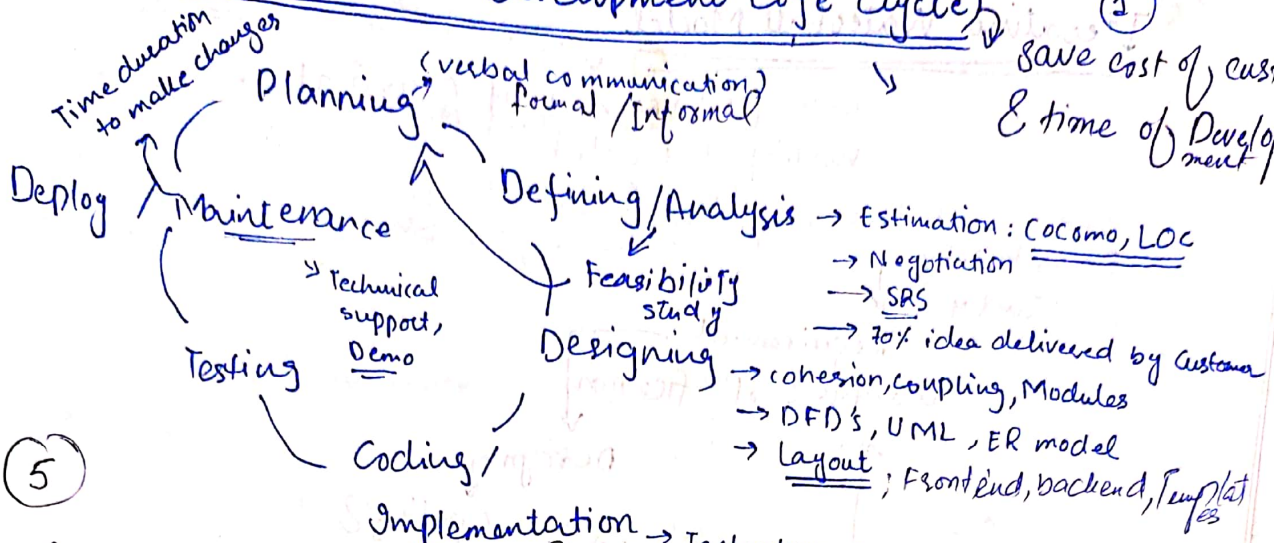
SE & SQA

www.bba
w3bzal

SDLC → Develop S/w through Phases (Software Development Life Cycle)

(1) Save cost of cust. & time of Development



Inception / Initiation → Planning → Executing (~~Designing~~) → Controlling & Monitoring → ~~Execution~~ → Closure

Develop software through phases: because

- ↳ customer invests money, time
- ↳ company wants to get multiple projects by satisfying customer.
- ↳ Genuine cost
- ↳ complete in Time Frame

2 Entities

1) Customer
2) Service provider

eg: See home after it's built

(1) Classical Waterfall Model

"Linear sequential life cycle Model"

: customer clearly know its requirements. works as a base for other Models

↳ sytematic way of steps

Platform Available
Team Available
Budget

Feasibility study → Req. Analysis & specifications → Design → coding & unit testing → System testing & integration → Maintenance

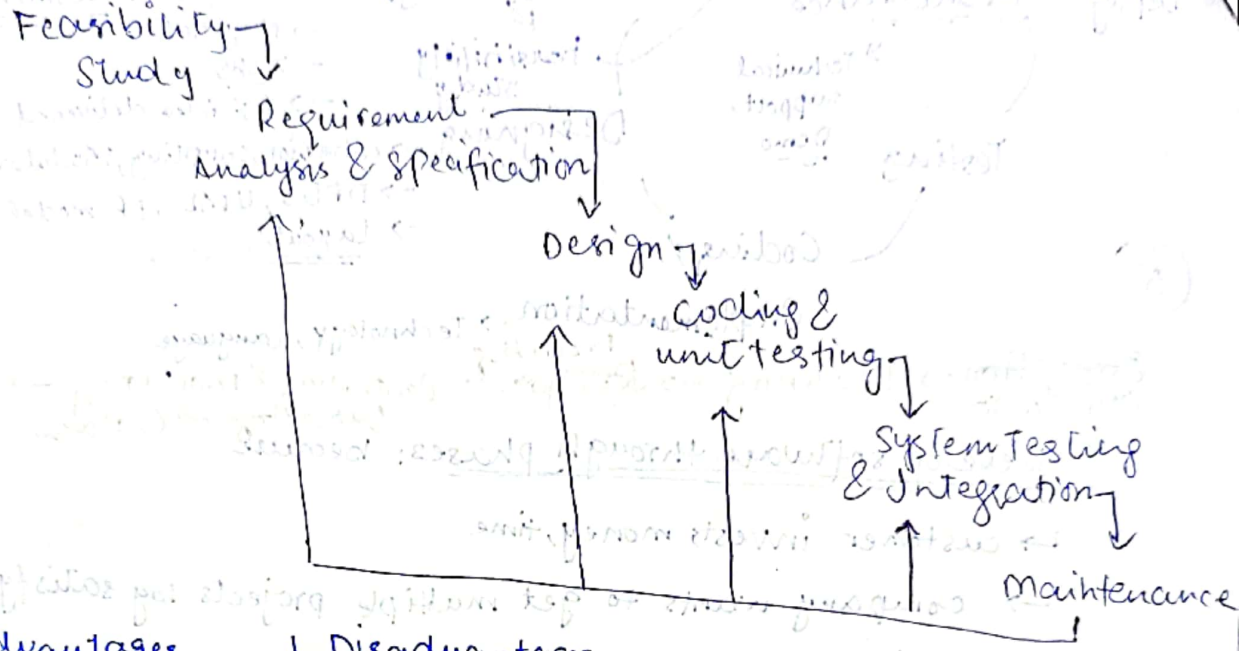
Maintenance: Fix tech, non-tech problems; 60% Effort; Fix Bugs etc

Advantages
- → Base Model
- → Simple & Easy
- → Small project

Dis-Advantages
- → No Feedback
- → No Experiment
- → No parallelism → No multiple Team work
- → High Risk
- → 60% Effort in Maintenance phase
- → Rigid; no flexibility

Iterative Waterfall Model

modified version of Classical Waterfall Model.



