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

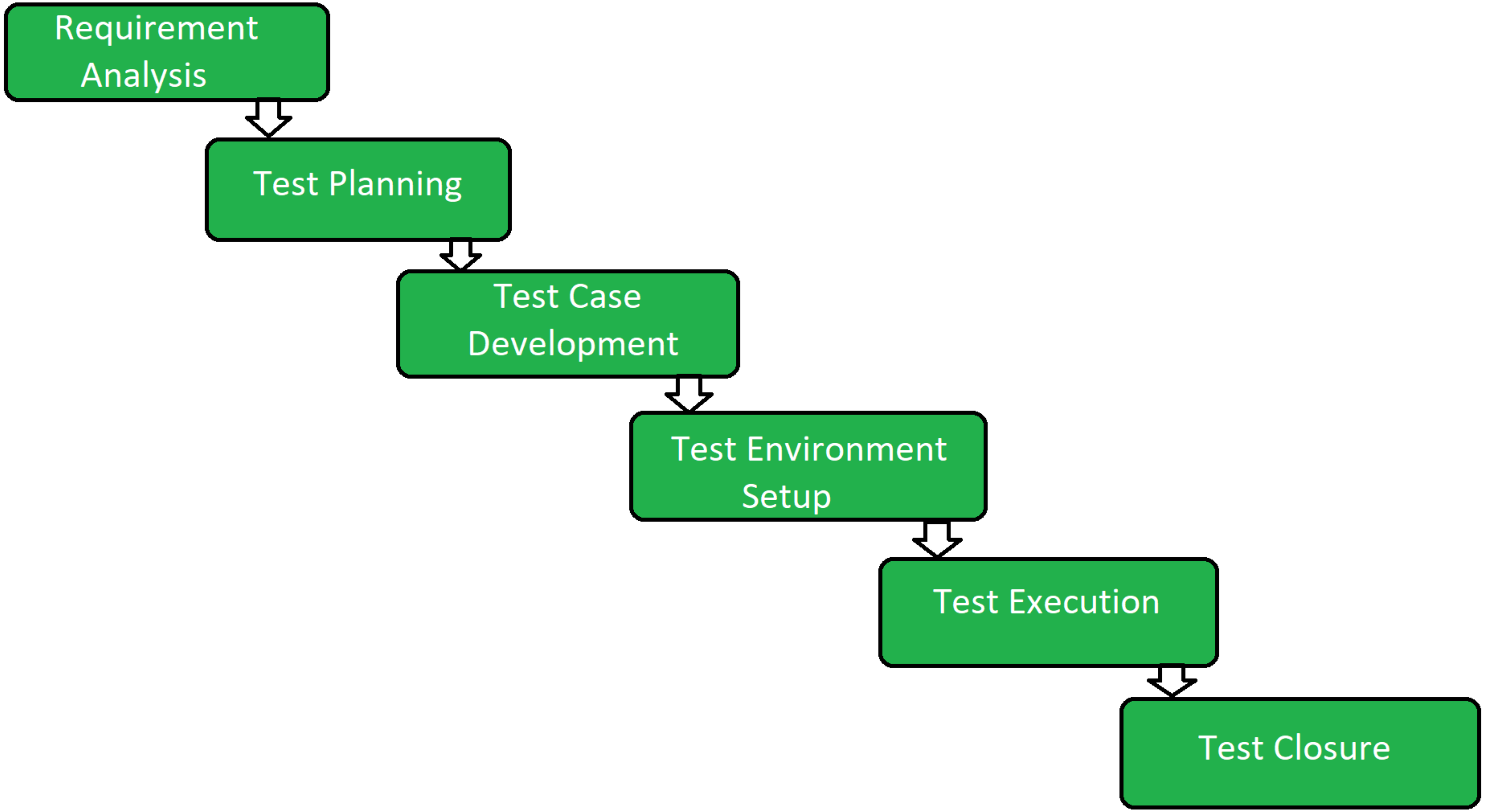
- Software Design: Classification, Software Design Approaches, Function Oriented Software Design, Structured Analysis- Data flow Diagrams and Structured Design, Introduction to Object Oriented Design.
- Coding and Testing of Software: Unit Testing, Block Box Testing, White Box Testing, Debugging, Program 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 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.


Requirement Analysis

Test Planning

Test Case Development

Test Environment Setup

Test Execution

Test Closure

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

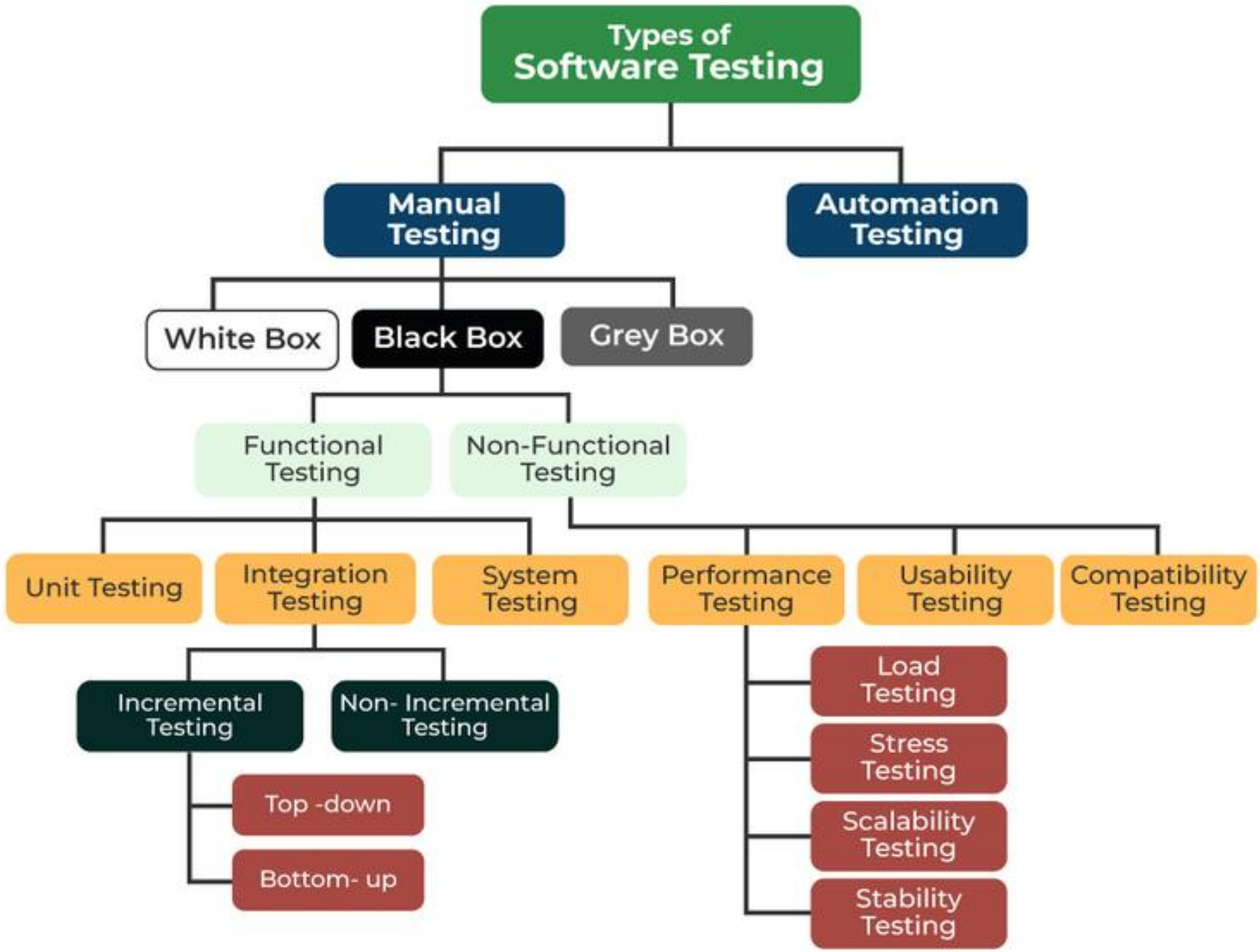
- All the tests should meet the customer’s requirements.
- To make our software testing should be performed by a third party.
- Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
- All the tests to be conducted should be planned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



