

INTRO TO SOFTWARE ENGINEERING

 Software :-

Software is a set of Programs, Procedures, data and documentation that tell a computer how to perform specific task.

characteristics:-

- Intangible → you cannot physically touch software.
- Highly Complex → It may contain millions line of code.
- Easy to Modify.

Type of Software

System

Helps run the computer hardware

- OS
- Device Drivers
- Utility Programs

Application

Software design for user to perform task

MS world
XL
chromie

→ Software Engineering



is the systematic, disciplined and measurable approach to developing, operating & maintaining high quality Software.

Needed

- 1→ Increasing size & complexity of Software
- 2→ To reduce development Cost
- 3→ To improve quality using Structured Method
- 4→ To complete Projects on time with proper Planning

Characteristics of Good Software:-

1- Correctness :-

Software must meet all requirements correctly.

2- Efficiency :-

Should use CPU, memory, storage efficiently.

3- Reliability.

4- Usability → Easy to learn and understand.

5- Maintainability → Easy to modify & update in future.

6- Security.

Protect data from unauthorized access.

7- Portability :-

Should work on different platforms.

Software Myths

software myths are false beliefs about software development followed by managers, users & developers.

Type of Myths

A → Management Myths

- Adding more people speed up development
- Once coding starts major changes will not be required.

B → User Myths.

- Software requirements change easily
- A general description is enough to start the project.

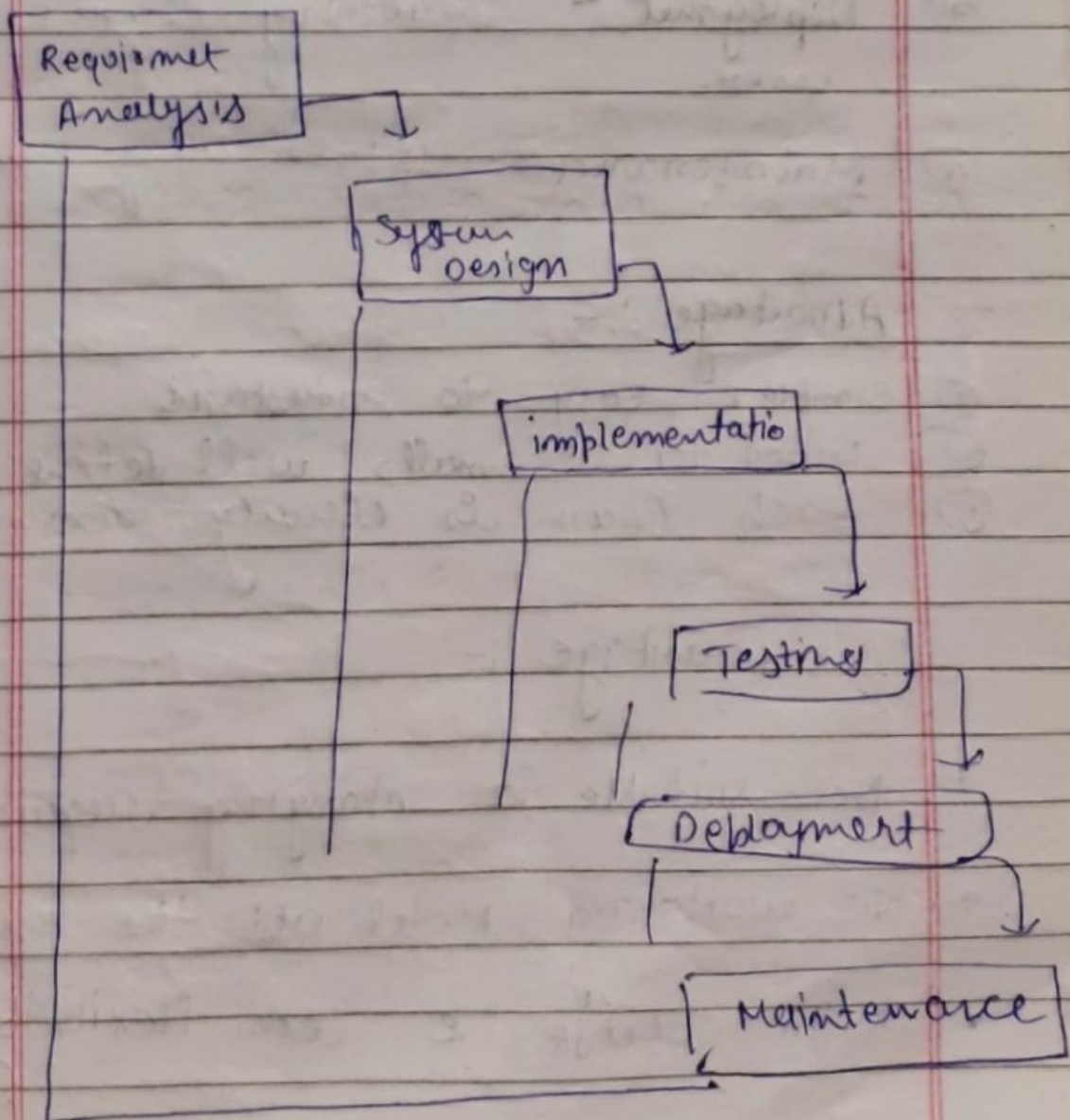
C → Developer Myths

- Working program is the only deliverable
- Project is complete when coding is complete

SDLC Model :-

1- Waterfall Model :-

The waterfall Model is the oldest and most widely used SDLC Model where development flows in step by step (linear) manner. Each phase must be completed before the next begins.



