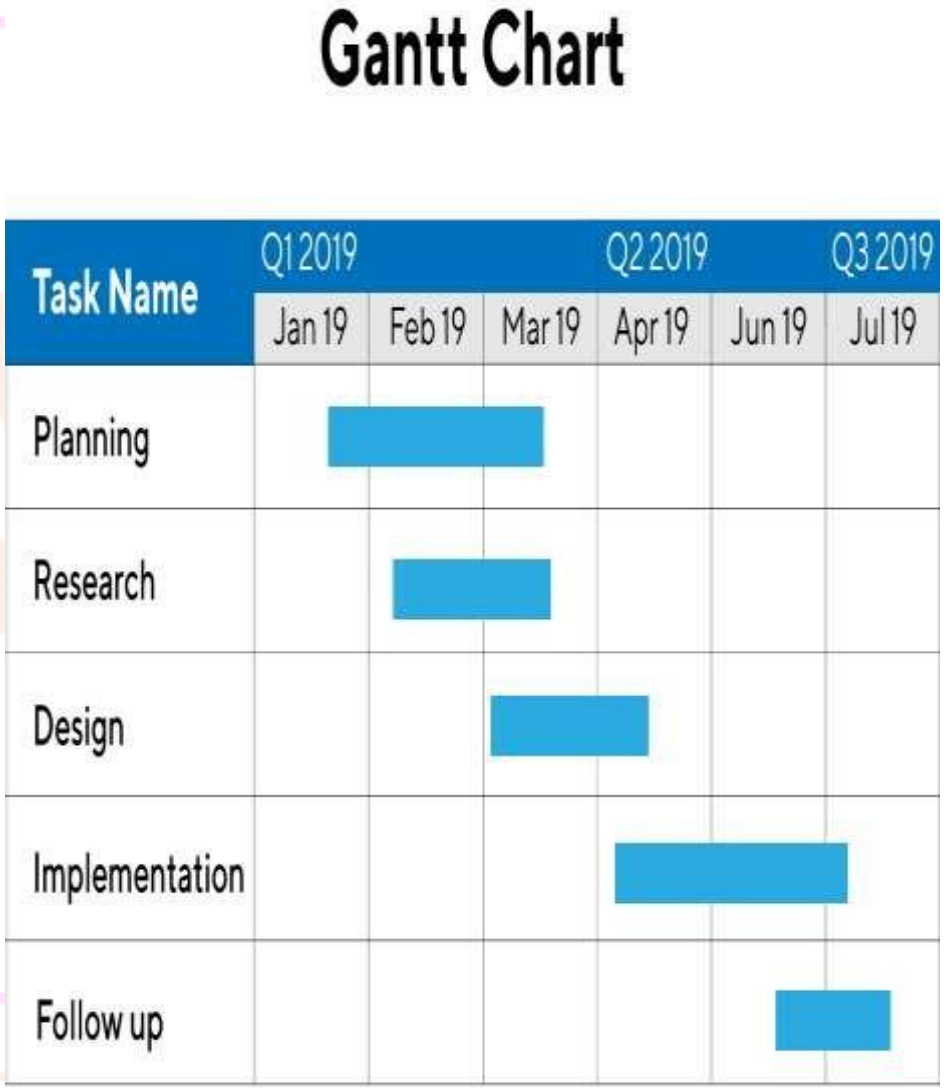


Project Evaluation Review Technique(PERT):

- PERT chart is a project management tool used to schedule, organize and coordinate tasks within the project.
- PERT methonology developed by the U.S. Navy in the 1950's to manage the polaris submarine program.
- PERT is an event-oriented technique
- PERT is a probabilistic model
- Grantt chart, PERT can be both a cost and a time management system.

Grantt Chart:

A grantt chart is a horizontal bar chart developedas a production control tool named after Henry L, Grantt an american engineer and social scientist ferquently used in project management.



Critical Path Method(CPM):

- CPM acts as the basic both for perparation of a schedule and of resource planning.
- The critical path determine the total duration of the project.
- CPM is an activity-oriented technique
- CPM is a deterministic model

Risk Analysis & Management:

- Risk analysis and management are a series of steps that help a software team to understand and manage uncertainty during the development process..
- A risk is a potential problem.
- Managers, Software enginners and customers participate in risk analysis & management.

Software Risk:

- According to webster risk is the possibility of suffering loss.
- Risk in a project or program is a measure of the ability to achieve objectives within cost, schedule and constraints.
- Types of software risk:

  Classification I

  Classification II

Classification I

- Project Risks
- Technical Risks.
- Business Risks.

Classification II:

- Known Risks.
- Predictable Risks.
- Unperdictable Risks.