Types of Software Testing

Manual Testing

Automation Testing

White Box

Black Box

Grey Box

Functional Testing

Non-Functional Testing

Unit Testing

Integration Testing

System Testing

Performance Testing

Usability Testing

Compatibility Testing

Incremental Testing

Non- Incremental Testing

Top -down

Bottom- up

Load Testing

Stress Testing

Scalability Testing

Stability 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

Black Box Testing

(Unknown Internal Code Structure)

+

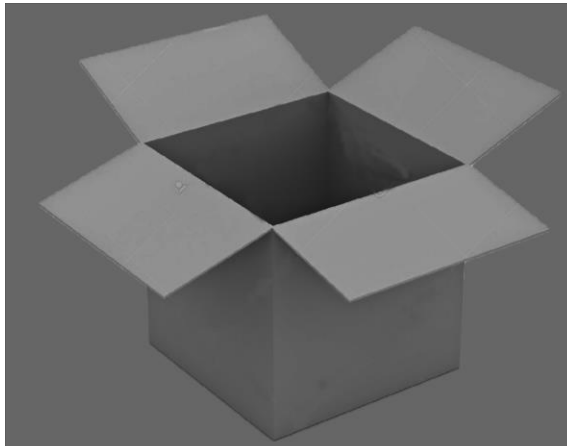
White Box Testing

(Known Internal Code Structure)

=

Grey Box Testing

(Internal Code Structure Partially Known)



- Both black box testing and white box testing are utilized (Mainly for database testing)

- In gray box testing techniques inner programming is partially known.

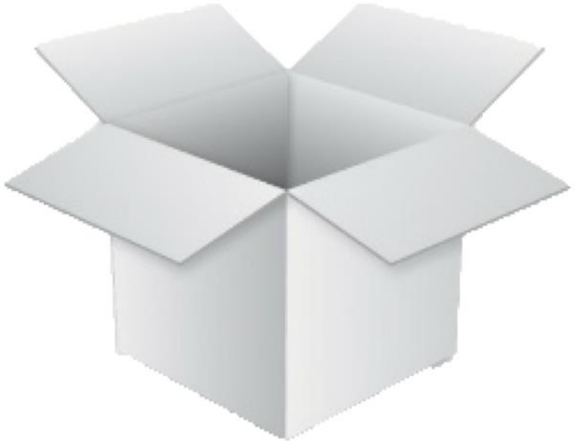
- Somewhat knowledge of internal working of application is known.

- 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.

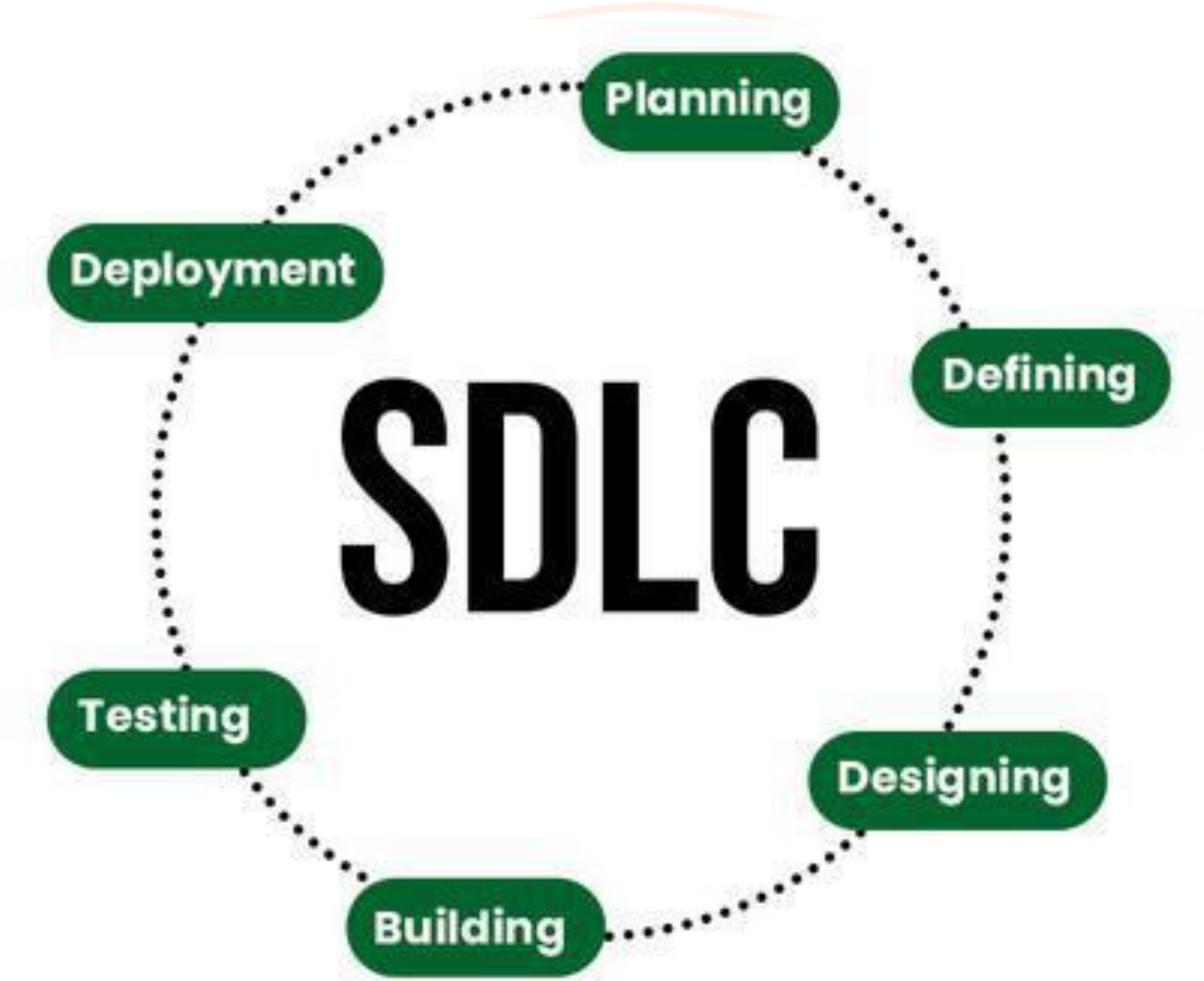
- Automated black box testing is not appropriate for calculation testing.