1 $\Rightarrow$ Phase of Waterfall Model

- ① Requirement Analysis:- Gathering user needs.
- ② System Design: System Structure, Architecture
- ③ Implementation (coding):- writing actual program.
- ④ Testing:- Finding & fixing errors
- ⑤ Deployment - Delivering system to users.
- ⑥ Maintenance - updates, bug fixes.

Advantage :-

- ① Simple & easy to understand
- ② Good for small, well define Project
- ③ Each Phase is clearly documented

Disadvantage :-

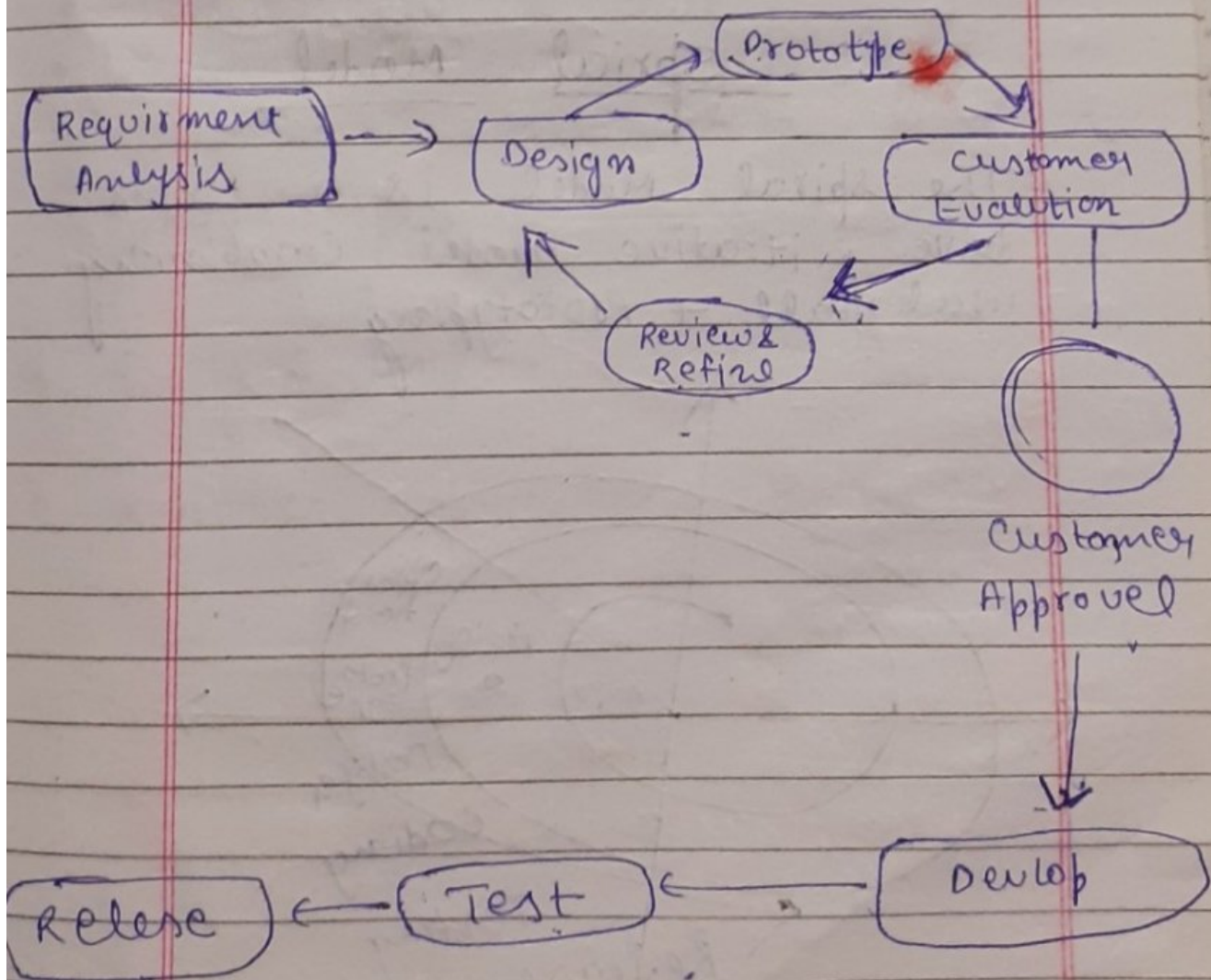
- Not suitable for changing requirements
- No working model until the end
- High risk & low flexibility

② Prototyping Model :-

Prototyping Model builds a working model (Prototype) early to clarify requirements before actual development starts.

