

SOFTWARE ENGINEERING

MSAJCE

UNIT I - INTRODUCTION

- Software Engineering paradigms –
Waterfall Life cycle model – Spiral Model
– Prototype Model – Fourth Generation
Techniques – Planning – Software Project
Scheduling, – Risk analysis and
management – Requirements and
Specification

UNIT II - SOFTWARE DESIGN

- Abstraction – Modularity – Software Architecture – Cohesion – Coupling – Various Design Concepts and notations – Real time and Distributed System Design – Documentation – Dataflow Oriented design – Jackson System development – Designing for reuse – Programming standards – Case Study for Design of any Application Project.

UNIT III - SOFTWARE TESTING AND MAINTENANCE

- Software Testing Fundamentals – Software testing strategies – Black Box Testing – White Box Testing – System Testing – Object Orientation Testing – State based Testing - Testing Tools – Test Case Management – Software Maintenance Organization – Maintenance Report – Types of Maintenance

UNIY IV - SOFTWARE METRICS

- Scope – Classification of metrics – Measuring Process and Product attributes – Direct and Indirect measures – Cost Estimation - Reliability – Software Quality Assurance – Standards – COCOMO model.

UNIT V - SCM

- Need for SCM – Version Control – SCM process – Software Configuration Items – Taxonomy – CASE Repository – Features – Web Engineering

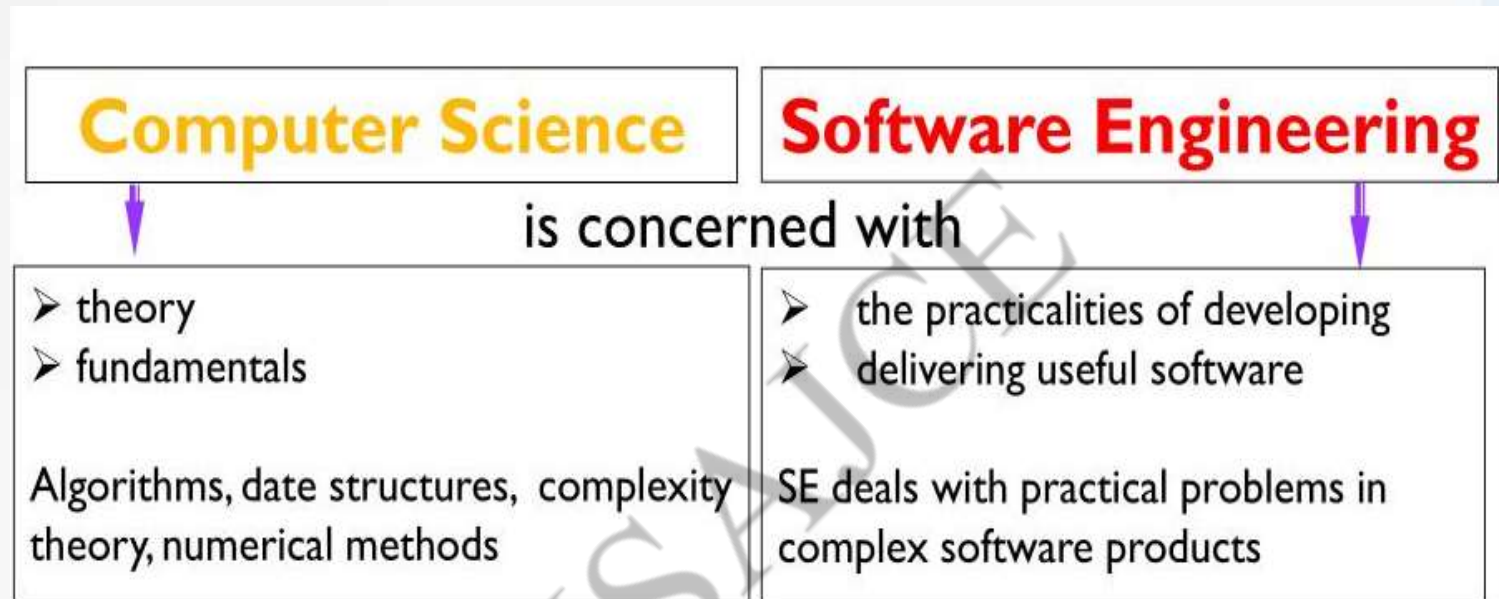
UNIT I - INTRODUCTION:

- Software is more than just a program code.
- A program is an executable code, which serves 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
- **What are the attributes of good software?**
 - Maintainability.
 - Dependability
 - Efficiency
 - Usability

What is the difference between software engineering and computer science?

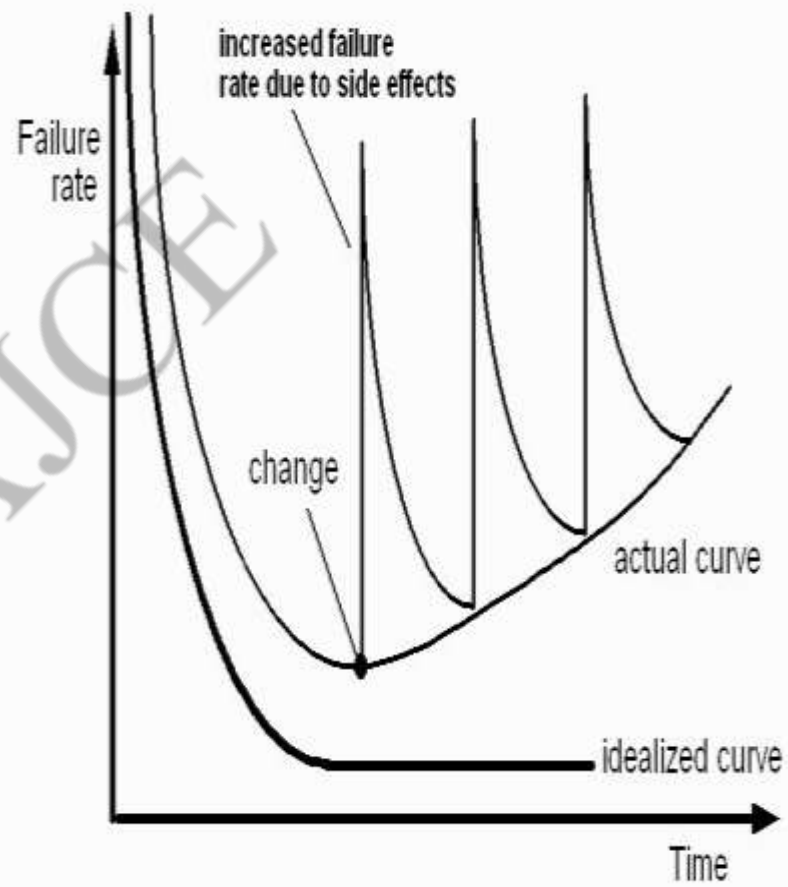
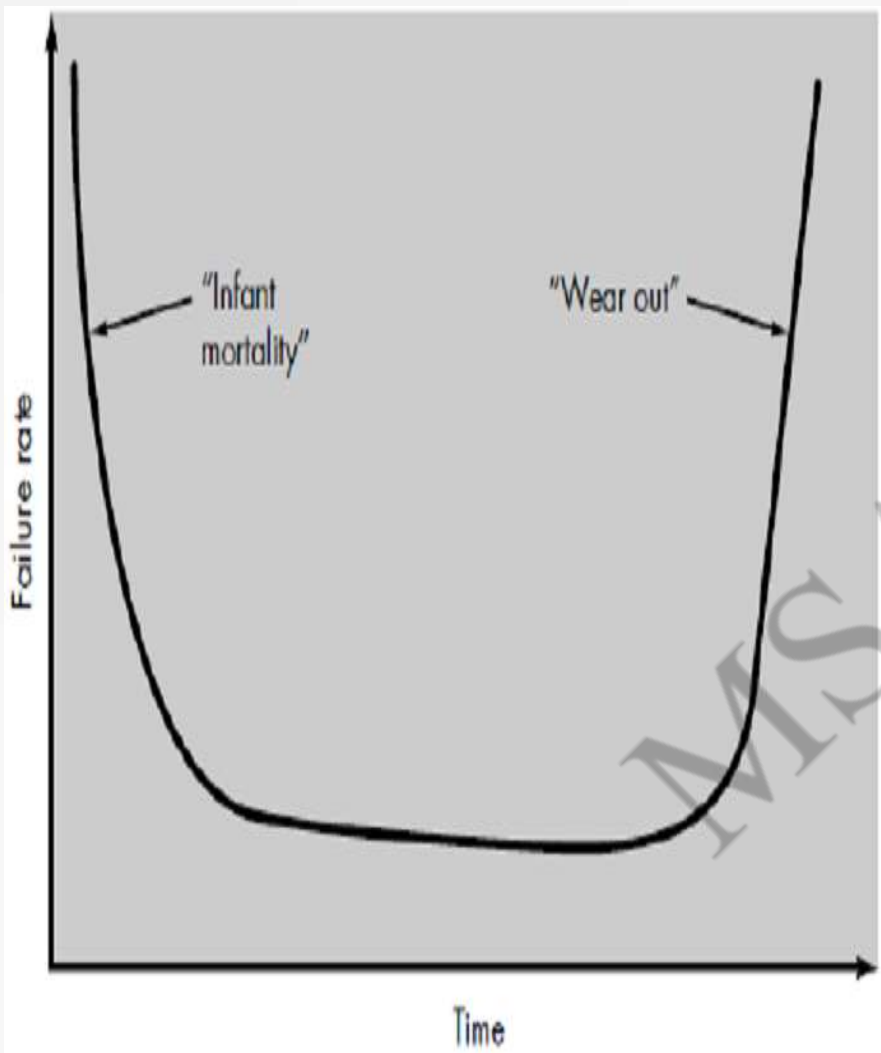