- 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.

- Software Development Life Cycle (SDLC)

- Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software. SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.



SDLC

Planning

Defining

Designing

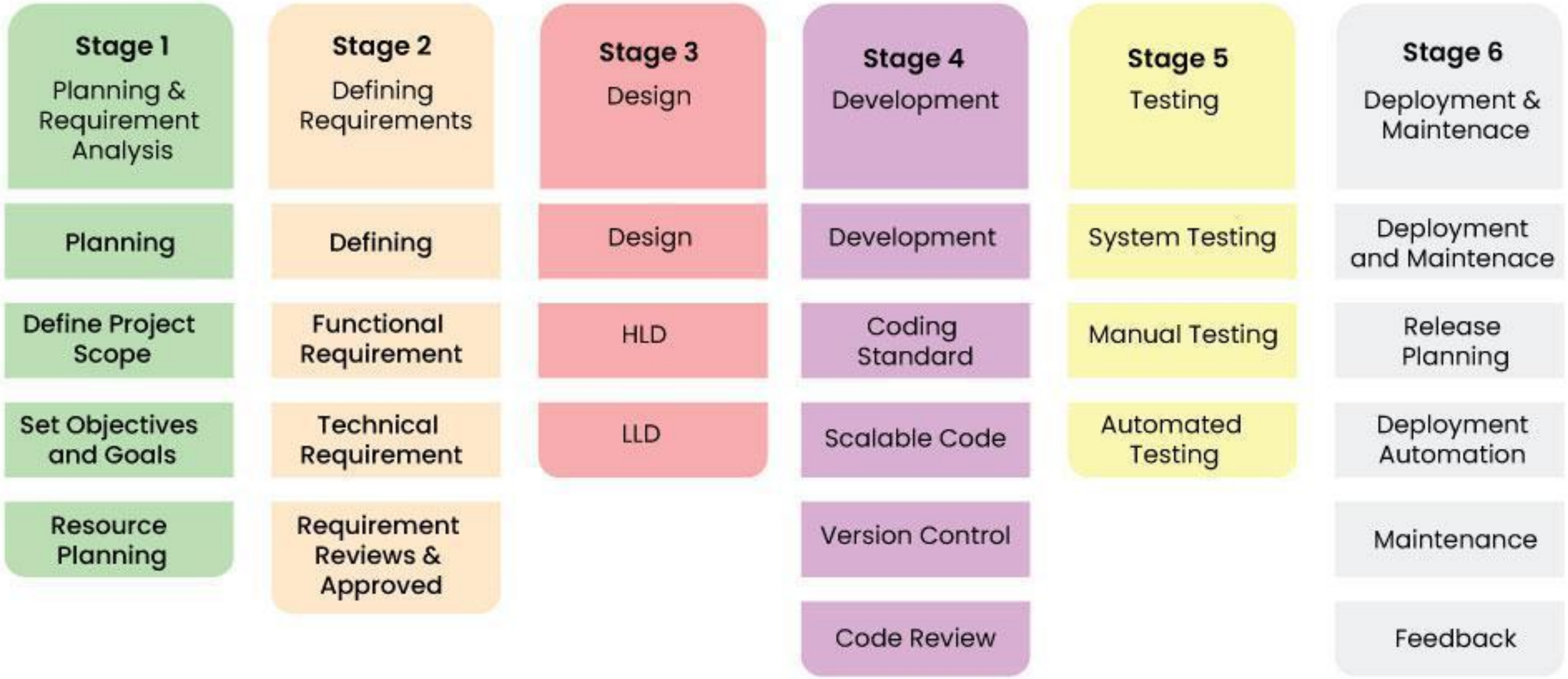
Building

Testing

Deployment

- Stages of the Software Development Life Cycle

- SDLC specifies the task(s) to be performed at various stages by a software engineer or developer. It ensures that the end product is able to meet the customer’s expectations and fits within the overall budget. Hence, it’s vital for a software developer to have prior knowledge of this software development process.



Stage 1 Planning & Requirement Analysis	Stage 2 Defining Requirements	Stage 3 Design	Stage 4 Development	Stage 5 Testing	Stage 6 Deployment & Maintenace
Planning	Defining	Design	Development	System Testing	Deployment and Maintenace
Define Project Scope	Functional Requirement	HLD	Coding Standard	Manual Testing	Release Planning
Set Objectives and Goals	Technical Requirement	LLD	Scalable Code	Automated Testing	Deployment Automation
Resource Planning	Requirement Reviews & Approved		Version Control		Maintenance
			Code Review		Feedback

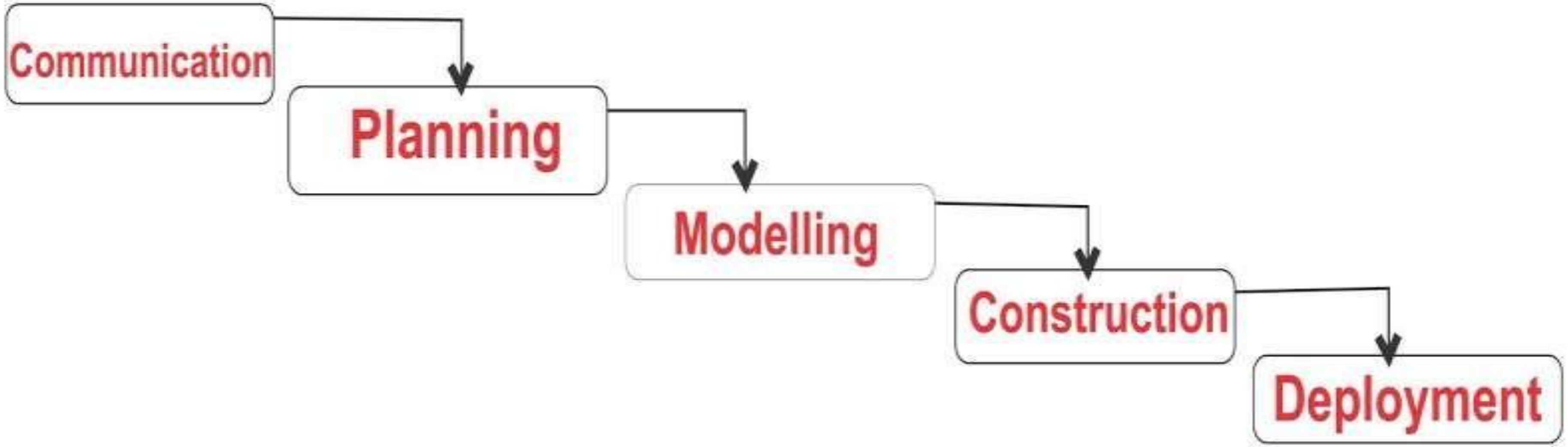
6 Stages of Software Development Life Cycle

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.



