GENIUS | Page No.
Date: / /

Unit - II (Requirement Engineering Process)

(#) Requirement gathering and analysis

- Requirement gathering is a crucial phase in software development lifecycle (SDLC) and project management.
- It involves collecting, documenting, and managing the requirement that defines the features and functionalities of a system or application.

Process of Requirement gathering

The six steps of Requirement gathering process are :-

Step-1 Assigning Roles

The first step is to identify and engage with all relavent stakeholder. stakeholder can include end-user, client, project manager, subject expert.

Step-2 Define Project Scope

Define the scope of project by outlining its objectives, boundaries and limitation.

Step-3 Conduct stakeholder interview

Schedule interview with key stakeholder to gather information about their need, preference and expectation.

Step 4 Document Requirement

This documentation can take various forms such as user stories, formal specification.

Step 5 Verify and Validate Requirement

Once the requirement are documented, its crucial to verify and validate them.

Step 6 Prioritize Requirement

Prioritize the requirement based on their importance to the project goal and constraints.

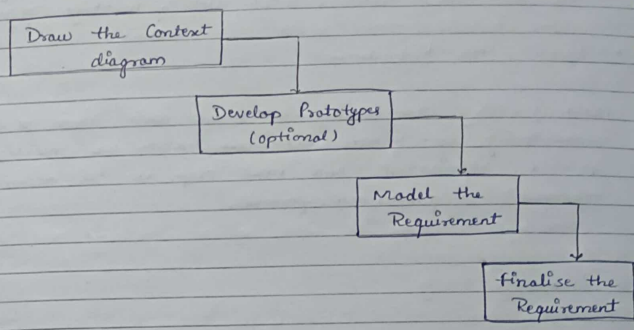
* Benefits of Requirement gathering

- Cost Reduction
- Customer satisfaction
- Improved communication
- Enhanced quality
- Risk management

Requirement Analysis

- We analyze, refine the gathered requirement to make consistent and unambiguous requirement this activity reviews all requirement and may provide a graphical view of entire system.

- The various steps of Requirement Analysis are :-



→ Draw the Context diagram

The context diagram is a simple model that defines the boundaries and interface of proposed system with the external world.

→ Development of Prototype

One effective way to find out what the customer want is to construct a prototype. Prototype should be built quickly and low cost.

→ Model the Requirement

This process usually consist of various graphical representation

of function, data entities. The graphical view may help to find the correct, incosistent, missing requirement

→ finalise the Requirement

After modeling the requirement, we will have a better understanding of system behaviour. Now we finalize the analyzed requirement and next step is to document these requirement in prescribed format.

Feasibility Study

- A feasibility study evaluate the potential success of a project by analyzing its technical, economic, operational and scheduling aspect.

Types

① Technical feasibility
Examines whether the technology required for the project is available

② Economic feasibility
Analyzes the cost-effectiveness of the project.

③ Legal feasibility
Assess whether the project complies with legal and regulatory requirement.

④ Operational feasibility
Evaluates whether the software will fit smoothly into the organization existing process and whether user can adapt

⑤ Schedule feasibility
Looks at the time required to complete the project and whether it can be completed within the desired timeframe

Process
↳ Identify project goals
↳ Gather requirement
↳ Analyze feasibility
↳ Prepare feasibility report

Requirement Validation

- Requirement validation is the process of checking that the requirement are complete, consistent, feasible and verifiable It ensure that the requirement meet the need of users and stakeholder.

- Validating requirement is essential for reducing the risk of project failure. It helps to ensure that the development team is building the right product, which leads to higher customer satisfaction.

* Techniques

① Review

formal or informal reviews are conducted where stakeholder provide feedback. This process helps examine requirement

② Prototype

A prototype or mock-up is created this helps stakeholder understand how the requirement will be implemented

③ Modeling

Modeling techniques is used in understanding and validating the requirement

④ Walkthrough

Walkthrough session are organised where stakeholder discuss the requirement in detail and provide their input

⑤ Interviews and Surveys

Conducting interviews or surveys with stakeholder help clarify their need and expectation

Requirement Management

- Requirement management is a process used in software development to effectively handle requirement throughout the project lifecycle.
- The requirement management process is the process of managing changing requirement during the requirement engineering process and system development where the new requirement emerge as a system is being developed after it has gone into use.