Step in Prototyping Model :-

- 1- Collect initial requirement
- 2- Build quick Prototype
- 3- Show to user & take feedback
- 4- Refine Prototype
- 5- Develop final system



Advantage :-

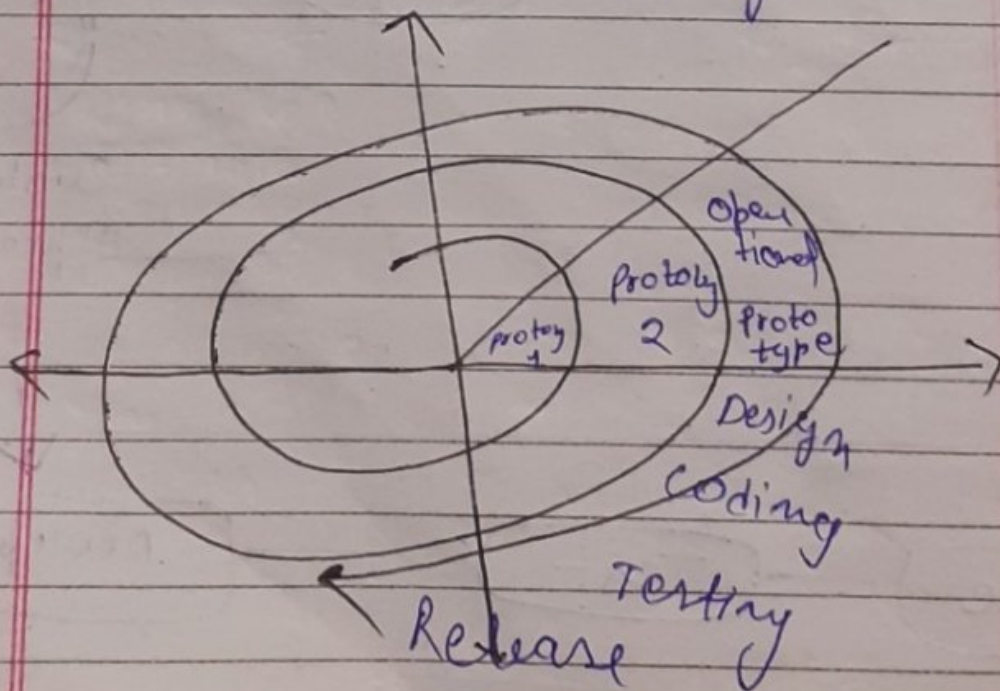
- ① user requirement become clear
- ② Reduces risk of failure
- ③ user gets early visibility of system

Disadvantage

- ① can increase development time
- ② Not suitable for large system

Spiral Model

The spiral model is a risk driven, iterative model combining waterfall + Prototyping



- ① Planning → Objective & Constraints
- ② Risk Analysis - identifying risk
- ③ Engineering - Development & testing
- ④ Evaluation - User Review

Advantage is:

- Best for large, complex Project
- Handle high risk
- Requirement evolve gradually

Disadvantage

- Costly & Complex
- Requires experienced developers.

SRS (Software Requirement Specification)

SRS is a formal document that describes what the software will do.

Importance

- Clears understanding b/w user & developer.
- Reduce development time.
- Helps in design, testing & maintenance.

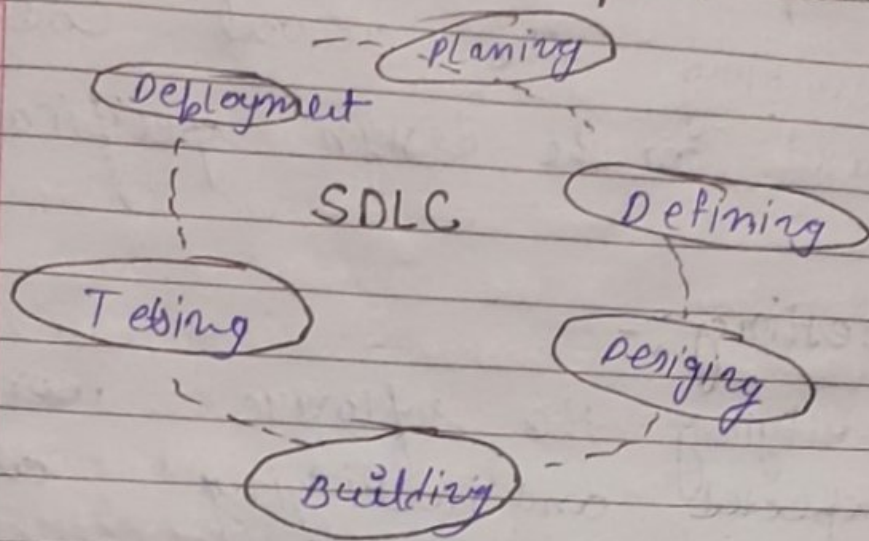
Characteristics of a good SRS

- 1 - Correct
 - Complete
 - Consistent
 - Verifiable
 - Modifiable

Component of SRS

- ① Functional Requirement
- ② Non " " "
- ③ System Constraints
- ④ Interface
- ⑤ Performance Requirement

*) SDLC (Software development life cycle)



SDLC is the structured process for developing high quality software.

