Software Construction & Development

Software engineering / Logic building → Implementation

Books:

Code complete: A practical handbook of Software construction by Steve McConnel Microsoft.

Reference Books:

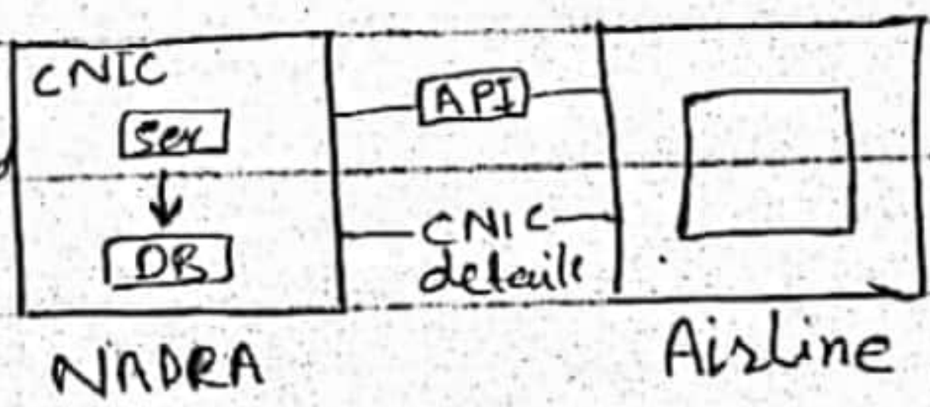
1. Working Effectively with legacy code,
2. Refactoring

Specification: How to achieve a requirment

user Requirment: What user needs.

user: Book ticket

user check: Not blacklisted



System Requirement:

Specification

user logic → NADRA System Req = Specification

Specification ⇒ How to implement a uses requirement or a process to implement a uses requirement.

User security check & Login ke Zerye = Specification.

SDLC

1. Domain of problem / understand problem (down)
2. Requirement → InConsistant words, contradict..
3. Design Analysis
3. Implementation
4. Testing
5. Deployment
6. Maintainance

Waterfall Model: First complete one (Phase Cycle) then goes on next.

→ Used for stable software / system

→ Not best for that application, whose requirments are changing after short period

→ Best for Large system / Software

Requirment Gathering

User
↓
What is requirment
. what user wants?

System
↓
How they are imple-mented.

Non functional Requirments:

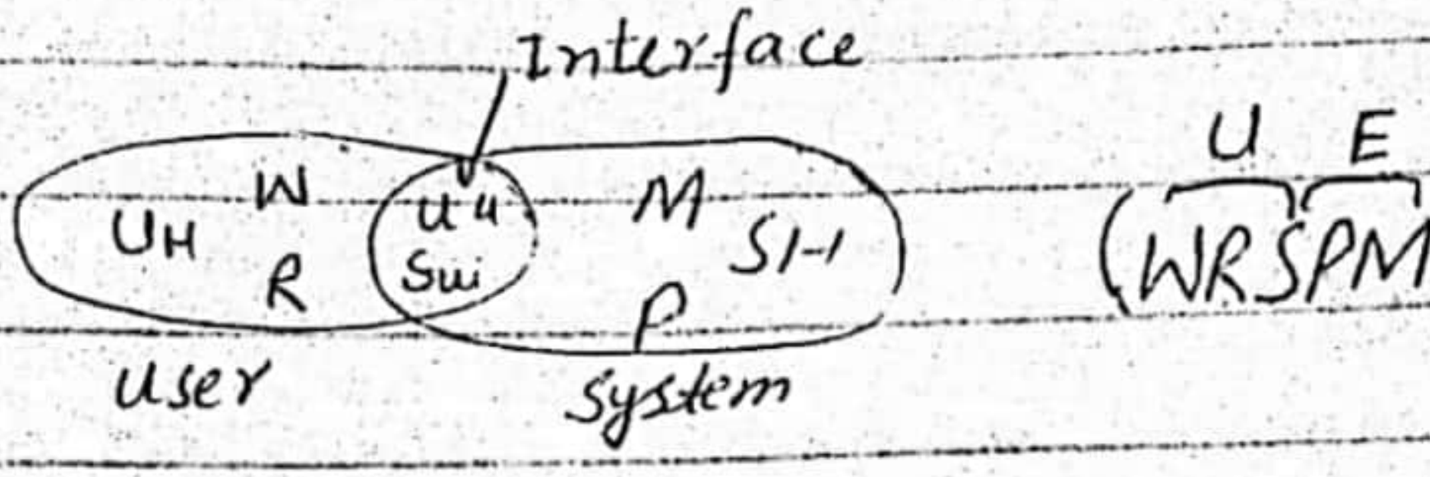
→ product Req. (Web based, app ba.. OS version)

→ Oaganizational (blacklisted online access

→ External Req. (

Why clients need that softwase?

How Project will behave



Ur => User info (Card, pin in case of banking)

Sui => System info (check, transfer, withdraw)

W => Problem domain / world

R => User Requirement

S => System Specification

P => Programming

M => Machine

S → R => SS satisfies user sequirment

P&M → S => P&M Satisfy system sequirment

Architecture:

Decompose an enterprice System into independent sub systems that have value in the System

→ How these sub-system interact.

