UNIT-02
SOFTWARE TESTING

⇒ Topics:

- ☆ Requirement Engineering Process:-
  - Elicitation
  - Analysis
  - Documentation
  - Review and Management of User Needs.
- ☆ Feasibility Study
- ☆ Information Modelling
- ☆ Data Flow Diagram
- ☆ DFD vs Flow chart
- ☆ ER Diagram
- ☆ Decision table, Decision tree
- ☆ SRS Document
- ☆ IEEE Standards for SRS
- ☆ Software Quality Assurance (SQA): Verification and validation.
- ☆ SQA Plans
- ☆ Software Quality Frameworks
- ☆ ISO 9000 Models
- ☆ SEI-CMM Model.

⇒ Requirement Engineering Process :

- It refers to the process of defining, documenting and maintaining requirement in the engineering design process.

- Requirement engineering videos provides the appropriate mechanism to understand what the customer desired, analyzing the need and assessing feasiblity, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into the working system.

- Requirement Engineering is the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system.

⇒ Process :-

1. Requirement Elicitation :

- This is also known as gathering of requirement
- It is related to the various ways used to gain knowledge about the project domain and requirements.
- The various source of domain knowledge include customers, business manuals, the existing software of same type, standards and other stakeholder of the project.

- The techniques used for requirements elicitation include interviews, brainstorming, Delphi technique, prototyping etc.

(ii) **Requirement analysis:**

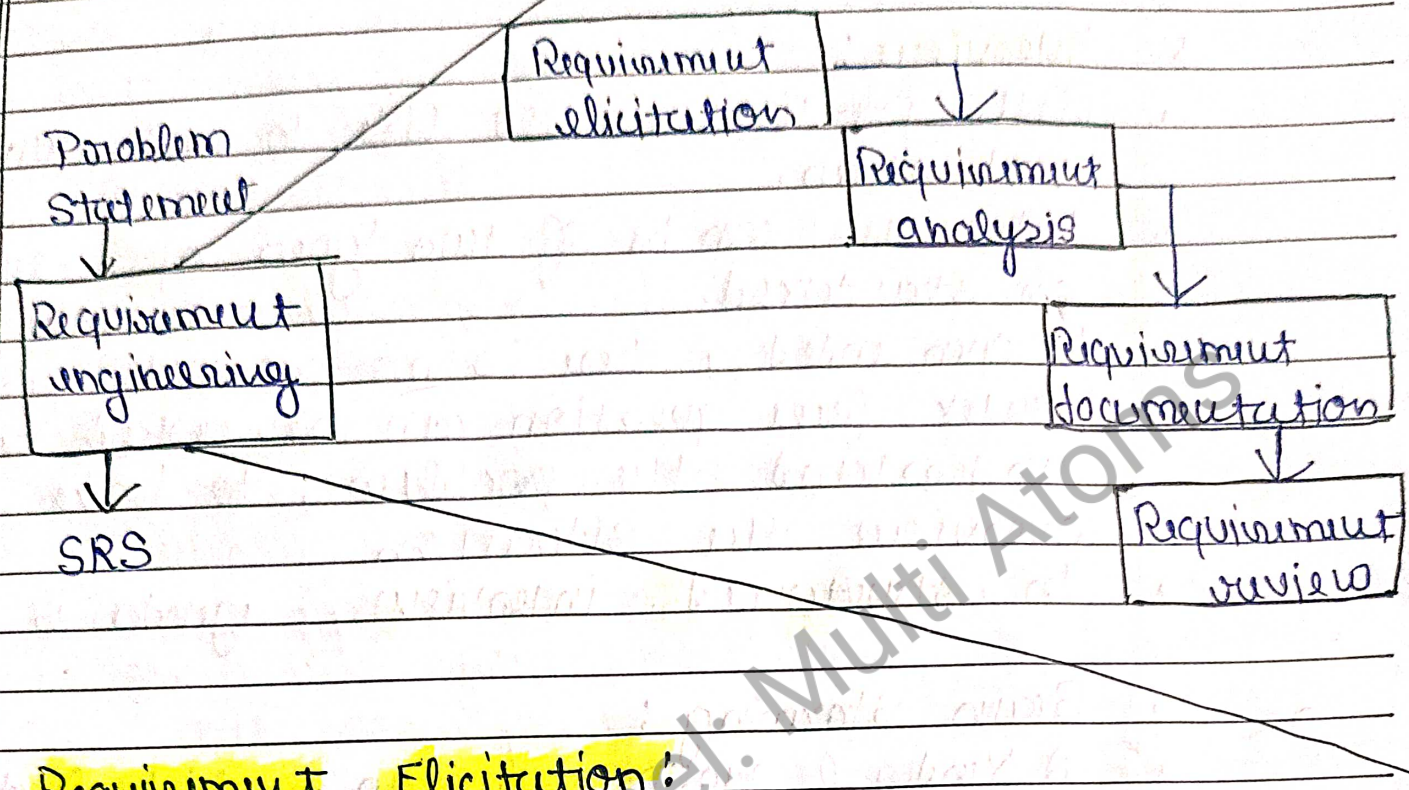
- The requirement are analysed in order to identify inconsistencies, defect, omission etc
- We describe requirements in terms of relation ship and also resolve conflict, if any.

(iii) **Requirement documentation:**

- This is the end product of requirement elicitation and analysis.
- The requirement documentation is very important as it will be the foundation for the design of the software
- The document is known as SRS.

(iv) **Requirement review:**

- The review process is carried out to improve the quality of the SRS.
- It may also be called as requirement verification.
- For maximum benefits, review and verification should not be treated as a device activity to be done only at the end of the preparation of SRS
- It should be treated as continous activity that is incorporated into the elicitation, analysis, and documentation.