Main Phase

- ① Requirement — In this phase developer meets the customer and collect all requirements of the software.
- ② System design:-

Here the system structured is planned, Database design, user interface design & architecture are prepared.

③ Implementation :-

writing the actual code based on the design specification

④ Testing :-

Verifying the software works as expected and identifying any bugs or errors.

⑤ Deployment :- Maint

② Provide ongoing support, bug fixes and updates after the software has been deployed

⑤ Maintenance :- Deployment :-

Providi

Releasing the finished software to the productive environment for end user.

releasing the soft in

Software Design & Planning

* Software Design :-

Software design is the process of creating a plan or blueprint for building software.

* It focuses on deciding how the system will work internally.

Features :-

- 1) Define Architecture
It describes the overall structure of the system
- 2) Modularity
- 3) Abstraction
- 4) Encapsulation
- 5) Reusability
- 6) Maintainability

Types of Software Design :-

- 1) Architectural design
- 2) High-level Design
- 3) Low level Design

1) Architectural $\Rightarrow$ Defines the overall system Architecture, Frameworks layers & Communication style.

2) HLD (gives the design of major modules data flow, database design and system interface)

3) Low level Design

Describe the internal logic of each module.

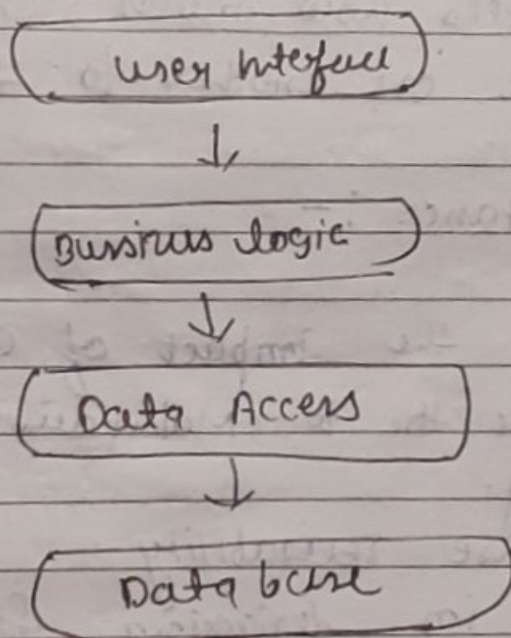
Importance $\Rightarrow$

- 1) Reduce development time & cost
- 2) Improves software quality
- 3) Ensure better communication
- 4) Helps in risk identification

Architectural Design

Architectural design is the high level design of a software system.

It defines overall structure of the software, the major components and how they interact with each other.



- 1) User interface $\Rightarrow$ handle user input/output
- 2) BL $\Rightarrow$ Contains the main logic of the system
- 3) Data Access layer $\Rightarrow$ Manages database connection & query
- 4) Data base layer $\rightarrow$ store data permanently

Coupling & Cohesion

Coupling :-

Coupling is the degree of dependency between software modules.

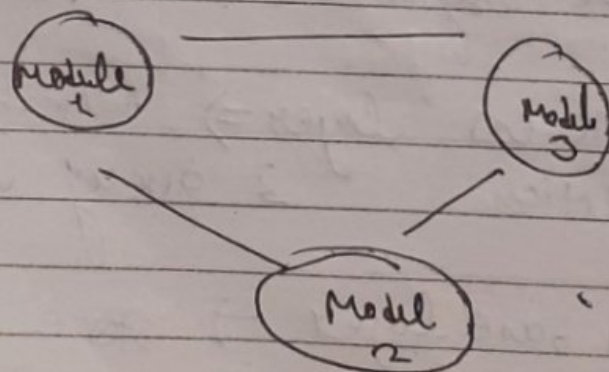
- It tells how much one module relies on another to function.

Importance :-

- Reduce the impact of changes
- easier to maintain & debug.
- increase reusability
- Help in designing flexible soft

Type

- 1) Content 2) Common 3) External
- 4) Control 5) Stamp 6) Data



Cohesion :-

is the degree to which the element of a module belong together.

Importance

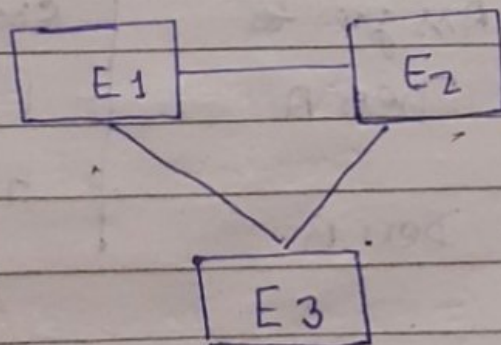
- 1) improve maintainability
- 2) increase reusability
- 3) facilitates testing & debugging

Type

- ① Coincidental ② Logical ③ Temporal
④ Procedural ⑤ Communicational ⑥ Functional

High cohesion $\Rightarrow$ Means Module does one task well

Low cohesion $\Rightarrow$ Module tries to do many unrelated things.