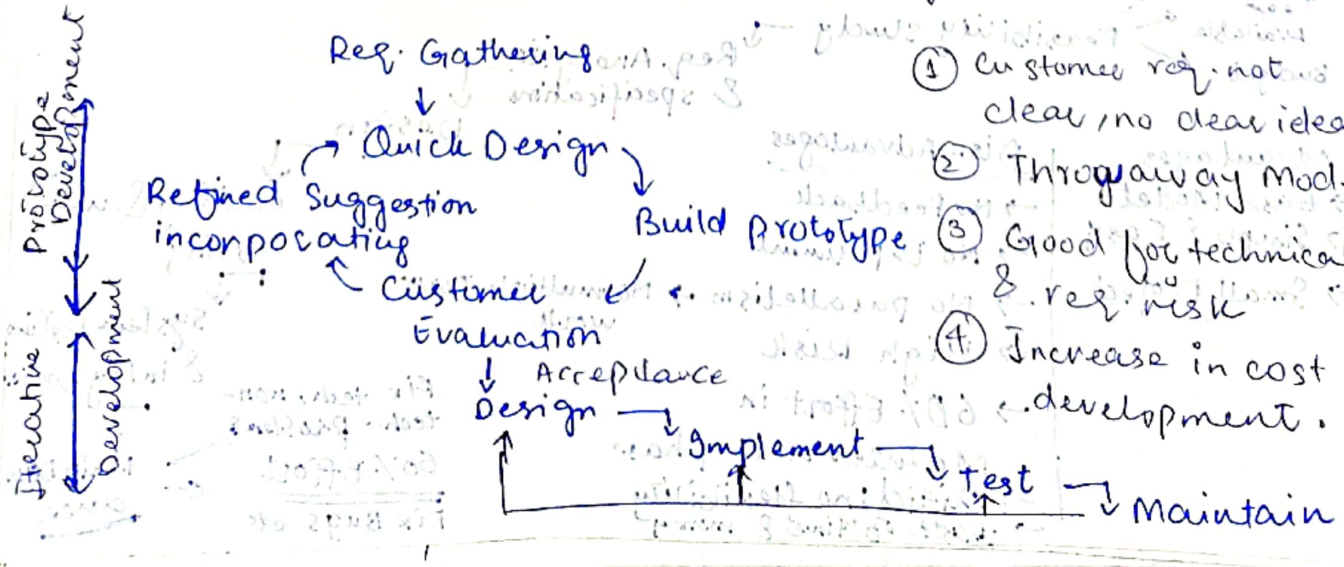
Advantages

1) Base Model
2) Simple & Easy
3) Small Projects
4) Feedbacks → Mistakes / bugs

Disadvantages

1) No phase overlapping / No parallelism
2) No intermediate delivery
3) Rigid (No changes) → fixed requirement
4) Less customer interaction

Prototyping Model → Dummy / Toy Model



1. Customer req. not clear / no clear idea
2. Throwaway Model
3. Good for technical & req. risk
4. Increase in cost of development.

SRS → Before Design

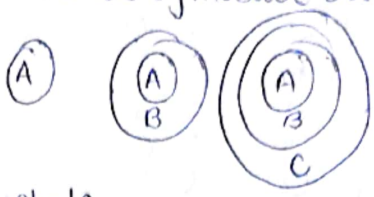
↳ Functional + Non Functional Req.

↳ Use cases: if student forgets password then what? → Apply for leave.

1. Introduction :-
   1) Purpose
   2) Intended Audience
   3) Scope
   4) Definitions → who is principle, Admin → In SMS School Mgmt system
   5) References
2. Description
   1 → User interface
   2) System Interface
   3) Constraints, Assumptions & dependencies
   4) User characteristics
3. System Features & Requirements
   1) Func. Req → how system work, Registeration → login → home page
   2) Use cases
   3) External Interface Req → online fee
   4) Logical database Req
   5) Non-functional Req → safety, security, Data Availability, portable, Reliable
      ↓ Related to Quality
4. Deliver for Approval

Incremental Model → 2 types → ① Staged Delivery Model ② Parallel development Model (3)

→ Module by Module Development



Eg: LMS → Student Module
→ Teacher Module
→ Dean Module
→ Attendance Module
→ Announcements

Requirements (SRS document created, All req. locked)

Build 1 → Design & Development → Testing → Implementation

Build 2 → Design & Development → Testing → Implementation

⋮

Build n → Design & Development → Testing → Implementation

1) Module by Module working; balanced cost wise, man power
2) Customer interaction maximum; less error chances
3) Large projects; Web, mobile application; updations
4) Beneficial for Early Release product Demand
5) Flexible to changes; updations

Spiral Model: Risk

↳ Risk Handling
↳ Radius of spiral = cost
↳ Angular dimension = progress
↳ Meta Model:
Use Multiple models
1) Classical waterfall model: Step by step
2) Iterative WF model: Taking Feedback
3) Prototype Model

Adv: ① Risk Handling ② Large Projects ③ Flexible ④ customer satisfaction

Disadv: ① Complex ② Expensive ③ Time ④ Too much Risk Analysis

if not proper Team ↳

Comparison of Various SDLC Models

Classical Waterfall

1) Basic
2) Rigid
3) Inflexible
4) Not for Real project
5) No Feedbacks
6) Lock on req.

Iterative Waterfall

(Product, Embedded Develo)

1) Basic
2) Problem is well understood
3) Feedbacks
4) Lock on req.

Prototype Model

(User-interface creation)

1) User req. not clear
2) Costly
3) No early lock on requirements
4) User involvement high
5) Re-usability

Incremental model

1) Module by module delivery
2) Easy to test & debug
3) Req. Lock

4) No fixed time to complete the next iteration.

Evolutionary Model

1) Large projects
2) No lock on req.

Same as incremental model

2) Fixed

RAD Model (IBM)

(Rapid Application Development)

(60-90 Days)

OOP

Powerful

1) Time & cost Constraint,
2) Re-usability
3) User at all levels.