Communication

Planning

Modelling

Construction

Deployment

WaterFall Model

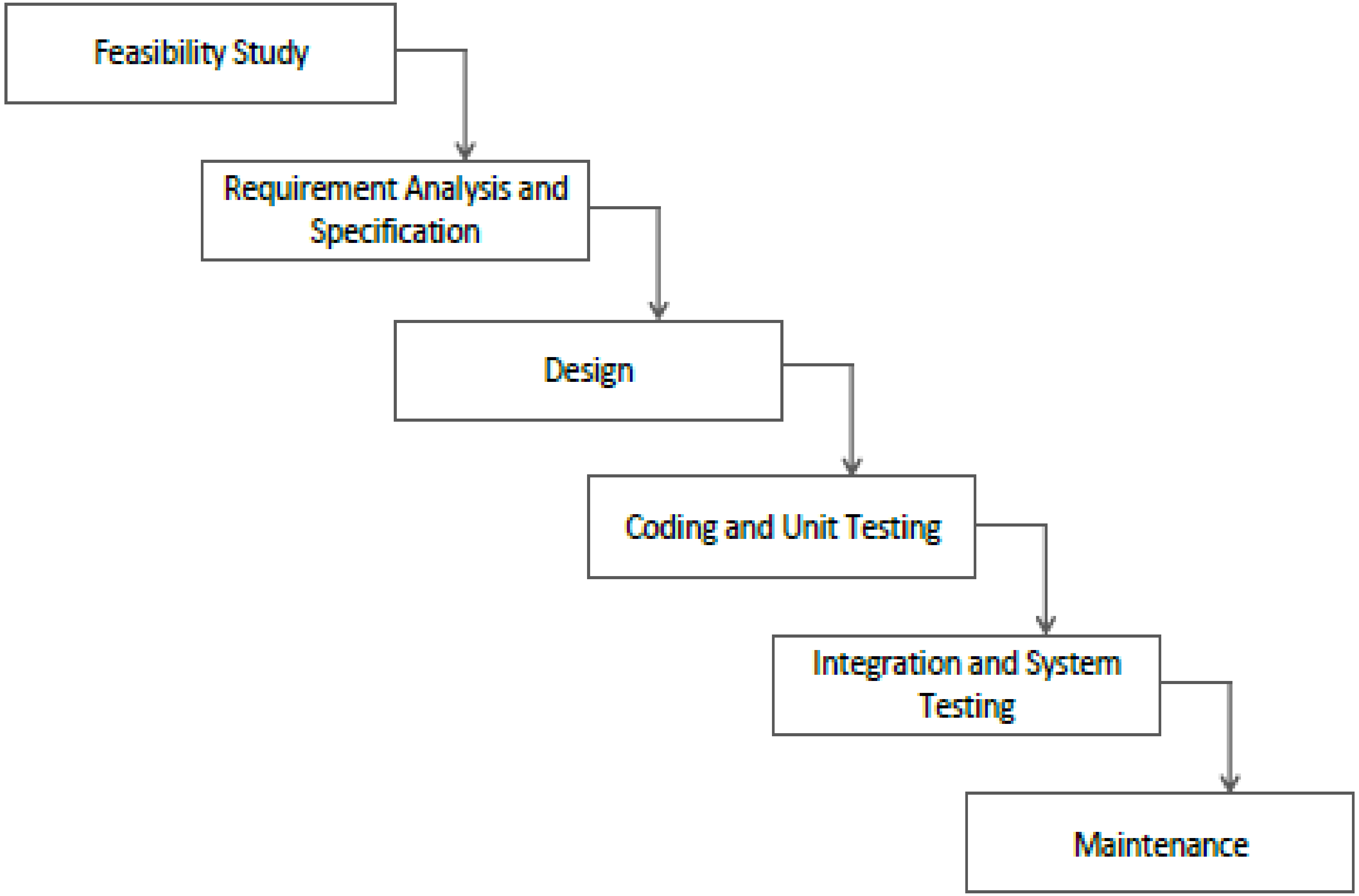
Advantages:

- 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
Requirement Analysis and Specification
Design
Coding and Unit Testing
Integration and System Testing
Maintenance

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:

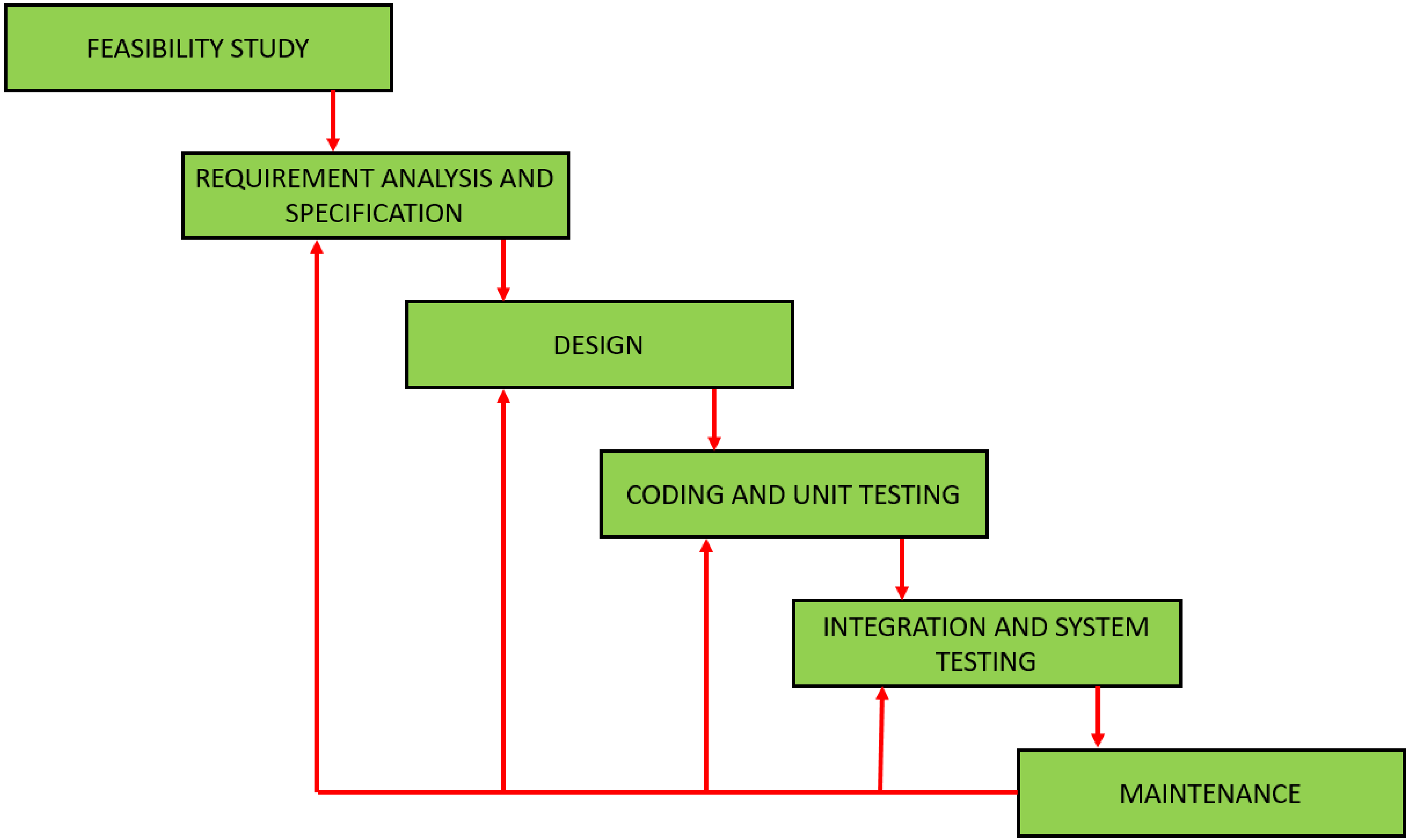
- ‘Integration of different modules is coded and unit tested.
  - $\alpha - Testing$
  - $\beta - Testing$
  - Acceptance Testing.