Computer science theories are currently insufficient to act as a complete underpinning for software engineering, BUT it is a foundation for practical aspects of software engineering

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.

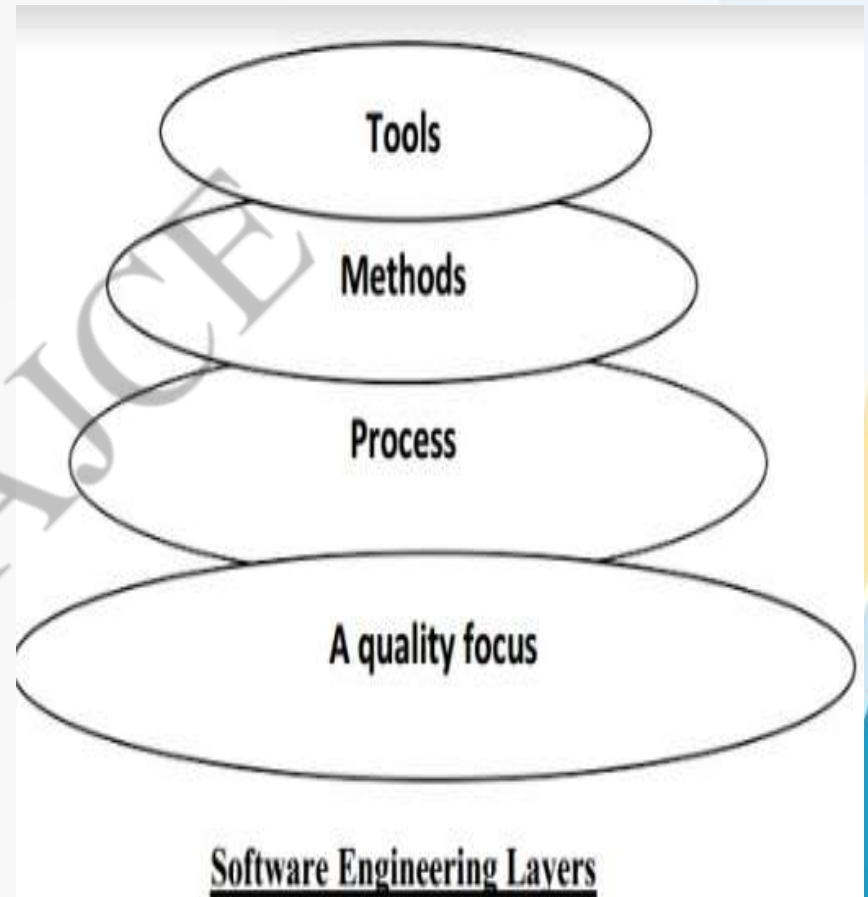


Software Applications Types:

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



What are the five generic process framework activities?

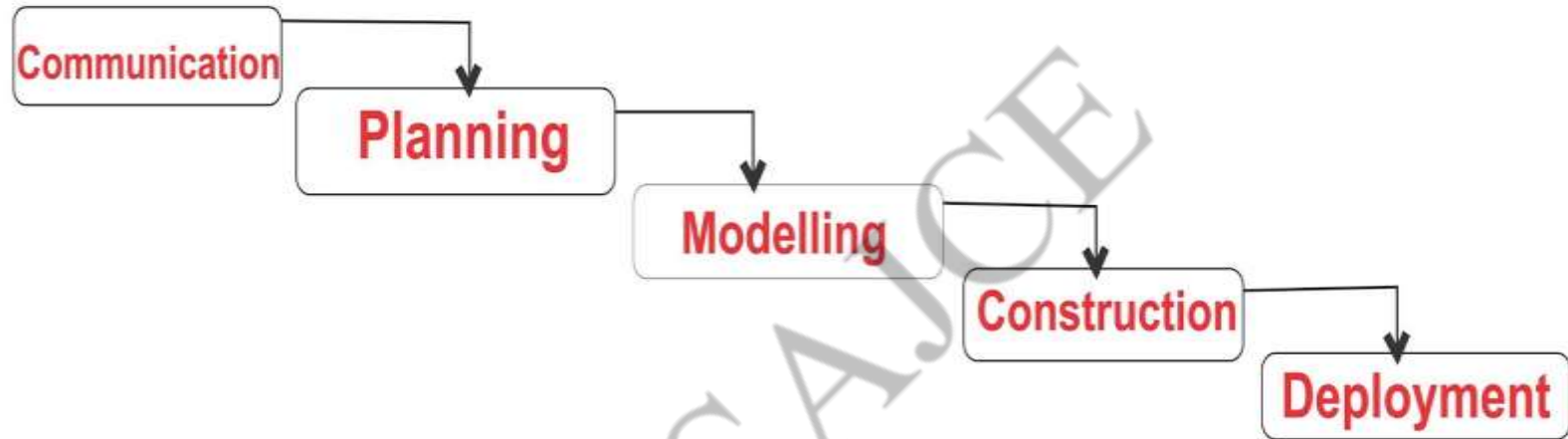
- The following generic process framework is applicable to the majority of software projects.
 - Communication.
 - Planning.
 - Modeling.
 - Construction.
 - Deployment.

