SOFTWARE ENGINEERING

Prepared by

Mr. D. Karthik,

Asst. Professor,

PG & Research Department of computer science,

Government Arts college, Ariyalur

UNIT 1

Content of this Unit

Introduction

What is Software?

- Software is a Collection of Similar tasks that provides a single interface to the user to obtain the results for a variety of requests.
- Software is the Controlling unit of an H/w Component.
- Computer Contains Memory device power unit, Capacitors, resistors, processor, monitor and cables.

Software Engineering

- “Software Engineering is the technological and managerial discipline concerned with Systematic production and maintenance of Software product with in the time and cost Estimate.”

Software

- Software is defined as the tool with designed order of Instructions, that performing tasks to provide the desired result.

  Software = Computer programs + Procedures + Rules + (Instructions) Associated Documents + Data

  Examples:

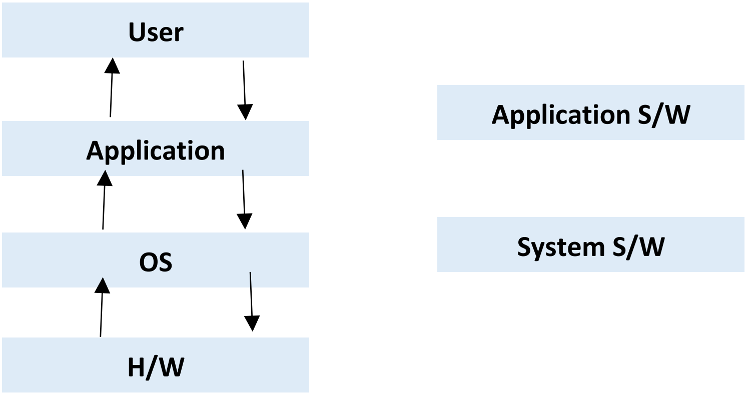
  1. MS Word
  2. MS Excel
  3. MS Power Point
  4. Linux
  5. Photoshop

Software was divided into two Categories

1. Application Software
2. System Software

❖ Application Software

- Application Software Consists of programs purely for the tasks of the users and takes the I/P and processing in the instructions Already defined.
  - **Example:**
    - **MS Office,**
    - **Notepad,**
    - **Calc.**



System Software

System software control the operations of the H/w Components.

Example:

Compilers and Debuggers

- Phases in S/w Development
  - i. Requirement Analysis
  - ii. Designing phase
  - iii. Coding and Testing phase
  - iv. Implementation phase
  - v. Maintenance phase

Requirement Analysis:

First phase of S/w development where in the customer and the Business Analysis interact the S/w Product.

- ✓ It helps to develop a report of customers' Requirements.
- ✓ It is easy to fix a problem in the requirements at the requirement Analysis stage.
- ✓ If we found on later stages of design and coding them it would cost of 10-100 times more to fix the same.
- ✓ Capturing the requirements fully and correctly at the early stage of the project is essential to make the project is profitable.

Types of Requirements:

- ✓ **1.Functional**
- ✓ **2.NonFunctional**
- ✓ **3.Interface**

Steps of Requirement Engineering phase

1.Feasibilty Study

2.Eliciting Requirements

3.Requirement Analysis

4.Specify and Validate Requirement

5.Requirements Management

Models of Requirement Analysis:

- ➢ Structured Analysis
- ➢ Object Oriented Analysis Model.

Designing phase:

1. Software Requirement is converted into design Models.
2. The Design phase acts as a bridge between the Requirement Analysis phase and the coding phase.
3. Designing of the Software depends on complexity level of the product, GUI required and End user requirements.
4. Designing tells the structure of the product.

Modularity, cohesion, coupling, and layering are the factors to be considered while designing the software.

Software Design includes:

- ❖ **Data Design**
- ❖ **Architectural Design**
- ❖ **User Interface Design**
- ❖ **Procedural Design**
- ❖ **Deployment level Design**

Coding and Testing phase:

- ❑ Documented Design is transformed into lines of codes in programming Language.
- ❑ After coding phase is completed the code have to be tested for ensuring its right functionality in the platform.

Implementation:

As we are able to get a developed software into the market and to the end users. It is also called as Deployment phase.

Maintenance

The process of modifying the production System after the delivery to correct the faults, improve the performance and adapt to the changing Environment is called as Software Maintenance.

Software Characteristics:

- ✓ Should be Complete: Meet all the requirements of Customers.
- ✓ Should be Reliable.
- ✓ Should be Accurate (precise).
- ✓ Should be Efficient.

Software Development:

- ➢ **Binary Instruction**
- ➢ **Development Tools**
- ➢ **Object Based Development**
- ➢ **Open Source Software**
- ➢ **Drag and Drop Customizations.**

- ❖ In-Built Software (Automobile, plane, Robot) reduce human participation.
- ❖ System-based Software (computes software, Driver Software).
- ❖ Application based software (service to the Requesting user).

Software Myths:

- ❖ Software is better than Implementing Device.
- ❖ Reusability Ensures better solution
- ❖ More Designers-more delay
- ❖ Additional features Add Efficiency.

Need of Software Engineering:

- ✓ It is the basic of any Software development
- ✓ It helps to produce reliable, reusable and Quality Software.
- ✓ Software Engineering understands the customer needs and development the Software.
- ✓ It helps to develop Software in efficient way.

Role of Management in Software Engineering:

Management of Software		
Participants	**Procedures**	**product**
Customer ,Team Members , Team Leads , Managers	Process ,Lifecycle , Final plan	Final Outcome

Software Process:

A set of interrelated actions and activities to achieve a predefined result is called as process.

1.Linear Sequential Model

2.Layered Technology

3.Prototyping Model

4.RAD Model

Linear Sequential Model

- ✓ The process of development is divided into sequence of Actions (steps) from Analysis, design, coding and testing, Implementation and finally Maintenance.
- ✓ It is also called predictive Life Cycle model and classical Life Cycle model.

Example

- ➢ Prototyping Model
- ➢ Rapid Application Development

Advantage

- ❖ Easy to Understand
- ❖ Easy to Implementation

Dis-Advantages:

It takes Longer time to finish the project because each step takes own time.

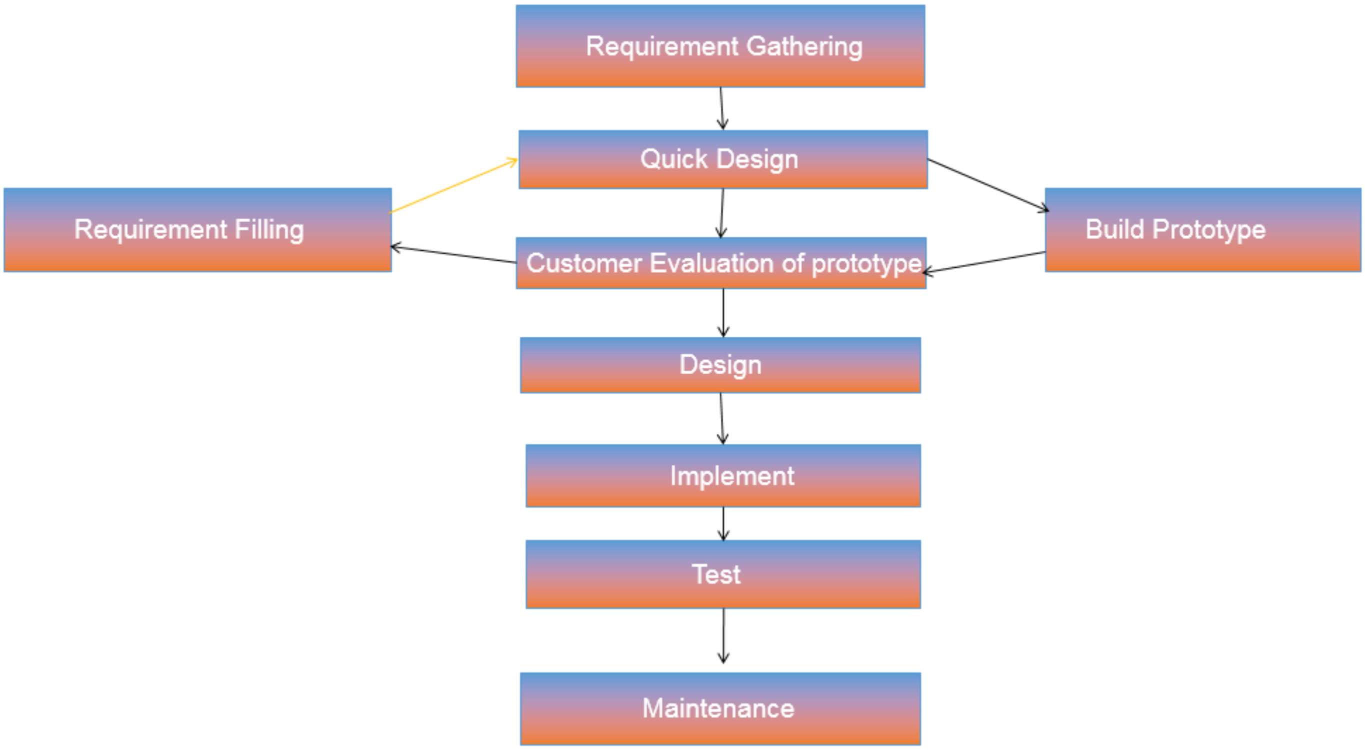
Layered Technology:

A Layered approach ensures a much stronger product,

- ✓ Quality Process
- ✓ Process
- ✓ Methods
- ✓ Tools

Prototyping Model:

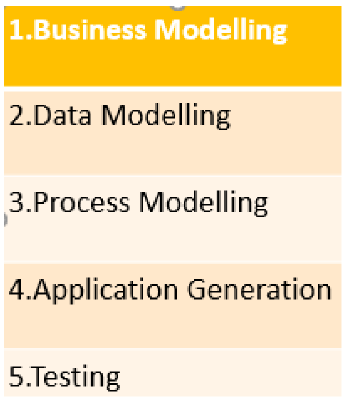
Sample will be created and displayed to the customer, until the customer satisfaction the prototype will be altered.



RAD Model:

Rapid Application Development Model includes the Prototype and iterative Designs.RAD Requires the 60 - 90 Days to deliver the Software Product.

RAD Model includes the following five phases for Development**.**



1.Business Modelling

2.Data Modelling

3.Process Modelling

4.Application Generation

5.Testing

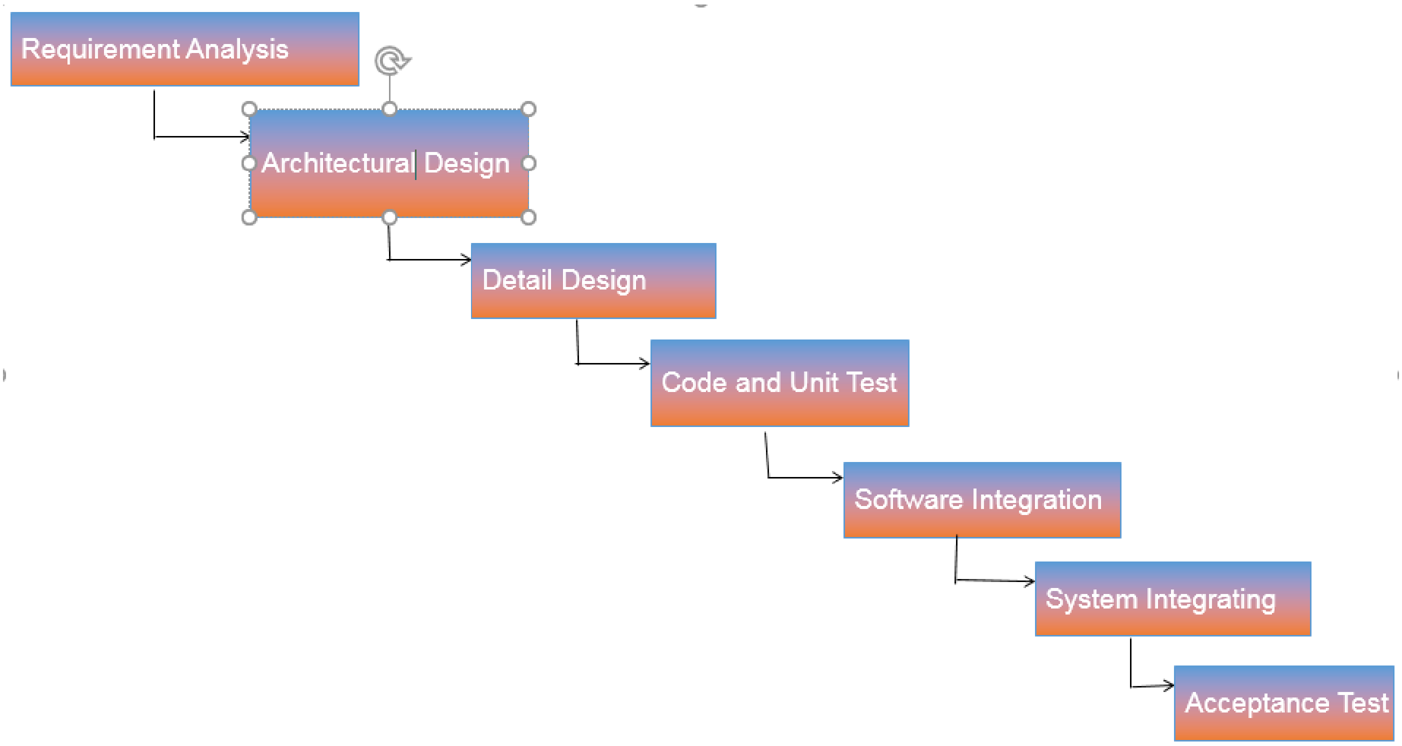
Software Process Models:

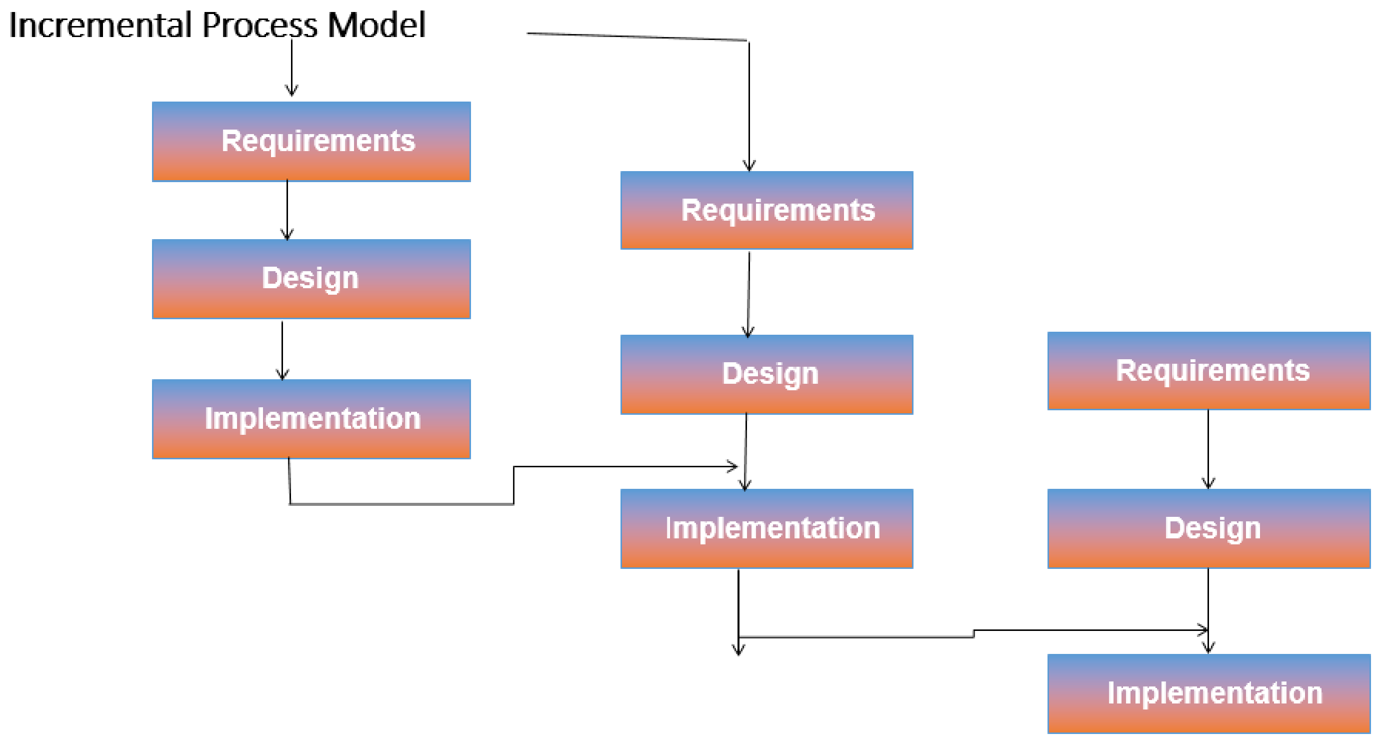
Software process models help the developer from to recognize the following:

1. Set of tasks, Sub tasks and Interface
2. Resources needed for every process
3. Adequate time Span

WATERFALL MODEL:

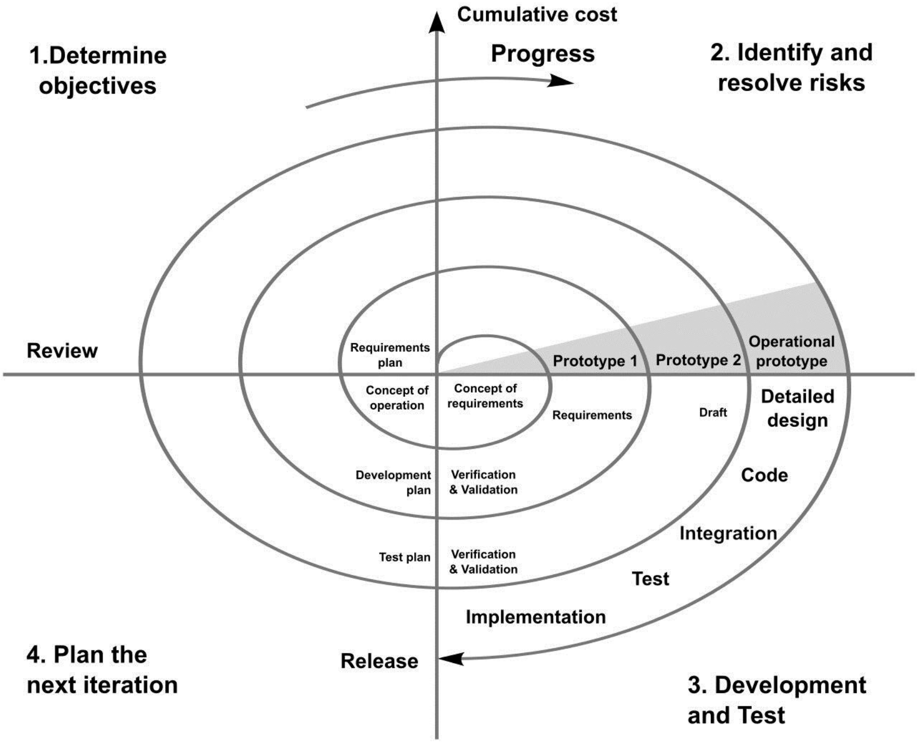
- ✓ All the Process owned one after another. It is devised by **Royee in 1970**
- ✓ All the process of the Waterfall Model ensure in Sequential Order. These process are documented and the Confirmed information are printed and stored for future References. This model is also called as **Document-driven Approach.**





- ❖ Multiple Streams being progressed independently and simultaneously. Separate terms can work individually to produce a product assigned. Finally Integrate that all.

SPIRAL MODEL:



- The Spiral Model was proposed by **Boehm,** Spiral model is also called as **Risk Spiral Model. This** model combines the Software development life cycle with Risk Management principle to control risks at early state of product.
- It is a Combination of **Prototype and Waterfall Model.**

ADVANTAGES:

- ✓ Risk Reduction
- ✓ Functionality can be added in the later phase.
- ✓ Software is produced early.
- ✓ It is good for large and critical project.

What is Prototype?

- Collect the information from the user to create requirement and it to show the user.
  - **Example :**
    - If a person take a sick leave we have faced Risk. So we alter a New person is Risk because He leave all the things is take so much time, so we Need to Add more than the Actual Strength is best.
- **Dis Advantages :**
  - It can be Costly to develop a Software model.
  - It is not used for Small project.

CHARECTERISTICS OF GOOD SOFTWARE:

- **Completeness :**
  - Developed Software products should cover all the stated and agreed requirements without missing out any.
- **Consistency :**
  - Operations of product should be stable and capable of handling the problems in implementation.
- **Durability :**
  - Customers may vary in technology requirement or geographical aspects and the product should respond to a diversity of environments.
- **Efficiency**
  - Product should not waste the resources.
- **Security**
  - Stored contents and the Input statements of Software should be Confidential.
- **Interoperability**
  - Communication Between other processes and Environments should be enabled to apply a product universally.

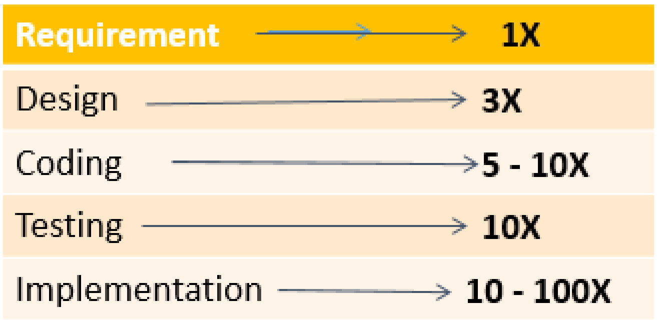
REQUIREMENT ENGINEERING PRINCIPLES:

- A Condition or capability needed by a user to solve a problem that must be met to Satisfy a standard, specification or other formally imposed documents.

Importance of Requirements:

- It Software projects get started without properly understanding the user's Needs, there will be misalignment at the end between the final result delivered and if Need to be lot of rework.

Cost of Correcting Error:



Requirement →	1X
Design →	3X
Coding →	5 - 10X
Testing →	10X
Implementation →	10 - 100X

Types of Requirements:

Requirements can be split into 3 types,

- Functional Requirements
- Non-Functional Requirements
- Interface Requirements

Functional Requirement:

- ✓ It defines what the system will do. These Requirements determines how the software will behave to meet user’s Needs.
- ✓ **For example,**
  - ✓ **Business logic, Data logic,**
  - ✓ **Data manipulation,**
  - ✓ **Processing logic**.

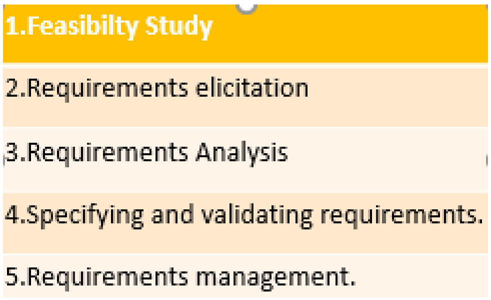
Non-Functional Requirement:

- ✓ As a End Users prospective include performance, Availability, usability and security and from a developer's point include reusability, testability, Maintainability and portability.
- ✓ **Example :**
  - ✓ The System should be Available 24X7(Availability).
  - ✓ The System should be save 1000 records in 2 second (performs).

Interface Specification:

- ❖ It tells the interaction requirement of one System to another.

Steps of Requirements Engineering:



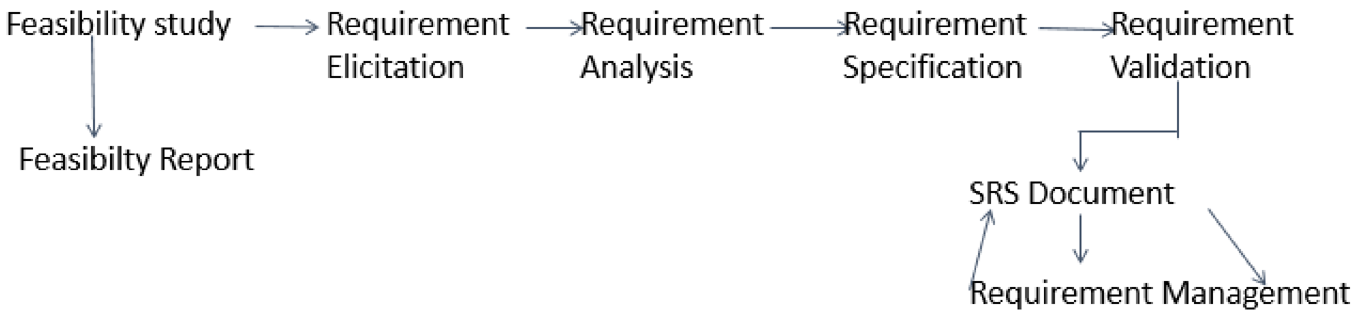
1.Feasibilty Study

2.Requirements elicitation

3.Requirements Analysis

4.Specifying and validating requirements.

5.Requirements management.



Feasibility Study:

- It is the step of Requirements Engineering process.
    - Operational Feasibility
    - Technical Feasibility
    - Economic Feasibility

Technical Feasibility:

- It is used to check the software, Hardware resources and software development platforms.

Economic Feasibility:

- To Calculate the cost for developing a System, including Software license, Hardware purchasing and Manpower Cost.

Requirements Elicitation:

- It is the Second step of Requirements Engineering Process.
- All the Requirements are identified and them by using these Sources and the user needs.
- The Stakeholder involved during this phase include End user, Managers, development and Testing Engineers, domain Expert and Business Analyst.

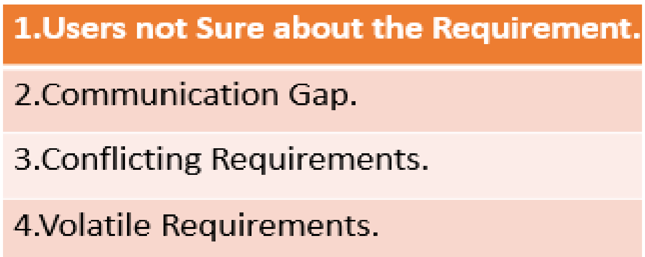
Identifying Stakeholders:

- At first find out to know the people who can contribute gather the requirements. If the not identifying the stakeholders we with do the rework at a later stage.
- To identify the stakeholder is the first step of Requirements Elicitation.

Characteristics of Stakeholders:

- If Stakeholders Positive approach only the project outcome is successful. Stakeholder's involvement is always situation-specific.

Problems in Eliciting Requirements:



1.Users not Sure about the Requirement.

2.Communication Gap.

3.Conflicting Requirements.

4.Volatile Requirements.

Techniques of Requirements Elicitation:

- Interviewing
- Focus Groups
- Facilitated Workshops
- Prototyping
- Questionaries'
- Brainstorming
- Direct Observation
- Apprenticing

- ✓ **Interviewing:**
    - Face- to -Face interactions with users through individual interviewing is the primary Source of requirements.
- ✓ **Focus Groups**
    - It is same as interviewing but has a Group of 6 to 10 people at a time. Let We have a discussing with Group of Members.
- ✓ **Types of Groups :**

1.Two-way focus Group.