→ principles & guidlines for the design evolution over time.

→ Technical interface b/w user & design

→ All components that are independent

Components module sy mil ks bnty h

They are independent

Design:

Aik subsystem k design ko software design ya system design kehty hain

Example:

persons or companies may own cars. Car owner ID is the ID either the person or company that owns the car. A car may have only loan one owner (person company). A car may have a loan multiple loan. A bank provides a loan to person or a company for the purchased of a car. Only the loan obtain a loan on the car. The car owned type and the loan customer type indicate whether the car owner holder is person or company.

Nouns	verbs
Person / Company	Owns
Car	provide
Owner ID	Obtain
Car owner	purchase
Loan, loan Customer type	indicate
Bank, car owner type	

Classes:

Person (Name, ID, Address

Company (Name, ID Address

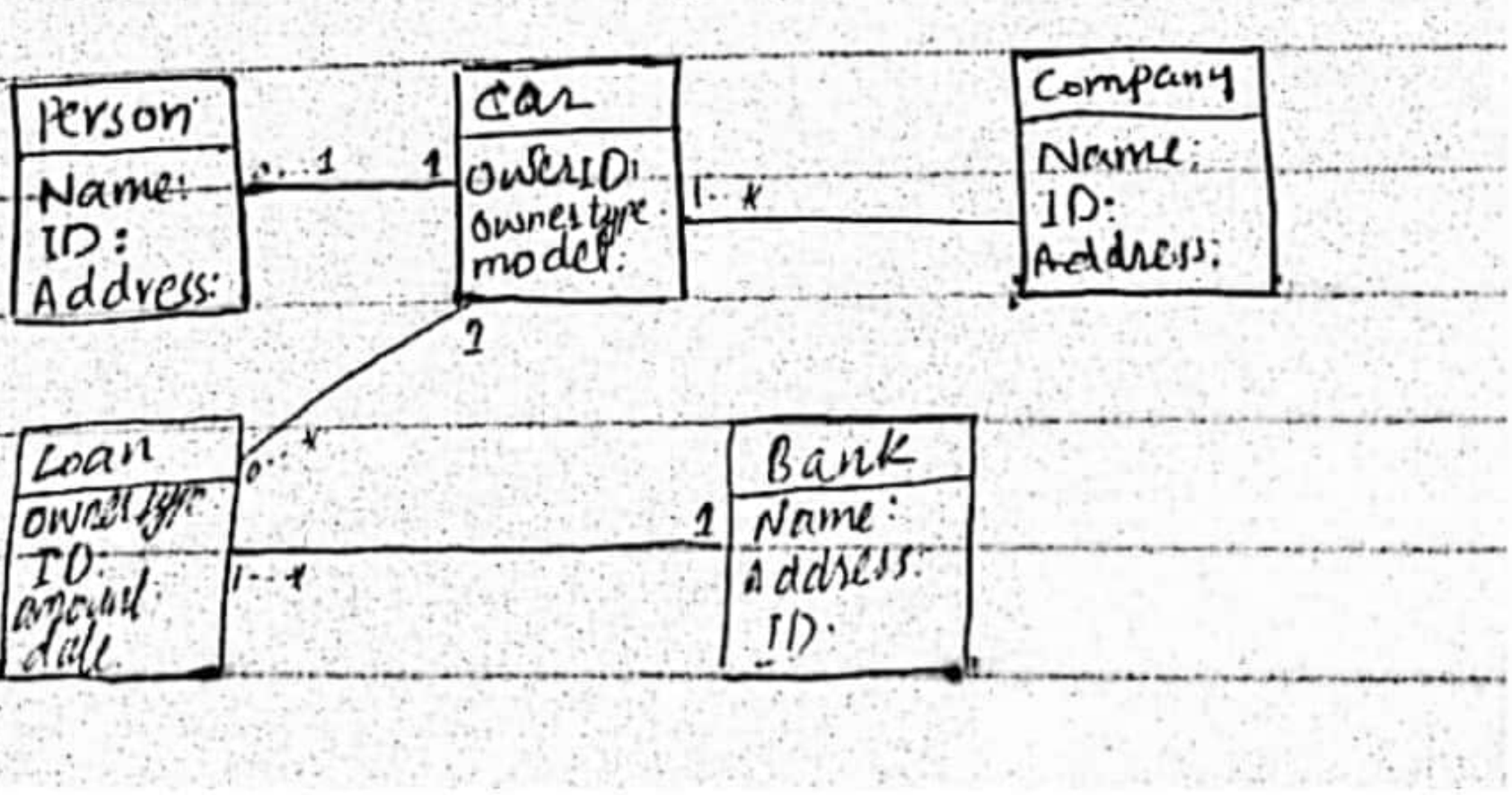
Car: (owner type). (OwnerID), model

Loan (customer-type, ID, amount, date, account

Bank (Name, Address)

payment Method (API)

Order Processing



A car can be owned by only one person

A person may own one, zero or multiple cars (0 ... *)

A car can have multiple loans (0...