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.



WaterFall Model

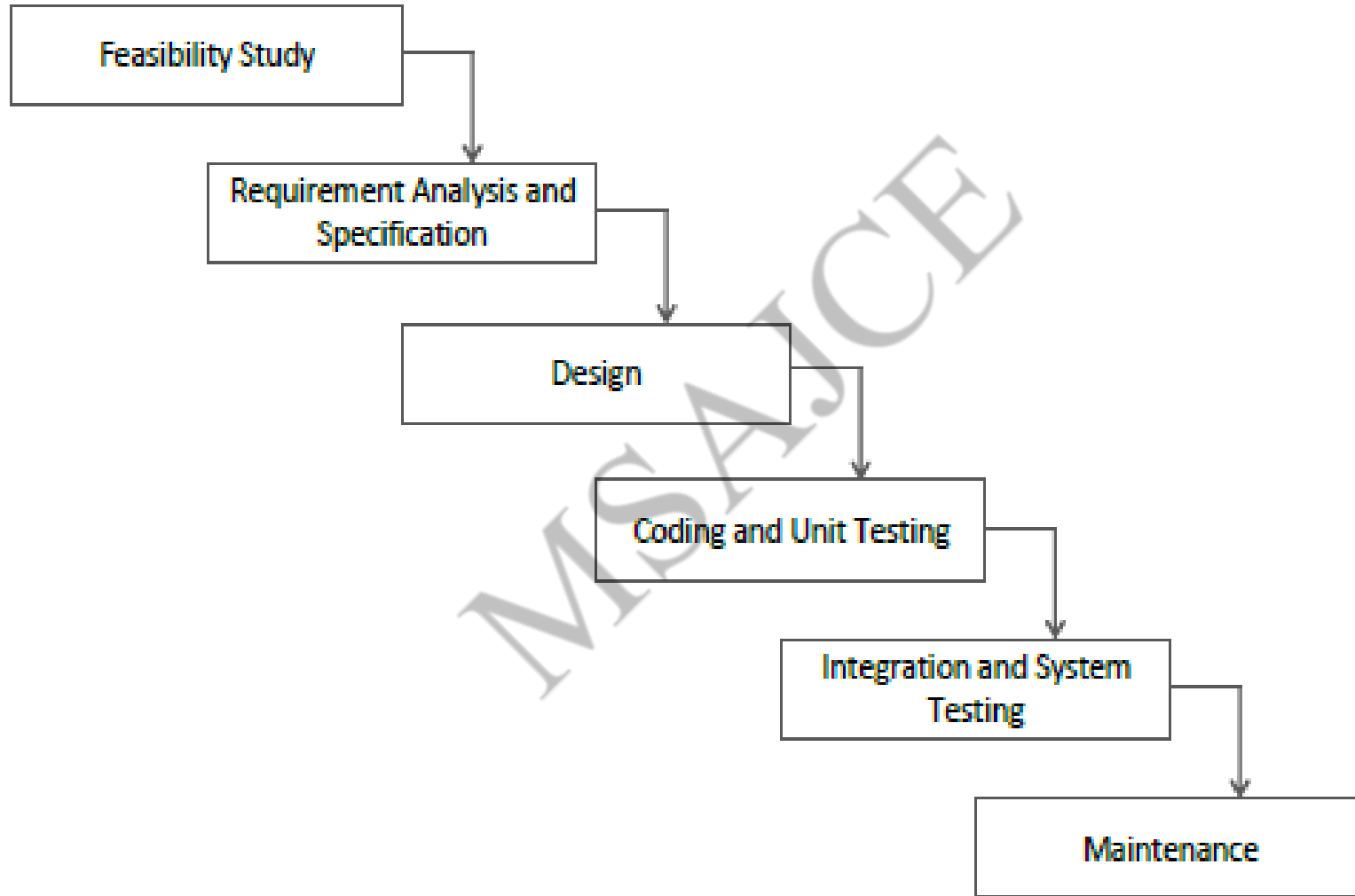
Advantages:

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

Disadvantages:

- One way street.
- It lack overlapping and interactions among phases.
- Model doesn't support delivery of system in pieces.

Phases of the Classical Waterfall Model:



Feasibility Study:

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

Requirements Analysis and Specification:

- It is understand the exact requirements 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:

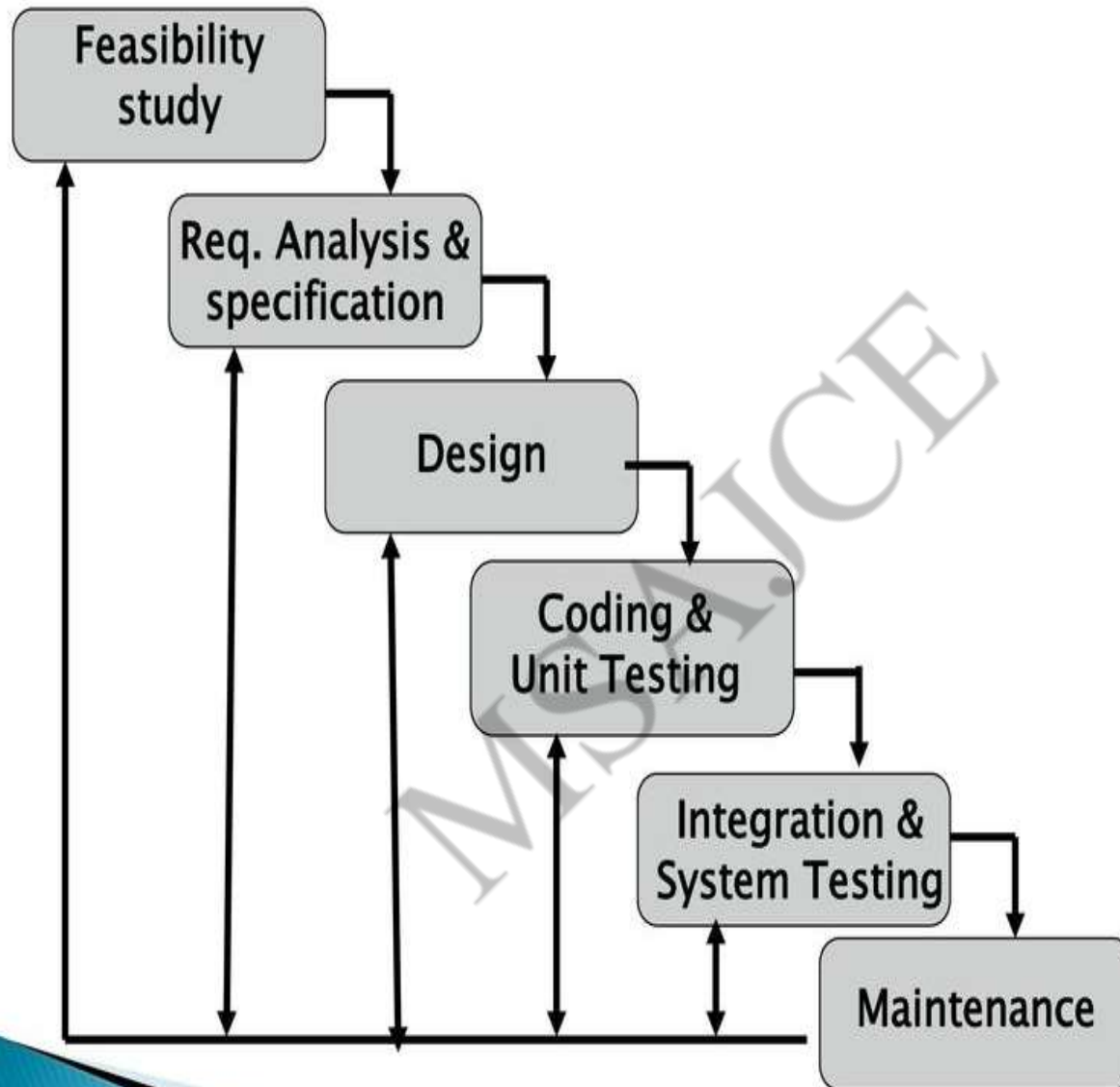
- 'Integration of different modules is coded and unit tested.'
 - α – Testing
 - β – Testing
 - Acceptance Testing.