Maintenance:

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

Iterative Waterfall Model

- The Iterative Waterfall Model is a software development approach that combines the sequential steps of the traditional Waterfall Model with the flexibility of iterative design. It allows for improvements and changes to be made at each stage of the development process, instead of waiting until the end of the project. The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model


FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Prototype Model:

Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. It offers a small-scale facsimile of the end product and is used for obtaining customer feedback.

This model is used when the customers do not know the exact project requirements beforehan In this model, a prototype of the end product is first developed, tested, and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms th basis for developing the final product.

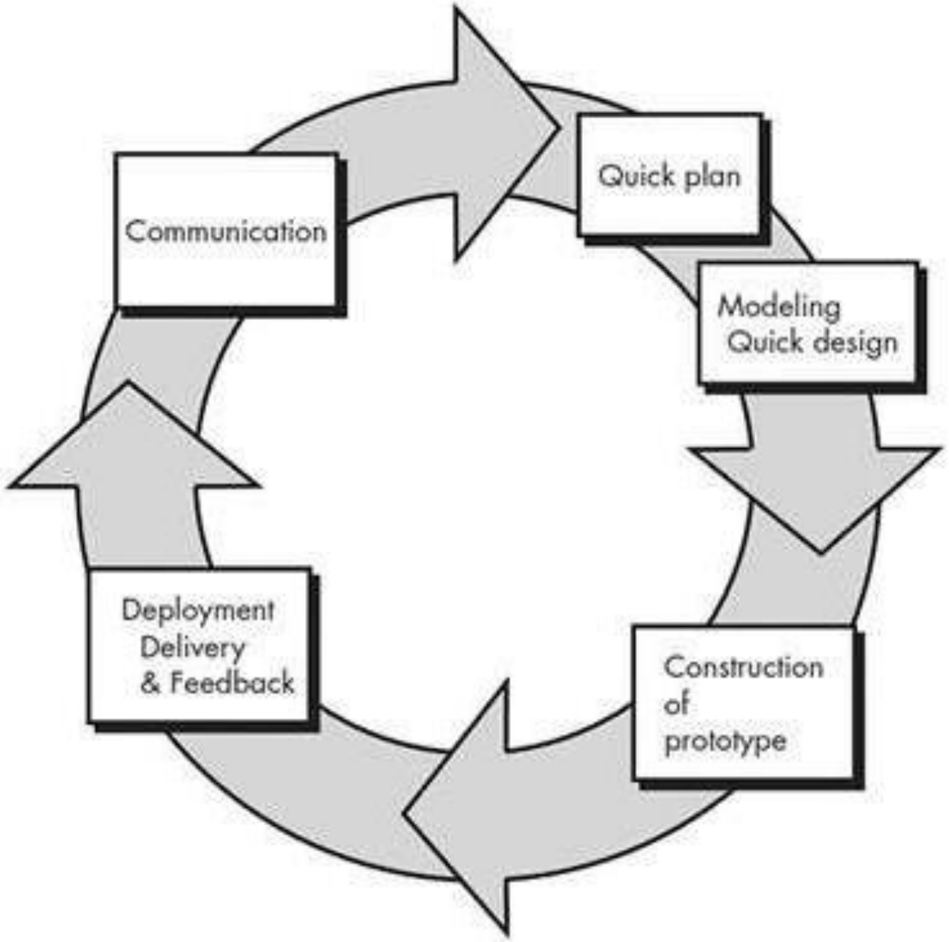


Figure: Prototype Model

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.

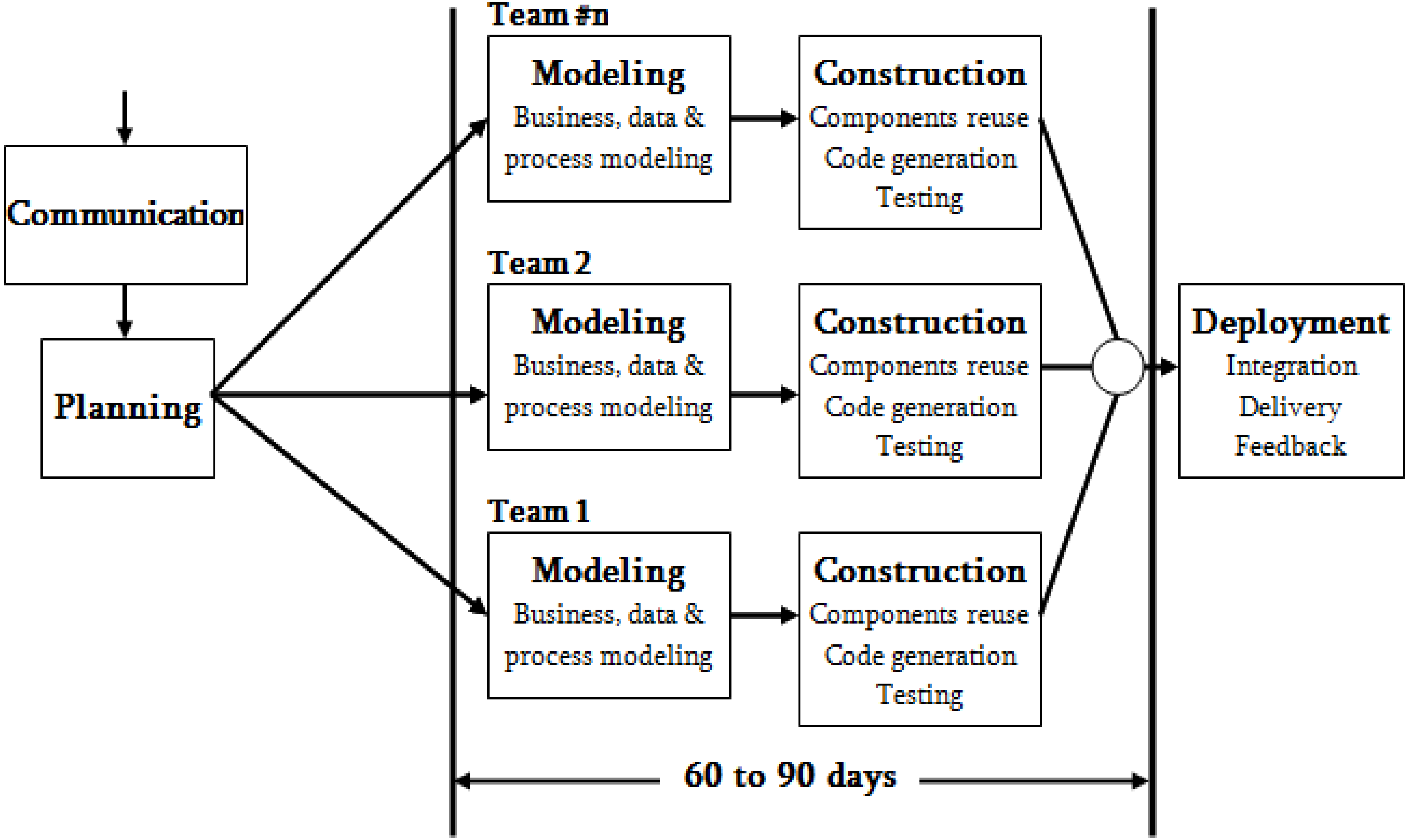


Figure : Flowchart of RAD model

Contents of RAD Pakages:

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

Advantages:

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

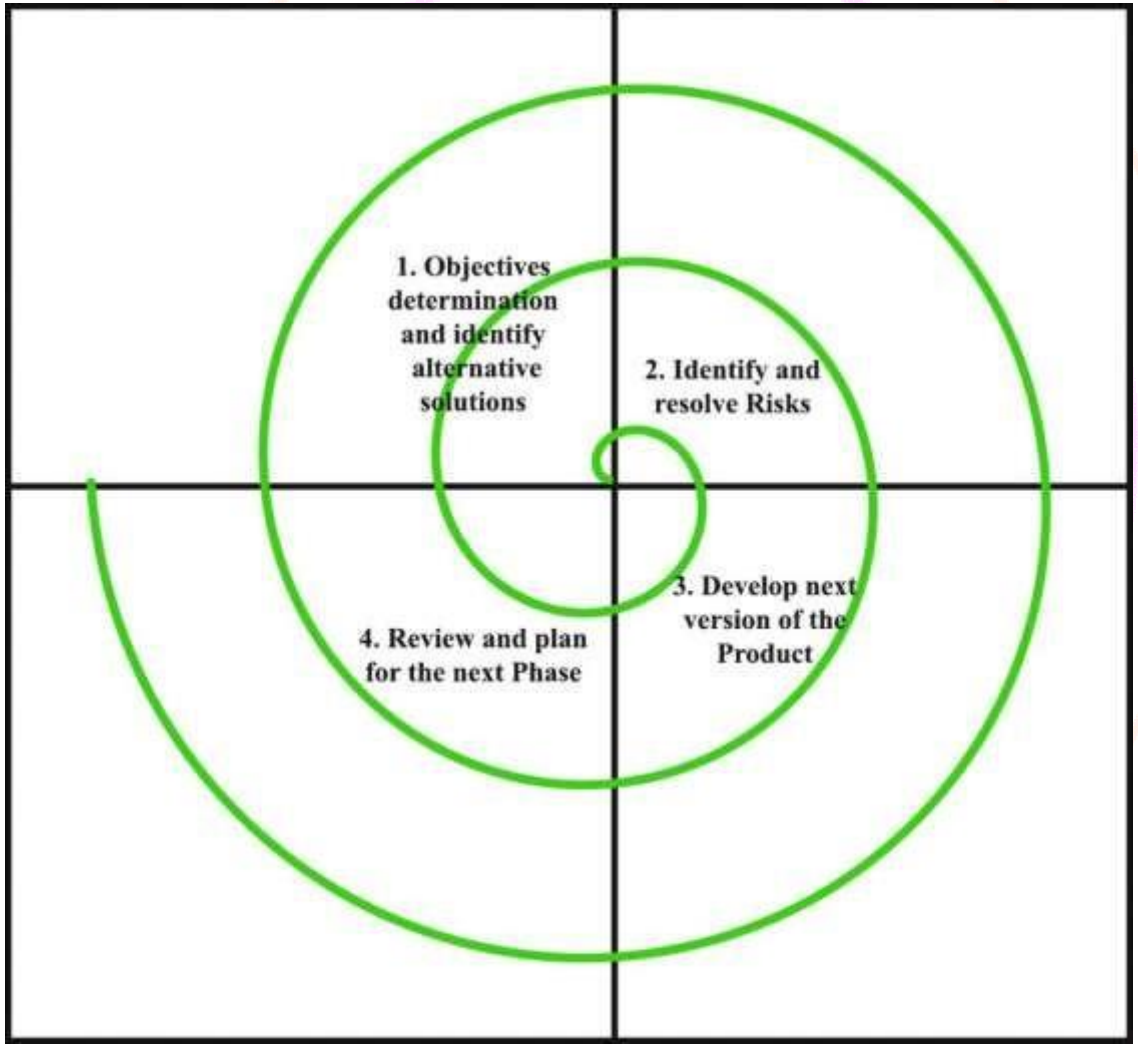
Disadvantages:

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

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 :


1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

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.

Planning:

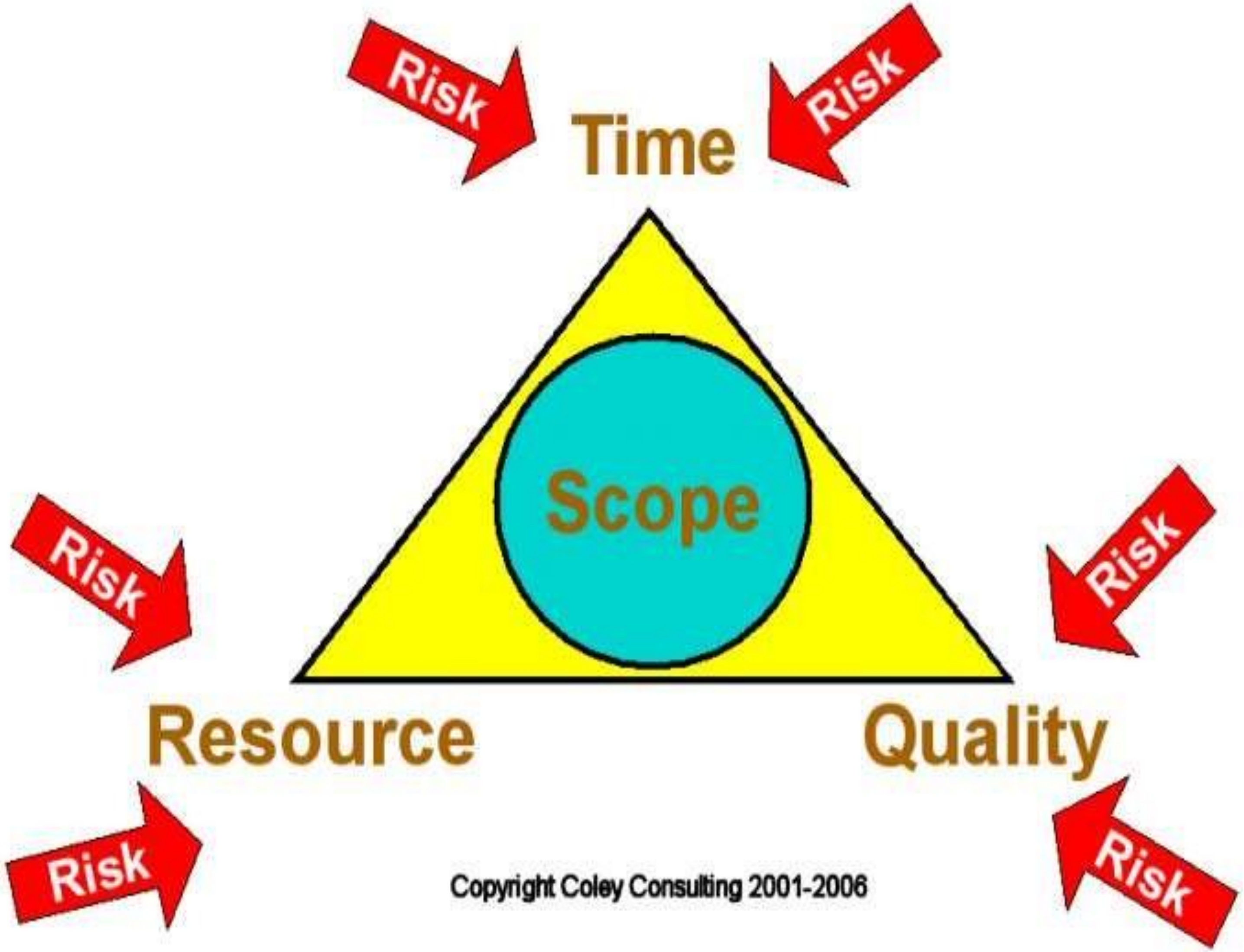
- Software planning process include steps to estimate the size of the software work products and the resources needed produces a schedule identify and access software risks.
- **During planning a project is split into several activities :**
- How much efforts is required to complete each activities?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