Advantage

- Recognizing the need for change in the requirement
- Improved team communication
- It help to minimize error at early stage of development lifecycle

Functional and Non-functional Requirement

- These are the requirement that the end user specifically demand as basic facilities that the system should offer these are represented or stated in the form of input to be given to the system, the operation performed and output expected this is called functional Requirement
- These are the quality constraint that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to another This is called non-functional Requirement

Difference between functional and non-functional Requirement

functional Requirement

- Describe what the system should do i.e. specific functionality
- Focuses on behaviour and feature of system
- Easy to measure in terms of output or result
- Directly related to user and business requirement
- Can be tested through functional testing
- Ex- User authentication, data input/output, transaction processing
- Define the action and operation of system

GENIUS | Page No.: | Date: / /

Non-functional Requirement

- Describe how the system should perform i.e, System quality
- Focuses on the performance, usability and other quality attributes
- Difficult to measure
- Related to user experience and system performance
- Evaluated through performance testing, security testing
- Ex- Scalability, security, reliability, maintainability
- Defines condition under which the system must operate

User Requirement

These requirement describe what the end-users want from the software system. User requirement are usually expressed in natural languages and are typically gathered through interviews, surveys, or users feedback.

User requirement provide information that serves as the basis for further specification, design and verification of manufacturing system.

System Requirement

These requirement specify the technical characteristics of the software system, such as its architecture, hardware requirement, software component and interface. System requirement are typically expressed in technical terms and are often used as a basis for system design.

GENIUS | Page No. | Date: / /

SRS document

- The SRS document is reviewed by the testing person or a group of persons by using any verification method.
- The SRS document is typically written by a business analyst or a system analyst in collaboration with stakeholder.

Goals of SRS document
- ↳ Problem breakdown
- ↳ Input to design specification
- ↳ feedback to the customer
- ↳ Product validation check

Advantages
- ↳ Clarity
- ↳ Consistency
- ↳ traceability
- ↳ Validation

Disadvantage
- ↳ Time-consuming
- ↳ Limited flexibility
- ↳ Limited user involvement
- ↳ Misinterpretation.

(#) Design Engineering

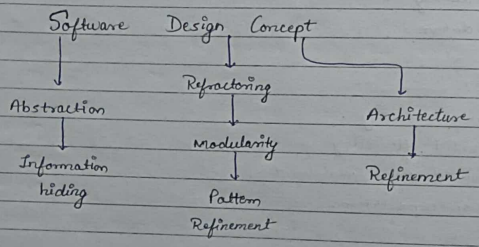
Software Design Concept

- Software design is the process of transforming user requirement into a suitable form, which help the programmer in software coding and implementation

- The software design concept simply means the idea or principal behind the design. It describe how you plan to solve the problem of designing software and thinking behind how you will design software.

  It allows the software engineer to create the model of the system software that is to be developed or built

- It provides a supporting and essential structure or model for developing the right software.



- Abstraction

  Abstraction simply means to hide the details to reduce complexity and increase efficiency or quality. Abstraction can be used for existing element as well as the component being designed. The two common abstraction mechanism are :-

  → functional Abstraction
  → Data Abstraction

- Modularity

  Modularity simply means dividing the system or project into smaller parts to reduce the complexity of system or project. It allows large programs to be written by several or different people. characterstics of modular are :-

  → Each module has single specified objectives
  → Modules can be separately compiled
  → Module should be easier to use than to build

- Architecture

  Architecture simply means a technique to design a structure of something. It is a concept that focuses on various element of the structure. These component interact with each other and use the data of the structure in architecture.

- Refinement

  Refinement simply means to refine something to remove any impurities if present and increase the quality. It is a process of developing or presenting the software or system in a detailed manner which means elaborating a system or software.

- Pattern

A pattern simply means a repeated form or design in which the same shape is repeated several times to form a pattern

- Information hiding

Information hiding simply means to hide the information so that it cannot be accessed by the unwanted party. Information hiding is achieved by designed the modules in such a manner that the information gathered or contained in one module is hidden and can't be accessed by any other module

- Refractoring

Refractoring simply means reconstructing something in such a way that it does not affect the behaviour of any other feature. Refractoring means reconstructing the design to reduce complexity and simplify it without impacting the behaviour of its function.

* Design Process

The software design process can be divided into the following three levels or phases of design

① Interface Design
② Architectural Design
③ Detailed Design

① Interface Design

Interface design is the specification of the interaction between a system and its environment. This phase proceeds at a high level of abstraction with respect to the inner working of the system

- Description of event or message that the system must produce
- Specification of data, format of data coming into and going out of the system
- Description of message from agent to which system must respond.