Automated tools & techniques

Spiral model

1) Risk, Not for small projects.
2) No early lock on requirements
3) Less experience can work

Agile model

1) Flexible
2) Advanced
3) Parallel
4) Process divided into sprints
5) End date for an iteration is fixed.

Every person is working at same level, Team work

Agility achieved by removing un-necessary activities that wastes time & effort.

AGILE MODEL

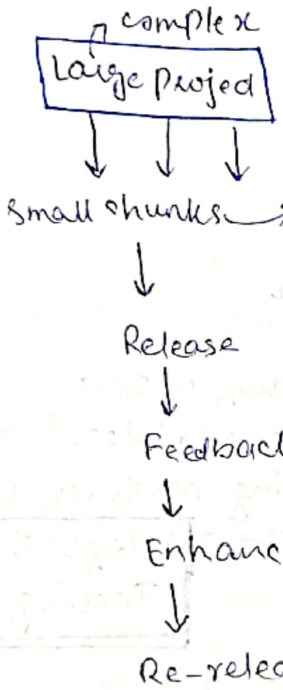
(5)

Latest Model

Used By Major Companies, Google, Adobe, Amazon

Agile means → Move Quickly

eg. update version of Laptop



Advantages:

1. Frequent delivery
2. Face to face communication with client
3. Changes
4. Time

Disadvantages:

1. Less documentation
2. Maintenance Problem

↳ End date for an iteration is fixed, it can't be changed. The development team have to decide to reduce the delivered functionality to complete iteration on time.

Scrum model in Agile Technology.

limited time → Sales, Pharmaceutical.

light weight, incremental, iterative

"Sprints"

development phases divided into stages.

Unit Testing / Tier 1 / Level 1 testing:

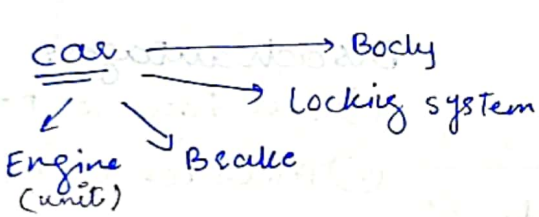
→ check small pieces of code to ensure if individual parts of program working well

↳ At first stage Testing → Multiple Teams

↳ Performed 1st

↳ Done by Developers

Unit → Individual part / Modules / Functions



- Unit Testing problem → undecideable

Integration Testing

→ According to SRS: Combine Modules, & while combining modules, testing done → Integration Testing

① After unit Testing

② Before System Testing

Types of Integration Testing:

① Big-bang → Integrate modules 1 by 1 & Test

② Mixed (Sandwich): Top-down + Bottom-up

③ Top-down → Top Functionality / Top priority Modules Tested first

④ Bottom-up → Least priority tested first

Software Testing ↔ Verification / Validation

check every phase

In process

SDLC Verification

1) Are you building it Right ?
2) Check wheather an artifact confirms its previous artifact.
3) Done by Developer.
4) Concerned with phase containment of errors.
5) Methods involves Review, Inspection, Unit & Integration Testing.
6) Static & Dynamic Activities.

Validation STLC

Software develop

1) Have you build the right thing?
2) Check final product against specification; SRS.
3) Done by Tester.
4) Aim is to make final product error free.
5) Involves System Testing
6) Only Dynamic → run code; drive car

Example : Car Manufacturing

only Review

4 phases:

1) Tyre
2) Engine setup
3) Body
4) Paint / Finishing

Types of Software Testing in Software Engineering

Software Testing

Test Design Cases Approaches

- White Box Testing
- Black Box Testing

Categorized into 4 Levels

- Unit Testing
  - In development phase.
- Integration Testing (Testing phase)
  - Big-Bang
  - Top-down
  - Bottom-up
  - Mixed
- System Testing (complete software)
- Regression Testing
  - Maintence phase
  - upgradation, seed Artificial errors

Based on who is Testing

- Alpha (Development Team)
- Beta (Friendly customers)
- Acceptance (customer)

Perfoemance/ Non-Functional (launch website)

- Volume
- load
- stress
- security
- configuration
- compatibility
- Recovery
- Install

Error Seeding in Software Testing

S = Total Errors seeded in a code

s = Total seeded errors detected during Testing

S = 20

s = 16

N = Total errors in a system

n = Total non-seeded errors

n = 200

$$\frac{n}{N} = \frac{s}{S}$$

$$N = \frac{n \times S}{s} = \frac{200(20)}{16} = 250$$

$$\text{UD errors} = \frac{n \times (S - s)}{s} = \frac{200(20-16)}{16} = 50$$

E - V + 2 , R + 1

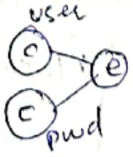
⇒ White box Testing Technique:

1) Path coverage : cyclomatic complexity
multiple entry & exit blocks check, find independant paths.
if → else if → else

⇒ Black Box Testing Technique:

1) Equivalence Class Testing → divide test data in groups
age > 18 AND age > 56. watch movies

2) Cause Effect Graphing



3) State based Testing

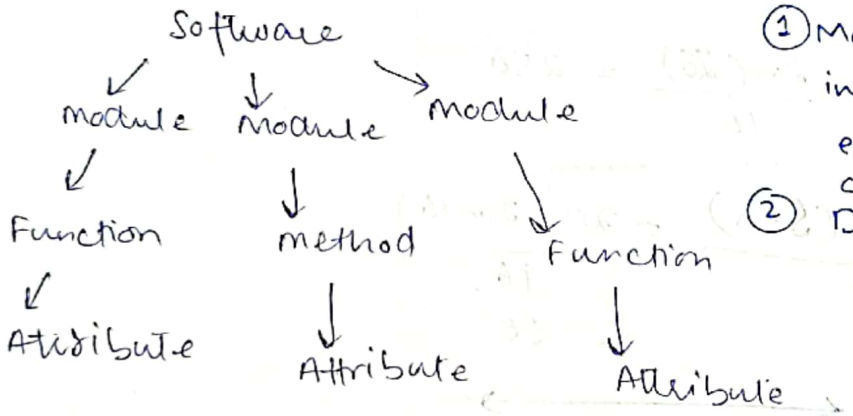
State Transition
→ multiple inputs
→ 3 wrong pin → card block
1 right pin → successful Transaction

After Design Phase :
SRS has been converted → Software Design document

⇒ Cohesion
⇒ Coupling

Module → Functions, Attributes
Class → Functions, Associate Data Structures

Modularity → Attribute of Good Design : Decomposition of a problem into modules.