Software matrix

Date _____
Page _____

Software Matrix is a tabular representation used to organize and analyze various aspects of software system.

⇒ used to compare track or map, software requirements, Modules testing, Performance

⇒ Help in decision-making, Project tracking

Project Matrix :- is used to track and manage the progress of a software project

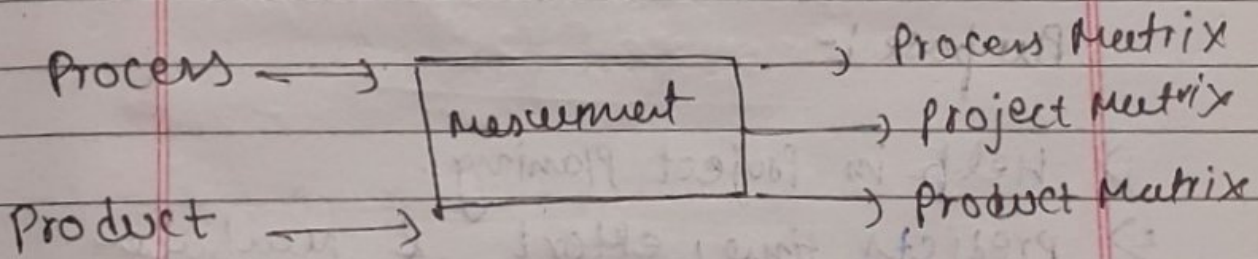
Example

Module / Task	Assigned to	Start Date	End Date	Notes
Login Module	Dev A	1/12/25	5/12/25	Completed
Payment Module	Dev B	2/12/25	6/12/25	In Progress

Product Matrix :-

is used to track & compare software product or modules against features or attribute.

- * useful for tracking product quality, features coverage & enhancement.



- 1) Not
- 2) Completed
- 3) In Progress



COCOMO

The Constructive Cost Model is a software Project estimation Model that Predicts the effort, time & cost required for software development based on the LOC (lines of Code)

Purpose :-

- 1) Help in Project Planning
- 2) Predicts time, effort & Manpower
- 3) Useful for budgeting & Scheduling

Types of Cocomo

- 1) Basic - Estimate effort using only Project Size
- 2) Intermediate :-
Considers size and other Cost drivers
- 3) Detailed - Considers size, Cost drivers, & Phase wise effort

Formula

Effort (Person-Months) =

$$a \times (KLOC)^b$$

Types of COCOMO Model

1) Basic Model :-

is used for quick & rough cost calculation for the software

It calculates the effort, time, & no of people required

$$(E) \text{ Effort} = a (KLOC)^b$$

$$\text{Time (T)} = c (E)^d$$

$$\text{people required} = \frac{E}{T}$$

2) Intermediate :-

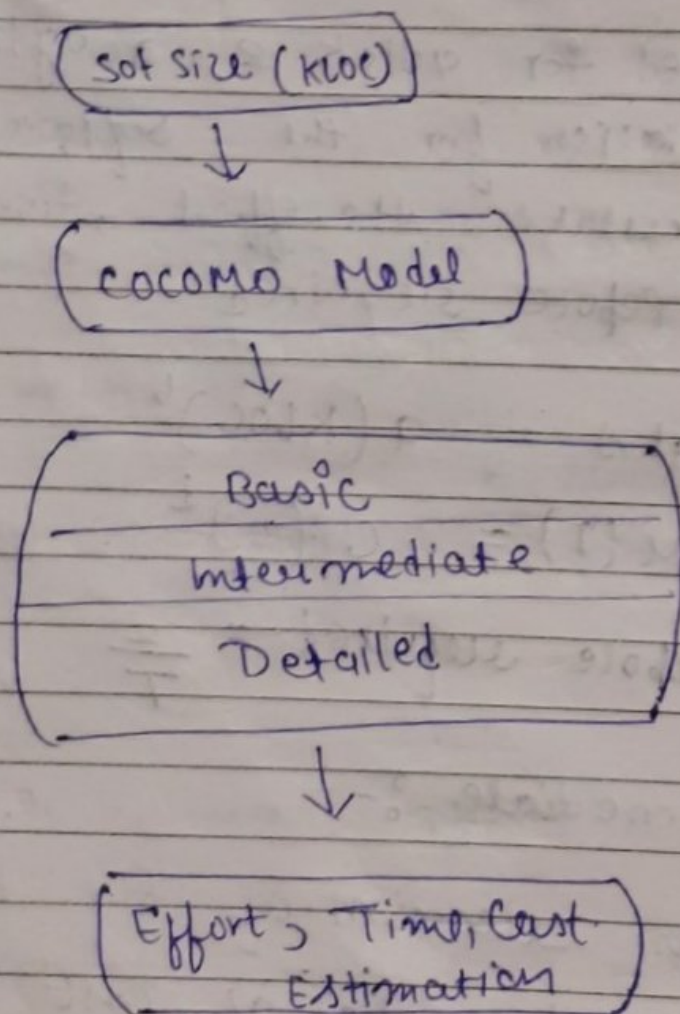
is an extension of the basic model & include a set of cost drivers to calculate the estimate with better accuracy.