② Architectural Design

Architectural design is the specification of major component of system, their responsibilities, interface and interaction between them

- Gross decomposition of system into major decomposition
- Allocation of functional responsibilities to component
- Communication and interaction between component

③ Detailed Design

Detailed design is the specification of internal element of all major system component. It may include

- User interface
- Algorithm and data structure
- Control interaction between units

GENIUS | Page No.: | Date: / /

* Function Oriented Design

- The basic abstraction, which are given to the user are real world function
- function are grouped together by which a higher level function is obtained
- Carried out using structured analysis and structed design.
- It is top down approach
- Begins by considering the use case diagram
- Mainly used for computation of sensitive application

* Object - Oriented Design

- The basic abstraction are not real world function but data abstraction are real world
- functions are grouped together on the basis of data they operate.
- Carried out using UML
- It is bottom up approach
- Begins by identifying objects and classes.
- Mainly used for evolving bussiness case.

GENIUS | Page No.: | Date: / /

* Structured Analysis and Structured Design (SA/SD)

- Structured analysis and structured design is a diagrammatic notation that is designed to help people understand the system
- The basic goal of SA/SD is to improve quality and reduce the risk of system failure.
- Structured analysis and structured design is a traditional software development methadology that was popular in 1980s and 1990s
- Steps involved in SA/SD process :-
  - ↳ Requirement gathering
  - ↳ Structured analysis
  - ↳ Data Modeling
  - ↳ Process Modeling
  - ↳ Input / output design
  - ↳ Structured design
  - ↳ Implementation and testing
- SA/SD is combined known as SAD and mainly focuses on 3 points
  - ↳ System
  - ↳ Process
  - ↳ Technology
- It involves 2 phase
  - ↳ Analysis - Data flow diagram, Data dictionary, state transition diagram and ER diagram
  - ↳ Design - Structure chart and Pseudo code

GENIUS | Page No.: | Date: / /

* DFD (Data flow Diagram)

- A Data flow diagram (DFD) is a traditional visual representation of the information flows within a System. It shows how data enters and leaves the system, what changes the information and where data is stored.

- The objective of DFD is to show the scope and boundaries of system as whole.

- Characteristics of DFD

① Graphical Representation
DFD use different symbols and notation to represent data flow within the system.

② Problem Analysis
DFD is are usiful in understanding the system and effectively used during analysis.

③ Abstraction
DFD provide an abstraction to complex model. It hides unnecessary implementation and show only flow of data.

④ Hierarchy
DFD provide a hierarchy of system. Ex- High-level diagram and low level diagram.

GENIUS | Page No.: | Date: / /

Types

① Logical data flow diagram

Logical data flow diagram mainly focuses on system process. It illustrate how data flows in the system. Logical flow diagram mainly focuses on high level process and data flow without dividing deep into technical implementation detail. Logical data flow diagram is used in various organization for smooth running of system.

② Physical data flow diagram

Physical data flow diagram shows how the data flow is actually implemented in the system. In physical data flow diagram we include additional detail such as data storage, data transmission, and specific technology. Physical DFD is more specific and close to implementation.

Components of Data flow diagram are :-

- Process
- Data flow
- Data store
- External entity

Symbols used in DFD

- Rectangle
- Circle
- Arrow

Advantages

- Helps to understand the functioning and limit of system
- Easy to understand because of graphical representation
- Represent detailed and well explained diagram

Disadvantage

- Sometime confuses the programmers
- Time consuming

* Coupling and Cohesion

- Coupling refers to the degree of interdependence between software modules. High coupling means that modules are closely connected and changes in one module may affect other module. low coupling means that modules are independent, and changes in one module have little impact on other modules.

- Cohesion refers to degree to which elements within a module work together to fulfill a single, well-defined purpose. High cohesion means that the elements are closely related and focused on single purpose, while low cohesion means that element are loosely related and serve multiple purpose.

- Both coupling and cohesion are important factors in determining the maintainability, scalability and reliability of software system. High coupling and low cohesion can make a system difficult while low coupling and high cohesion make a system easier to maintain and improve.

Difference

Coupling

- Coupling is the concept of inter-module
- Represent the relationship between modules
- Increasing coupling is avoided for software
- Represent interdependence among modules
- Loosely coupling gives the best software
- Modules are connected to other modules
- Coupling is created b/w two different modules

Cohesion

- Cohesion is the concept of intro-module
- Represent the relationship within a module
- Increasing cohesion is good for software
- Represent functional strength of modules
- Highly cohesive gives best software
- Module focuses on a single thing
- Cohesion is created between the same module

GENIUS | Page No.: | Date: / /

* UML Diagram

- Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprint used in other field of engineering. It is not programming language it is rather a visual language.

- We use UML diagram to show the behaviours and structure of a system

- We need UML to visually represent and communicate complex system design better understanding and collaboration among stakeholder.

Different View of Software using UML

① User View

This view capture the view of the system in terms of functionalities offered by the system to its user. Defines numerous functionalities of the device.

② Structureal View

This view defines the structure of the problem. The structure of the problem is defined on the basis of classes and object which is important to understand the working of system.