⇒ **Requirement Elicitation:**

- Requirement elicitation is the activity during which software requirements are discovered, expressed, and revealed from system requirements
- Requirement elicitation is the process of gathering information about the needs and expectations of stakeholders for a software system.

⇒ **Requirement elicitation Activities:**

→ Knowledge of the overall area where the system is applied.
→ Interaction of system with external requirement
→ Detailed investigation of user needs

Methods of requirement elicitation :-

1. **Interview :**
   - Both parties would like to understand each other.
   - Interview can be of two types open-ended or structured
   - In open ended, there is no preset agenda, context free question can be asked to understand the problem, to have an overview the situation
   - In structured interview, agenda is pre-set.

2. **Brain Storming :**
   - A kind of group discussion, which lead to ideas very quickly and help to promote creative thinking
   - Very popular now a days and is being used in most of the organizations
   - All participants are encouraged to say whether whatever idea come to their mind and no one will be criticized for any idea no matter how goofy it seems.

3 **Delphi technique :**
   - Here participants are made to write the requirement on a piece of paper, then these requirement are exchanged among participants who gave their comments to gets a revised set of requirement.
   - The process is repeated until the final consensus is reached

Page No.
Date / /

3. FAST (faciliated application specification technique)
- This approach encourage the creation of joint team of customer and developer who work together to understand correct set of requirements

4. Use case approach:
-> There are structured description of the user requirement.
-> It is a narrative which describe the sequence of events from user's perspective
-> Use case diagrams are graphical representation to show the system at different levels
-> They are some times supportive my the activity diagrams, to understand the work flow

5. Prototyping: This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholder and to validate requirement

=> Problems in Requirement Elicitation:
1. less knowledge of technology.
2. Use of different language by developer and user
3. lack of skills in developer
4. User is not providing some information due to some reason.

Page No.:
Date : / /

⇒ **Feasibility Study :**

- A feasibility study specifies whether the proposed software project is practically possible or not.

- The objective behind the feasibility study is to create the reason for developing the software that is acceptable to users, flexible to change and conformtable to established standards.

- **Types of feasibility study :**

1] Technical Feasibility
2] Operational Feasibility
3] Economic Feasibility
4] Ugal Feasibility
5] Schedule Feasibility.

- **Technical Feasibility :** Technical feasibility evaluates the current technologies, which are needed to accomplish customers requirements within the time and budget.

- **Operational Feasibility :** Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements

Page No.
Date / /

- **Economic Feasibility**: Economic feasibility decides whether the necessary software can generate financial profits for an organization

- **Legal Feasibility**: Legal feasibility makes sure that your product complies with all regulations and doesn't break any law.

- **Schedule Feasibility**: Mainly timelines / deadlines is analyzed for proposed project.
- How much time the team will take to complete the final project.

⇒ **Requirement analysis**:

- In this phase we analysis all the set of requirements to find any inconsistency or conflicts
- In requirement gathering phase our all concentration was on getting all the set of requirements but now, we see how many requirements are contradictory to each other or requires further exploration to be considered further.

- Different tools can be used Data flow Diagram, Control flow Diagram, ER diagram

Date : / /

⇒ Data Flow Diagram :-

① In software engineering, a data flow diagram is a graphical representation of the flow of data within a system

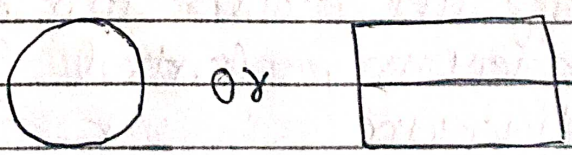
② DFD stand for Data flow diagram, it is also known as "Bubble chart" through which we can represent the flow of data graphically in an informatio

③ By using DFD we can easily understand the overall functionality of system because diagram represents the incoming data flow / out going data flow and stored data in a graphical form

④ It describes how data is processed in a system in term of input and output.

⇒ Component of a DFD:

- Process : Represent function or transformations that are performed on data.
- Process are represent by circle or oval.

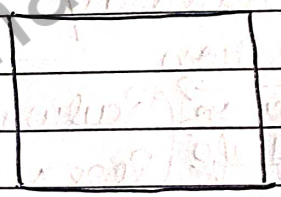


- Data Stores : Represent where data is stored

within the system. They are shown as rectangle with two parallel line inside

- Data Flows : Represent the movement of data between process data stores and external entities.
- They are depicted as arrow indicating the direction of data flow

- External Entities : Represent sources or destinations of data outside the system. They are shown as squares in the diagram



⇒ General guidelines and rules for constructing DFDs

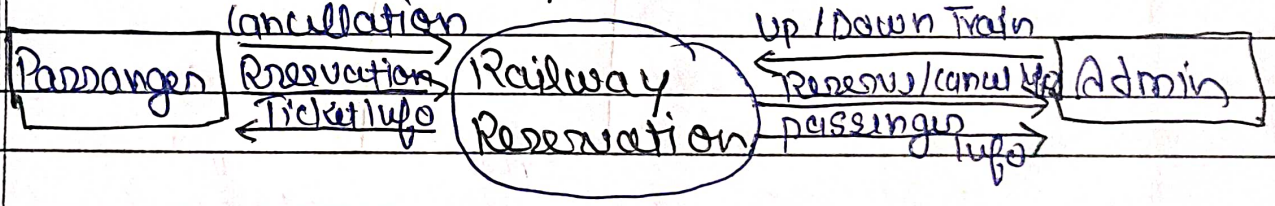
- Choose meaningful names for process and external entities
- Number the processes.
- Avoid complex DFD.
- Remember that a DFD is not a flow chart
- All names should be unique.
- Make sure the DFD is internally consistent.
- Processes are always running, they do not start or stop.