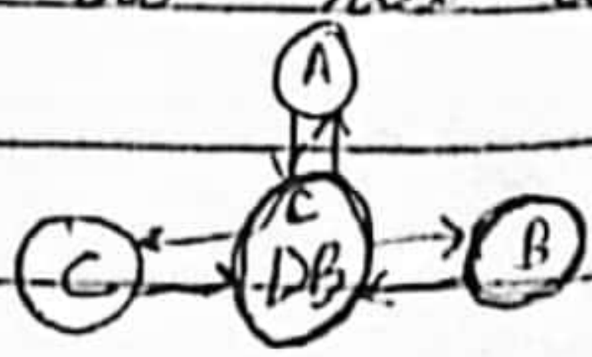
A loan can only be consumed by a single.

- Domain Modeling
- Requirement Elicitation
- Software Architecture

Software Architecture Model:

1- Pipe and Filter (same input & output : e.g comp
2- Blackboard (central database, different comp
3- Layered
4- Client Server (Request & Response based)
5- Event based (similar to blackboard

Blackboard: It will communicate with centralized DB and do not comm with each other



Concerns:

i. System Structure (decompose into components)

ii. Interaction of components

iii. Modulas decomposition

Modularity:

Breaking down of a component and reassembling / intesaction

Modules: It should have following properties

- Coupling (Should be low coupled)
- Cohesion (Should be highly cohesion)
- Information Hiding
- Data Encapsulation

Coupling ⇒ The interaction b/w two classes

i. changing in one class effects the other class.

ii. classes should loosely bound (Change should not effect)

Cohesion: Aik class aik kaam krey.

Single goal achieve krana he.

Tight Coupling: Module A can directly access Module B data members (Content coupling)

(Common coupling):

Module A and B are relied.
Same global data.
External coupling:
Module reling on externally imposed
format (protocol / interface)

Loose Coupling:

Data Coupling: Module A only pass p
meters for sequesting functionalty
module B

Message coupling: Module A sends mess
to module B

No Coupling

Medium Coupling:

Control Coupling: Module A controls th
logical flow of module B by passing in
formation or by using flags

Data Structure Coupling: Module A &
rely on same composite data structure
changing data structure directly affect
the other module.

Weak Cohesion

Temporal: Coincidential

Different parts of module are lo

just because they are in single

Temporal:

Different parts / code / functions are actuate at the same time.

procedural: One part follows the other in time.

Logical association: similar parts / functions are grouped. They are similar but perform different things.

Medium Cohesion:

: All elements operate on some inputs and produces some output.

Sequencial : One part output serves as input to other part.

Strong Cohesion:

Object: Each operation in module can manipulate object attributes.

functional: Each part of the function is necessary for execution of single well defined behavior.

Complele SDLC

1- Domain

2. Requirment Elicitation

3- Softwasc Architecture

4- Softwase Design

5.. Coding & Debugging

6- Unit Testing

7- Integration Testing

8- System Testing

10- Corrective Maintainance

Types of testing:

Two types of testing

i- White box

ii. Black box

Black box: Implementation details nai

Input de kr output check krty

White box: function / class level pr

krty hain k ye theck kaam kr

→ User

→ developer

→ Modular level pr → code check ho

V and V process

Validation:

Deployment

Physical Enviroment

Hardware

Documentation

Training

DB related Concerns

Third party Software

Software

Failure / Maintaince

Cold back up: sirf machine ps hota he,

24 hours mai train hota.

Worm Stand by: Software install hota ha ps db

handle nae hoti

Hot fail over: phly kaam nai krti jab failure

aati ha tb ye ready hoti he

Key Construction Decisions

Language Identification: ye language use krne

sy humein kia faida ho ga

What is purpose to use it ?

Developer ko bi pta hota us language ka.

Programming Conventions

(Variable / Classes / declaration)

→ Global conventions (Jo sab use krty.

→ Project / company based (dor convention

must follow them)

Current technology ware: platform indepen

Coding

Selection of major construction practi

Coding

Team Work

QA

Tools

Fundamental of SC

→ Reuseable Software

→ Construction for Verification

→ You have to reduce complexity

→ You have to simplify the problem

→ You have to provide flexibility like

one function has to update ther

other functions should not be affe

→ Business constraints and implementation

Whole Past → diamond

i Predictive Vs Adaptive

ii Iterative Vs Incremental

Predictive ⇒ requirements are clear and they are not going to change. Clear what he wants

Adaptive ⇒ Idea clear ha pr requirment clear nai sab kch chahiye hota pr ye nai pata ke chahiye kya

He wants everything but not sure what actually he wants.

Incremental Process:

→ Both predictive and adaptive model

→ Small parts are combine to form a Large one

→ Work independently.

→ parts involved

Adaptive ⇒ feedback le kr improve krna.

predictive ⇒ Feedback nae hota.

e.g: Bike

Iterative Model:

With the passage of time, ...
koi sahein.