$$E = a (KLOC)^b * EAF$$

$$\text{Time (T)} = c (E)^d$$

Type of Software Project

- 1) organic $\Rightarrow$ Small, Simple Software
- 2) Semi detected $\Rightarrow$ Medium Complexity
- 3) Embedded — Complex, hardware / Software dependent



Software Maintenance & Risk Management



Software Maintenance :-

is the process of modifying & updating a software system after it has been delivered.

to correct faults, improve performance or adapt to it a change environment

Purpose

- 1) fix bugs & error
- 2) improve performance

Types :-

- 1) Corrective :- fixes errors or bugs found after the software is delivered.

2) Adaptive :-

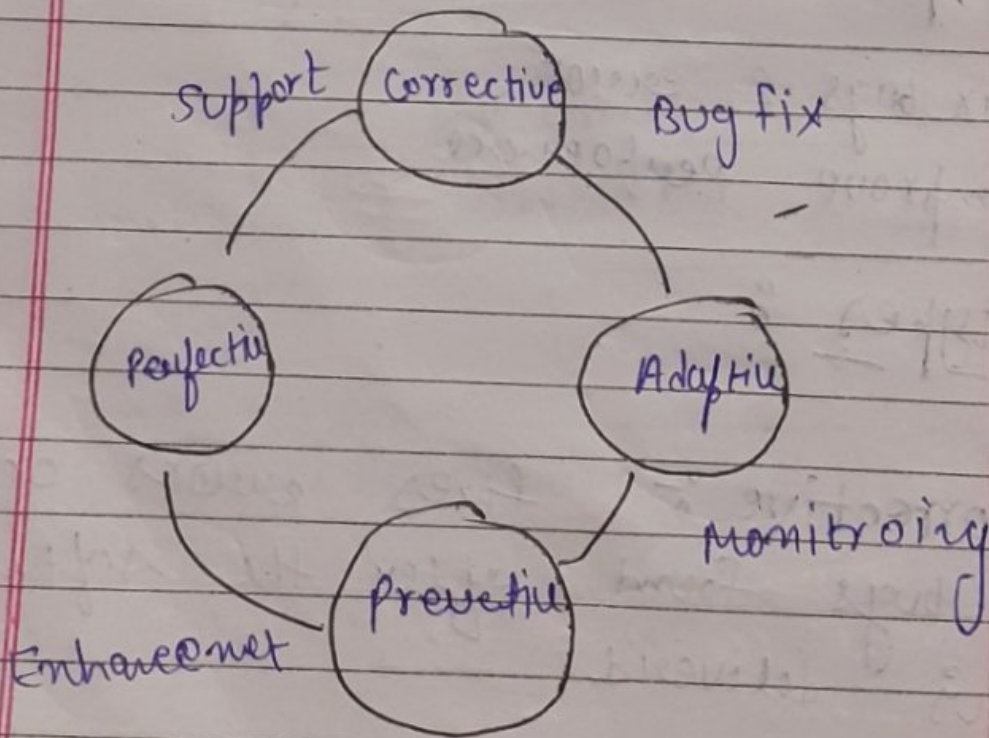
Modifies software to work in a change environment

3) Perfective ;

improve performance, efficiency or usability.

4) Preventive :-

prevent future problems by improving maintainability



Risk Management :-

Risk Management is the process of identifying, analyzing & mitigating risk in a software project to ensure its success.

helps $\Rightarrow$ Reducing project failures, delays & extra costs.

Steps of Risk Management

- 1) Risk identification :-
find all possible risk in the project
- 2) Analysis $\Rightarrow$ Analyze probability of risk occurring & impact project
- 3) Planning :- Plan to avoid risk
- Risk Evaluation :-
decide which risk are most critical & need attention

Risk Treatment

Plan how to deal with risk

Strategies

Avoid

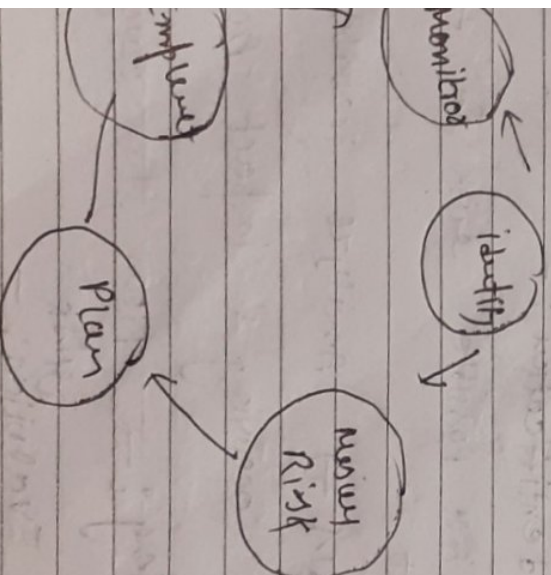
Mitigate

Transfer

Monitoring

keep track of risks and effectiveness of mitigation.