- All data flows are named.
- Number of processes at each level should be small (not more than 7)
- All data flows entering or outgoing from context DFD process should also be maintained in its leveled DFD.
- It should not contain loops.
- It should not contain crossing lines

⇒ **Levels of DFDs:-**

- Level 0 DFD: Also known as the context diagram, it provides an overview of the entire system, showing external entities and major processes.

Example: Railway Reservation



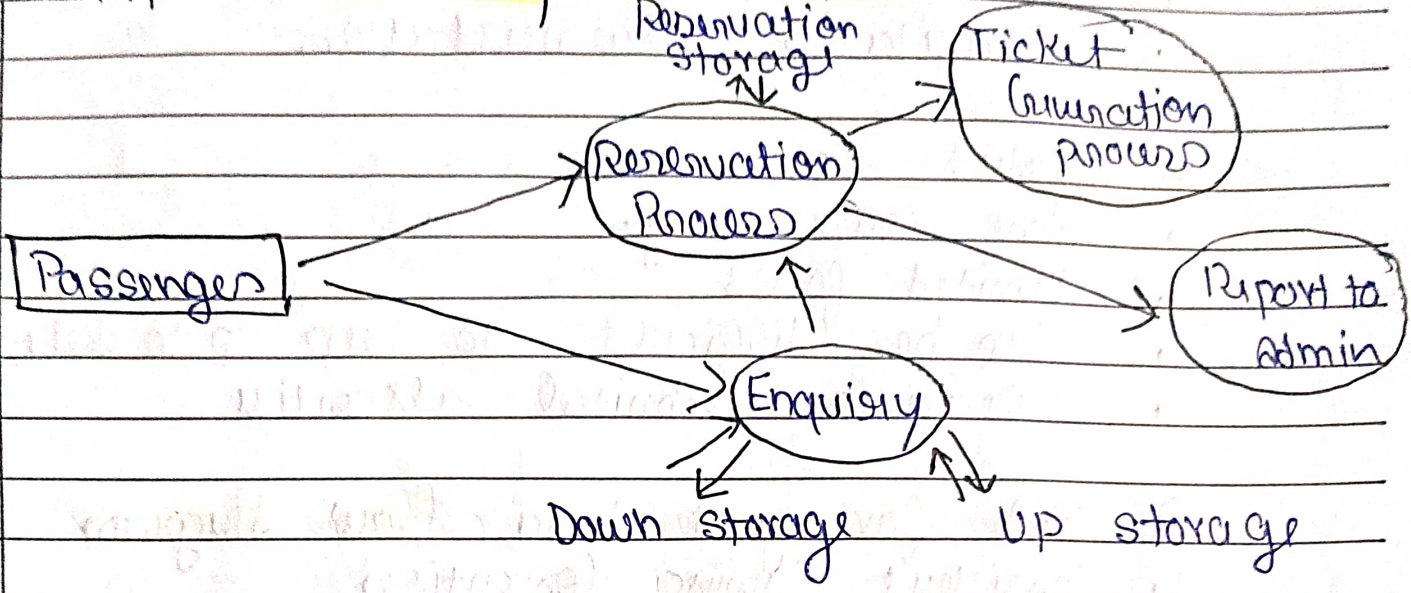
- It represent the entire system as a single bubble with input and output data indicated by incoming / outgoing arrows.

- 1-level DFD: In 1-level DFD, the context diagram is decomposed into multiple bubbles / processes.