① Modules are almost independant of each other
② Divide & conquer called Peinciple.

Modularity

Modules should display
1) High cohesion → Togetherness, within module various functions work (Intra module)
2) Low coupling → within modules dependency (Intermodule)
↑ restrictions

Logical cohesion : Error handling, data input, output etc → Attendance mark, view, delete (Attendance System)

Temporal cohesion → Same time

Diff b/w White Box & Black Box Testing | eg. car

White Box → info of internal structure (how created?) → opaque / driven

Black Box → simple input → output (correct or not)

White Box	Black Box
1) Developers can perform white box Testing	1) Test Engineers perform black box Testing.
2) What the software is supposed to do, also how it does	2) What software suppose to do, but not, how it does.
3) Understanding of Programming Lang.	3) No need to understand Programming Lang.
4) Check source code, Logic	4) Check functionality of Product.
5) Control Flow Testing, condition coverage Technique	5) Boundary value Analysis
6) Used extensively at unit testing, also in integration & System Testing	6) Unit Testing, Integration Testing, System Testing, Acceptance Testing

Boundary value Testing:

Technique of Black box Testing

40 — 80

39, 81 Invalid

Min Max

Valid & invalid partitions ?

To design white box Test cases.

Condition → cover all conditions Coverage Technique: WBT

Predicate coverage Technique

```
read a,b,c;
if(a==0 || b==0)
{ print 1;
}
else
{
  if(a==0 || b==0)
  { print 2;
  }
}
```

a=0, b=0

a=0 T

1 condition check
25% coverage

a=1 or b=0
F OR T

50% coverage

To make test cases in WB

max. statements Cover

Statement coverage

Desirable coverage

path testing

White Box Testing :- → Control Flow → Add beneficiary
→ Data Flow → Add beneficiary or not
→ Branch Testing

- Clear Box / Glass Box / Transparent Box Testing / Structural Testing

I/O → [ ] → O/P

↳ Internal structure & working of an application

Tools:
Veracode, cppunit

Diff b/w Test Case & Test Scenario?

Test Scenario	Test cases
1) High level documentation	1) Low level documentation; detailed documentation
2) Consist of single line	2) Consist of Multiple components, each component have various steps
(3) A person who have knowledge of product, hard to execute	3) No knowledge of a product, have lot a data to
(4) what to test	(4) How to test
(5) Req → Test Scenarios (SRS)	(5) Test scenarios → Test cases
eg → validation of login	① connected to gmail ② Net ③ Display ④ Enter ⑤ Sign in

6

(3) Benefits of Automated Testing:

↳ Technique for automating the manual testing process.

→ Manual testing is replaced by collection of automated testing tools.

Beain X

1) Difference B/W QA & QC

QA	QC
↳ Part of Quality management plan.	↳ Inspection phase of QA.
↳ Ensures product is ready to go through manufacturing Process.	↳ series of test
↳ verify that quality req. will be fulfilled as the products are made.	
↳ (1) Pro- Active → Prevent defects.	(2) Re- Active → Identify defects
(2) Process oriented → Prevent quality issues	(2) Product Oriented. → identify quality issues in a manufactured Product.
(3) System	(3) Part → such as Raw material
(4) Roadmap for creating high Quality product.	(4) Verification of product.
(5) Entire team	(5) Dedicated Personnel

Advantages of Automated Testing:-

1) Improves coverage of Testing;
   Automated Test cases execute faster than manual execution.

2) Reduce dependency of testing on availability of test Engineers

3) Takes less resources.

4) Produce repository of different-tests → Increase knowledge.

5) Provides round the clock coverage; 24*7

6) less chances of error; reliable

7) Act as test data generator; produce max. test data to cover large input.

Disadvantages:

1) Expensive

2) Becomes inconvenient & burdensome as to decide who would automate & who would train.

3) Limitization; many organizations not prefer test automation.

4) Req. Additionally trained & skilled people.

5) Only removes mechanical execution of testing process, but creation of test cases still req. testing professionals.

X - ✓

DataBase DataMapping

i-> CRUD Testing: (BlackBox Testing)
Create Update Delete Read any Resource.

→ Testers prepare Crud Matrix

Req. Objects	C	R	U	D
Customer	-			
Order				
Payment				
Vendor				
check				
Register				
Stock				
Back Order				
Index				

(8) Major Attributes while writing Test Cases?

Steps to achieve Test scenarios

Serial No	Test Scenario	Precon	Steps to Reproduce
1- Login	validation of login functionality with correct credentials	connected to wifi	1) Launch browser 2) Enter valid Gmail 3) Enter U-name & click on next button 4) Enter valid Pwd 5) Click on submit button
2- Compose Mail	Validation of composing Email		
3- Delete Bin	Validation of deleting mail & checking if it's present in Bin		

Expected Behaviour: HomePage should be displayed post login

Actual Behaviour: upon entering correct Username & Pwd HomePage displayed.

Result: Pass

Automated: Yes

Bug-id

→ Short span of time

flexible

Agile Model:

→ Req. keep on changing.

Iterative + Incremental Approach

↓ process keeps on repeating

↓ → features Modules keep on Adding

↳ Company follow Agile Model to give fast delivery.

↓

customer satisfaction

Adv. / Principles

↳ Req. change Allowed at any time

↳ Releases very fast

2 → customer satisfaction

2 → good com b/w Customer

↳ Easy

Dis

↳ less focus on design & documentation

2 → diff. to handle long term project

2 → Does by Senior Testers

↳ Scrum

PM

① Product backlog : what are the things required.

② Sprint : project for 1 year

③ Sprint Planning → conducted by Scrum Master or Product Owner

hold meeting b/w Developers & Testers & check from product backlog which things req. in next sprint

(4) Scrum Master : Responsible for delivering product to customer within planned period.