INTERM



Type of Risk Management

1) Strategic

focus on long-term business goals & strategy

2) Operational

⇒ Deals with risk from internal processes, people or system

3) Financial

Handle risk related to money investment or cash flow

4) Compliance

Ensure adherence to laws, rules & regulation.

5) Reputational

protect the company's image & public trust

6) Project

manages risks specified to projects to meet time, cost & quality goals.

S-OFCRP

Verification :-

is the process of checking whether the software is being & will correctly according to the specified requirements.

- It is the static process
- checks documents, design, requirement code structure
- Ensure the software is being made correctly
- Save time & cost

Verification Activities

- ① Reviews
- ② Walkthrough → Developer Explain the design to team members
- ③ Inspection → Formal checking of documents for error
- ④ Static Analysis → using tools to find errors in code without executing it.

② Validation

is the process of checking whether the final software product meet the user's needs & expectations

Are we building the right Product

- dynamic Process
- focus on the functionality & correctness
- Done After verification

Activities

- Functional testing
- Acceptance testing
- System testing
- Performance



ISO 9000

ISO 9000 is a set of international rules that tell companies how to maintain good quality in their work.

- Focuses on Process quality
- follows standard procedure

Component of ISO

ISO 9000 → basic concept & Terminology

9001 → quality system management requirement

9004 → Performance improvement guidelines

19011 → Auditing guidelines

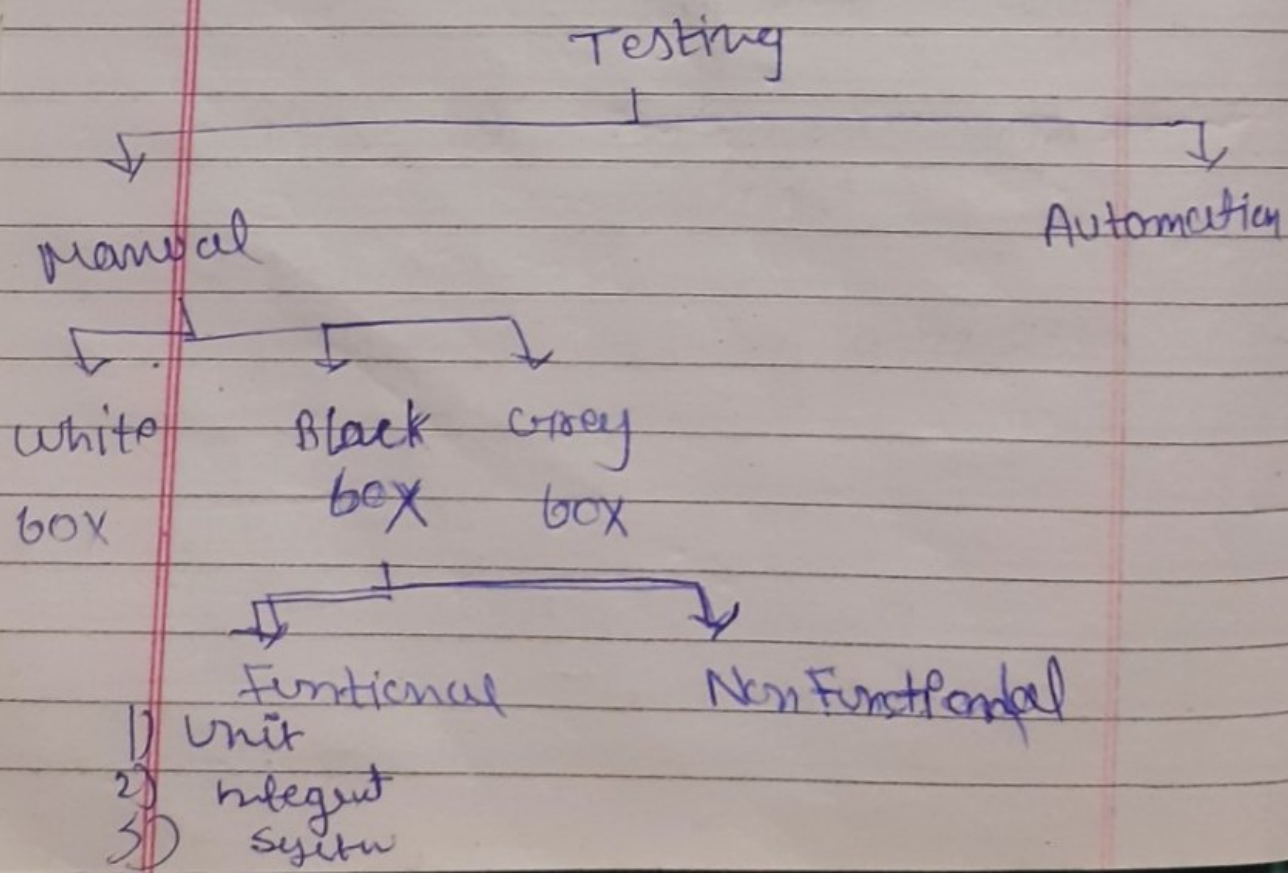
- 1) Better process control
- 2) Improved communication
- 3) High customer trust
- 4) continuous improvement