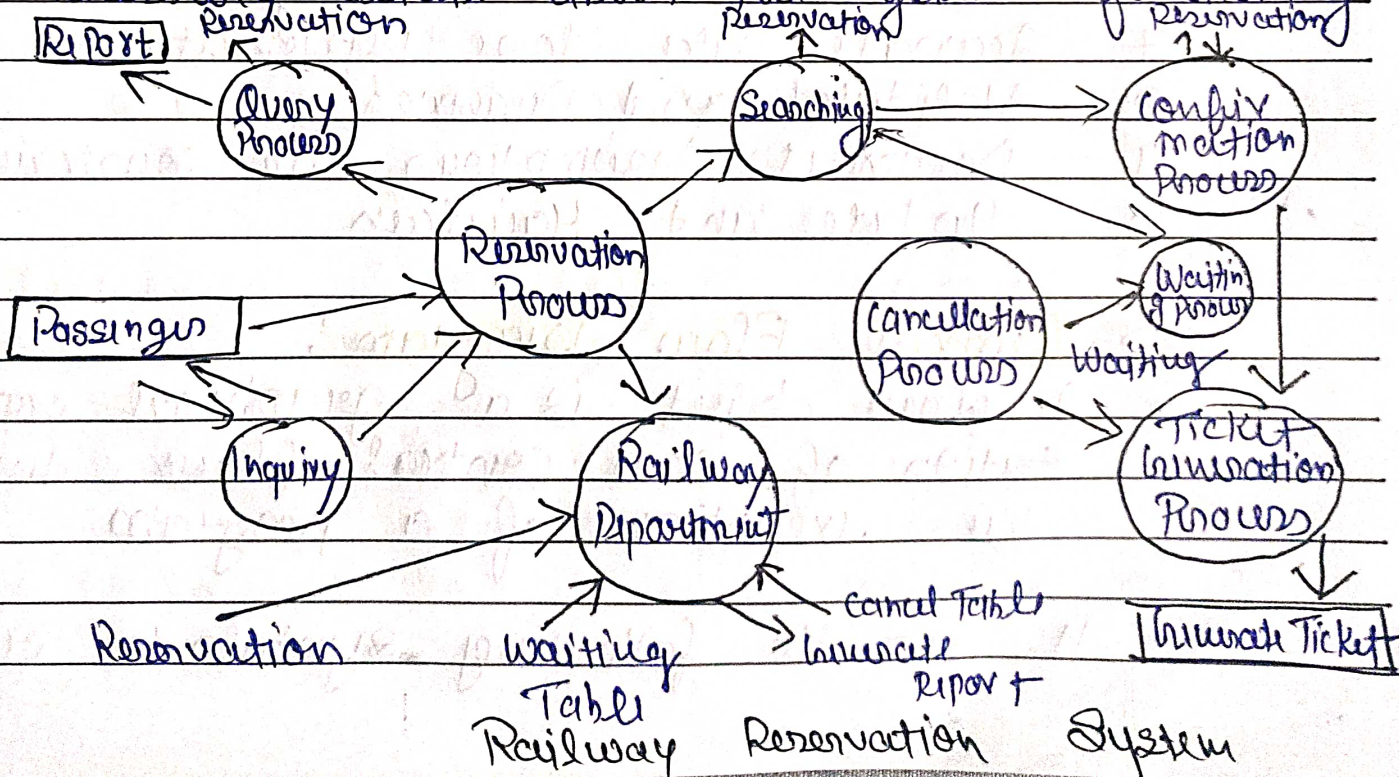
⊙ In this level, we highlight the main

functions of the system and breakdown the high level process of 0- level DFD into sub processes

Example : Railway Reservation



- 2- level DFD : 2- level DFD goes one step deeper into parts of 1- level DFD
- It can be used to plan or record the specific/ necessary detail about the system functioning



Date : / /

Adv :

- Easy to understand
- Improves system analysis
- Support system design
- Enables testing and verification
- Facilitates documentation

Disadv :

- Time - consuming
- Limited focus
- Can be difficult to keep up to date
- Require technical expertise

⇒ **Rule for good data flow diagram**

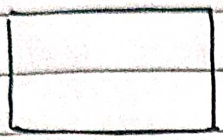
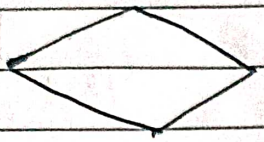
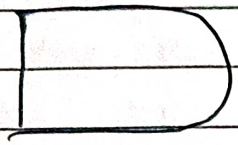
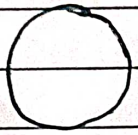
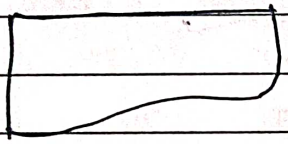
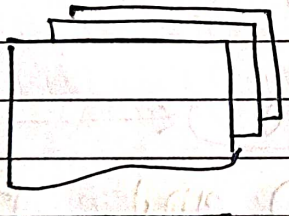
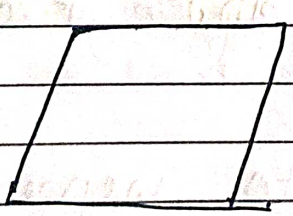
1. Consistent Naming Convention
2. Correct Symbol usage
3. Balanced level of detail
4. No cross - layer connections
5. Clear Data flow Direction
6. Minimal Data redundancy
7. Accurate Data store Representation
8. Validated and Reviewed
9. Document Assumptions and constraints
10. Update and Maintain

⇒ **Control Flow Diagram :**

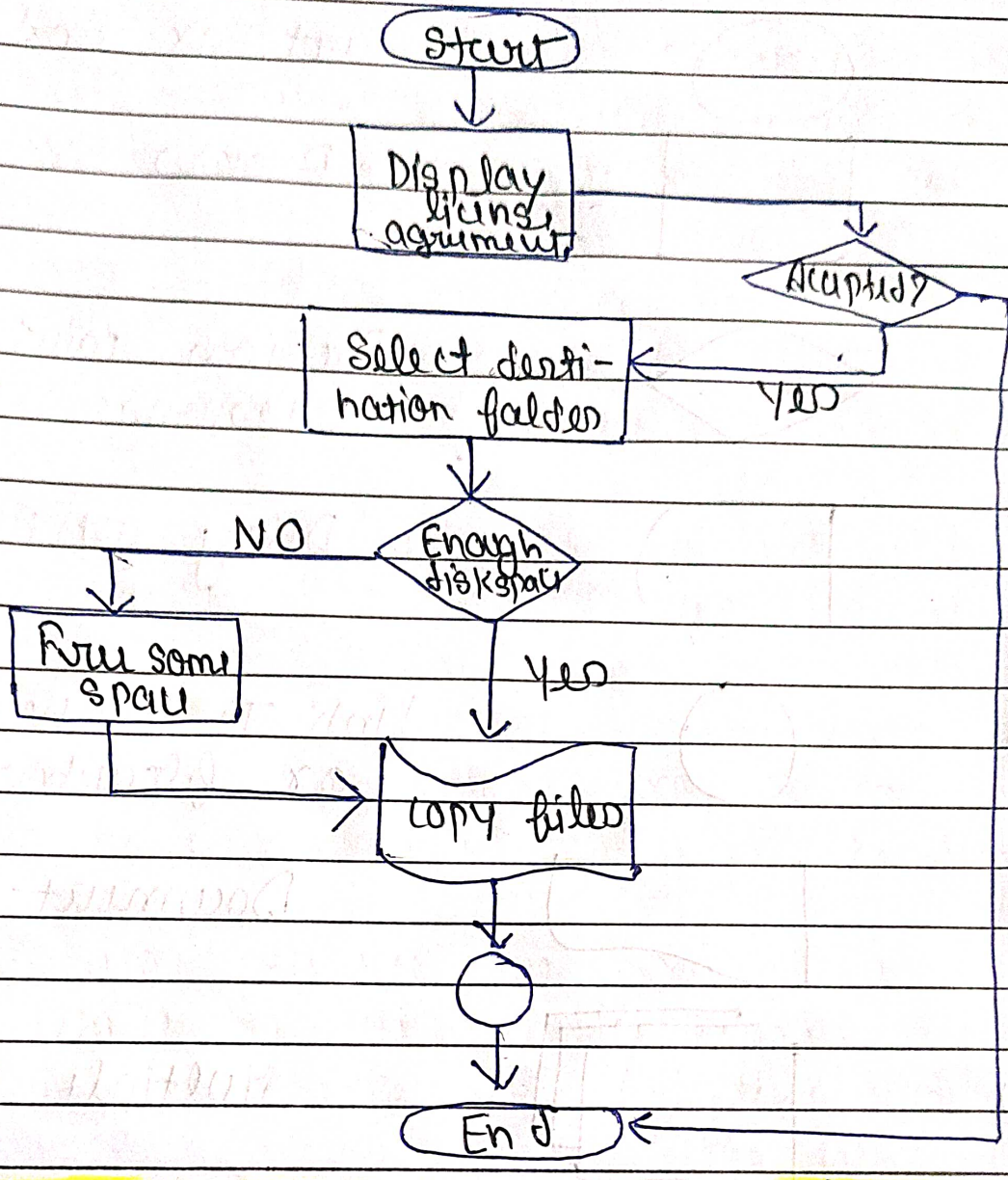
- A flow chart is a graphical representation of how control flow during the execution of a program
- It uses the following symbols to represent