Tracking All

→ Sr. Tester, Sr. Deve, PM

(5) Scrum Meeting: Daily Standup Meeting (10-15 m)

→ what did y, Tom, Any obstacle -

Principle of ST?

1) Start Testing at early stage: Req Gather
2) find defect
3) Highly impossible to give error free S/w
4) shouldn't do exhaustive testing

com. [.]

↓

Accept decimal also

same kind type of input 100.62✓

multiple times.

(5) Test-based (what testing)

eg. web based ← load/performance testing

(6) Pesticides-Paradox

contains more defect Testing same Testcase

(7) Defect-clustering

(10) Top Qualities of Software Testers

(11) Regression

(12) Severity & priority ?

↙ impact of bug on business workflow

↘ importance given to fix the bug

(13) Big Bang Method

↑ Non-Incremental integration Testing

↘ don't know ~~about~~ who is Parent & who is children module

↓ How will they do Testing ?

mix all modules & check data flow.

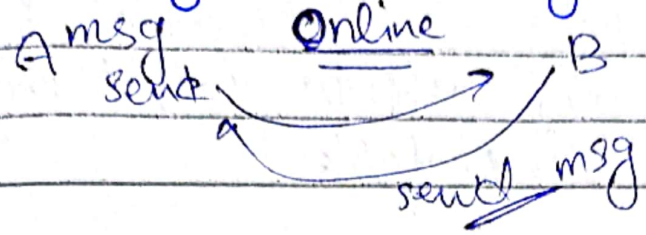
(14) Principles of Software Testing

1. find defect
2. Not possible to launch defect free s/w
3. Should not do exhaustive Testing
4. Pesticides Paradox

In App → 4 modules

5. Defect Clustering → After every release test same test cases

(5) Integration Testing scenario?



(6) Formal Testing ~~Type~~ Type of Smoke Testing

80 cases

15 pass

5 fail

Document Test cases.

Informal → No documentation
No Test Scenario
case
Traceability Matrix
Report

(7) Useability Testing:

↳ do Not check application,

↳ we check user-friendly of Application

(8) Agile Model

→ Scrum Master

→ Product Owner

(9) Release Note

Unexpected

(1) Purpose of Testing : defect free customer req.

→ 100% error free S/W can't be delivered.

(2) Got minor defects, will you release S/W?

No → define Severity.

Team Lead

(3) When developer will say defect is not reproducable

→ Tester is able to see the defect but developer not;

→ improper build

→ platform mismatching

→ improper Test Report by tester

→ Technology not support that features.

(4) 2 positive & 2 Neg. Test Scenario of whatsapp

→ 8 people can do call

→ Black Box Design Technique | Boundary Value | 0-10 | Error Analysis

(14) Test case Design Technique

(15) No Testing → product going to production?

(16) When do you feel to stop the testing?
optimal testing → Not receiving any bug.

(17) Test case Review Process
↓
Peer, Team lead, PM

1) Agile model? → Sprint → is a cycle

2) Performance Testing? what are we exactly Testing in Performance Testing

↳ Check the stability & Response time of an application by applying load.

3) Functional vs Non Functional Testing
Component Level Testing
Neg. Testing
Performance

4) Smoke Testing Vs Sanity Testing
when a new build is launched, when Stable
changes can be still made,
Perform smoke Testing.

5) Principles of S/W Testing (7)
1) 100% testing can't be done
2) Pesticides Paradox

6) Severity : impact of bug on Customer business workflow.
eg. Login pressed → Show blank page.

(13) Retesting : APP fail first, & now again testing

(7) Priority: problem need to be fixed otherwise customer can't login.

(8) Major Attributes while writing Test cases?
Test case Id, Test Scenario.

(9) System Test case? End-End testcase?
↓
Any thing customer do on applica.
login → search item → Add to card → payment → Order placed.
Main flow of APP

(10) Ad-hoc Testing

(11) Dis-advantages of not writing Test cases?
- in-consistency, loose quality

Adv → creative ideas
↳ Analytical thinking
Next time → don't have to think so much scenarios