→ Adaptive

Waterfall Model:

predictive Mod

Domain Modeling

Requirement Analysis

team be ea

Architecture

or client ko h

Design

ho k wo kia

Implementation

chalta ha.

Testing

Integration

1 Deployment

2. Maintainece

3.

4 V-Model:

5 . predictive

SASHMI Model:

6 → It a time multiple phase

active

→ team lead krna hota h

→ It is also predictive

Iterative Model:

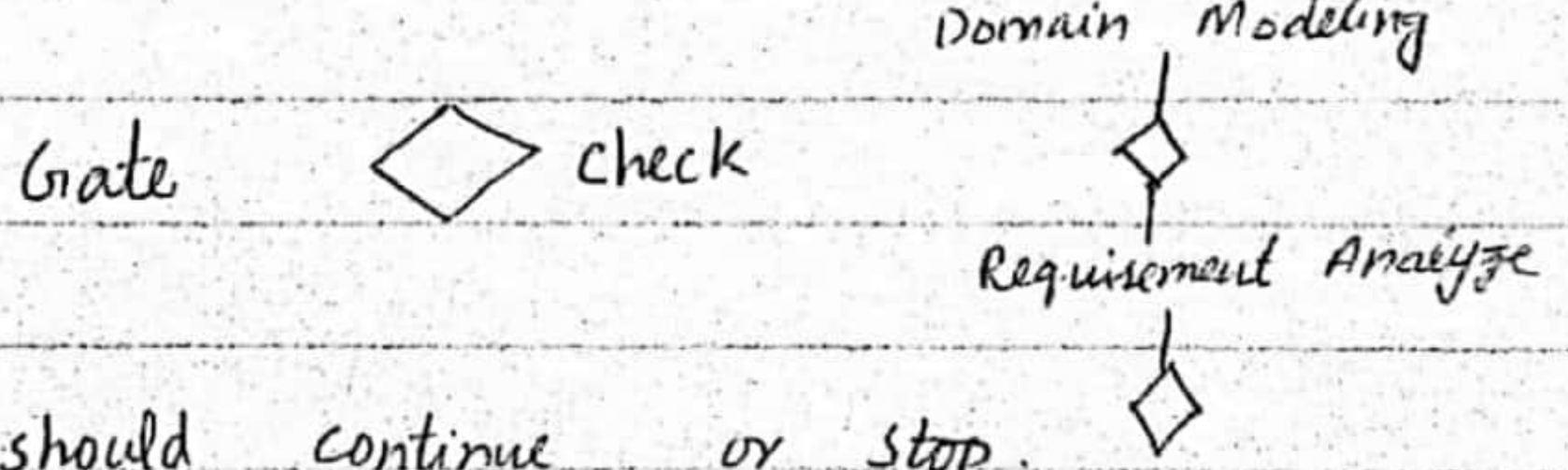
__ Not clear about requirement (Adaptive Unified Process