Page No.
Date: / /

a system control flow.

1.  Start or End Point
2.  Process Step
3.  Decision point and Response path
4.  Delay or wait
5.  Link To Another page or flowchart
6.  Document
7.  Multiple Document
8.  Data.

⇒ Flowchart of Software installation



⇒ Difference b/w DFD and Flowchart.

- Flowchart;
- The main objective is to represent the flow of control in the program
- It has only a single type of arrow is used to show the control flow in the flow chart.
- It is the view of the system at a lower level.

Page No.
Date / /

- Three symbol represent a flowchart
- It deals with the physical aspect of the action.
- It show how to make the system function
- It is not very suitable for a complex system
- Flowchart types are
  - → System flowchart
  - → Data flowchart
  - → Document flowchart
  - → Program flowchart

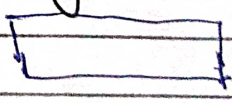
- Data flow diagram :
- The main objective is to represent the process and data flow between them.
- It is the view of the system at a high level.
- It defines the flow and process of data input, data output and storing data.
- Five symbol represent a DFD.
- It deals with the logical aspect of the action
- It defines the functionality of the system
- It is used for complex systems
- DFD types are
  - Logical DFD
  - Physical DFD

Page No.:
Date : / /

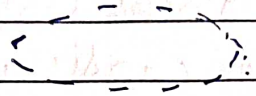
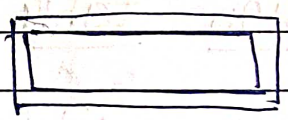
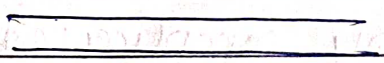
E-R Diagram :-

- ER Diagram a non-technical design method works on conceptual level based on the perception of the world.
- Diagram created using this ER-modeling method are called Entity-relation ship Diagrams or ER diagram.
- Three main constructs are data entities, their relationship and their associated attributes
- Entity :- An entity may be an object with a physical existence.
- Attribute :- are the properties that define by the entity type.
  For ex "Roll_No, Name, DoB
- Relationship : A relationship type represent the association between entity types

Component of ER diagram:

S.No	Component name	Symbol	Description
1.	Rectangles		Represent entity set

Page No.
Date: / /

2.	Ellipses		Represents a attributes
3.	Diamonds		Represent relationship sets
4.	Lines		Link attributes to entity sets and to relation
5.	Double ellipse		Represent multivalued attributes
6.	Dashed ellipse		Represent derived attributes
7.	Double Rectangle		Represent ~~desired~~ weak entity set
8.	Double lines		Represent total participation of an entity in a relationship set

Page No.:
Date : / /

Adv:
- Simple
- Effective
- Easy to Understand
- Integrated
- Useful in decision making
- Easy conversion
- Flexible

Disadv:-
- loss of information.
- limited relationship.
- No industry standard.
- Data inconsistency.
- Missing cardinalities.
- Difficult to modify.

⇒ Decision table :

- A decision table is a tabular representation of conditions and actions.
- It is used when the process logic is very complicated and involves multiple conditions
- It is not possible to represent the process logic efficiently with structural English.
- The main parts of decision table are:

(a) Conditions stubs: It lists all the conditions will ~~triggers~~ relevant to the decision

(b) Action stubs: It lists all the possible action that will take place for a valid set of conditions

c) Rules: It specifies which set of conditions will trigger which action.

⇒ **Decision tree**:-

- A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decisions.
- Decision tree are similar to decision tables, but they are more graphical and they resemble a tree
- It is drawn sideward, with the root of the tree at the left-hand and the leaves at the right-hand
- There are three types of nodes in a decision tree which are:-

a) Decision node
b) chance node
c) Terminal node

⇒ **Difference b/w Decision table and Decision tree**

- **Decision table**:
- Decision table are a tabular representation of conditions and actions.
- We can derive a decision table from the decision tree.
- It helps to clarify the criteria
- In Decision Tables, we can include more than one or condition.
- It is used when there are small number of properties

- It is used for simple logic only.
- It is constructed of rows and tables.
- The goal of using a decision table is the generation of rules for structuring logic on the basis of data stored in the table

- **Decision tree:-**

- Decision trees are a graphical representation of every possible outcome of a decision
- We can not derive a decision tree from the decision table.
- It helps to take into account the possible relevant outcomes of the decision.
- In Decision trees, we cannot include more than one 'or' condition.
- It is used when there are more number of properties.
- It can be used for complex logic as well.
- It is constructed of branches and nodes
- A decision tree's objective is to provide an effective means to visualize and understand a decision's available possibilities and range of possible outcome.