(12) White Box Black box
Re Modify ← Regression Testing
delete
add
bug-fixing

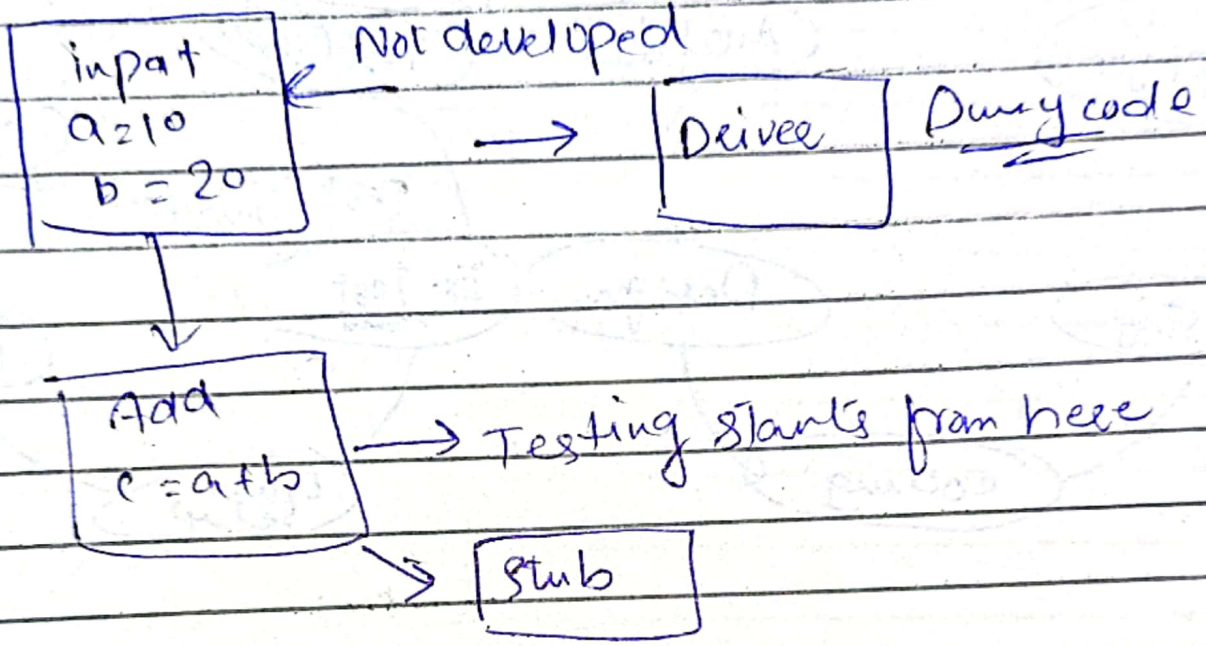
Shallow wide Testing → Smoke

Narrow & deep → Regress Testing → Sanity

eg. car

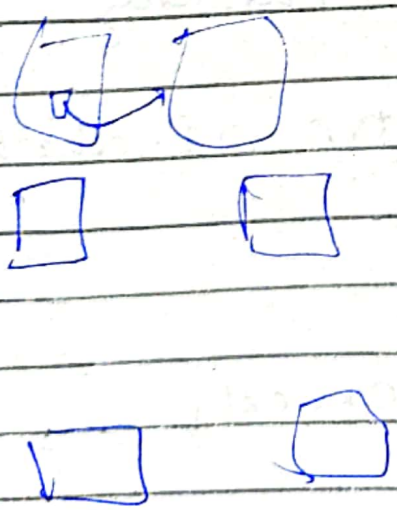
Smoke	Sanity
↳ Testing at high level, no depth	↳ Focuses on 1 or few areas of functionality
↳ Basic critical func. working properly?	↳ New functionality / bug fixes working perfectly?
↳ Usually Automated	Regression (Sanity)
↳ Positive testing	Stlc
→ Develop & Tester	↳ Usualy not automated
1st Smoke Testing	↳ +ve +-ve
until s/w → Stable ↳ general health check	→ By Testers ↳ when stable ↳ Specialized checkup
↳ Daumented	↳ Not documented
eg. Gmail Inbox ↓ mail	mail can be fwd or not? can be replied or not