2.Dual moderator focus Group Model 1.

3.Dual moderator focus Group Model 2.

4.Respondent moderator focus Group.

5.Client participant focus Group.

6.Mini focus Group.

7.TeleConference focus Group.

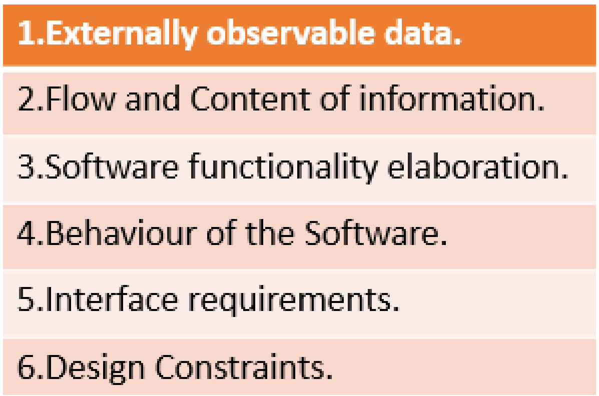
8.Online focus Group.

- ❖ **Two - Way Focus Group :**
    - ✓ Interaction happen between the two focus group as per the predefined rules and proper discipline.
- ❖ **Dual Moderator focus Group Model 1 :**
    - Two moderators do the job, one looks into the discipline and the other looks into the Content flow.
- ❖ **Dual moderator focus Group Model 2 :**
    - Moderator take completely opposite stands. if one says that something is possible , the other says why it is not possible.
- ❖ **Respondent moderator focus Group :**
    - Respondents are asked to act as the moderator temporarily.
- ❖ **Client Participant focus Group :**
    - Client responsibilities participate in the discussion.
- ❖ **Mini Focus Group :**
    - Group members contain only four (or) five only. It helps to avoid confusion and manage and Gather the requirements quickly.
- ❖ **Teleconference Focus Group :**
    - Telephone Network is used to facilitate the discussions.
- ❖ **Online Focus Group :**
    - Computers Connected via the Internet.
- ❖ **Facilitated Workshops :**
    - A Workshop is a Very Quick and best way to gather requirements, Compared with all other techniques.
    - Large Number of people involved in the Workshop.

- ❖ **Prototyping :**
  - ❖ This technique aims to get users to express the requirements. Prototyping is used to get feedback from users. It gives the initial version of the System.

Types of Prototype:

- ❖ **Proof - of - Principle Prototype :**
  - ✓ A few working codes are part of this Prototype.
- ❖ **Form Study Prototype :**
  - ✓ Only Visual appearance is considered and functionalities are not considered.
- ❖ **Visual Prototype :**
  - ✓ It focus about appearance, color, font and features.
- ❖ **Functional Prototype :**
  - ✓ It simulates the actual functionality of the work.
- ❖ **Questionaries' :**
  - ✓ It can be used to collect Input from multiple Stakeholders Quickly. These are finally considered as Requirements.
- ❖ **Brainstorming :**
  - ✓ It is used to collect a large number of requirements in a short period of time.
- ❖ **Direct Observation and Apprenticing :**
  - ✓ It is used to gather a Knowledge what is actually done by the user of the System.
- ❖ **Requirements Analysis :**
  - ✓ It is the third step of the requirements Engineering Process.
  - ✓ Analysis focus the following stages,



1.Externally observable data.

2.Flow and Content of information.

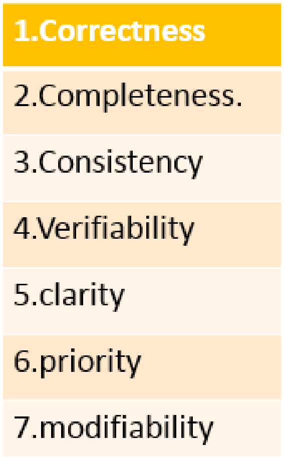
3.Software functionality elaboration.

4.Behaviour of the Software.

5.Interface requirements.

6.Design Constraints.

- ❖ **Specifying Requirements :**
  - ❖ It is used to specify all the requirements in a very clear, concise and unambiguous manner.
  - ❖ A Good SRS Document can be the following :



1.Correctness

2.Completeness.

3.Consistency

4.Verifiability

5.clarity

6.priority

7.modifiability

Validating Requirements:

- The main aim of requirements Validation is to ensure that the customer Needs are Complete, clear and Consistent.
- The different Stakeholders involved in Validation Process,
    - Customer
    - End User
    - Domain Expert
    - Architect
    - Test Engineer.

Requirements Management:

- Requirement Management Plan
- Requirement Identification.
- Requirements Change
- Requirements Status tracking
- Requirement Traceability.

----------- Complete ------------

UNIT – 2

Content of This Chapter

- Requirements Analysis Modeling
- Analysis Modeling Approaches
- Structured Analysis - Object Oriented Analysis
- Design and Architectural Engineering
- Design Process and Concepts
- Basic Issues in Software Design
- Characteristics of Good Design
- Software Design and Software Engineering
- Function Oriented System vs Object Oriented System
- Modularity, Cohesion, Coupling, Layering
- Real Time Software Design
- Design Models
- Design Documentation.

Requirement Analysis Modeling:

The Analysis Stage acts as the bridge between design and requirement and it happens after the requirement phase.

Aim of Requirement Analysis Stage:

- ✓ Clearly illustrate the user Scenarios
- ✓ Explain the functional activities in detail.
- ✓ Classify the design problems and their relationships.
- ✓ Define System behavior and class Structure.

STRUCTURED ANALYSIS:

The domain of Structured Analysis Contains three modeling approaches

They are,

- ✓ Data Modeling
- ✓ Functional or flow-Oriented Modeling
- ✓ Behavioral Modeling

1. Data Modeling:

- It is an Analysis technique that deals with the data Processing part of an Application.
- It is fully describes how the data transforms.
- The visual representation, which is called the Entity Relationship Diagram helps the Engineers Understand and design Data Elements.
- Some of the Notations to draw the E-R Diagram is ,
- **Data Object, Attributes and Relationships :**
  - Data Objects hold only the data Components and they should be no intent to hold the process that deal with the data inside Objects.
- **Cardinality and Modality :**
  - One-to-One Relationship
  - One-to-many Relationship
  - Many-to-Many Relationship

E-R Diagram:

- ✓ The Relationships of the Objects Present in a System are represented graphically through the ERDS.
- ✓ **Components of ERD :**
  - ✓ Data Objects
  - ✓ Attributes
  - ✓ Relationships
  - ✓ Various indicates describing the relationships.

Data Dictionary:

- ✓ Information all about the Data Objects is called Data dictionary.

2. Functional Modeling Or Flow - Oriented Modeling:

- ➢ It is the Second Model of structured Analysis. Functionality flow is also known as the Dataflow.
- ➢ **Data Flow Diagrams :**
  - ➢ It describes the Data flows between one Object to the other.DFD tells the relationships between the Various Components in a program or system.
- ➢ **Components of DFD is :**
  - ➢ Entities
  - ➢ Processes
  - ➢ Data Stores
  - ➢ Data Flows

Entities:

- ✓ An Entity provides data to the system is called a Source and an Entity which receives data from the system is called a Sink Entity represented by rectangle.

Process:

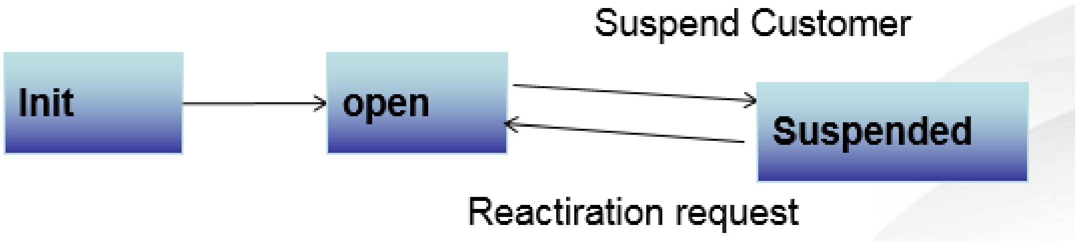
- ✓ A Process is the manipulation or work that transforms data while it flows through the system.

Data Store:

- ✓ At the time of Data Processing the process may store the Data intermittently for future use.

Data Flow:

- ✓ It describes the movement of data between the entities.



Behavioral Modeling:

- It is the third Step of Structural Analysis Approach.
- State Transition Diagram is one Example of Behavioral Modeling :

STATE TRANSITION DIAGRAM:

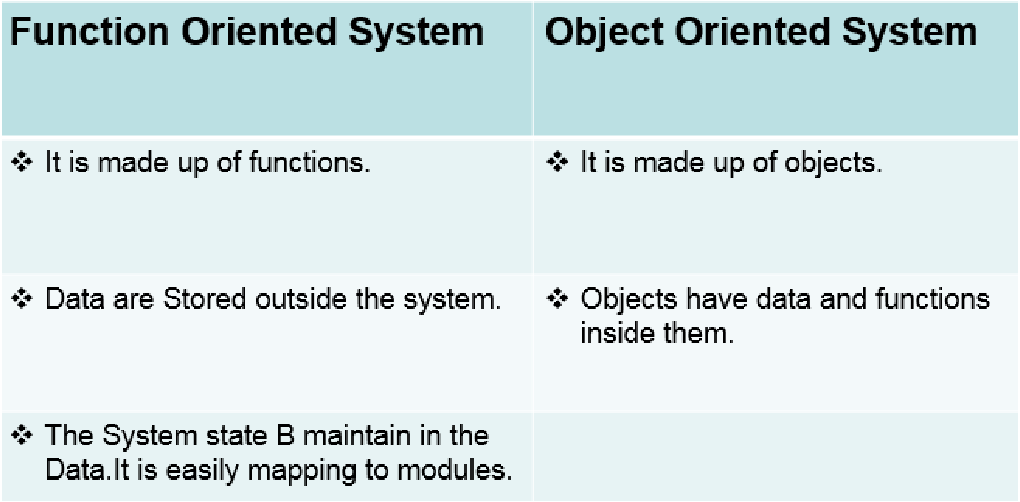
- State Transition Diagram represents the system states and other events that trigger state transitions or state change.
- A State can be an Observable mode of Behavior.
- State Transition Diagram are Created for objects which have significant dynamic behavior.

Design and Architectural Engineering:

- Each design product is verified and validated before moving to the next phase of Software development.
- Design Activities are subdivided into high-level design activities and Low-level design Activities.
- **Basic Issues in Software Design :**
    - Requirements model is wrongly represented then the design model also will go wrong.
    - Mapping each functional requirement into design representation is difficult.
    - Error function Handling is also a difficult part of design.
    - Balancing Clarity and Code safety is difficult.

Characteristics of a Good Design:

- ✓ Should be able to convert all the requirements into design.
- ✓ Should be easily understandable and maintainable.
- ✓ Should be easy to change the design.
- ✓ Should be easily scalable.



Function Oriented System	Object Oriented System
❖ It is made up of functions.	❖ It is made up of objects.
❖ Data are Stored outside the system.	❖ Objects have data and functions inside them.
❖ The System state B maintain in the Data.It is easily mapping to modules.	

Modularity:

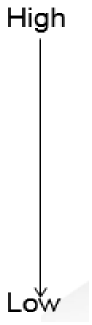
- ✓ The possibility of dividing a single project into smaller units called modules is termed modularity.
- ✓ It helps in reusability and maintainability.
- ✓ There are two main Characteristics :
  - ✓ Cohesion
  - ✓ Coupling
- ✓ **Cohesion :**
  - ✓ It refers to "How Strongly" one module is related to the other, which helps to Group Similar items together.

There are several types of Cohesion

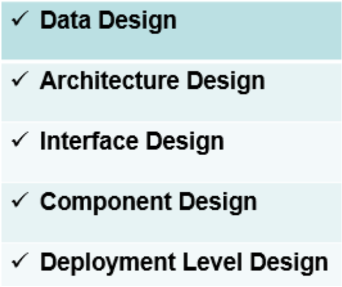
- ❖ Coincidental Cohesion
- ❖ Logical Cohesion
- ❖ Temporal Cohesion
- ❖ Procedural Cohesion
- ❖ Communicational Cohesion
- ❖ Sequential Cohesion
- ❖ Functional Cohesion
- ❖ **Coincidental Cohesion :**
  - ❖ Elements or Modules are grouped together for different purposes.
- ❖ **Logical Cohesion :**
  - ❖ Elements or Modules are grouped together based on Common logic.
- ❖ **Temporal Cohesion :**
  - ❖ Temporal Cohesion (Grouping) Occurs during run time at a particular time of program Execution.
- ❖ **Procedural Cohesion :**
  - ❖ It Occurs on Certain Procedure.
- ❖ **Communicational Cohesion :**
  - ❖ It occurs modules access a Common Database.
- ❖ **Sequential Cohesion :**
  - ❖ Output of the one module is Input of another modules.
- ❖ **Functional Cohesion :**
  - ❖ It occurs as all the functions contribute to a single task of the module.