Software Testing

is the process of checking a software system to find defects and ensure that it works correctly according to user requirements.

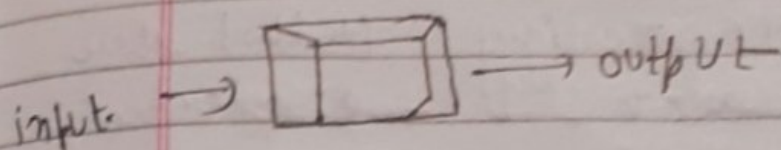
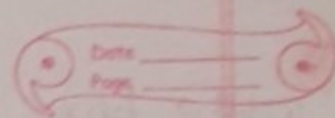
⇒ help ?

- 1) The software is reliable
- 2) error free
- 3) Performed the require function
- 4) Ensure quality





Black Box Testing



is a software Testing method where the tester checks the software without looking at the internal code or logic.

only input & output are tested

Advantage

- 1) Simple & Easy to Perform
- 2) Work even if code is hidden
- 3) Good for large Application

Disadvantage

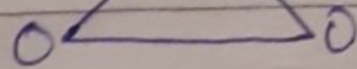
- Limited coverage
- Hidden errors in logic
- Test case design may be inefficient

Example -> logi Page

Regression

Functional Testing

Types



NFT

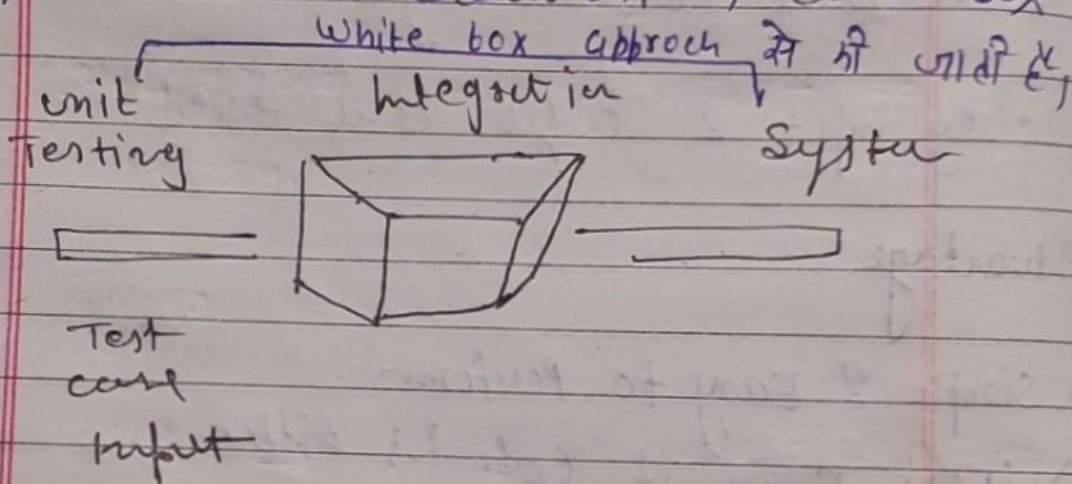
(Non Functional)



White box Testing

is a software testing method where tester check the internal structure, logic & code of the program

Also called Glass box, clear box



Advantage:-

- 1) Optimize possible
- 2) hidden error detected
- 3) Thorough testing of code
- 4) High coverage

Disadvantage

- 1) Complex
- 2) time consuming
- 3) Costly process
- 4) Not suitable for large system
- 5) Focus only on code

Levels of Testing

Level of Testing refers to the different stages in which a software is tested.

- Each level tests the software from a different angle. Make sure it works correctly.

1) Unit Testing

is the first level of testing. At this level, individual units or small parts of the code are tested.

2) Integration

After unit testing the tested are combined together & then their interaction is tested.

3) System

entire software system is tested

4) Acceptance :-

last level of testing before releasing the product to customer.

Types

1) Alpha $\Rightarrow$ done by developers tester at the developers site

2) Beta $\Rightarrow$ done by real user at their own place

1) CM with its Level

2) Describe the quality parameter used in software sys.

3) Software Reliability

4) Feasibility Study and its type

5) Software quality assurance & write important guidelines

6) Debugging & approach

4) Software & Program

5) function & non func req's

6) Cost estimation

quality parameter used in softw

Prototype