Maintenance:

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

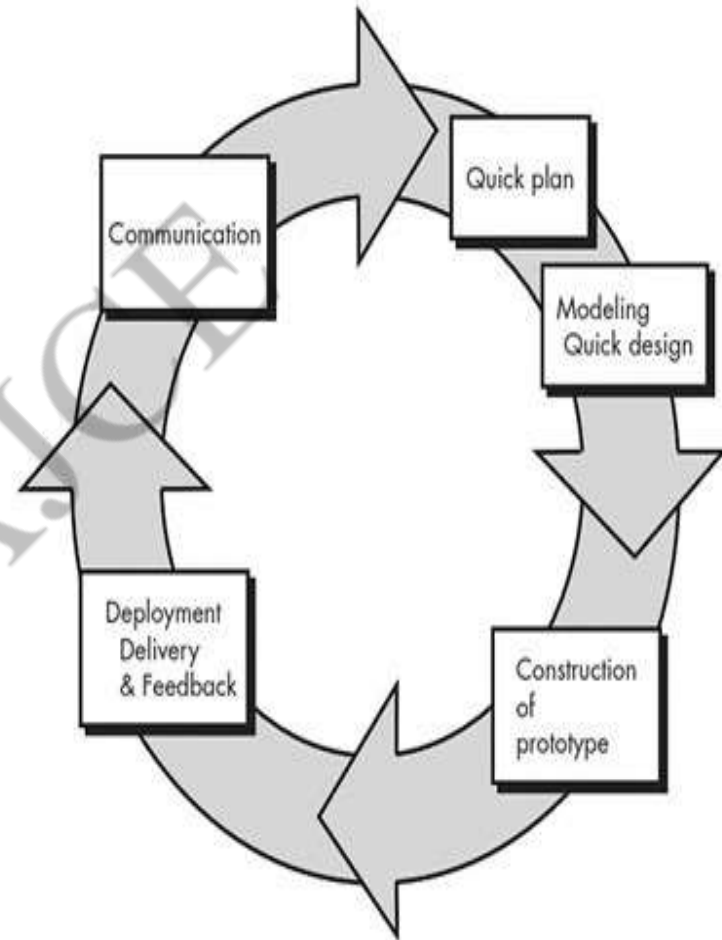


Figure: Prototype Model

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.

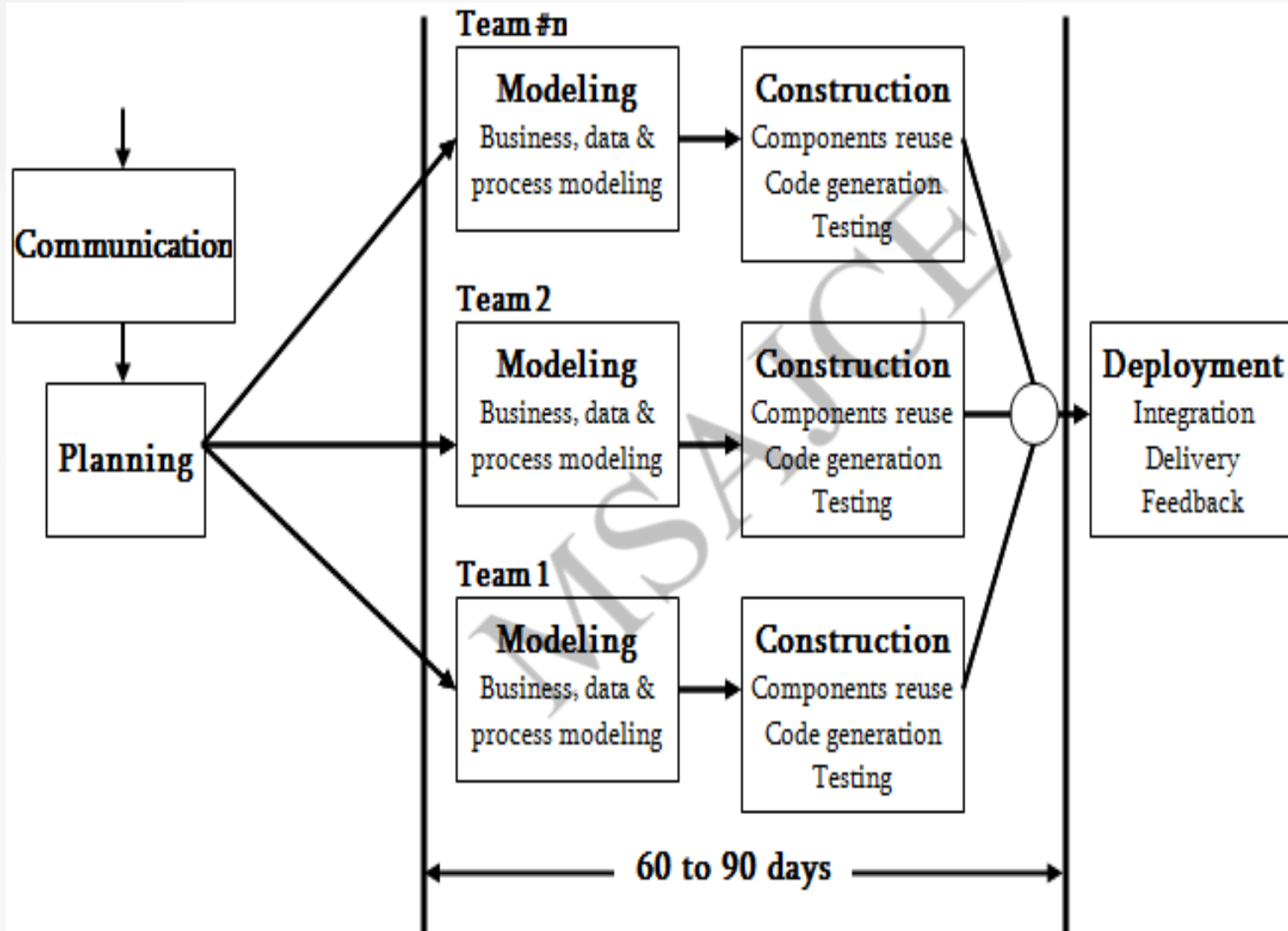


Figure : Flowchart of RAD model

Contents of RAD Packages:

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

Advantages:

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

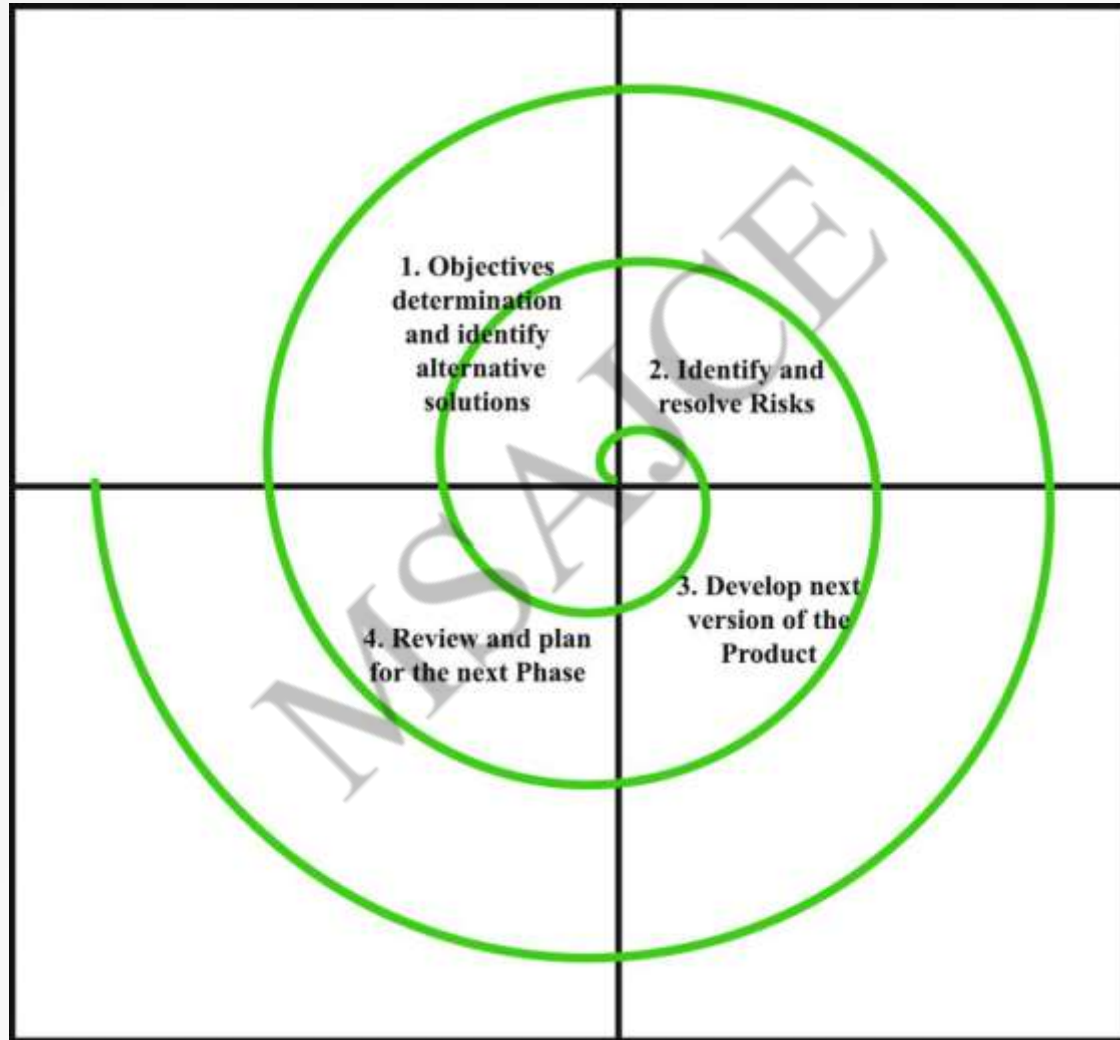
Disadvantages:

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

Spiral 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 spirial model :



Advantages :

- Risk Identification at early stage.
- Suitable for high risk projects.
- Flexibility for adding functionality.

Disadvantages:

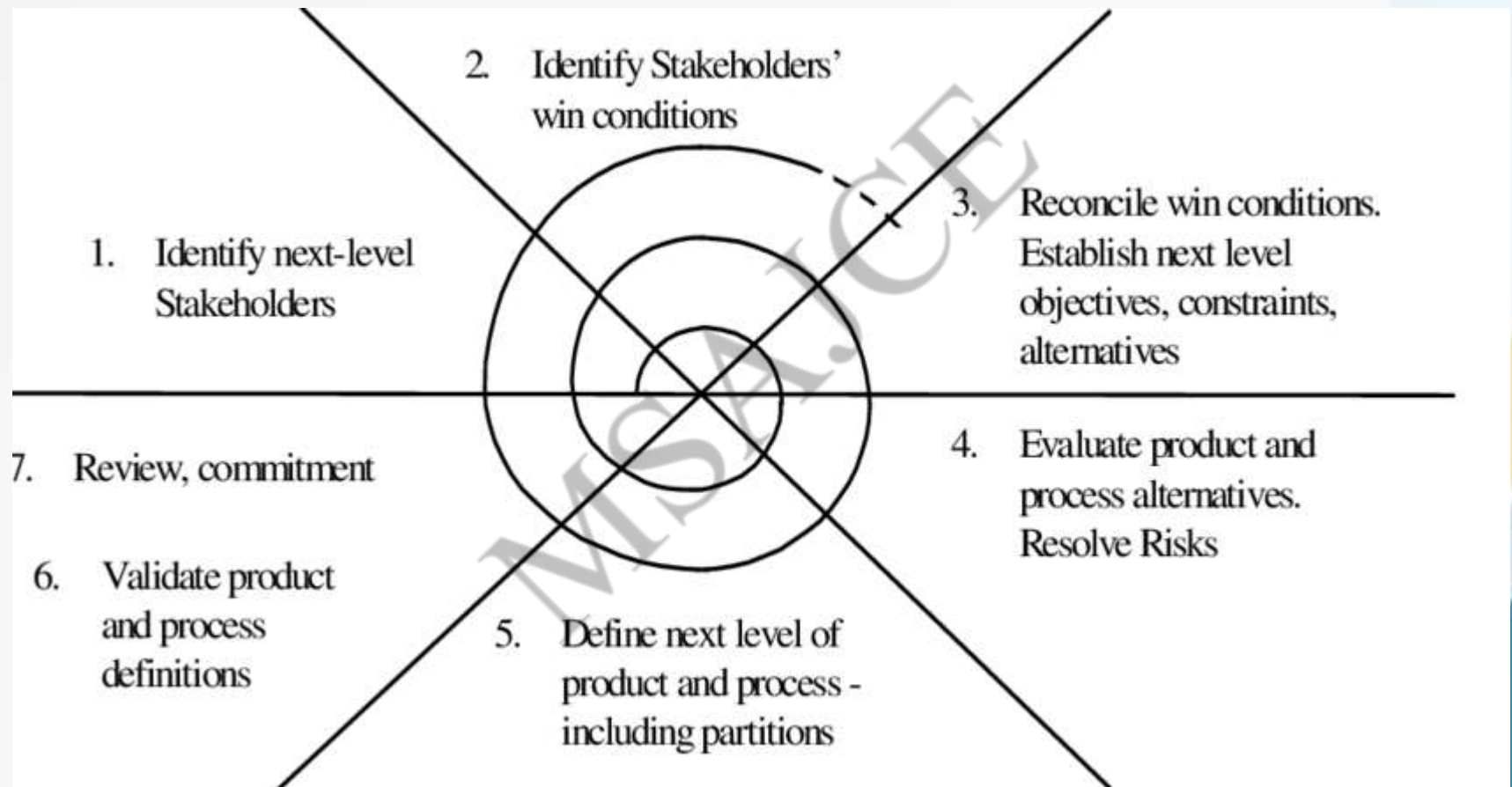
- Costly.
- Risk dependent.
- Not suitable for smaller projects.
- Difficult to meeting budget.