Stub & Driver



Stub ← Top-down

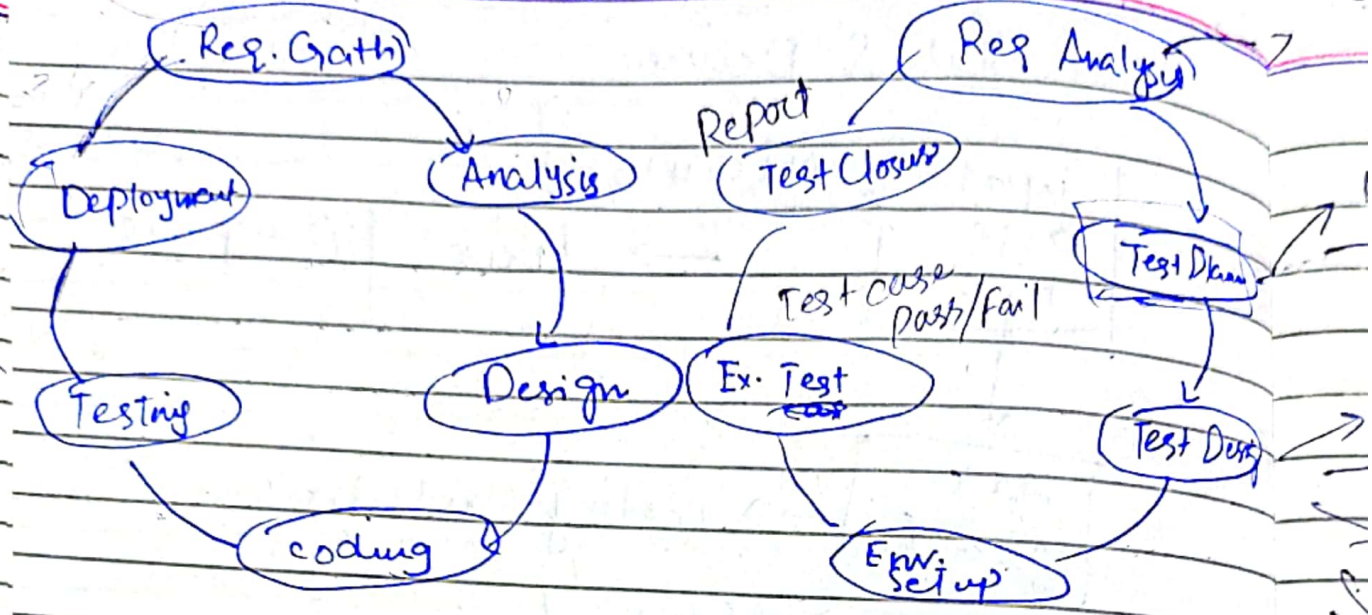
Bottom up → Driver



SDLC ← STLC ← BLC/DT

Bug life cycle / Defect Tracking

SDLC vs STLC



Bug Life cycle / Defect Tracking

1) Wrong Implementation

Home → About us → Feedback

2) Missing info

UN

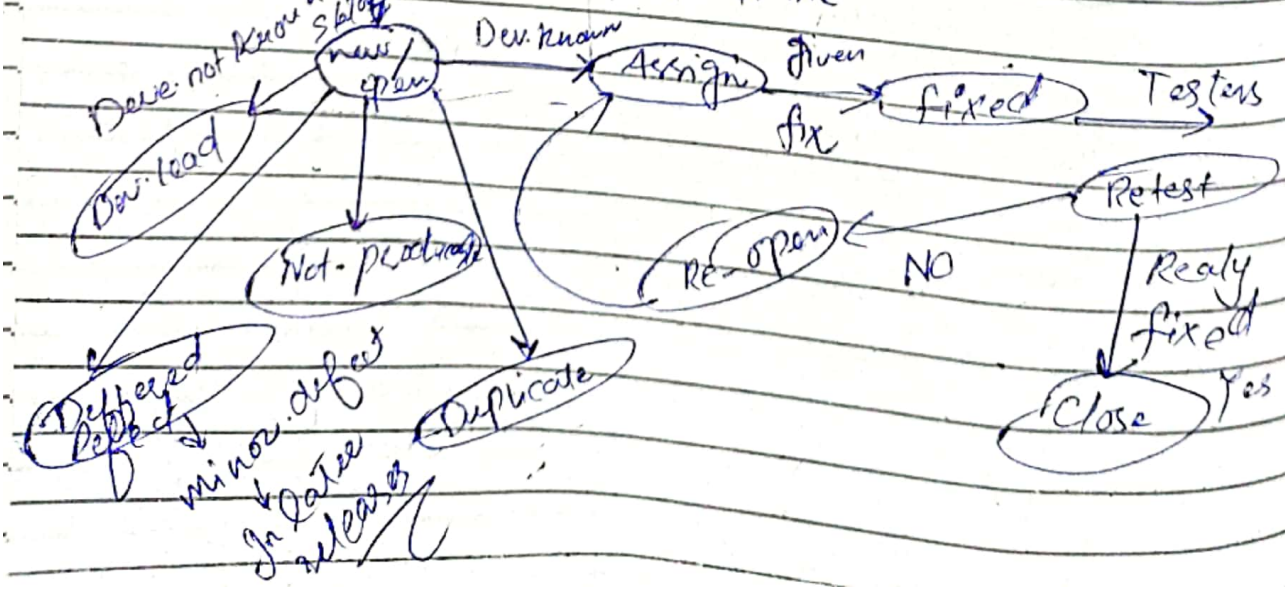
Pwd

Login

cancel

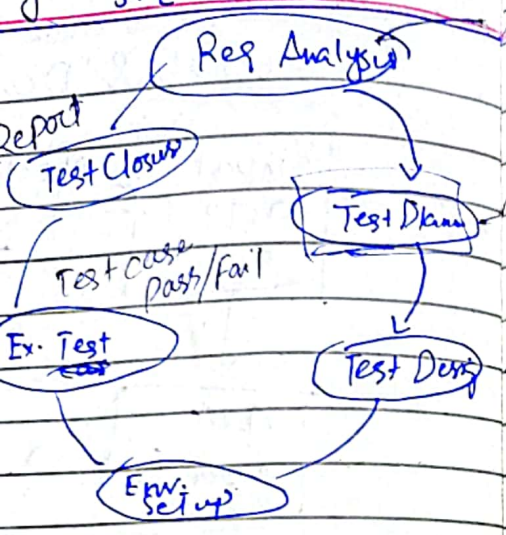
3) Extra Info

Raise defect for 1st time



BLG/DT
↙ ↘
life cycle STLC Defect Tracking

complete knowledge by PM to Testers & Developers



Which engineer
How many engineers
→ Tool

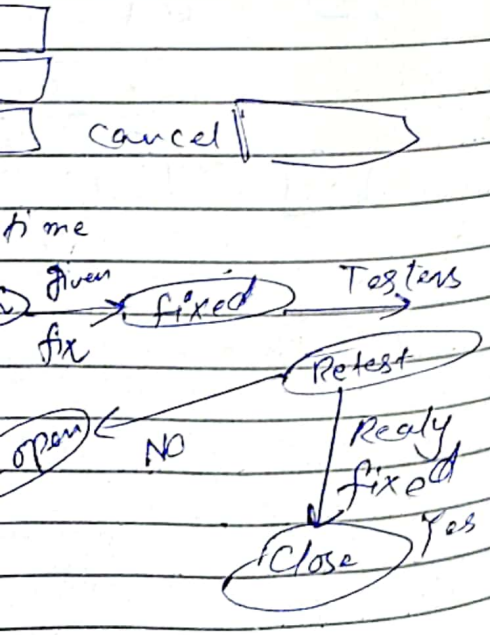
→ Modules given to Testers
→ Test Scenarios
Test cases
Review
fix review
Approval
Neg./positive

Defect Tracking

Traceability Matrix → Test Req. got Atleast one test case

1) Fwd / Horizontal → Before exe. 1 test case → R
2) Bwd / Vertical → After ex. According to
3) Bi-directional

About us → Feedback



Over Bug life cycle ?
Test Cases Vs Test Scenarios

i2c

1

1. Functional vs Non functional Req.
2. Test Cases of Mobile
3. Stub → in code
4. Smoke *
5. Sanity *
6. White box, black box, Grey box
7. Normalization
8. Stages of Normalization
9. Queries:
   1) Distinct projects
   2) Count
   3) Max Salary
   4) left join
   5) Self join
10. Trigger function
11. SDLC uses
12. STLC uses
13. Bugs
14. Errors
15. 3rd highest Salary
16. QA & QC & their methods.
17. Unit Testing, Alpha, Beta
18. Verification Validation
19. Functional Non func. requirements of bug.
20. Exploratory Testing ? If go to company documents missing. then what are demerits ? Data inconsistency & Bug can be considered low quality as Feature & feature ~~as a bug~~

Engineering → Principles, rules to create sth new.

SPM → Estimation of cost, Risk, time
Software Design → Coupling, Cohesion, UML, DFD, Class Diagram

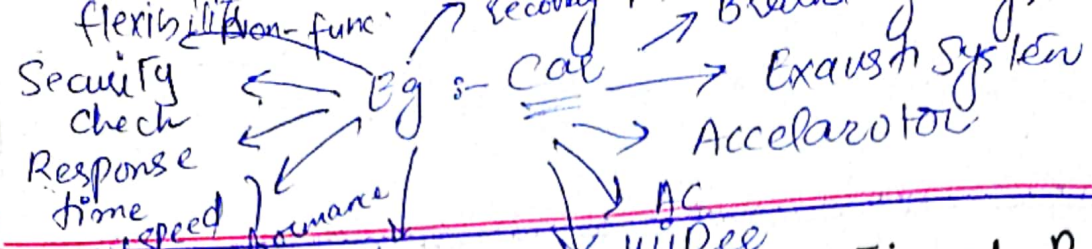
SDLC:
→ customer invests money
→ company wants to make name & become known
→ Satisfy customer, time framework
eg. Youtube Channel Tutors
Customer — Service Provider
→ cost, Platform, Team Available
• feasibility study → in Defining / Analysis →
Senior SE → collect basic info from customer
technical, Economical, Operational feasibility
Planning: cost, time

ER

1) Planning: basic, market, provide info
In mind

bag.
my
re.