Page No.
Date: / /

SRS DOCUMENT :

- A software Requirement Specification (SRS) is a document that describes the nature of a project, software or application.
- In short, the purpose of this SRS document is to provide a detailed overview of our software product, its parameters and goals.
- This document describes the project's target audience and its user interface, hardware and software requirements

⇒ SRS structure and its parts:

1. Introduction
2. General description
3. Functional requirement
4. Interface requirement
5. Performance requirement
6. Design constraints
7. Non-functional Attributes
8. Preliminary Schedule and Budget
9. Appendices

- Introduction:

(i) Purpose of this Document:—
At first main aim of why this document is necessary and what's purpose of document is explained and described

(ii) Scope of this document :-

- In this overall working and main objective of document and what value it will provide to customer is described and explained
- It also includes a description of development cost and time required

(iii) Overview -

In this, description of product is explained, It's simply summary or overall review of product.

- General description :-

→ It includes objective of user, characteristic, feature, benefits, about why its importance is mentioned.

→ It also describe features of user community and application domain.

- Functional Requirement :-

→ It Describes the possible effects of a software system, in other words, what the system must accomplish.

→ Each functional requirements should be specified in following manner:

- Description
- Criticality
- Technical issues
- Cost and schedule
- Risks

Page No.
Date / /

- Interface Requirements:

→ In this software interfaces which mean how software software program communicates with each other or users either in form of any language, code or message are fully described and explained.

→ Each functional requirements should be specified in following manner:
GUI
API
CLI

- Performance Requirements:

→ In this, how a software system performs desired functions under specific condition is explained.

→ Specifies speed and memory requirement

- Design Constraints:

→ In this constraints which simply means limitation or restriction are specified and explained for design team.

→ Examples may include use of a particular algorithm hardware and software limitations, etc.

- Non-functional Attributes:

→ In this non-functional attributes are explained the are required by software system for better performance

Page No.:
Date : / /

- An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, etc

- Preliminary schedule and Budget:
→ In this initial version and budget of project plan are explained which include overall time duration, required and for development of project.

- Appendices:
In this, additional information like references from where information is gathered, definition of some specific terms, acronyms, abbreviation etc. are given and explained

Goals of SRS document :-

1. Feedback to customer
2. Problem decomposition
3. Input to design specification
4. Production validation check.

Quality characteristics of a good SRS :-

1. Complete :- The SRS should include all the requirement for the software system, including both functional and non-functional requirements

Page No.
Date / /

- Consistent: The SRS should be consistent in its use of terminology and formatting and should be free of contradictions.
- Unambiguous: The SRS should be clear and specific and should avoid using vague or imprecise language.
- Traceable: The SRS should be traceable to other documents and artifacts, such as use cases and user stories, to ensure that all requirement are being met
- Verifiable: The SRS should be verifiable, which means that the requirement can be tested and validated to ensure that they are being met.
- Modifiable:- The SRS should be modifiable, so that it can be updated and changed as the software development process.
- Prioritized: The SRS should prioritize requirements, so that the most important requirements are addressed first
- Testable: The SRS should be written in a way that allows the requirements to be tested and validated
- High-level and low level: The SRS should

Page No.
Date : / /

- Provides both high-level requirement and low-level Requirement.
- Relevant
- Human-readable.
- Aligned with business goals.

IEEE Provides Standard for SRS

1. Introduction
   - Purpose
   - Scope / Intended Audience
   - Definition, Acronyms and Abbreviation
   - References / contact information / SRS team number
   - Overview

2. Overall Description
   - Product Perspective
   - Product functions
   - User characteristics
   - General Constraints
   - Assumptions and dependencies.

3. Specific Requirement
   - External interface requirement
   - functional requirement.
   - performance requirement
   - Design Constraints
   - Logical database requirement
   - Software System attributes

Page No.
Date / /

4. change Management process
5. Document approval.
- Tables, diagram and flowchart
- Appendices
- Index.

☆ Importance of SRS:
- Clear communication.
- Basis for Agreement.
- Guidance for development.
- Quality Assurance
- Change Management
- Project - planning and estimation.

Adv:
- Clarity and Understanding
- Alignment with stakeholders
- Basis for Development.
- Quality Assurance
- change management
- legal and Regulatory compilance

Disadv:
- Time-consuming
- scope creep
- Ambiguity and incompleteness
- Maintenance overhead
- Rigidness
- communication barrier.

Page No.:
Date : / /

Software Quality Assurance

- Definition: SQA is a process ensuring software products meet quality standards and requirements
- objective: Prevent defects, improve quality, and insure customer satisfaction.
- Process: Implement quality control and management activities throughout develop ment
- Techniques: Uses code reviews, inspection, audits and testing to evaluate software quality.
- Standards: Adheres to international standard like iso 9001, IEEE 730, and CMMI
- Quality Attributes: Focuses on reliability, maintainability, usability, efficiency, and functionality
- Continous Improvement: Monitor process and products to identify a implemnt improvements
- Metrics: Employs metrics like defect density and code coverage to evaluate quality

Page No.
Date: / /

- Training : Emphasizes skill development for developers and testers to meet quality standards
- Documentation : Requires proper documentation for transparency and traceability.

⇒ **Validation / Black Box :-**

- Validation ensures that the software product meets the end-user requirements and is fit for its intended purpose, "Are we Building the right product".
- Ensures that the product met its design specifications.
- Focused on satic analysis techniques. checks for consistency, completeness, correctness
- Code reviews
- Static analysis tool.
- Inspection of requirements, design and code documentation.