Win-Win Spiral Model:

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

Disadvantages:

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



Fourth Generation Techniques:

- **Introduction:**

- The tools in automatically generate source 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:

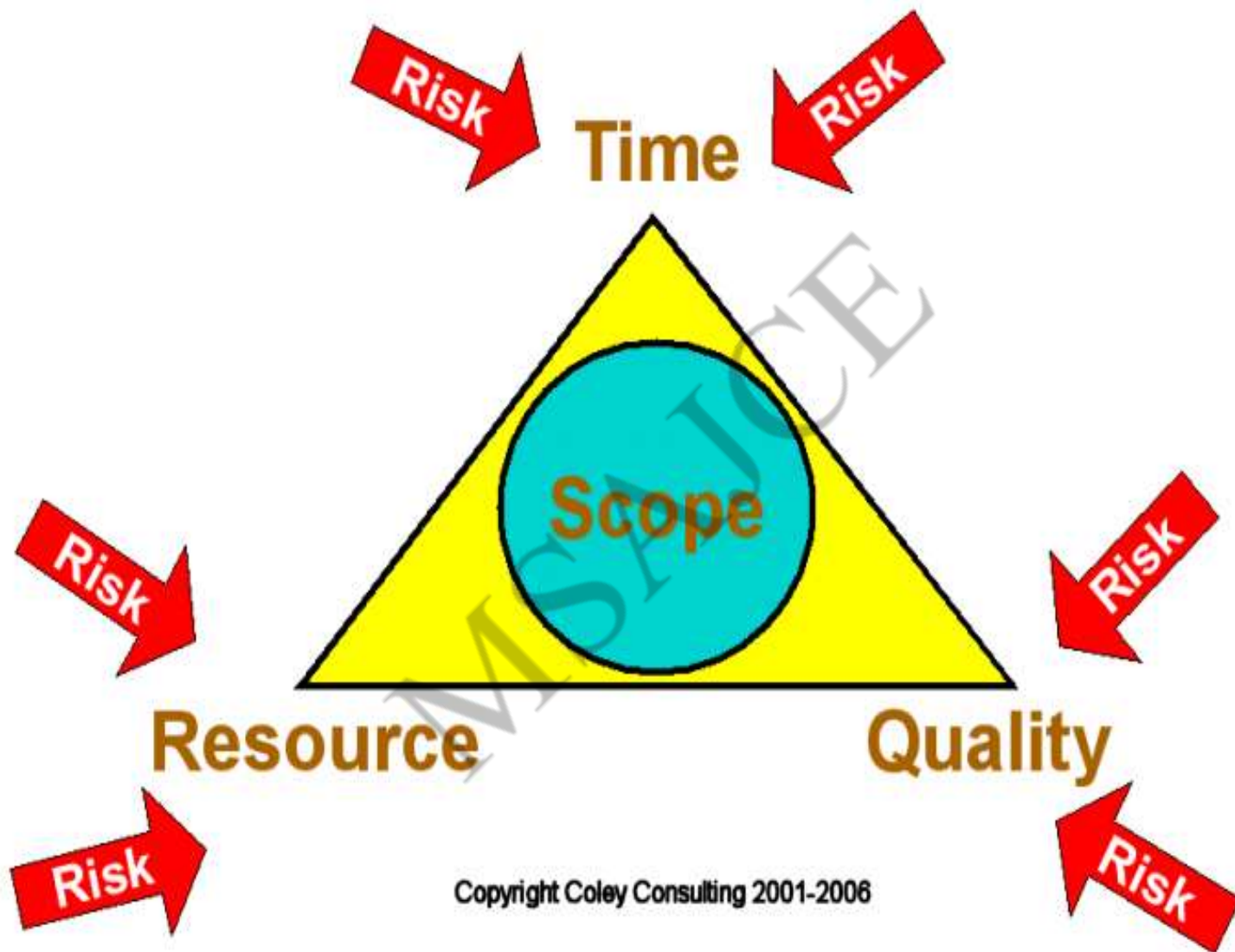
- Software planning process include steps to estimate the size of the software work products and the resources needed produces a schedule identify and access software risks.
- **During planning a project is split into several activities :**
- How much efforts is required to complete each activities?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



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.

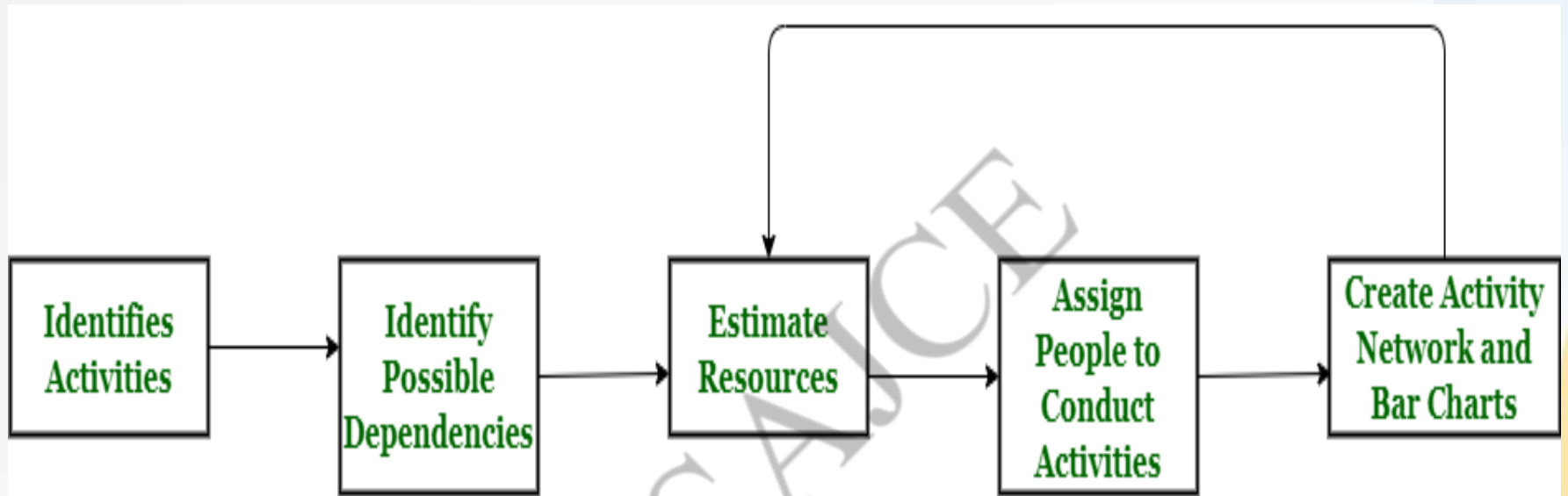
Project planning.

Major issues in planning a software project:

- Software requirements 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 scheduling involves seperating the total work involved in a project in seperate activities and judging the time required to complete the activities.