③ Behavioural View

This view defines how object interact with each other just to realize the system behaviour.

④ Implementation View

This view captures the various important component of system and their interdependence.

⑤ Environmental View

This view describe the implementation of numerous specific portion of hardware.

Types of UML diagram

↳ Class Diagram

The most widely use UML diagram is the class diagram It is the building block of all object oriented software system. Class diagram also helps us identify relationship between different classes or object.

- Class notation in UML

Car	← Name
- make : String	
- model : String	← attribute
- year : int	
+ start() : void	
+ stop() : void	← Operation
+ drive() : void	

- Class diagram shows the static structure of a software system, classes, attributes, methods and relationship.

↳ Object diagram

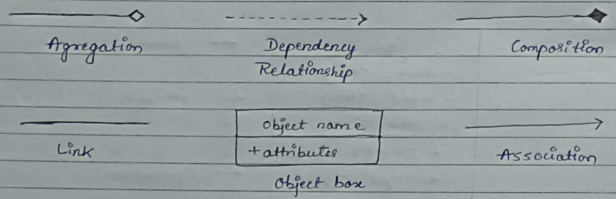
- Object diagrams are a visual representation in UML that illustrate the instance of class and their relationship within a system at a specific point in time. They display object, their attribute and the link between them.

- An Object diagram is the Unified Modeling Language (UML) is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.

- An object diagram is similar to a class diagram except it shows the instance of classes in the system.

- Object diagram Notation

  The object diagram in UML uses specific notation to represent instance of class and their relationship at particular time.



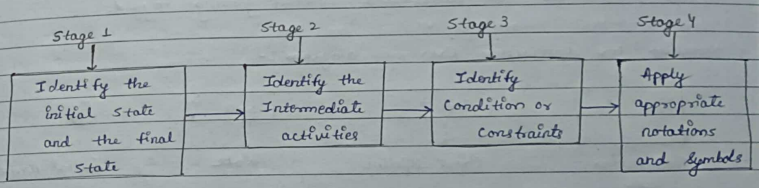
- Uses
  - ↳ Test case design
  - ↳ Training and documentation
  - ↳ Debugging and troubleshooting

↳ Activity diagram

- Activity diagram shows the steps involved in how a system works, helping us understand the flow of control. They display the order in which activities happen and whether they occur one after the other or at the same time.

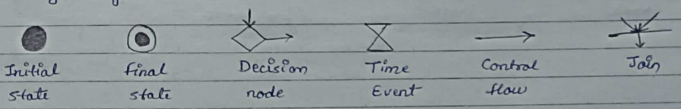
- An activity diagram starts from an initial point and ends at a final point, showing different decision paths along the way.

- Steps to draw an Activity diagram :-



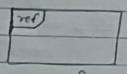
- Activity diagram are often used in business and process modeling to show how a system behave over time.

- Activity diagram notation



↳ Interaction Overview diagram

- Interaction Overview diagram (IOD) in UML provide a high-level view of the interactions between various component or object in a system. they are used to visualize the flow of control and interaction within a system.
- They help with requirement analysis, documentation, communication and system design.
- Interaction diagram give a general overview of system behaviour that stakeholder can understand without getting bogged
- Notation for Interaction Overview diagram :-



Interaction use

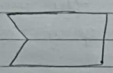
Object lifeline

Note

Initial node

Decision Node

fork/Join node



Recieve Signal node

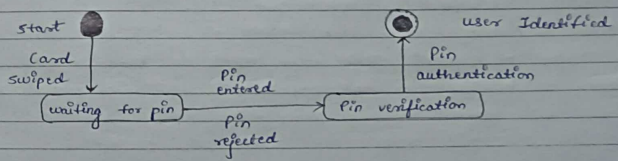
Control flow

↳ State Diagram

- A state diagram is a UML diagram which is used to represent the condition of the system or part of the system at finite instance of time
- State diagram is also known as state machine diagram or state chart diagram
- We can say that each and every class has a state but we don't model every class using state machine diagram
- Steps to draw state machine diagram :-

Step 1 - Identify the system
Step 2 - Identify initial and final state
Step 3 - Identify possible states
Step 4 - Label triggering event
Step 5 - Draw the diagram with appropriate notation

- Example - A state machine diagram for User Authentication



GENIUS | Page No.:
Date: / /

Steps to Create UML diagram

step 1 - Identify the purpose
step 2 - Identify elements and relationship
step 3 - Select the appropriate UML diagram type
step 4 - Create a Rough sketch
step 5 - Choose a UML Modeling tool
step 6 - Create the diagram
step 7 - Define Element properties
step 8 - Add Annotations and comments
step 9 - Validate and Review
step 10 - Refine and Iterate
step 11 - Generate documentation