Classification I:

- **Project Risks:** The projct schedule will slip and that costs will incnrease.Project risks identify schedule, resource, customer and requirements problem.
- **Technical Risks:**The product quality and the timeliness of the schedule if a technical risks is real then implementation may become difficult or implssible.
- If identify potential design, implementation, interface, verfication and maintenance problems.

Business Risks:

- Market Risk.
- Strategic Risk.
- Management Risk.
- Budget Risk.

Classification II:

- **Known Risks:** That can be uncovered after careful evaluation of the project plan.

•**Predictable Risks:**Predictable Risks are extrapolated from past project experience.

•**Unperdictable Risks:** Unperdictable Risks are the joker in the desk, they can extremely difficult to identify in advance.

Risk Principles:

- Global Perspective.
- Forward looking view.
- Open communication.
- Integrated management.
- Continuous process.
- Shared product vision.
- Team work.

Risk Strategies:

- Reactive Risk Strategies.
- Proactive Risk Strategies.

Risks in software development projects:

- Poorly defined requirements.
- Client requirements changes.
- Poor techniques for cost estimation.
- Dependence on skills of individual developers.

Risk Management Process:

- The risk management activities includes identify, analysis, plan, track and control risks.
  - Risk Assessment.
  - Risk Control.

Risk Assessment:

- Risk assessment is the determination os qualitative value of risk related to a concrete situation and recognized the risk.

- Risk identification.
- Risk analysis.
- Risk Prioritization.

Risk Identification:

- Risk identification is a systematic attempt to specify to the project plan.

Risk Item Checklist:

- Product Size.
- Bussiness Impact.
- Customer Characteristics.
- Process definition.

Activities in risk identification phase:

- Identify Risks.
- Define Risk Attributes.
- Document.
- Communicate.

Risk Analysis:

- The risk identify all items are analysed using different criterias.
- **Activities in risk analysis:**
- Group similar risks.
- Determine risk derivers.
- Determine sources of risks.
- Estimate risk.

Risk Prioritization:

- The project focus on its most server risks by assessing the risk.
- Let (r) is the likehood of a risk coming trace.
- (s) is the consequence of the problem associated with that risk.
- P=r*s.

Risk Control:

- Risk control is the process of managing risks should outcomes.

- Risk Management Planning.
- Risk Resolution.
- Risk Monitoring.

Risk Management Planning:

- It is a plan for delaing with each significant risk.

Strategies in risk management planning:

- Risk Avoidance.
- Risk monitoring.
- Risk management and contingency planning.

Risk Resoution:

- Risk resolution is the execution of the plan for dealing with each risk.
- The risk has triggered the project manager need to execute the action plan.

Outputs of risk resolution phase:

- Risk status.
- Acceptable risks.
- Reduced rework.
- Corrective actions.
- Problem prevention.

Risk Monitoring:

- Risk monitoring is the continually reassessing of risks as the project proceeds and conditions change.
- RMMM Plan:
- Risk Mitigation , Monitoring and management in the software project plan or the risk management steps are organized.

Requirement Engineering Process:

- **Introduction:**
- Requirement engineering is the sub-discipline of software engineering that is concerned with determine the goal, functions and constraints of software system.

Requirements:

- Requirements management is a systematic approach to eliciting organizing and documenting the requirements of the systems.

Types of Requirements:

- System Requirements.
- User Requirements.

System Requirements:

- System requirements set out the systems functions, services and operational constraints in detail.
- It may be part of the constract betwwen the system buyer and the software developer.

Types of system requirements:

- Functional requirements.
- Non-functional Requirements.
- Domain Requirements.

Functional Requirements:

- The customer should provide statement of service. It should be clear how the system should react to particular inputs and how a particular system.

Problem of Functional Requirements:

- User Intention.
- Developer Interpretation.
- Requirements completness and consistency:

Non-Functional Requirements:

- The system properties and constraints various properties of a system can be: realiability, response tiime, storage requirements.

Types of Non-Functional Requirements:

- Product Requirements.
- Organizational Requirements.
- External Requirements.

Domain Requirements:

- Requirements can be application domain of the system, reflecting, characteristics of the domain.

Problem of Domain Requements:

- Understandability.
- Implicitness.

User Requirements:

- User requirements are defined using natural language lables and diagrams because these are the representation that can be undestood by all users.

- Client Managers.
- System End Users.
- Client Engineers.
- Contract Managers.

Problem of User Requirements:

- Lack of Clarity.
- Requirements Confusion.
- Requirements Mixture.

Software Requirement Specification:

- Software Requirements document is the specification of the system.
- It is not a design document.
- Requirements document is called SRS.

Users of SRS:

- Users, Customer and marketing Personnel.
- Software Developers.
- Test Engineers.
- Project Managers.
- Maintenance Engineers.