Coupling:

- ✓ Coupling measure the Strength of the relationships between Entities and Modules.
- ✓ It is very easy to measure coupling Both Quality and Quality of Code.
- ✓ They are ,
  - ✓ Content Coupling
  - ✓ Common Coupling
  - ✓ Control Coupling
  - ✓ Stamp Coupling
  - ✓ Data Coupling
  - ✓ Uncoupled Coupling



- ✓ **Design Models :**

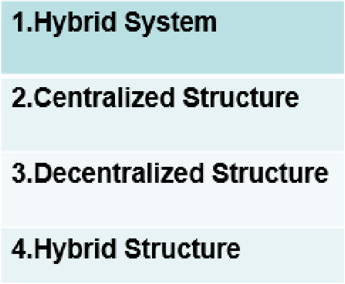


- ✓ Data Design
- ✓ Architecture Design
- ✓ Interface Design
- ✓ Component Design
- ✓ Deployment Level Design

❖ **Architectural Design :**

- ❖ Architectural design is derived from requirement Analysis and it is also based on already available Architectural patterns and styles.

❖ **Distributed System Architecture :**



1.Hybrid System

2.Centralized Structure

3.Decentralized Structure

4.Hybrid Structure

❖ **Interface Design :**

- The DFD helps both Architecture and User Interface Design.
- It Creates the Communication path between the System and User.

❖ **Component Design :**

- It is usually done after User Interface Design.

❖ **Deployment -Level Design :**

- The Deployment-Level is explain with Diagrams.

UNIT 3

Concept of this Chapter

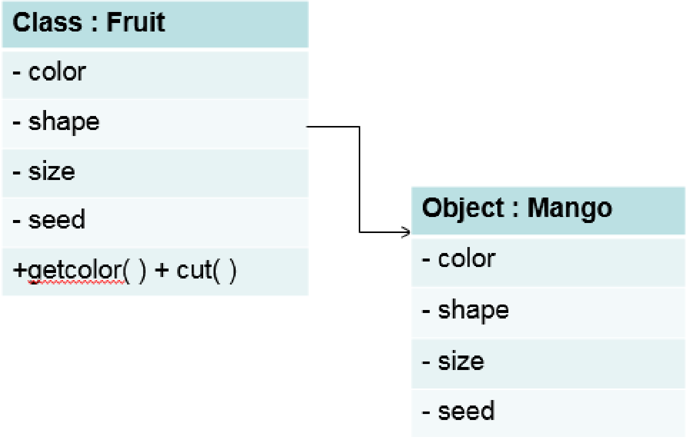
- Object Oriented Concepts
- Fundamental Parts of Object Oriented Approach
- Data Hiding and Class Hierarchy Creation
- Relationships
- Role of UML in OO Design
- Design Patterns
- Frameworks
- Object Oriented Analysis
- Object Oriented Design
- User Interface Design
- Concepts of User Interface
- Elements of User Interface
- Designing the User Interface
- User Interface Evaluation
- Golden Rules of User Interface Design
- User Interface Models
- Usability.

Object Oriented Concepts:

- ✓ Design principles can be categorized in Function - Oriented design and Object Oriented Design.

Functional Parts of Object - Oriented Approach:

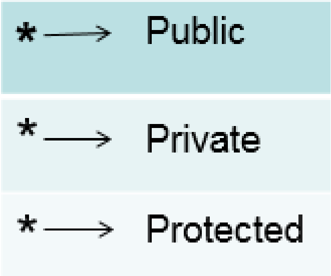
- **Object :**
  - Everything in this Universe can be considered as an Object.
    - Example :
      - **a fruit - Mango**
        - Mango is a member of large group of Objects called fruit.
        - Fruit is a class and Mango is a Object.
- **Class :**
  - Collection of Objects that form a Common name and behavior is called **Class.** The Class Contains the Data and the processes. That are capable of manipulating the Data.



- **Attributes :**
    - Characteristics that represent the state of an Object are called Attributes. The Attribute may be of simple data type such as Numeric, text and Binary.
- **Methods :**
    - The method implements the behavior of an Object. The Collection of method that exhibits what the Object is going to function is called Behavior.
- **Messages :**
    - The Objects interact with each other through messages .The messages act as the Stimulus to involve the methods within a class.

Data Hiding and Class Hierarchy Creation:

- **Encapsulation :**
    - It refers to protecting of Data from the users. Users need not know what is happening inside the Object.

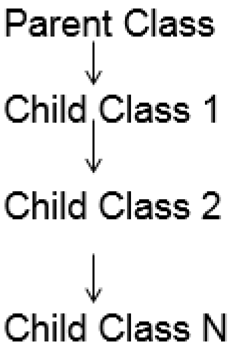


* → Public

* → Private

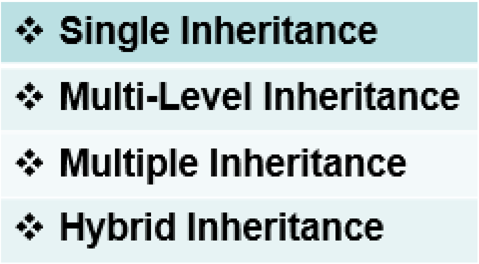
* → Protected

- **Inheritance :**

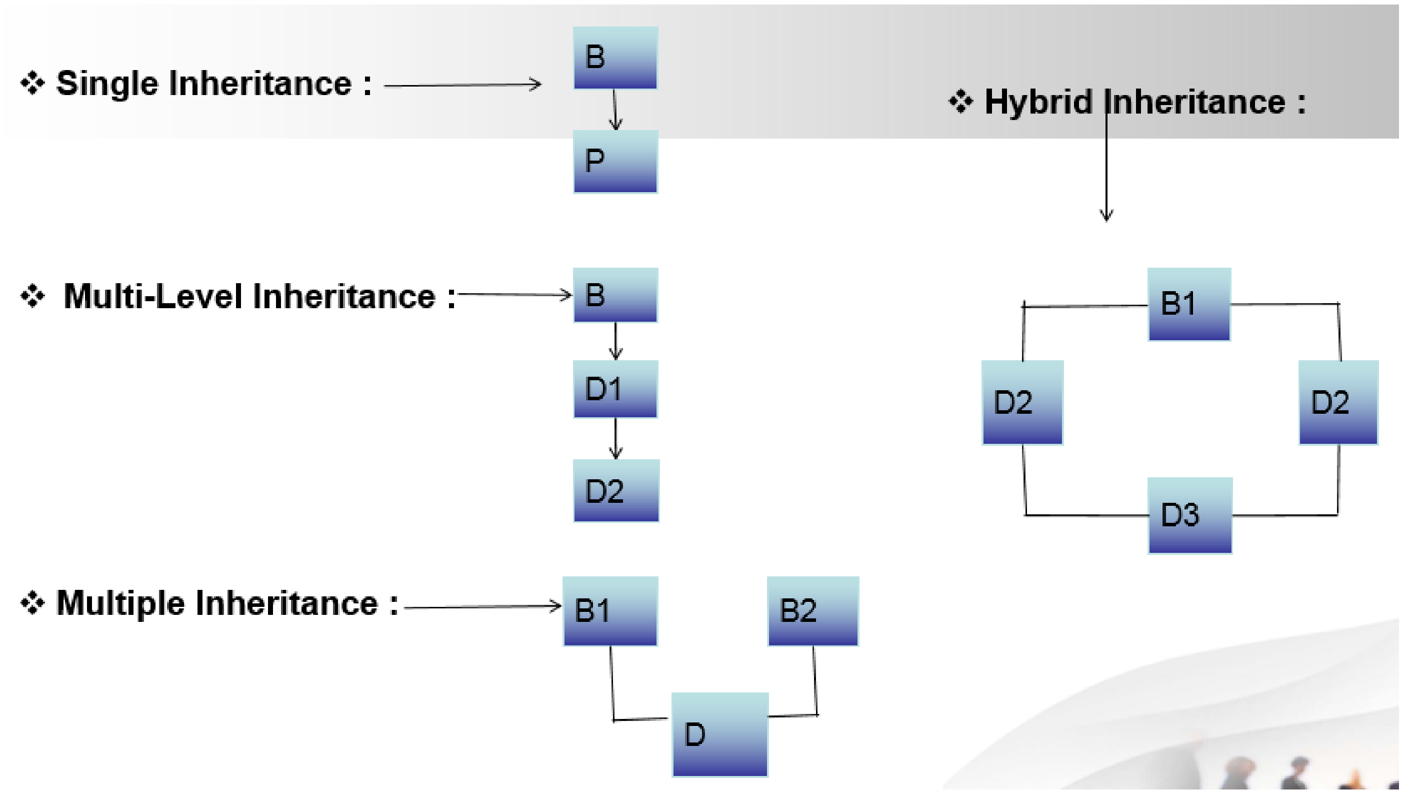


✓ Process of deriving a new class from an Existing class. It provides reusability and it helps to understand the problem clearly.

- **Types of Inheritance :**



- Single Inheritance
- Multi-Level Inheritance
- Multiple Inheritance
- Hybrid Inheritance



❖ Single Inheritance :

❖ Hybrid Inheritance :

❖ Multi-Level Inheritance :

❖ Multiple Inheritance :

- **Polymorphism :**
  - Object that can take different form is termed as Polymorphism.
  - Types of Polymorphism :
    - Types of Polymorphism :
    - Compile time Polymorphism
- **Abstraction :**
  - Only necessary information is exposed to other Objects. Abstraction Concepts Supports Inheritance and Polymorphism.
- **Relationships :**
  - Relationships between Objects can be denoted in three forms.

1. Is - a Relationship

2. Has - a Relationship

3. Uses - a Relationship

- **!. Is - A Relationship :**
  - It describes the Inheritance relationship between the classes.
  - Example: A Car is a Vehicle.
    - Object car has all the Characteristics of the General Object Vehicle.

- **2. Has - A Relationship :**
  - Aggregation is also called as the “has - a” Relationship. When a class has another Object as the data member or Attribute then it is said to be of “has - a” Relation.
  - Example :
    - A Car has a Wheel.
- **3. Uses - A Relationship :**
  - The method of One Class may use the Object of another Class.
  - Example :
    - A Car uses petrol for functioning. But, petrol is an Object of the Class fuel.
- **Role of Unified Modeling Language (UML) in OO Design :**
  - The Objects are real - World entities that exist around us and the Object .or basic concepts such as Abstraction, Encapsulation, Inheritance, and Polymorphism all can be represented easily using UML.
- **Design Patterns :**
  - General reusable Solution to more frequently / commonly Occurring Software design Problems is called as Design Patterns.
  - Design Patterns has four Essential Parts :
    - Pattern Name
    - Problem
    - Solution
    - Consequences
- **Pattern Name :**
  - Each Pattern should have a meaningful Name. Which describes in a word or to the Problem, domain it addresses and the Solution.
- **Problem :**
  - This explains the Problem, and one Context So that One Designer is able to understand when to apply this Design Patterns.
- **Solution :**
- It contains a description of how a General arrangement of classes solves the design issues.
- **Consequences :**
  - There are three types of Design Patterns :
    - Creational Patterns
    - Structural Patterns
    - Behavioral Patterns

1. Creational Patterns:

- ✓ This group of Patterns helps in abstracting the Object instantiation Process. Unless the system is made independent of how the Objects are created and composed. These Patterns use Inheritance to group the Small repeatable behavior of Objects to create logically related Classes.

2. Structural Patterns:

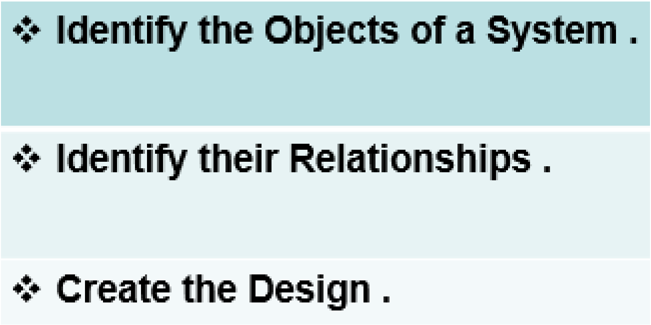
- ✓ These Patterns describe how to compose the Classes and Objects to build larger Structures.

3. Behavioral Patterns:

- ✓ This group of Patterns abstracts in Communication between the Objects. The Object Composition is used to build these Patterns and describe the Cooperation of a group of Objects, to perform a task not to alone.

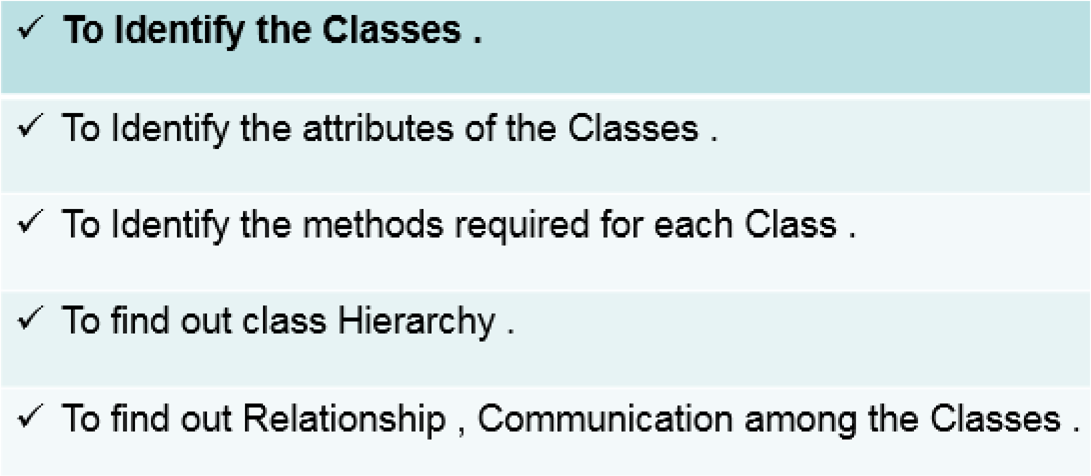
Frameworks:

- ✓ A more Specific Solution description for design problems is found in the form of frame work. A Single framework many consist of multiple design Patterns.
- ✓ **Object Oriented Analysis and Design :**
  - ✓ **Purpose of OOAD :**



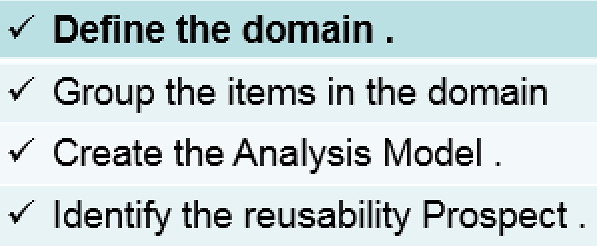
- ❖ Identify the Objects of a System .
- ❖ Identify their Relationships .
- ❖ Create the Design .