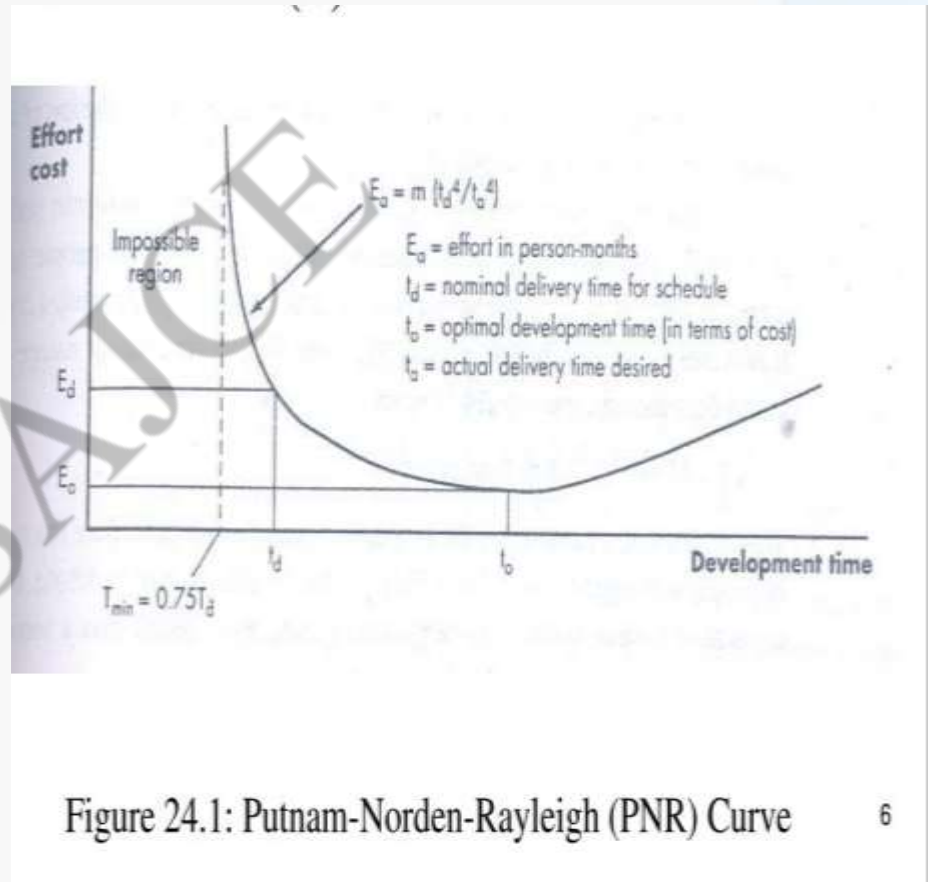
Project Scheduling Process

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.



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 accomplished 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.

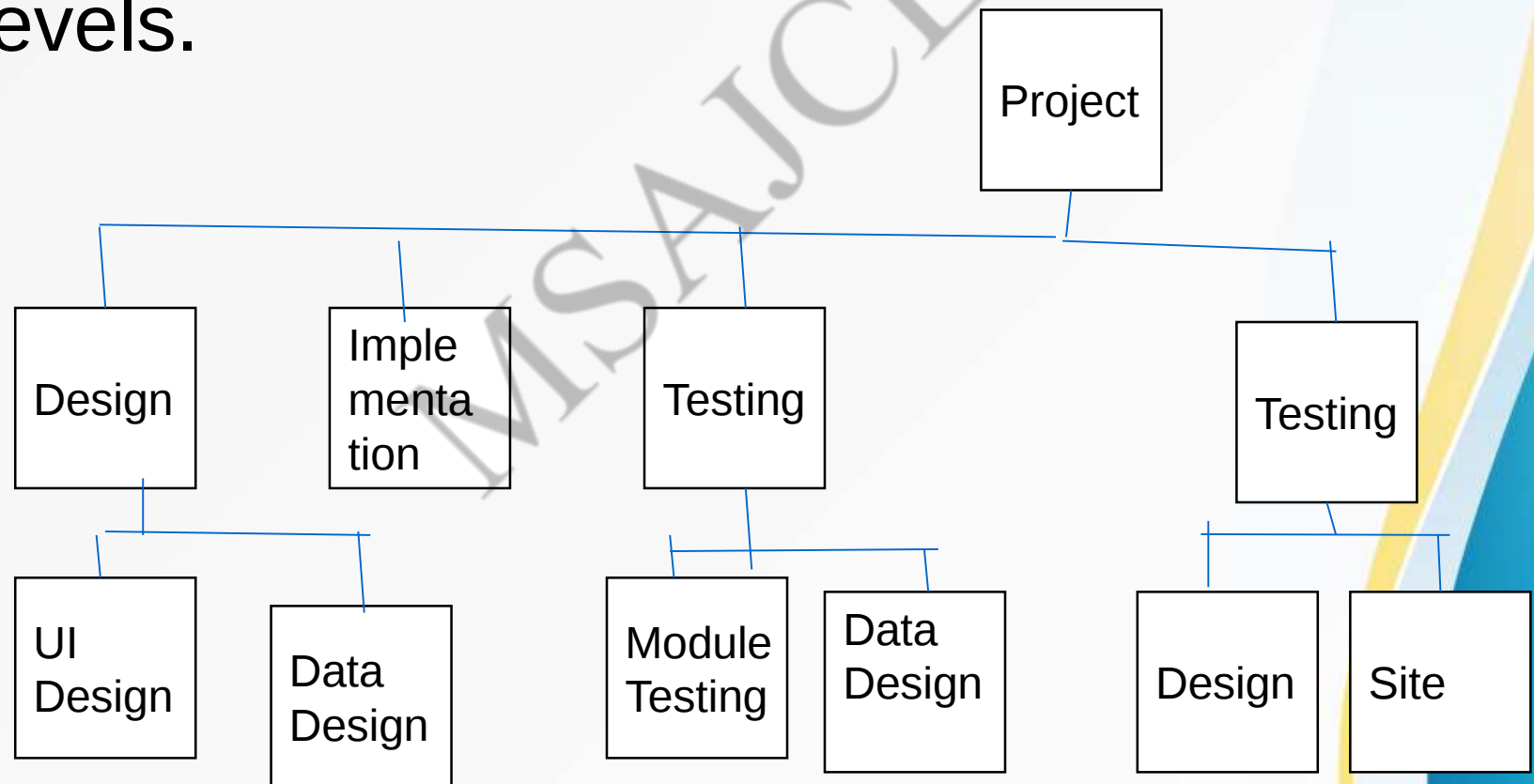
- **Customer Reaction:** Customer reaction to the concept feedback on a new technology concept and target specific customer applications.

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).
 - Gantt Charts.
 - Critical Path Method(CPM).

Work Breakdown Structure(WBS):