- Rational
- Enterprise
- spiral Model

GATE CHECK

AGILE

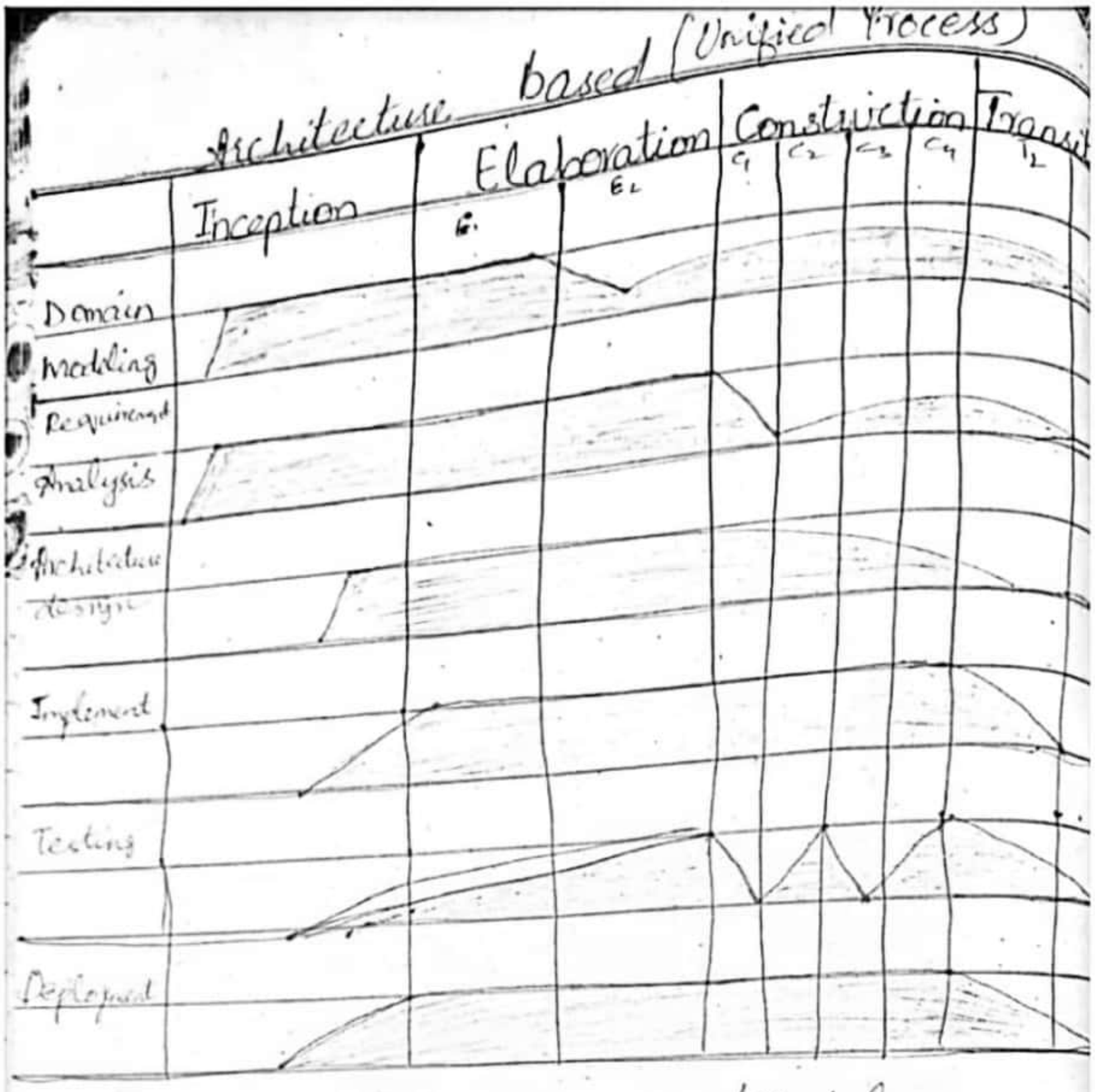
GATE CHECK



→ we should continue or stop.

→ Har step k baad aik gate hota ha or hum analgze krty hain k humein agla step krna chahiye k nae

→ Jab human pata na lw ke ye cheez humein faida dy gi ya nai



→ Frame Work, not a process/Model

→ Not Completely adaptive and predictive

→ It specifies and elaborate Software architecture. → use for architecture design.

Iterative design

Spiral Model

Define / determine / objective

Identify

Plan

Implement / Test

→ Fo risk

Agile Model 10-10-2023

→ Mind Set → 70% of Agile Methodology

→ (DSDM, FDD, Scrum, Crystal, XP, Custom) → Agile development Model

4 values [Principles by 17 Members

i- Individuals and Interactions over process & tools

ii - Working Software over comprehensive documentation.

iii - Customer Collaboration over contract Negotiation

iv - Responding to change.

Agile Principles

1- Highest priority is to satisfy customer through early and continuous delivery of valuable software

2 Welcome changing requirements over late in development. Agile processes harness change for the customer's competitive advantage.

3- Deliver working software from a couple of weeks to the couple of months with the preference to the shorter timescale.

4- Business people and developers must work together daily throughout the project.

5- Build the projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6- The most efficient and effective method of conveying information to and within

a development team is face to face
conversation.
7- Working Software is the primary measure
of progress.
8- Agile process promote sustainable developme
the sponsers, developers, and users should
be able to maintain a constant pace
indefinitely.
9- Continuous attention to technical excellen
and good design enhances agility - cost
exploration.
10- Similarity - the art of maximizing the
amount of work not done - is essential
11- The best architectures, requirements
and design emerge from self-organizi
teams (PSP → Personal Software Process).
12- At regular intervals, the team reflect
on how to become more effective
then tunes and adjusts its behavio
accordingly.

PSP:

A self improvement process for softwa
engineers.

1- Bottom up approach to

improve engineering practice. (personal level) -
2- Starting point is by training individuals skill and tools for work.
3- The improvement principles are not just only for software industry but neutral to all industries.

IDEA

1- Individuals should be able to plan, deliver, monitor and improve the quality and timeliness of their own work.
2- Use data to justify refute unreasonable demands.
. Say 'Yes' with confidence
. Say 'No' with data & options.
3- Learn from experience, use data from one piece of work to improve the next.

Principles

. Measure Stuff
. Measure Size
. Measure Time
. Measure Efforts
. Measure Defects (time included in ~~(functio)~~ finding solutions and point of introduction).