- ➢ **Objections of OOA :**



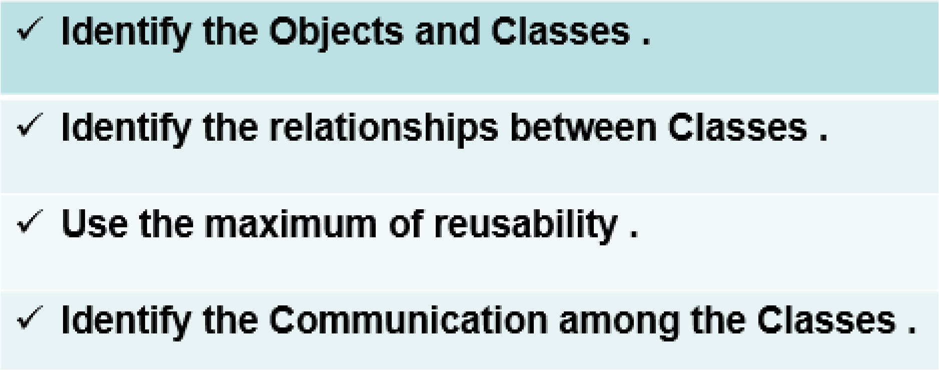
- ✓ To Identify the Classes .
- ✓ To Identify the attributes of the Classes .
- ✓ To Identify the methods required for each Class .
- ✓ To find out class Hierarchy .
- ✓ To find out Relationship , Communication among the Classes .

- **Domain Analysis :**



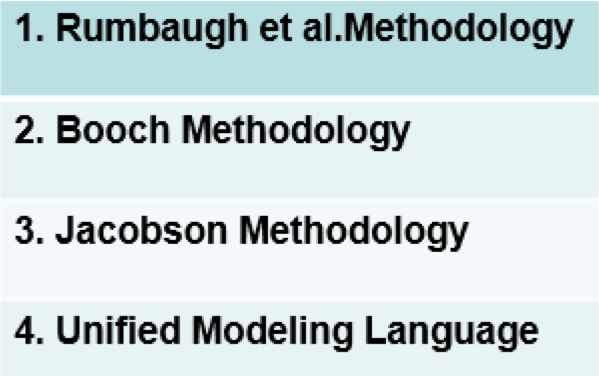
- ✓ Define the domain .
- ✓ Group the items in the domain
- ✓ Create the Analysis Model .
- ✓ Identify the reusability Prospect .

- **Use Case Modeling :**
  - The use Cases are pictorial representation in order to capture the System requirements clearly.
- **Object Oriented Design :**
  - After completing the analysis of the requirement, the next step is to translate it to the design of the proposed system.
- **Importance of OO Design :**



- ✓ Identify the Objects and Classes .
- ✓ Identify the relationships between Classes .
- ✓ Use the maximum of reusability .
- ✓ Identify the Communication among the Classes .

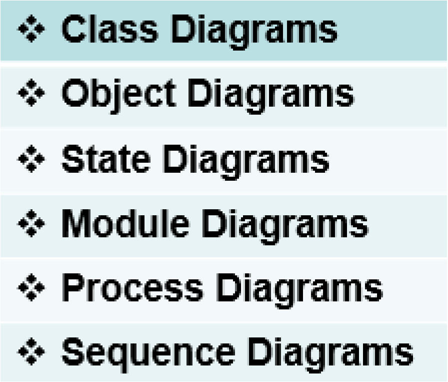
- **OOD Modeling Techniques :**
  - ✓ Modeling techniques provide a set of tools and guidelines which help to analysis and Designers in performing robust requirement analysis and design.
  - ✓ Some of the Powerful Modeling methodologies are :



1. Rumbaugh et al.Methodology
2. Booch Methodology
3. Jacobson Methodology
4. Unified Modeling Language

1. Rum Baugh et al. Methodology:

- ✓ Four Phases of OMT have to be carried out repeatedly to develop a useful object model for a System.



- ❖ Class Diagrams
- ❖ Object Diagrams
- ❖ State Diagrams
- ❖ Module Diagrams
- ❖ Process Diagrams
- ❖ Sequence Diagrams

2. Brooch Methodology:

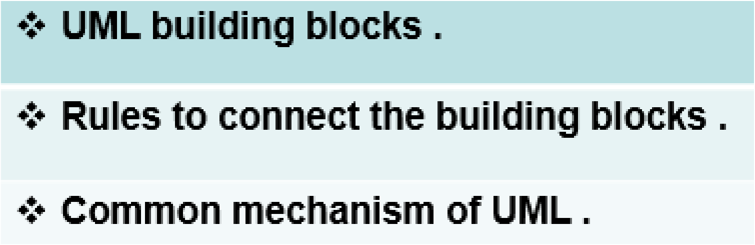
- ✓ This method consists of following diagrams:

3. Jacobson Model:

- ✓ Jacobson's Model include the Methodologies such as Object-Oriented Business Engineering (OOBE), Object-Oriented Software Engineering (OOSE) for Software development.

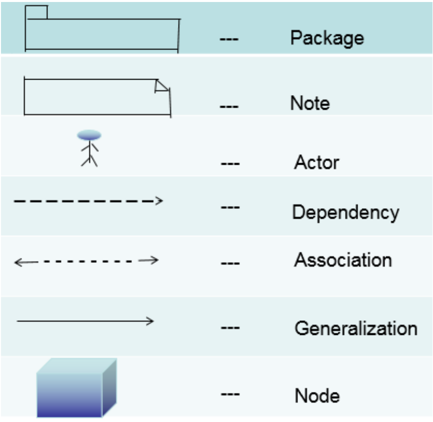
4. Unified Modeling Language:

- ✓ UML uses a set of Static and Dynamic Visual models to capture the Structure as well as behavior of the system. UML is used to develop all possible system in a simple way.
- ✓ **Major Elements of UML :**



- ❖ UML building blocks .
- ❖ Rules to connect the building blocks .
- ❖ Common mechanism of UML .

- ❑ **Common UML Notations :**



□	---	Package
□	---	Note
웃	---	Actor
- - - - - - - - - ->	---	Dependency
<- - - - - - - - - ->	---	Association
———————>	---	Generalization
■	---	Node

- ✓ **Relationship :**
  - ✓ In UML we provide the relationship among elements. Different types of relationships in UML are :

    1) Dependency

    2) Association

    3) Generalization

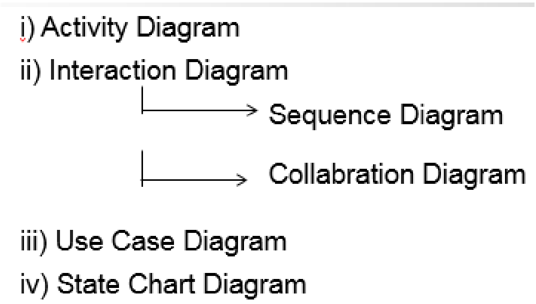
    4) Extensibility
- ✓ **Dependency :**
  - ✓ It describes the dependent elements and the direction of dependency. Represented by dotted Arrow.
- ✓ **Association :**
  - ✓ It describes number of elements taking part in an interaction.
- ✓ **Generalization :**
  - It represents the Parent-child relationship.
  - UML plays on important role in defining different perspective of a System.
    - o Design
    - o Implementation
    - o Process
    - o Deployment
- ✓ **Types of UML Diagram :**

1. Structural Diagram:

i) Class Diagram

ii) Object Diagram

2. Behavioral Diagram:



i) Activity Diagram
ii) Interaction Diagram
→ Sequence Diagram
→ Collabration Diagram
iii) Use Case Diagram
iv) State Chart Diagram

3. Implementation Diagram:

i) Component Diagram

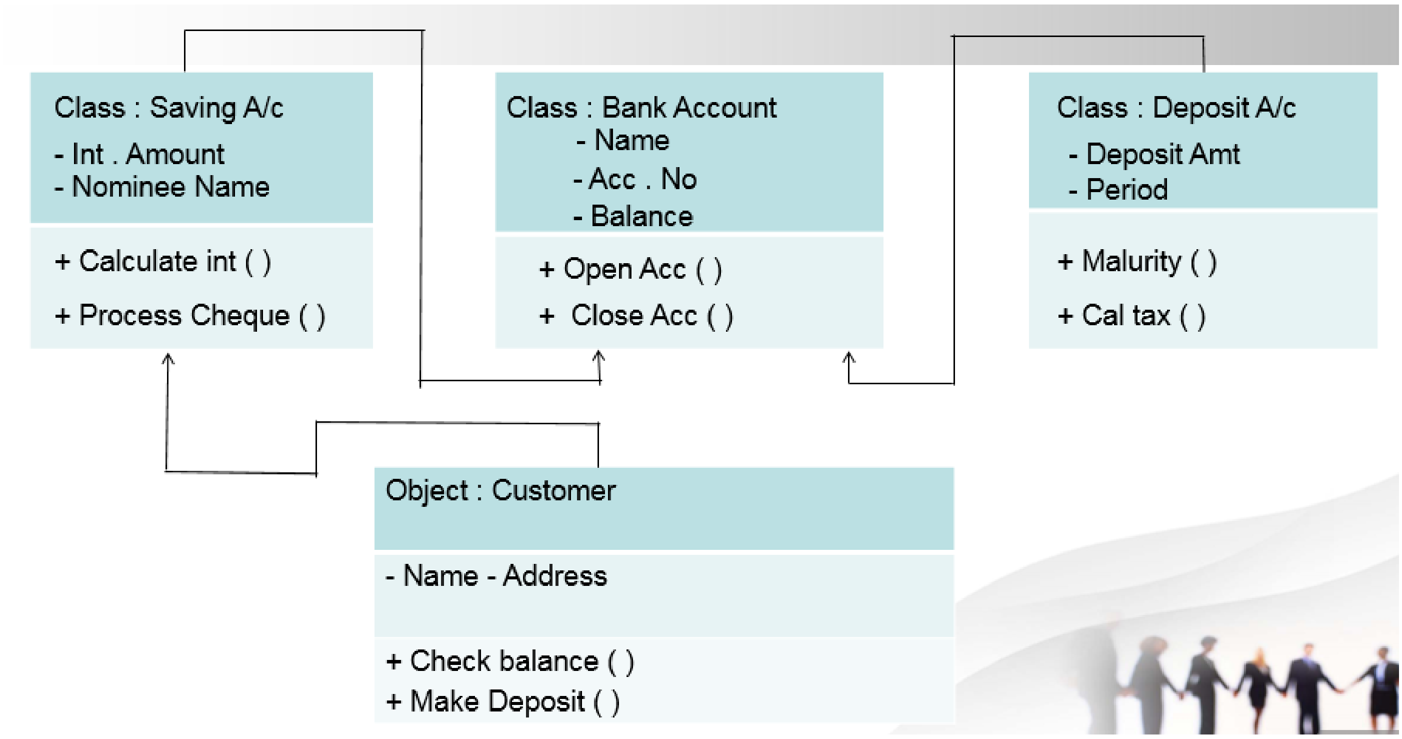
ii) Deployment Diagram

4. Architectural Diagram:

i) Package Diagram.