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

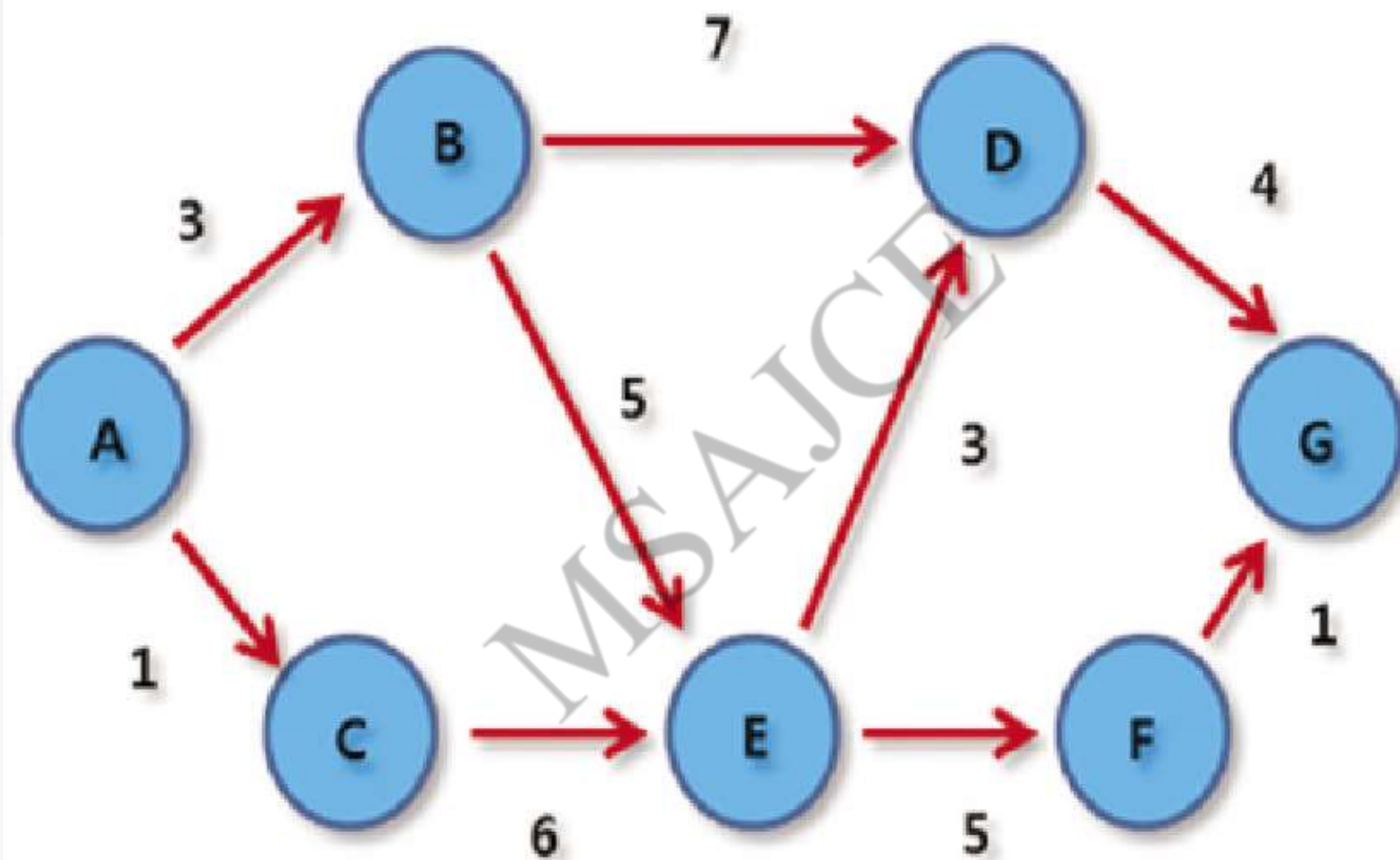


Features of WBS:

- 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.

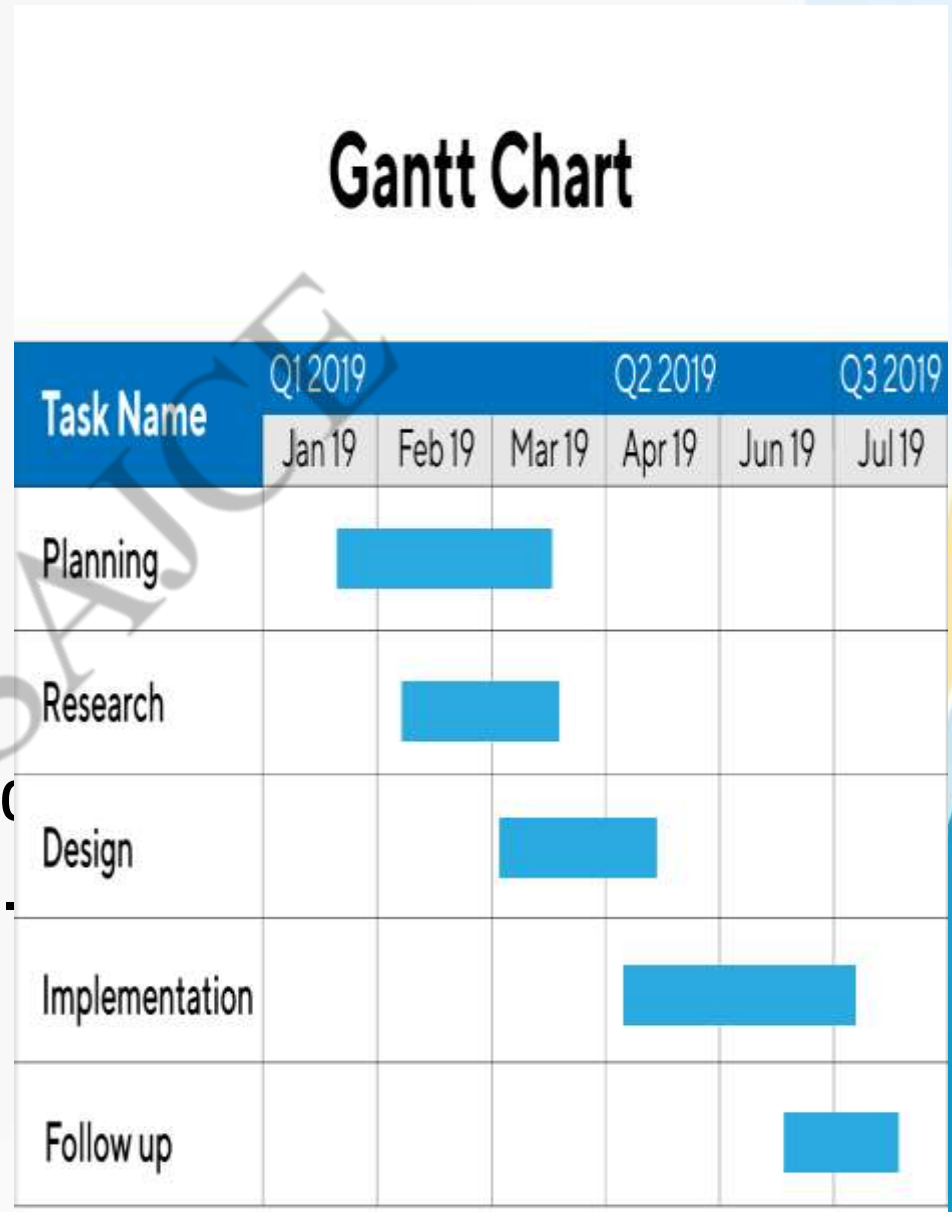


Project Evaluation Review Technique(PERT):

- PERT chart is a project management tool used to schedule, organize and coordinate tasks within the project.
- PERT methodology 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 developed as 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 preparation 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 engineers 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.
- Unpredictable Risks.

Classification I:

- **Project Risks:** The project schedule will slip and that costs will increase. 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, verification 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.

- **Unpredictable Risks:** Unpredictable Risks are the joker in the deck, they can be 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 of 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.
- Development environment.

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 drivers.
- Determine sources of risks.
- Estimate risk.

Risk Prioritization:

- The project focus on its most server risks by assessing the risk.
- Let (r) is the likelihood 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 dealing with each significant risk.

Strategies in risk management planning:

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

Risk Resolution:

- 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 contract between 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 completeness and consistency:

Non-Functional Requirements:

- The system properties and constraints various properties of a system can be: reliability, response time, 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 Requirements:

- Understandability.
- Implicitness.

User Requirements:

- User requirements are defined using natural language labels and diagrams because these are the representation that can be understood 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.