Humphreys

- Measure consistently
- Use correlation to judge usefulness of time to predict future performance.
(Use coculation)

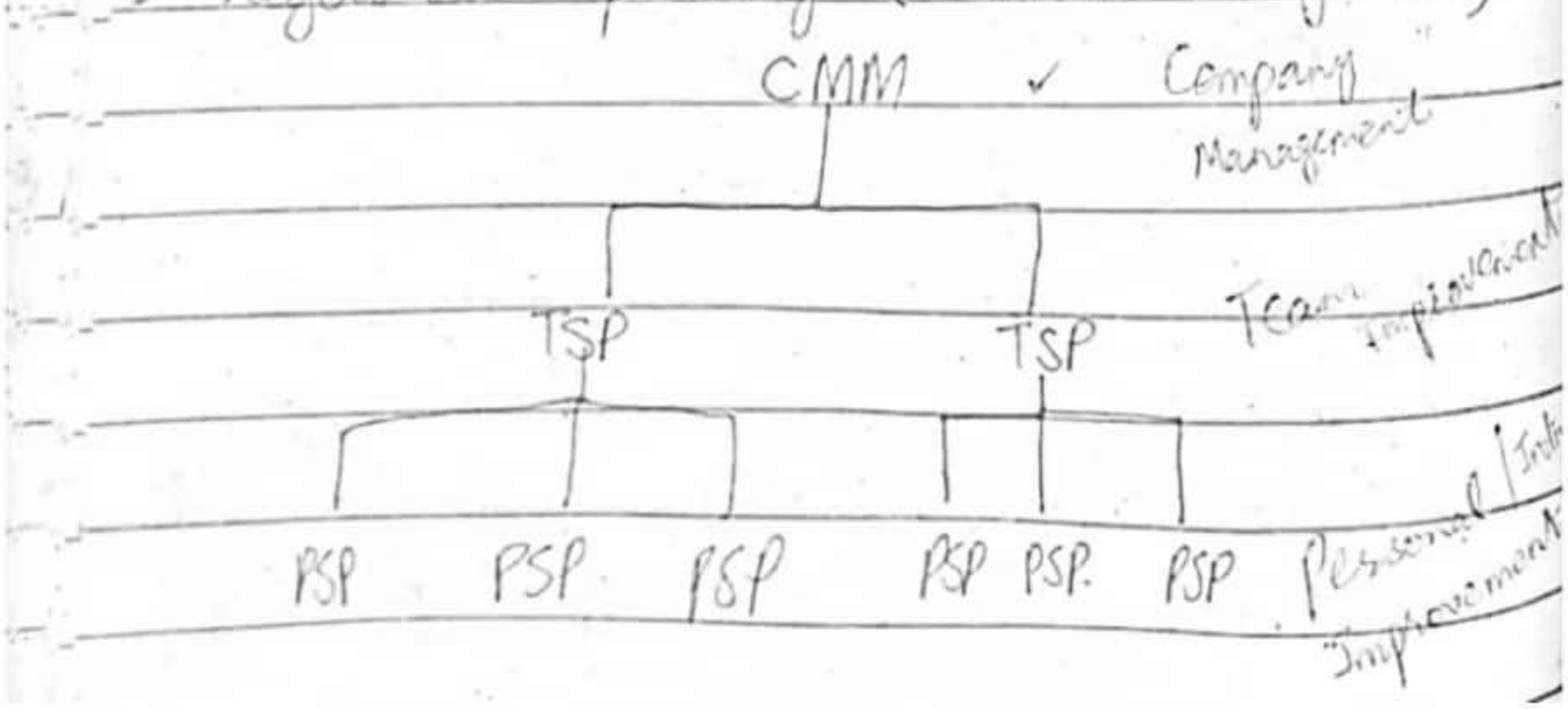
TSP: (Team Software Process)

(Idea)

- The team should be self-directed.
- Definite tasks assign to team individ that plays a role in achieving a goal.
- Communication
- Team members are dependent to achi a common goal.

Principles

- Measure the task for each individual / Analyze each member performance.
- Regular meetings. You have Scripts.
- Rigourous planning (Achievable goals).