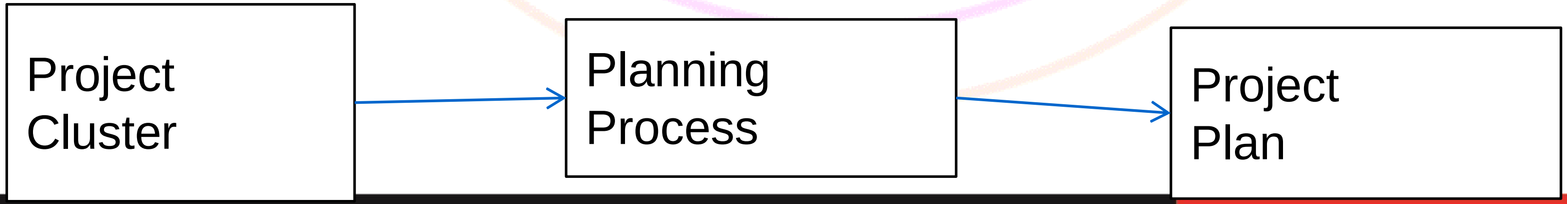
- Scope.
- Resource.
- Time.
- Quality.
- Risk.



Copyright Coley Consulting 2001-2006

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.

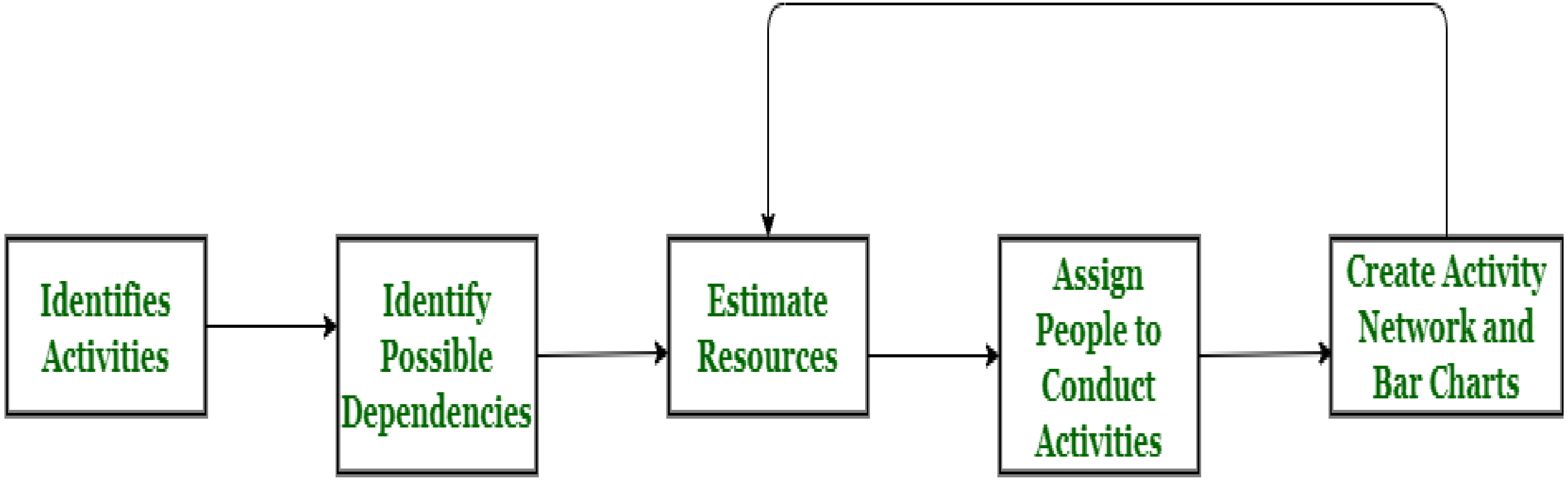
Project planning.

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.


Identifies Activities
Identify Possible Dependencies
Estimate Resources
Assign People to Conduct Activities
Create Activity Network and Bar Charts

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.

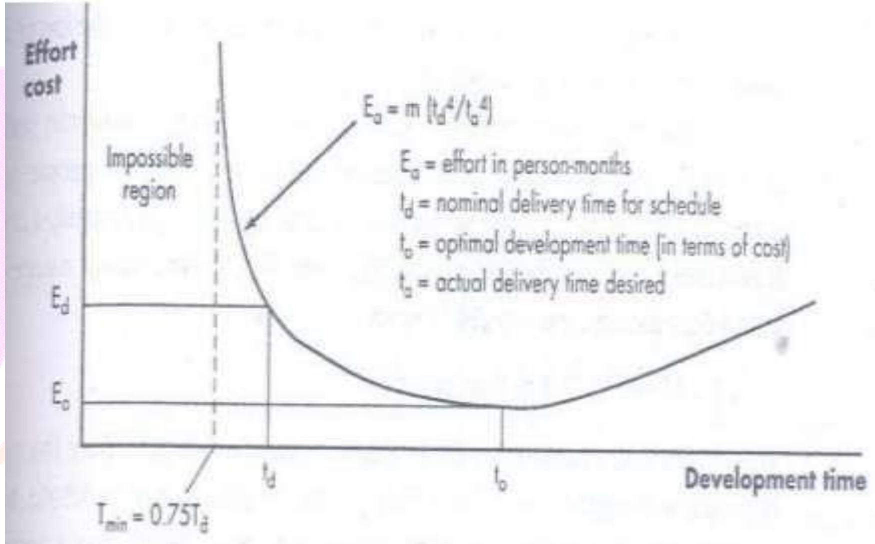


Figure 24.1: Putnam-Norden-Rayleigh (PNR) Curve

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.