⇒ **Verification / White Box :-**

- Verification ensures that the software product is designed and developed according to the specified requirements and standards.
"Are we Building Product Right.
- Process of ensuring that the product met its

its design specifications.

- Focuses on static analysis techniques.
- check for consistency, completness, correctness
- code reviews
- Static analysis tools
- Inspection of requirement, design and code documentation.

Software Quality Factors :-

- The various factors, which influence the software, are termed as software factors.

- They can be divided into two categories
- The first category of the factors of those that can be measured directly such as the number of logical errors

- Second category clubs those factors which can be measured only indirectly. For example, maintainability but each of the factor is to be measured to check for the content and the quality control.

- Several model of software quality factors and their categorization have been

Page No.
Date / /

suggested over the years. The classic model of software quality factors, suggested by (Mccall in 1977)

- The 11 factors are grouped into three categories
- Product operation factors :- Correctness, Reliability, Efficiency, Integrity, Usability
- Product revision factors : Maintainability, Flexibility, Testability.
- Product transition factors : Portability, Reusability, Interoperability.

⇒ Difference between Verification and Validation

- Verification

→ It includes checking documents, design, codes and program
→ Verification is the static testing.
→ It does not include the execution of the code
→ Method used in verification are reviews, walkthroughs, inspections and desk-checking
→ It check whether the software conforms to specification or not.
→ It can find the bugs in the early stage of the development.
→ The goal of verification is application and software architecture and specification.
→ Quality assurance team does verification.

→ It comes before validation.
→ It consist of checking of document, files and is performed by human.
→ Verification is for prevention of errors
→ Verification is also termed as white box testing or static testing as work product goes through reviews.
→ Verification find about 50 to 60% of the defect.
→ Verification is about process, standard and guideline.

⇒ Validation:

→ It includes testing and validating the actual product.
→ Validation is the dynamic testing
→ It include the execution of the code
→ Method used in validation are Black Box Testing, white Box Testing and non - functional testing
→ It check whether the software meets the requirement and expectations of a customer or not.
→ It can only find the bugs that could not be found by the verification process
→ The goal of validation is an actual product.
→ Validation is executed on software code with the help of testing team

Page No.
Date / /

→ It comes after verification.
→ It consist of execution of program and is performed by computer.
→ Validation is for detection of errors
→ Validation can be termed as black box testing or dynamic testing as work product is executed
→ Validation finds about 20 to 30% of the defects
→ Validation is about the product.

⇒ **SQA Plan :**

- SQA plan defines the QA process in detail, containing the procedures, tools and techniques employed to ensure a product align with defined standards or the software requirements specification (SRS).
- The SQA plan provides a roadmap for instituting SQA
- It serves as a template for SQA activities to the SQA team
- It includes purpose purpose and scope of the document and indicate those software activities that are covered by quality assurance.
- It list all the applicable standards,

review and Audit details, test, records training, risk management etc.

SQA plan template.

I. Purpose of plan
II. Reference
III. Management
- organisation
- task
- Responsibility

IV. Documentation
- Purpose
- Required Software Eng. document.

V. Standard practice and conventions
VI. Review and audits
- purpose
- Review Requirement

VII. Test
VIII. Problem reporting and corrective Action.
IX. Tools Techniques and methodologies.
X. Code control.
XI. Media control.
XII. Supplier Control.
XIII. Record collection and maintenance.
XIV. Training
XV. Risk Management.

Page No.
Date / /

⇒ **SQA framework :-**

- The SQA framework encomparses various process and activities aimed at ensuring that software products and processes meet specified quality standards.

- Software Quality Framework connects the customer view with the developer's view of software quality and its treats software as a product.

- Quality Planning : This phase involves define quality goals and objectives for the software project.

→ It includes identifying quality attributes, standards, and metrics that will be used to measure and evaluate the quality of the software.

- Quality Assurance : Quality Assurance activities focus on the processes used to develop and maintain the software.

→ This includes establishing standards and procedures, conducting review and audits, throughout the software development lifecycle.

- Quality Control : Quality control activities involve monitoring and evaluating the

Date : / /

actual product to ensure that it meets quality standards

- Quality Improvement :- Continuous quality improvement is emphasized in the SQA framework.

→ This involves analyzing quality data, identifying areas for improvement, implementing corrective and promoting a culture of quality within the development team.

- Process Improvement : Process improvement activities focus on enhancing the software development processes to increase efficiency, effectiveness and quality.

- Documentation and Reporting :- Comprehensive documentation and reporting are essential component of the SQA framework.

Page No.
Date / /

ISO 9000 Models in S.E :-

- ISO 9000 is a series of international standards related to quality management and assurance.

- In the context of software engineering, these standards provide guidelines for implementing effective processes and ensuring high-quality software product

- Key Component of ISO 9000 :

→ Quality Management System (QMS) :

- Defining policies and objective for quality, Documentation of procedures and process, Monitoring, measurement, and analysis of process, improvement opportunities and corrective actions.

→ Management Responsibility :

- Top management commitment to quality Establishment of a quality policy Ensuring adequate resources for QMS implimentation Reviewing QMS performance regularly.

→ Resource Management :

- Provision of necessary resources (human, infrastructure, work environment) competence and training of personel infrastructure maintenance and improvement

Page No.:
Date: / /

⇒ Product Realization:

- Requirements, determination and communication Product design and development verification, validation and testing Release, delivery and post-delivery support.

⇒ Measurement, Analysis and Improvement

- Monitoring and measurement of process and products internal audits to ensure compliance corrective and preventive actions continual improvement of the QMS

⇒ ISO 9000 Principles:

1. Customer focus
2. Leadership
3. Process approach
4. Continual improvement

⇒ ISO 9001: This standard applies to the organizations engaged in design, development, production, and servicing of goods

⇒ ISO 9002: This standard applies to those Organization which does not design product but are only involved in the production

- ISO 9003 : This standard applies to organizations that are involved only in the installation and testing of the product
  For ex : Cisco companies

⇒ Why ISO certification required by Software Industry ?