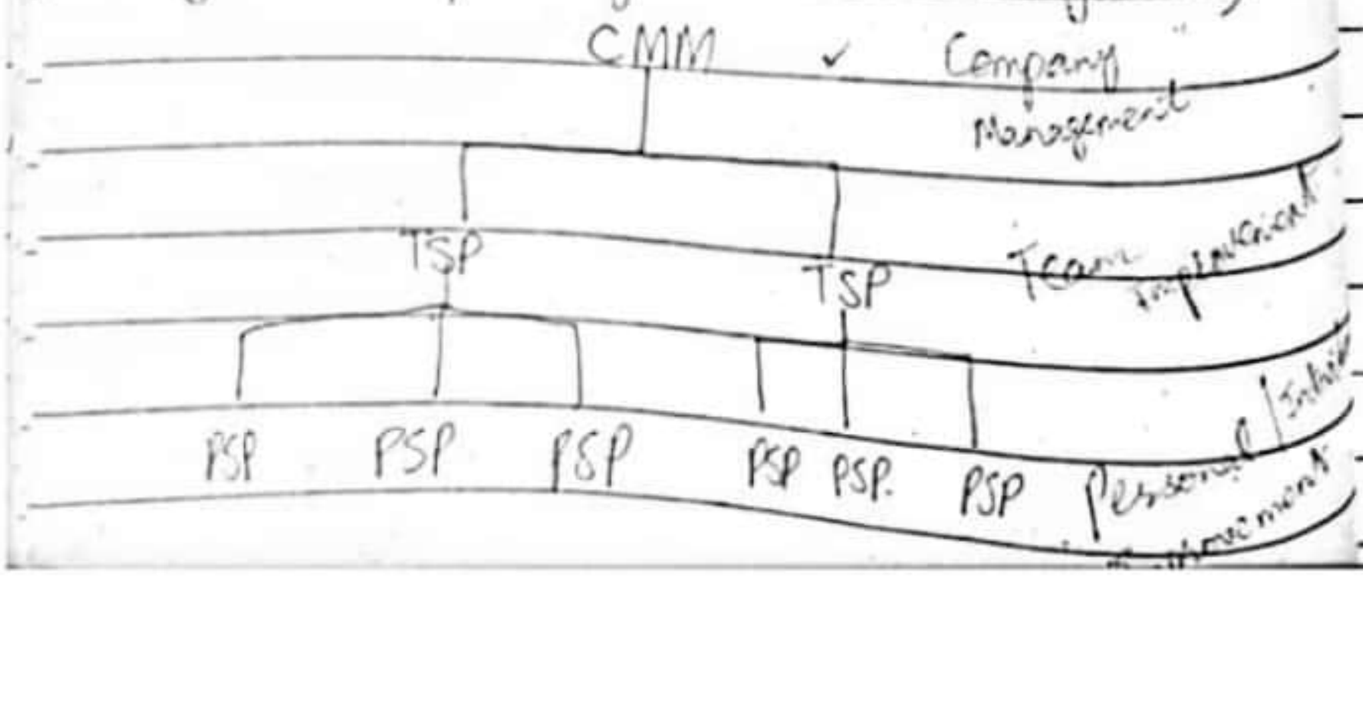


• Measure consistently
• Use correlation to judge usefulness of
• time to predict future performance.
(Use correlation)

TSP: (Team Software Process)
(Idea)

• The team should be self-directed
• Definite tasks assign to team indivi
that plays a role in achieving a sing
goal.
• Communication
• Team members are dependent to achie
a common goal.

Principles
- Measure the task for each individual
/ Analyze each member performance.
• Regular meeting • You have Scripts.
• Rigorous planning (Achievable goals).



SCRU

• Backlog of
• 1 to 4 wee
deploy ho jaa
Define → De

Sprint

1 • Product Backl
2 • Sprint Pla
3 • Sprint Back
Daily Meeti
→ What pr
→ Problems
→ Assign w
→ Time
→ Team
4 • Finish Produ
5 • Sprint Revi
6 • Burn up/d

ERROR ⇒ Ille
in abnormal
→ Syntax
→ Logical
→ Runtim

SCRUM MODEL

- Backlog of product
- 1 to 4 week tak aik requirement deploy ho jani chahiye.

Define → Develop → Test → Deploy

Sprint

1. Product Backlog
2. Sprint Planning Meeting
3. Sprint Backlog

Daily Meeting

→ What previous tasks

→ Problems

→ Assignment

→ Time

→ Team Members

4. Finish Product (first sprint ka finish).
5. Sprint Review
6. Burn up/down chart.

17-10-2023

ERROR ⇒ Illegal actions/operations that results in abnormal/malfunctioning of a program.

→ Syntax ERROR

→ Logical ERROR (Programmer side)

→ Runtime/unchecked ERROR

↳ Also known as Unchecked exception

ERROR Handling Technique

- Non-functional attributes
  - → Correctness
  - → Robustness ⇒ ability to handle correct incorrect inputs.

* Descriptive Techniques * Password (7 → 8)
* Corrective Techniques

Technique Approach

→ • Active (handling error before or after occu
• Passive [unique and correct username (krna. or ye system ki database sy che kr k btaye ga k correct he k nae)

Techniques

- → Returning a neutral value.
- → Substitue the next valid value.
- → logs and time stamps (kisi file mein
- → ERROR codes and messages (~~input~~) send
- → Shut Down (Not crash)

Exception

- Undesirable situation that can arise in the program.
  - → Checked (Compiler can recognize), Compile time
  - → Unchecked (Compile can't recognize/determine (Runtime) beforehand, before execution

Checked (File not found) Uncl...

Exception Handling

→ Try catch (check set of instruction that can cause exception → Try Handling routines for the exceptions if they occurs → Catch).

→ Throw ⇒ New InputMismatchException("This input pattern is not excepted").

→ Throws → Class a signature . multiple at a time (plenty found . etc) → Caller will handle.

→ Try . catch . final (final has scenario main execute he ga.)