- **Structural Diagram :**
  - It describe different Components of a System that interact with each other to generate the System's behavior.
- **Class Diagram :**
  - Components of a Class Diagram are :

- **Collection of Classes**
- **Interfaces**
- **Associations**
- **Collaborations**
- **Constraints .**



Class : Saving A/c
- Int . Amount
- Nominee Name

+ Calculate int ( )
+ Process Cheque ( )

Class : Bank Account
- Name
- Acc . No
- Balance

+ Open Acc ( )
+ Close Acc ( )

Class : Deposit A/c
- Deposit Amt
- Period

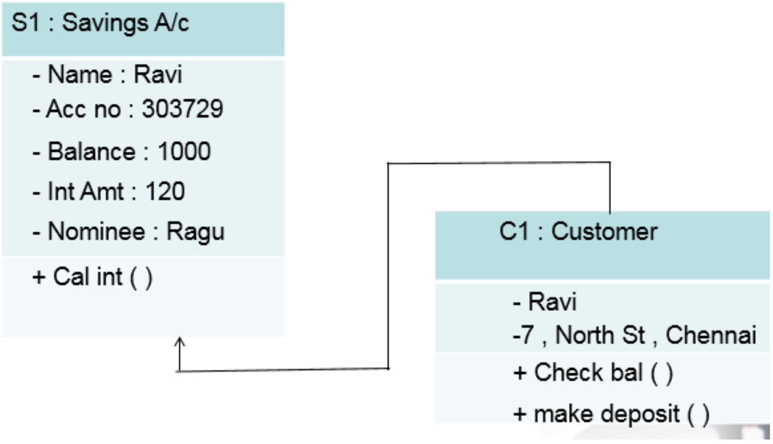
+ Malurity ( )
+ Cal tax ( )

Object : Customer

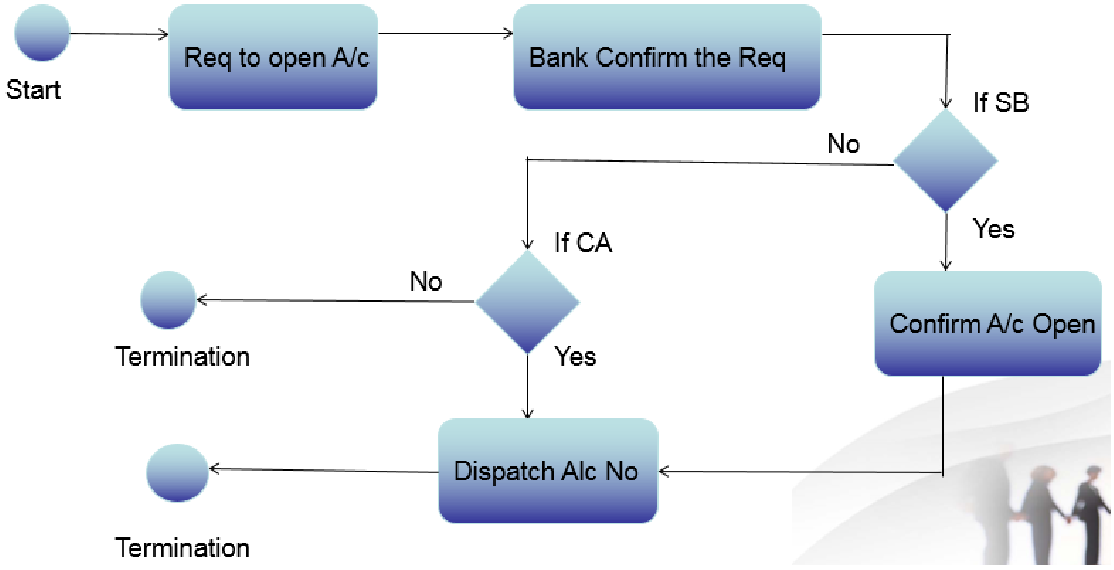
- Name - Address

+ Check balance ( )
+ Make Deposit ( )

- **Object Diagram :**
  - It is a closely related to Class Diagram. An Object Diagram can be interpreted as the
  - Instance of a Class Diagram.



- **Activity Diagram :**
  - Activity Diagram shows the Dynamic behavior of the System.



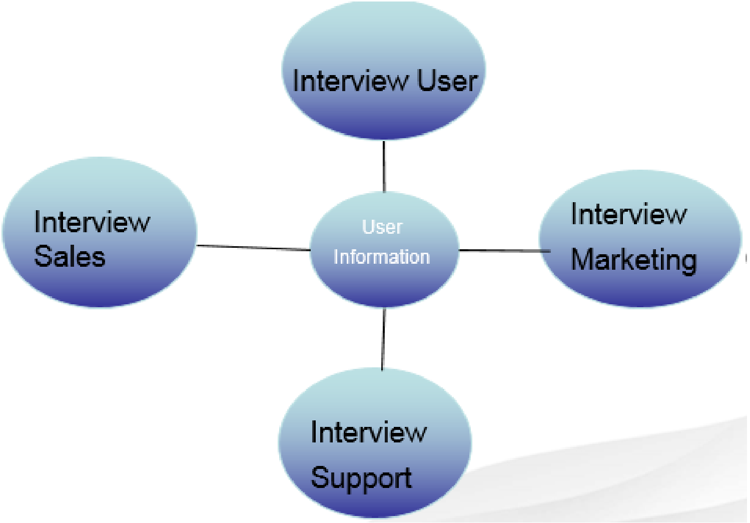
- **Sequence Diagram :**
  - It describes the interaction among the object in a System in a time Sequence.
- **Collaboration Diagram :**
  - It focuses more on showing the structure of the interaction rather than the sequence of interactions.
- **Use Case Diagram :**
  - The Use Case diagram are used to represent the interaction of the System Components with the users are the external system.
- **State Chart Diagram :**
  - The dynamic behavior of a system and effect of external and internal stimuli can be effectively represented using the State Chart Diagram.

- **Implementation Diagram :**
    - It give details of the system that need to be considered while deploying the system.
- **Architectural Diagram :**
    - It represents the pack of components and interaction among different object package.
- **User Interface Design** :
    - First look into the User Interface (UI) should impress the end users.
- **Components of UI :**
    - End Users
    - User Interface
    - System
    - A UI helps the users to interact with a system.

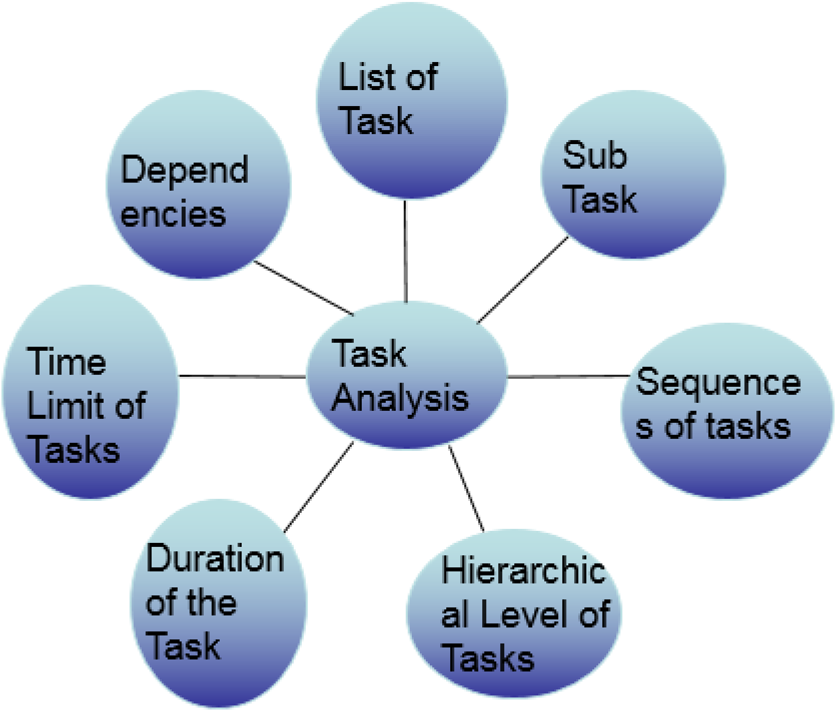
Elements of the UI:

1. Users
2. Tasks
3. Contents
4. Environment

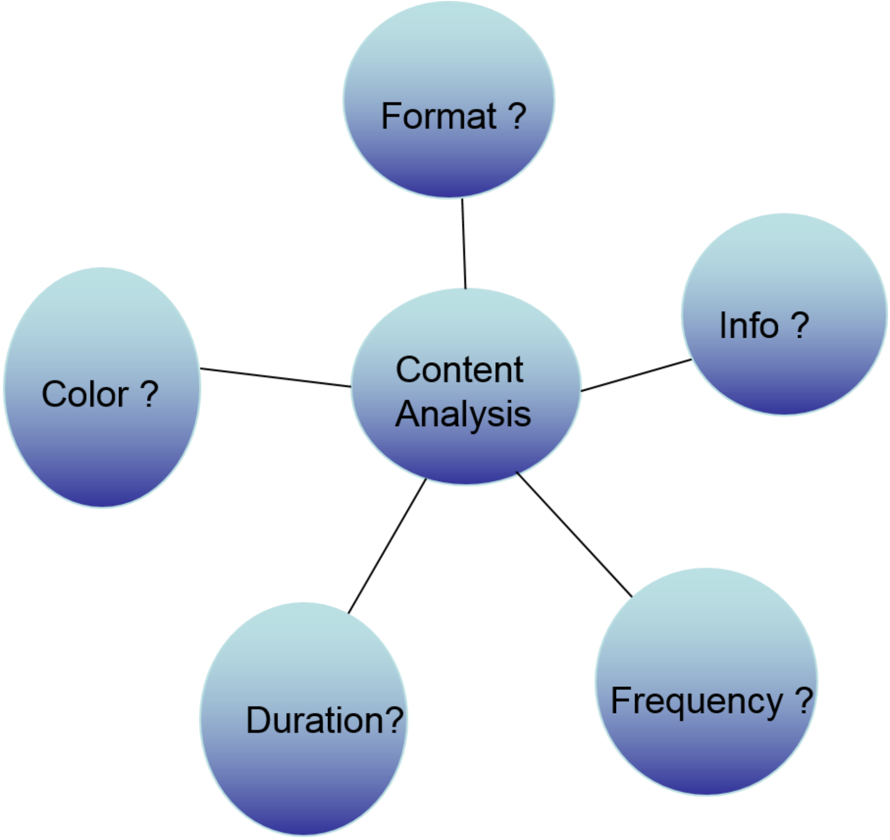
User Analysis:



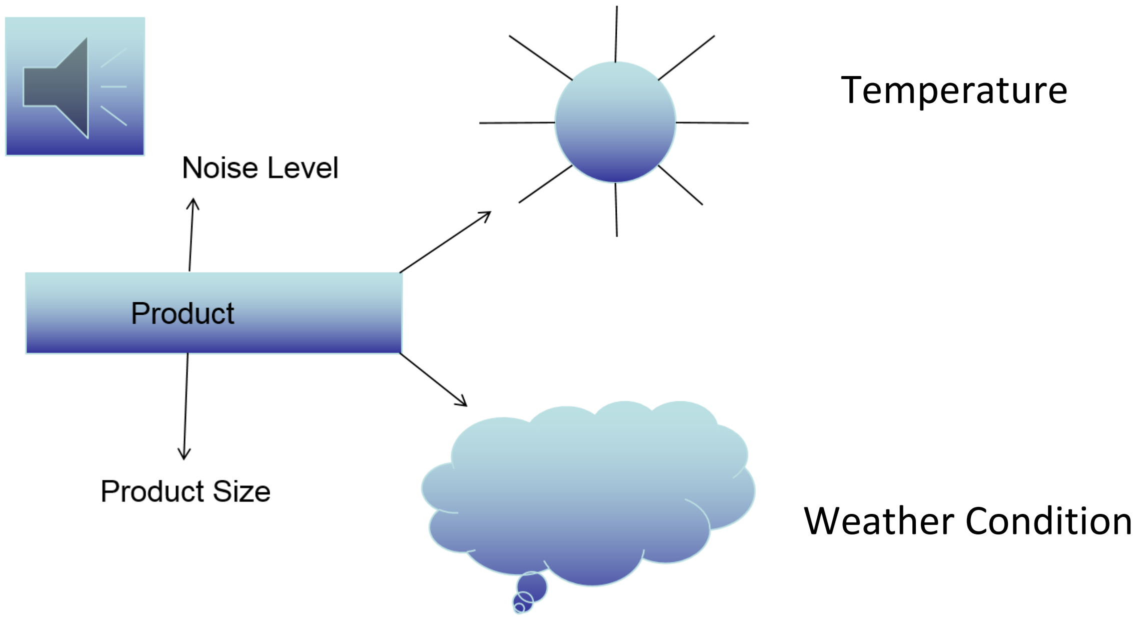
Task Analysis:



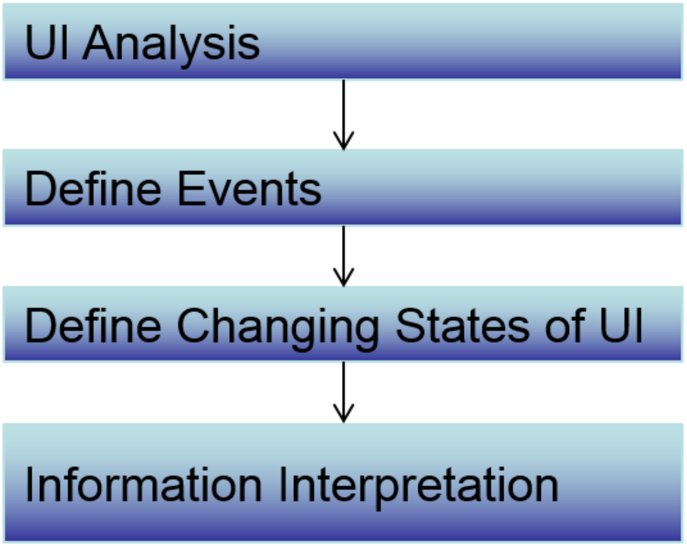
Content Analysis:



Environment Analysis:



Designing the user Interfaces:


UI Analysis

Define Events

Define Changing States of UI

Information Interpretation

- **Golden Rules of User Interface Design :**
  - Place Users in Control
  - Reduce Users Memory Load
  - Make the interface Consistent.
- **User Interface Models :**

✓ **User Profile Model**

✓ Design Model

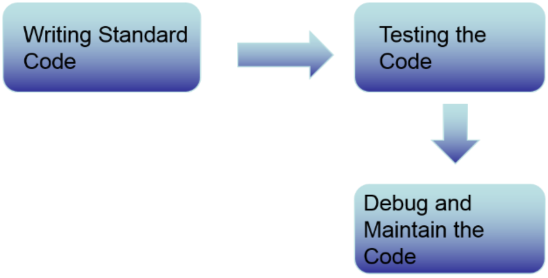
✓ Implementation Model

✓ Users Mental Model

UNIT 4

SOFTWARE CODING:

- "Conversion of detailed design Specification into the actual Software Code is called as Coding".



Writing Standard Code

Testing the Code

Debug and Maintain the Code

- **General Programming Principles :**

1. **Validity (Prg , Give the Correct Result is called Validity)**
2. Consistency
3. Maintainability (Prg , must be easlity Changeability)
4. Readability
5. Usability (Usable for the Specific Purpose) without any trouble .