SRS :- Functional & non-functional Req.
↓ input for designing phase



1) Functional & Non functional Req.

SRS

working/operation

No. of resources → choice of resources other than working.

Accessibility

expected req.

flexi

Recovery

cost

security

performance Req

storage → response time

Interoperability Eg. Google server

eg. Elavator → F take up or certain floor

→ N.F speed

System → P → Calculations
→ Use-cases
→ Tech.
→ Data Processing

Hospital → F → Equipments
→ Entries / Exits
→ Doorways / Stairs

Hotel → Room, Swimming pool, Dining Area
Banquet Hall.

→ undecidable

↳ Unit Testing : 1st place

↳ → Developer.

Individual part

eg: Car ; locking system, fuel system, braking system

↳ Integration Testing : Tester

↳ SRS document → evaluation

Types:-

1) Bigbang: join 2,3 modules & then join

2) Top-down: Top priority tested first eg. Fee in Student

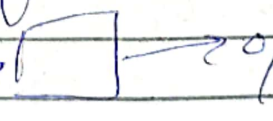
3) Bottom-up

4) Mixed (Sandwich)

↳ System Testing: → Testers

↳ End-to-End System Specification

* Black Box Testing



2 Types:

eg Movie ① α, β, Acceptance

② Func / Non func.

Data Flow Testing:

1) Statements in which variables are defined.
2) Statements in which variables are used.

eg: 1) read a, b, c
2) if (a > b)
3) x = a + 1
4) print x;
else
5) x = b - 1
6) print z;

	Define	Use
a	1	2, 3
b	1	2, 5
c	1	NA
x	3, 5	4
z	NA	6

1) Variables declared not used.
2) Variables used not declared.
3) Variable defined multiple times
4) Deallocating variable before used.

Boundary Value Analysis: (BBT)

Invalid Case | Valid | Invalid Case

Best use of

eg: Fuel of car at speed 40 - 80

39, 40, 80, 81

No code access required.

Large suited programs

difficulty in making test cases

Tester know limited info about app.

Black box Testing:
Closed-box Testing, Data Driven Testing

1) Defect : variation from Actual/Expected Result → issue in development phase & corrected

2) Bug) Testers find mismatch in Testing phase.
↳ In formal name for defect.

3) Failure : After deployment, customer finds issue.

4) Error : coding mistake → unable to run / compile code
↳

Acceptance Testing:- Formal Testing based on user req. & functional processing.
↳ kind of black box Testing.
→ last level of software testing.
✓ User AT

Alpha Testing: User Acceptance Testing
→ Before release to real world.

objectives :-
fix bugs that weren't found before

SRS → Test Cases & Test Plans →
Execute Test plans → Log Defects →
Retest once issues fixed

2 phases:
1) In-house developers
2) Testers

↳ require QAS/testing environment
→ Evaluating Test Cases.

Beta Testing :-
↳ doesn't require QAS
↳ black box Testing

Tool : zephyr, User Testing

↳ duplication of errors
↳ Time consuming.

Test from user point of view rather designer point of view

→ semi-transparent → tester can partially see

→ suited for ~~web App~~

Gray Box Testing:- Not suitable for Algorithmic Testing.

Test case efficient design

black box + white box

unknown of internal structure

know internal structure

Graybox

↓

partially know Internal Structure

↙

Data structures & Algo

↓

make test cases

risks identify by developer

causes of failure

1) Regression Testing
2) Matrix Testing
3) Pattern Testing
4) Orthogonal Testing

↓ BBT → Few test cases & large test data

Regression Testing:

→ updation

→ fixing 1 defect cause no issue to other functionality

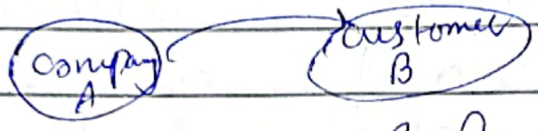
⇒ Translucent box Testing.

QA	QC
→ Process	→ Product
→ Preventive Approach ↳ proactive Approach eg. life jacket before boating	→ Reative Approach ↳ find defects & then solve → Testing product meeting quality
↳ Audit: process quality meeting standards or not	
→ Managerial Tool	→ corrective Tool
→ Development phase	→ testing.
→ All team members involved	→ Testing team involved

Test Sys R
function rel

confidence Testing / Build verification Testing

Smoke Testing
↳ Testing Basic & critical ~~ppt~~ part of app before Regression Testing



30 Days

White box Testing → done 20 Days
10 days left.

WBT → Test Engineers
↙ ↓ ↘ System
Unit/component Integr

Sanity Testing: → func.

Join

Trigger
highest point
pop-up

Yes

80-90

3.08

Query, mysql → DB

DB CN OS

7/10 8/10 5/14

↳ Stub Testing:- Dummy Modules / Actual

Monday — 2-5 pm

2-5 pm

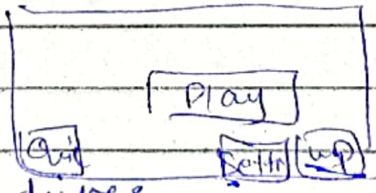
Wednesday 2-5 pm

Tuesday

Cour

Stub Testing:- Dummy Modules for Testing an app when actual modules not available for testing

Example:



Dummy Procedures

```
Sub PlayButtonPressed()
    Console.WriteLine("Play Button Press")
End Sub

Sub Su
    Stub Testing Code
If ButtonId = "Play" then
    Call PlayButtonPressed();
```

Adhoc, Gorilla, monkey

Positive

↳ performed for expected result/conditions, as per req.

Not 100% Test coverage → 100% Test case coverage

↳ doesn't ensure defect free s/w

↳ Done before Neg. Testing.

↳ Anyone can do

↳ check valid set of values

→ values 0–10
0, 1, ... 9

Neg. Testing

↳ performed for unexpected conditions, not performed as per req.

↳ ensure defect free s/w

↳ ensure good quality product

↳ only professional

check invalid set of values.

→ a, b, c, space, blank, special