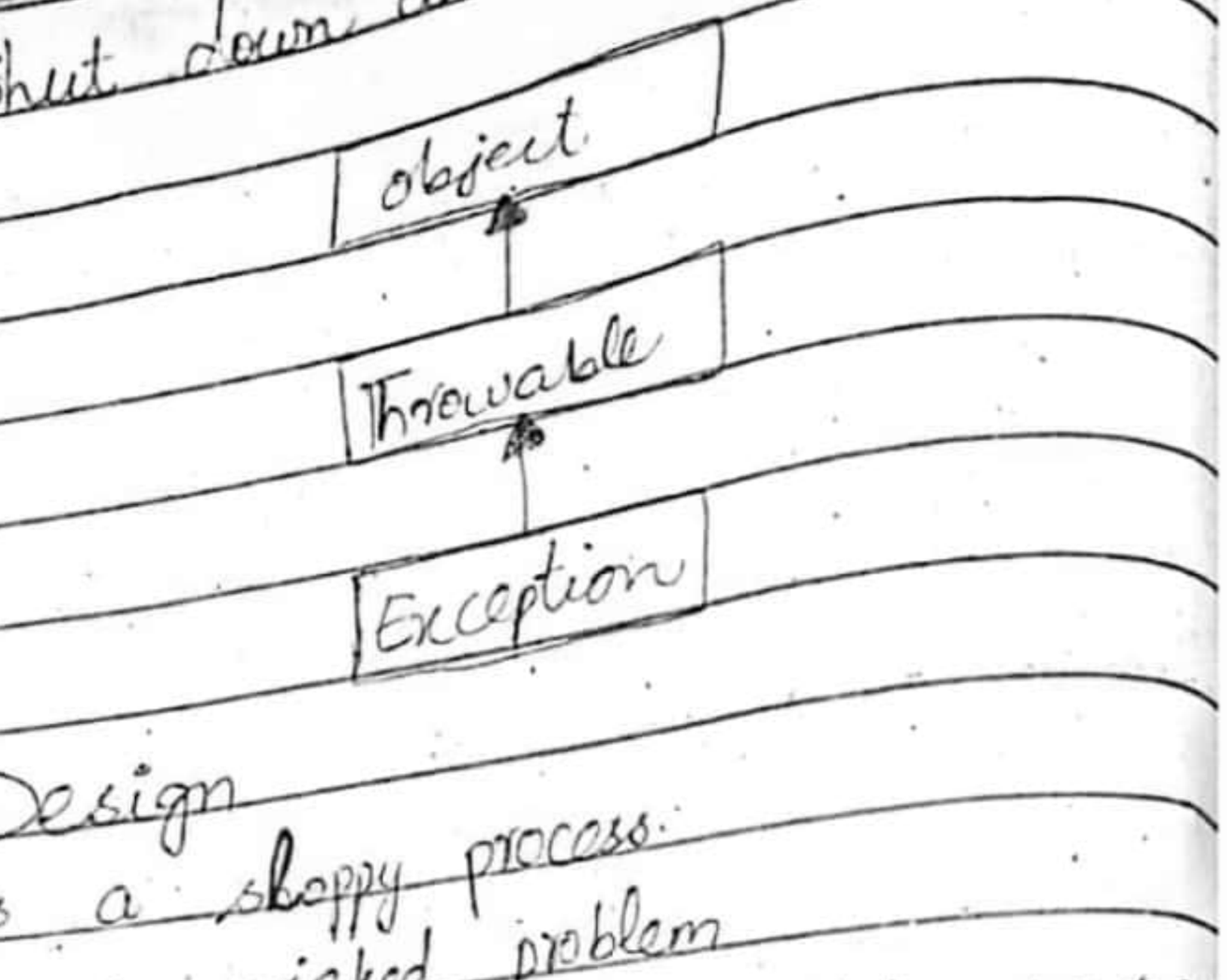
Fault Tolerance

Collection of techniques that increase software reliability by detecting errors and then recovering from them if possible or containing their effects of recovery if possible.

→ In ready state not running state.

① Back Up
② Retrying
③ Auxiliary Code (time tracking of events).
④ Voting Algorithms
⑤ Replacing erroneous input with plenty input (bogous but acceptable).

6. Shut down and restart



k Design

ii - is a sloppy process.

q - is a wicked problem

a - is about trade off, priorities is restricted

e - is non deterministic is it is heuristic process
(hit & trial)

n - is emergent

i Characteristics:

a - Minimal Complexity

c - Easily maintainable

t - Loose coupling

I - Extensibility

- Reusability

- High fan-in (a class is used by many other classes).

- Portability

- Leanness (nothing unnessary is implemented)

→ low to medium fan-out (a class use low to medium number of other classes)

- Stadification (Sorted format)

* Keep levels of decomposition stratified so that you can view the system at any single level and get a consistent view.
- Standard techniques

Principles

Abstraction → Hiding implementation details
Abstract , interface

Encapsulation → attributes & functions are contained in a single object.
access → Information hiding (hiding internal data from other classes).

- Separation of concerns

Polymorphism → overloading + Overfolding
(Same class) (inheritence)

Inheritence → Object

Modularity: Breaking down of a component and reassembling it.

- Design measure
- Coupling
- Cohesion
- Information Hiding
- Separation of concerns (private, getter setter).

SOLID principle :

S → Single responsibility

O → Open close

CamScanner

L → Liskov's principle / substitution (100% use
(Defining) Every subclass/derived class
should be substitutable for their parent

I ⇒ Interface Segregation
(A client should never be forced to implement
an interface that it doesn't use.

D ⇒ Dependency inversion
(High level classes should not depend upo
Low level classes instead both use abstrac