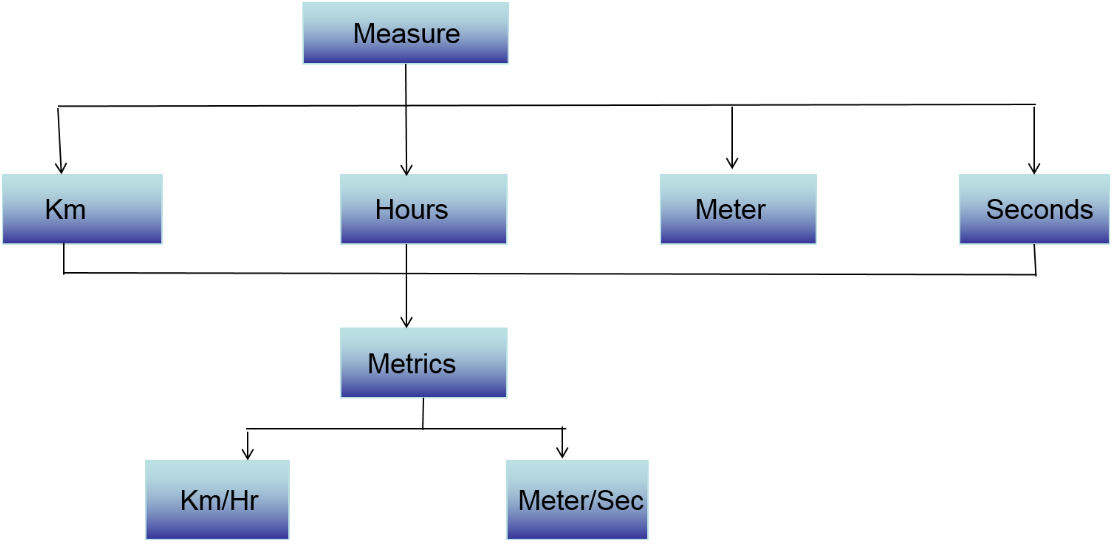
- **Key Concepts in Software Coding :**

  - **Structured Programming**
  - Information Hiding
  - Coding Standards and Guidelines
  - Top - Down and Bottom - up Coding
  - Iterative Life Cycle model
  - Code Sharing
  - Code Review

Software Measurement and Metrics:

- Software Measurement is a quantities attribute of a Characteristic of a Software product.
- Example :
  - Lock (No of Lines of Code)
- **Types of Measure :**
  - Direct Measure (length of Source Code)
  - In-Direct Measure (Calculated from other direct and indirect Measurements).
  - True Performance Measure (users use easily with their own language).
  - Substitute Performance Measures.

- **Metrics :**
  - A Metric indicates the nature and / or strength of an attribute usually by mems of a number.

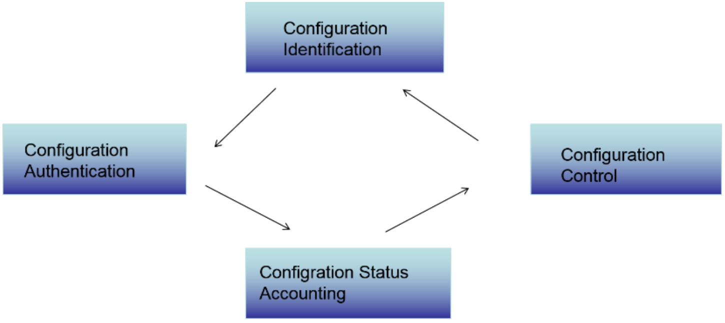


- **Black Box and White Box Metrics :**
  - If the Inside and Outside logic of a metrics is not known then it is **Black Box Metrics.**
  - If the Inside and Outside logic of a metrics is clearly known then it is called as **White Box Metrics.**

- **Types of Software Metrics :**
  - Product Metric
  - Process Metric
  - Resources Metric
  - External Metric
  - Internal Metric
  - Direct Metric
  - Indirect Metric

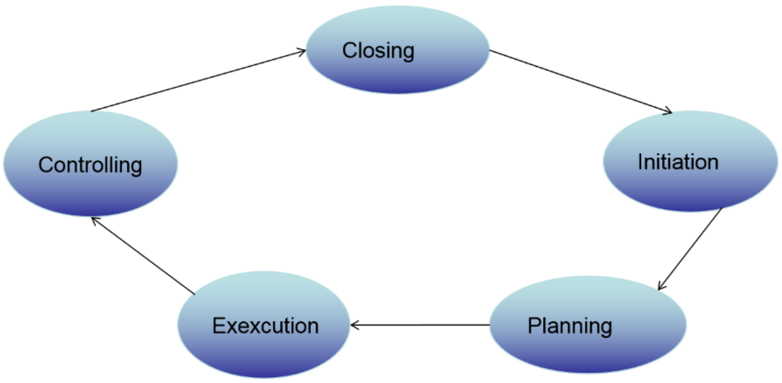
Software Configuration:

- The traditional SCM identifies four Procedures

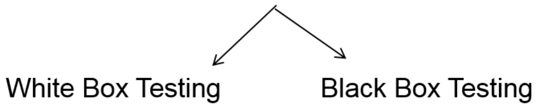

Configuration Identification
Configuration Authentication
Configuration Control
Configration Status Accounting

Project Management:

- ✓ **Process :**
  - A Process is a set of interrelated activities performed to create prespecified result.
- ✓ **Project Management Lifecycle :**



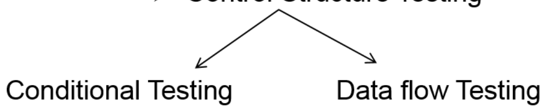
- ❖ **Software Testing :**
  - ❖ Application Software Testing :
    - ▪ Example :
      - Banking , Insurance , Payroll
  - ❖ System Software Testing :
    - ▪ Example :
      - OS , Compiler , Interpreter
- ❖ **Verification Testing :**
  - ❖ It is to prevent bugs and is used for acceptance of product. Verification happens internally and Validation is done by the customer.
- ❖ **Types of Software Testing** :


White Box Testing
Black Box Testing

- ✓ **White Box Testing :**

It Covers the ,

- ➢ Code Coverage
- ➢ Control Structure Testing


Conditional Testing
Data flow Testing

- ➢ Loop Testing

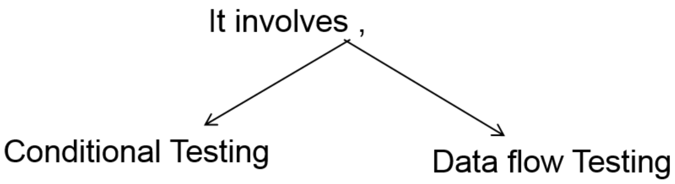
- □ **White Box Testing Techniques :**

  1. Code Coverage
  2. Control Structure Testing
  3. Loop Testing

1. Code Coverage:

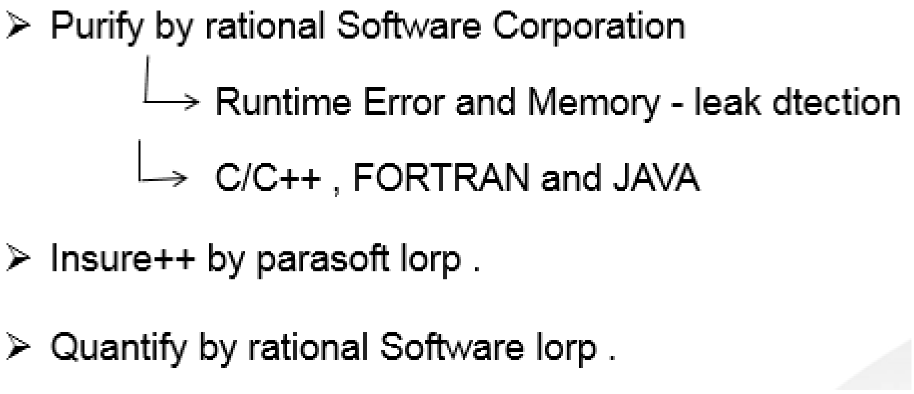
- ❖ Each Segment of code Control Structure is executed at least once in code Coverage. Each Branch in the Software code is taken in each possible direction at least once.

2. Control Structure Testing:



3. Loop Testing:

- ✓ Simple Loop
- ✓ Nested Loop
- ✓ Concatenated Loop
- ✓ Unstructured Loop
- ✓ **White-Box Testing Tools :**



- ➢ Purify by rational Software Corporation
  - → Runtime Error and Memory - leak dtection
  - → C/C++ , FORTRAN and JAVA
- ➢ Insure++ by parasoft lorp .
- ➢ Quantify by rational Software lorp .

- ✓ **Black-Box Testing :**
  - ○ Testing the App, bib the external or end user by Validating the Output.
  - ○ They are not mandatory to know about the Mechanism such as program.
  - ○ The tester has no access to the Source code.
- ✓ **Techniques of Black Box Testing :**

  i) Equivalence Partitioning

  ii) Boundary value Analysis

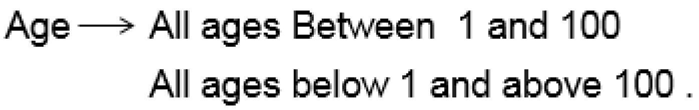
  iii) Decision Tables

i) Equivalence Partitioning:

- Equivalence Portioning is a technique used to reduce the number of test case to a manageable size.
- Example :
  - If we check the input value from 0 up to 10,000 then, we split it to four class's .Positive int, Negative into, Zero, and functions.

ii) Boundary value Analysis:

- This technique is a logical next step. after identifying the equalance Classes
- Example :

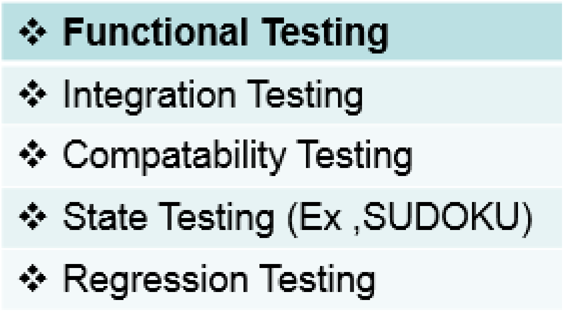


Age → All ages Between 1 and 100

All ages below 1 and above 100 .

iii) Decision Tables:

- Decision tables document complex business Roles that a System must implement.
- To identify and list all possible "Conditions" (Input) and all possible "Actions" (Output).
- **Black Box Testing Errors :**
  - Incorrect Or missing Functions
  - Interface Errors
  - Errors in Data Structure
  - Initialization and Termination Errors
- **Types of Black-Box Testing :**



❖ **Functional Testing**

❖ Integration Testing

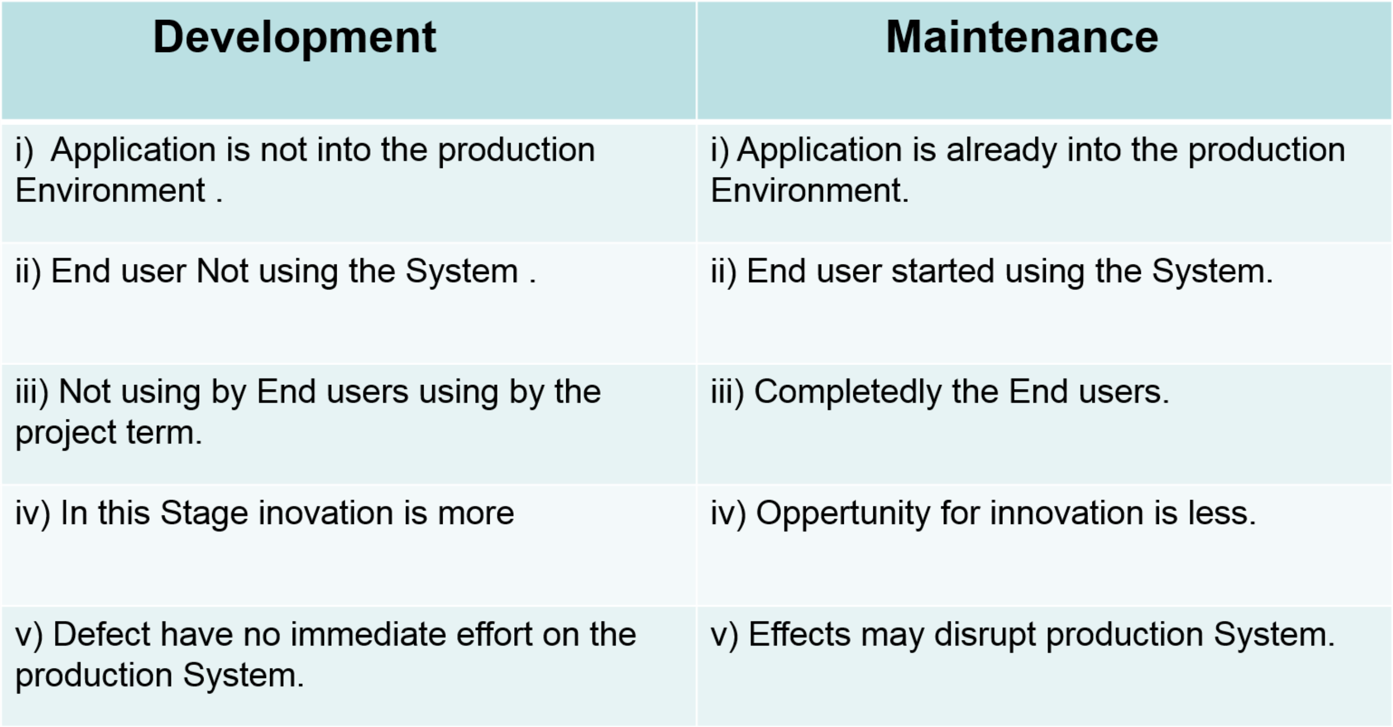
❖ Compatability Testing

❖ State Testing (Ex ,SUDOKU)

❖ Regression Testing

- **Software Maintenance :**
  - It is the Core aspects of Software Engineering, After the System Development the System gets deploys into the production Environment.