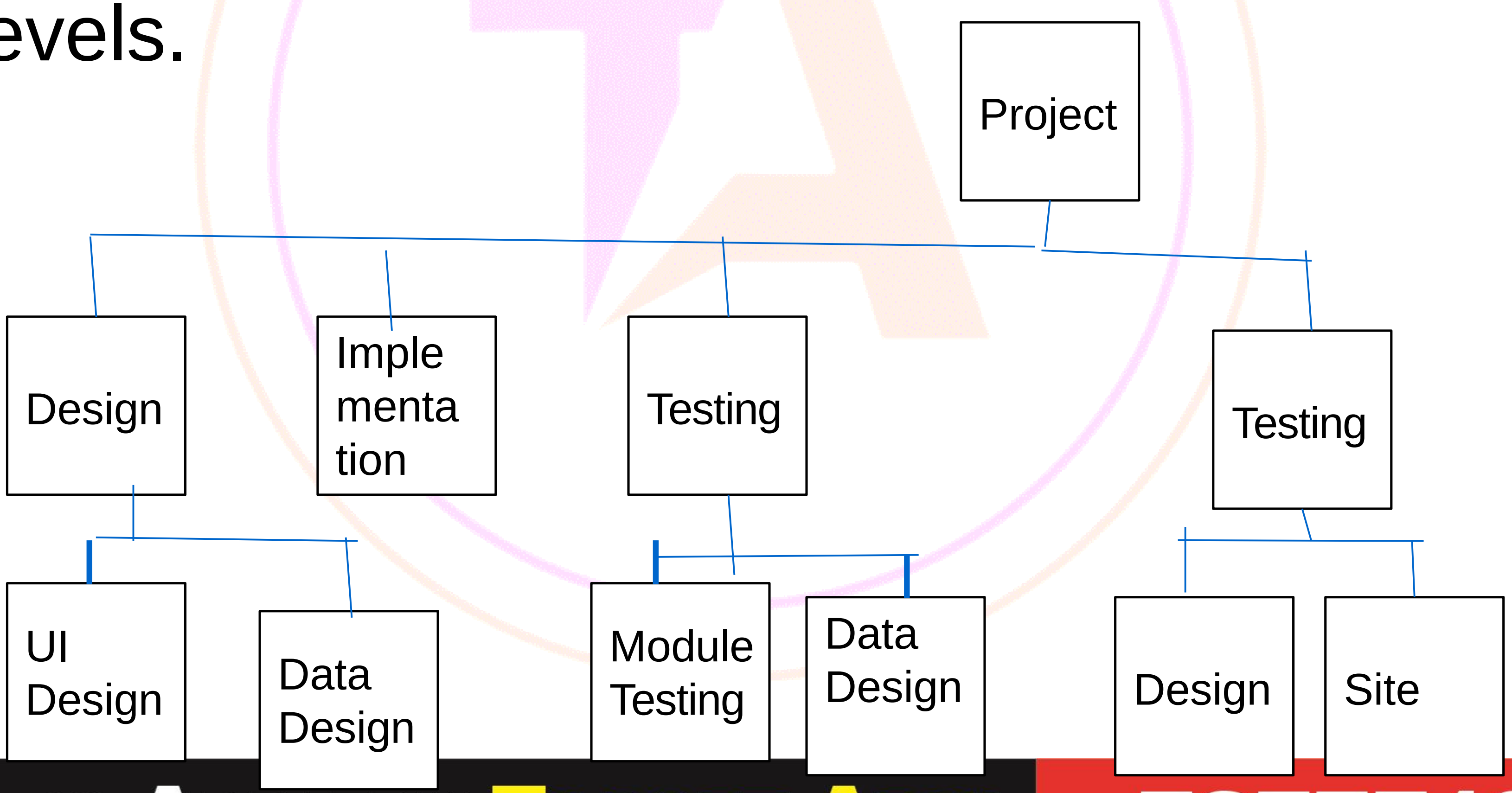
- **Customer Reaction:** Customer reaction to the concept feedback on a new technology concept and target specific customer applications.

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:

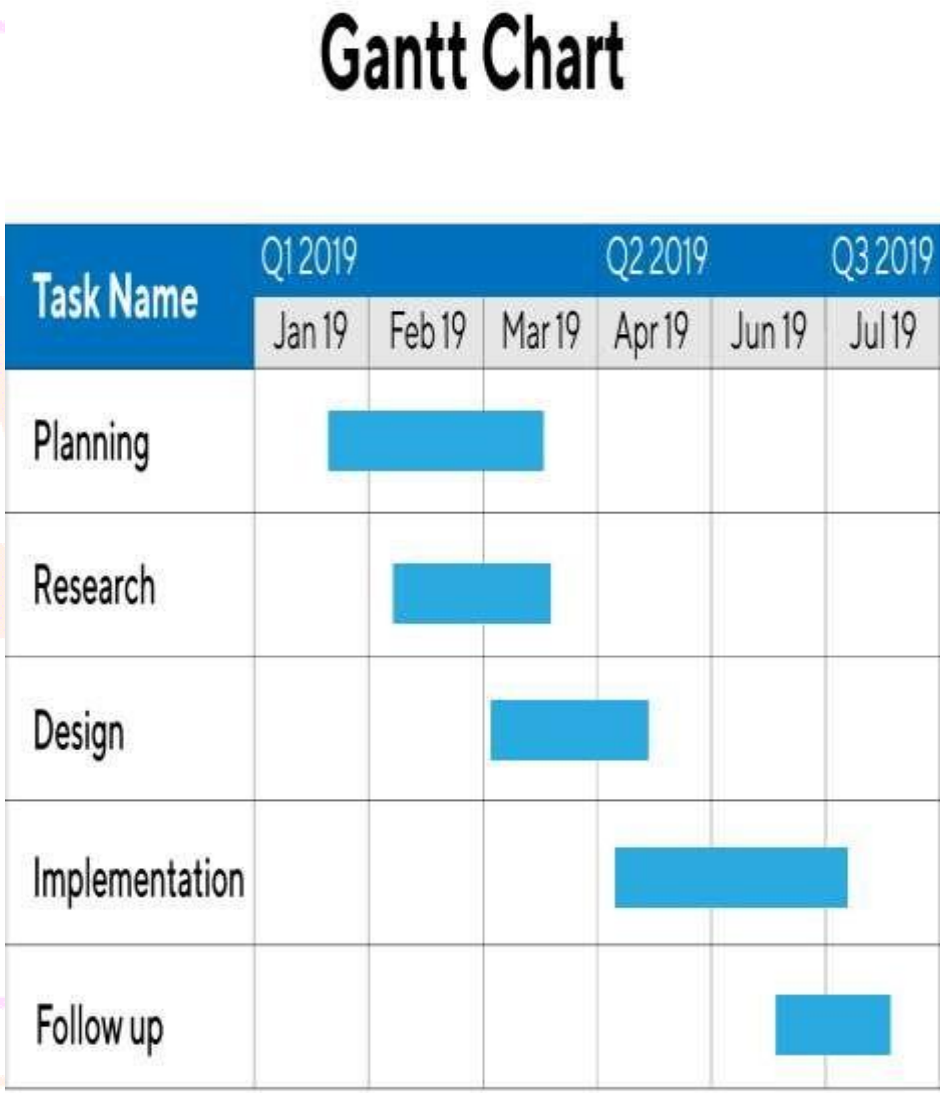
- Network of boxes and arrows.
- Shows different tasks making up a project.
- Represents the ordering among the tasks.

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.