- This certification has become a standards for International bidding
- It helps in designing high-quality repeatable software products
- It emphasis need for proper documentation.
- It facilates development of optimal processes and totally quality measurement.

⇒ Features of ISO 9001 Requirement :-

- Document control
- Planning
- Review
- Testing
- Handling

⇒ **SEI - CMM Model :-**

- Capability Maturity Model (CMM) was developed by the software Engineering institute (SEI) at Carnegie Mellon University in 1987.
- It is not a software process model.
- It is a framework that is used to analyze the approach and techniques followed by any organization to develop software product.
- It also provides guidelines to enhance further the maturity of the process used to develop those software product.

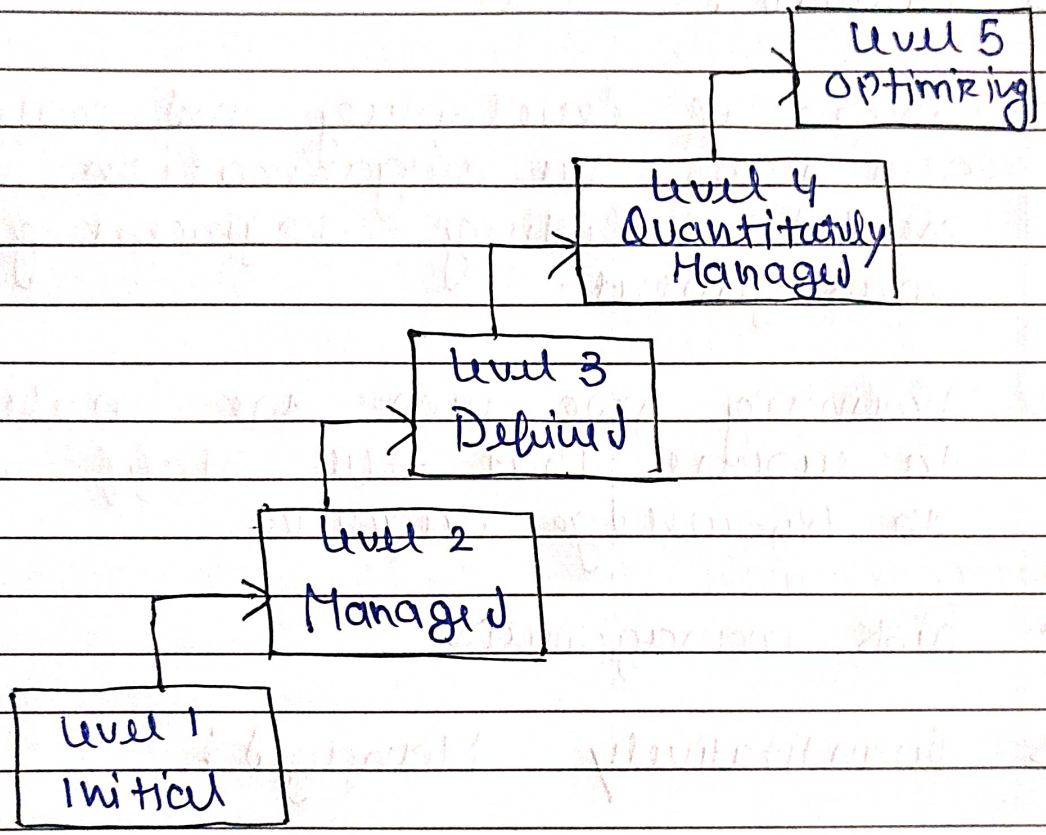
✱ **Importance of CMM :**

- Optimization of Resources : CMM helps business make the best use of all of their resources, including money, labor and time.
- Comparing and Evaluating : A formal framework for benchmarking and self-evaluation is offered by CMM
- Management of Quality : CMM emphasizes quality management heavily
- Enhancement of Process : CMM gives busines

a methodical approach to evaluate and enhance their operations

- Increased output: CMM seeks to boost productivity by simplifying and optimizing processes.

⇒ Characteristic of the Maturity levels :-



⇒ Initial (process unpredictable and poorly controlled)

- No engineering management, everything done on ad hoc basis
- Software process is unpredictable with respect to time and cost.
- It depends on current staff, as staff change so does the process

Page No.:
Date : / /

⇒ Repeatable (basic project management)

- Planning and managing of new project are based on the experience with the similar projects.
- Reliantic plans based on the performance based on the previous project.

⇒ Defined :

- Process of developing and maintaining s/w across the organization is documented including engineering and management.
- Training program are implemented to ensure that the staff have skills an knowledge required
- Risk management

⇒ Quantitatively Managed :

- Organization set quantitative goals for both product and process.
- Here process is predictable both with respect to time and cost.

⇒ Optimized (Continous Process improvement)

- Here organization analysis defect to determine their causes and goals is to preventing the occurrence of defects
- Here company continously improve the process performance of their projects.

⇒ **Information Modelling :-**

- Information modelling is the process of creating a conceptual representation of information in a system.
- It involves identifying the important entities, attributes, relationship, and constraints in a system and representing them in a way that is easy to understand and communicate.
- Information modling is used in software engineering to develop a clear understanding of the system being developed and to facilitate communication between stakeholders

Adv:

- High precision and accuracy.
- Requires less labor.
- Robustness against external force and error accumulation
- Accurate dimension can be obtained just by knowing the coordinates and distance between the two reference point.

Disadv:

1) The coordinate measuring machine are very costly.

2) CMM are less portable.