Development Vs Maintenance



Development	Maintenance
i) Application is not into the production Environment .	i) Application is already into the production Environment.
ii) End user Not using the System .	ii) End user started using the System.
iii) Not using by End users using by the project term.	iii) Completedly the End users.
iv) In this Stage inovation is more	iv) Oppertunity for innovation is less.
v) Defect have no immediate effort on the production System.	v) Effects may disrupt production System.

Maintenance Activities :

- **i) Adaptive Maintenance:**
  - Adjusting the System adaptiously based on the Environment Changes is called as Adaptive Maintenance.
- **ii) Corrective Maintenance :**
  - It is the Process of identifying, isolating and repairing the faults'.
- **iii) Perfective Maintenance :**
  - It is the System upgrade tired of work which makes the System to work more efficiently. Example: Upgrading Hardware, Software.
- **iv) Preventive Maintenance :**
  - Activities there are aimed at increasing the maintainability.
- **Reverse Engineering :**
  - It is analyzing the Software System to identify and understand the overall System Components, design and 1 requirements flow of the System.
- **Re-Engineering :**
  - In this process the Source Code is getting changed for easy maintenance and functionality of the System.

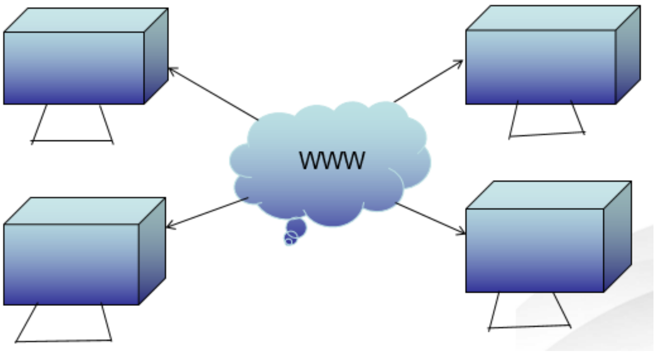
UNIT 5

Content of this chapter

- ✓ Web Engineering
- ✓ Introduction to Web
- ✓ General Web Characteristics
- ✓ Web Application Categories
- ✓ Working of Web Application
- ✓ Advantages and Drawbacks of Web Applications
- ✓ Web Engineering
- ✓ Emerging Trends in Software Engineering
- ✓ Web 2.0
- ✓ Rapid Delivery
- ✓ Open Source Software Development
- ✓ Security Engineering
- ✓ Service Oriented Software Engineering
- ✓ Web Service
- ✓ Software as a Service
- ✓ Service Oriented Architecture
- ✓ Cloud Computing
- ✓ Aspect Oriented Software Development
- ✓ Test Driven Development
- ✓ Social Computing.

Web Engineering:

- **"Web is a Collection of Computer Connected by a Network ".**
- Web helps us to place any information and documents in web in electronic formats and retrieve the same from Web.



- **Popular Web Browsers :**
  - Microsoft Internet Explorer
  - Google Chrome

Web Application Categories:

- Client side Web Application
- Server-side Web Application
- Middleware Web Application

❖ Client-Side Web Application:

- This Web Application runs at the Client-side in browser. Resource are coded at the Client-side. Client Application are coded using technologies such as HTML, CSS and JAVA Scripts.

❖ Server-Side Web Application:

- This Web Application runs at the Server only. Server Side Application are coded using technologies such as JAVA, ASP and PHP.

❖ Middleware Web Application:

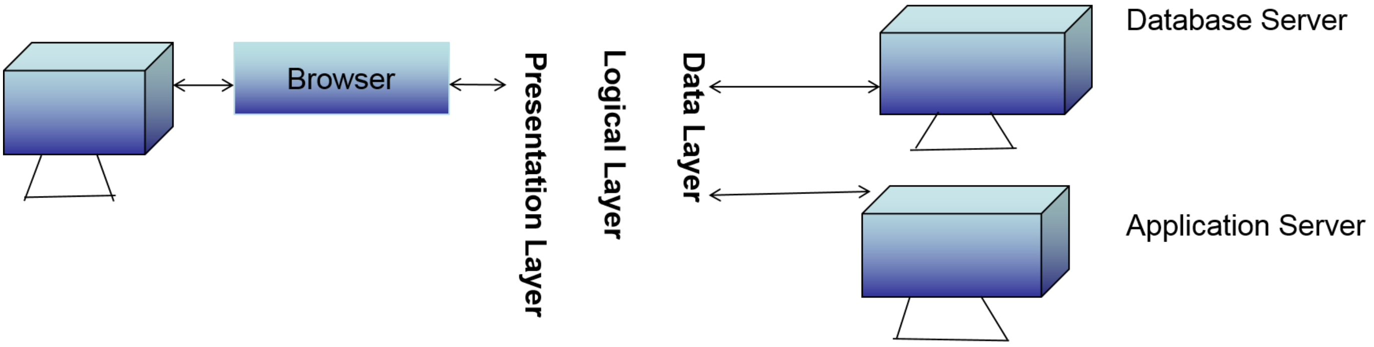
- This Web Application acts in between the Client and Server. Technologies include Java Beans , Java Servlet etc.,
- Web Application has Consists of Three Layers ,

❖ **Presentation Layer**

❖ **Application Layer**

❖ **Data Layer**

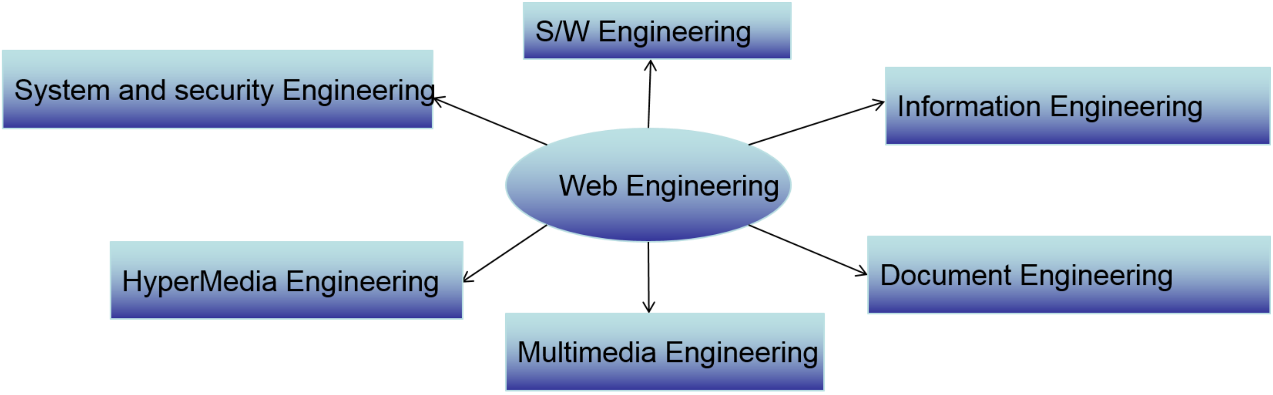
- **Presentation Layer :**
  - Presentation Layer Consists of a basic browser which runs HTML kind of pages.
- **Application Layer**
  - Application Layer does a basic Calculation and business logic and it is also called as “Logical Layer”.
- **Data Layer :**
  - Data Layer Consists of Database and other Storage devices.



➢ Advantages of WEB Applications :	➢ DrawBacks of WEB Applications :
✓ It can be accessed from anywhere in the world.	✓ We Applications are prone to get hacked easily by hackers.
✓ Only browser is needed at the Client side to connect with the Web Applications.	✓ Data theft is possible easily in Web Applications.
✓ Database and Objects are running at the Server end,so load at the Client side is always low.	✓ Application also can be copied easily in the Web.

Web Engineering:

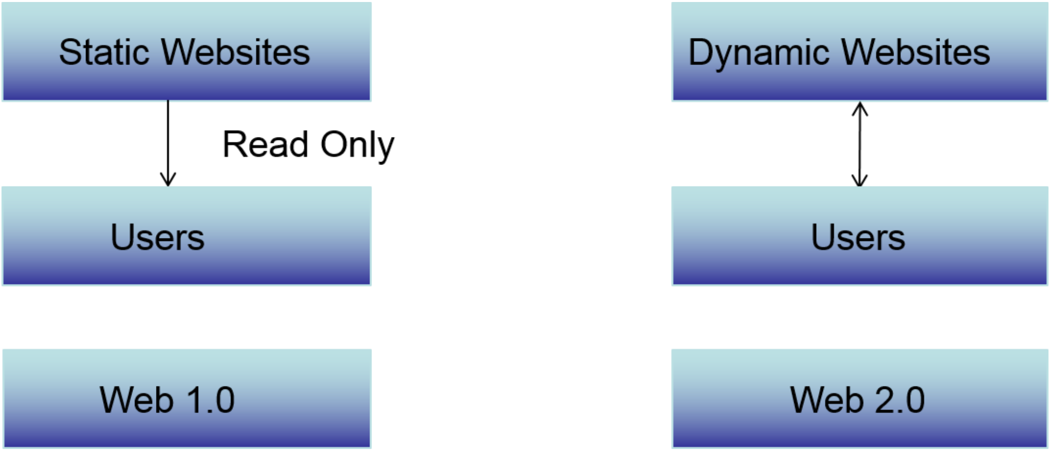
- It is the application of Systematic disciplined and Quantifiable approaches to the design, Production, deployment, operation, maintenance and Evolution of Web.
- Web Engineering is also defined as the process of Creating High-Quality Web-based Application.



- **Goals of Web Engineering :**

i) Develop high-Quality Web Application as per the requirement .
ii) Develop effective and efficient Web Application .
iii) Continuosly adopt to the requirement and Environment Change .
iv) Use Web Engineering process Model .

- □ **Engineering Trends in Software Engineering :**
    - ○ Web 1.0 is used to create with static Web pages (HTML), it is just one-way Communication.
    - ○ Web 2.0 is Considered as Collaborative technology used to create dynamic Web pages.



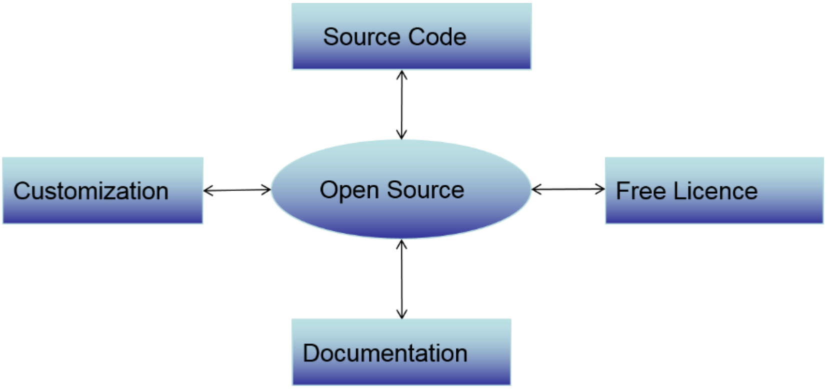
- ✓ Web 2.0 uses Social media for Interactions.
- ✓ Example for Web 2.0
- ✓ Principle: Gmail, Google maps.

Rapid Delivery:

- ➢ Software Engineering lifecycles are being followed in parallel rather then in Sequence.
    - ▪ Examples for Rapid Delivery include :
        - ○ SCRUM Methodology
        - ○ XP Methodology
        - ○ CANBAN Methodology

Open Source Software Development:

- ✓ OS not only gives the executable Code to the users but also the Source Code to the Users.
- ✓ The users can customize the code further based on his Needs.



Example of Open Source Software Products are as follows:

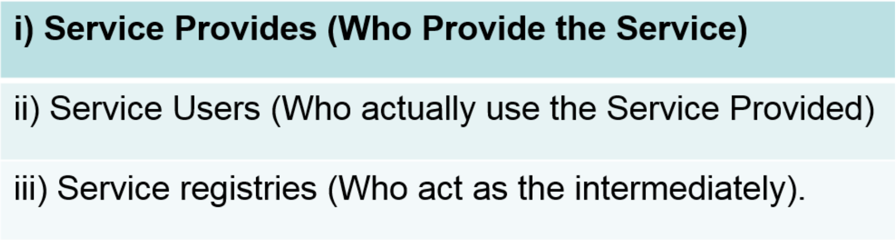
- ○ Mozilla firefox, PHP, Linux

Security Engineering:

- Security Engineering is a specialized field of Engineering that focuses on the Security aspects of the Software System.
- Default Downy and default permit are two fundamental principles of Security Engineering aspects.

Web Service:

- Web Service Description Language (WSDL) which is also XML Based.
- Three types of Players in Service-Oriented interfaces :



i) Service Provides (Who Provide the Service)
ii) Service Users (Who actually use the Service Provided)
iii) Service registries (Who act as the intermediately).

- **Cloud Computing :**
  - IaaS
  - SaaS
  - PaaS
- **Social Computing :**
  - It means using the Social Software to Connect with Society. There are two main Components of Social Computing.
    - Contents
    - Connections
- Contents are getting shared through Connections.
  - Example :
    - Blog, Facebook.

--------------------------- Complete ---------------------------

Thank You