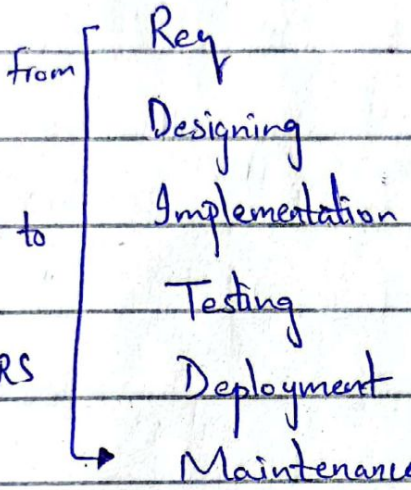
SCD-1 (07/07/23)

1) Software Engineering.

↳ 19% of softwares got rejected due to failure in req gathering.

↳ Understand domain



SCD-2 (08/07/23)

Domain:

↳ This involves req gathering:
System Req / Specification.

SDLC

↳ In this, firstly we find the domain, problem on which we are working:

Req:

User Req:

System Req:/Specification

Non-functional Req:

Waterfall Model:

↳ We follow steps, like firstly gather req then design.

+ point ↳ Each step is performed deeply.

- point ↳ Can't implement where requirements change frequently.

SCD-3 (12/09/23)

Software Engineering — Coding

1. Understand Problem Domain.
2. Req Gathering
3. Architecture.
4. Design.

2 Req Gathering

- Functional Req.
  - a) User Req. (Only what is Req). ← Post an update
  - b) System Req. (How req are implemented). ← must login to post

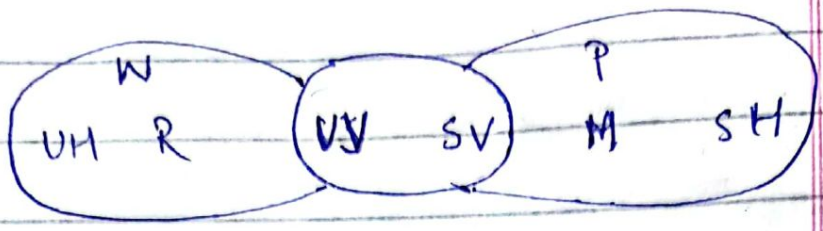
FUR: post

FSS: login, Internet, Unblock → email/password ~~phone/pin~~

- Non-Functional Req.
  - Product Req. (OS version, hardware version).
  - Organizational Req. (NADRA verification)

- External Req (External companies req)

WRSPM



U → User S → System.

W → World (Domain)

R → Requirement.

S → Specification.

P → Program.

M → Hardware Machine.

S → R · Specification specifies req

P & M → S

③ Architecture.

↳ Decompose an enterprise system into independent sub-systems that have value in the system.

→ How these systems interact?

→ Principles and guidelines for the design evolution over time.

Bad Architecture ⇒ Bad Design.

4- Design.

SCD-4 (13/07/23)

- Component is build by modules.
- Module is not independent.

Persons or companies may own cars. The car owner ID is the ID either the person or the company that owns the car. A car may have only one owner (person or company). A car may have a loan or multiple loans. A bank provides a loan to a person or a company for the purchase of a car. Only the loan owner may

obtain a loan on the car. The car owner type and the loan customer type indicate whether the car owner/loan holder is a person or a company.

- **Make a list of Noun**

Class:

1) Person — name, CNIC, address, ph #
2) Company — name, address.
3) Car — Type ID, owner ID, (P, C) . 1 — purchase
4) Loan
5) Bank.

Q: Why the client needs that software?
Q: What problems he/she wants to resolve?

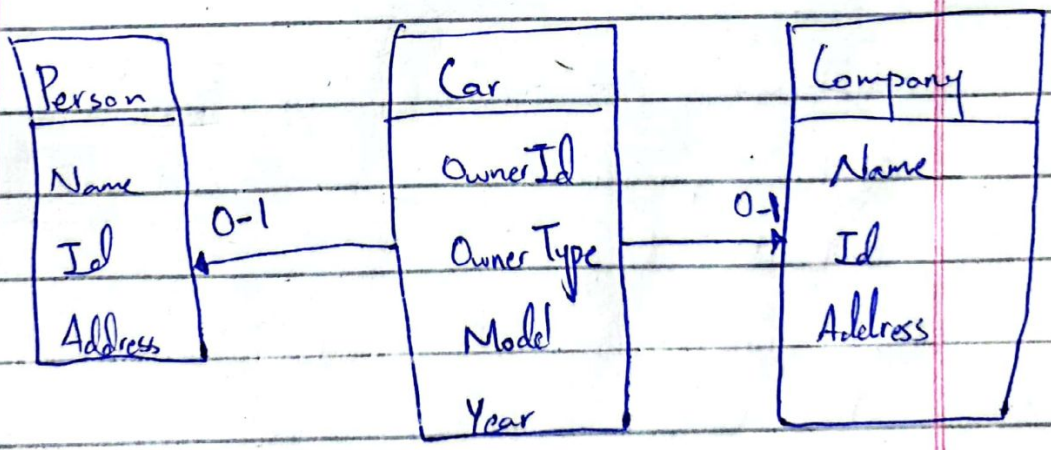
SCD-5 (19/09/23)

Scenario:

(Previous)

⇒ Nouns are used for classes.

Person	Company	Car	Loan	Bank.
Name	Name	OwnerId	CustomerType	Name
Id	Id	OwnerType	Id	Address
Address	Address	Model	Amount	BankId
		Year	Date	
			AccountId.	



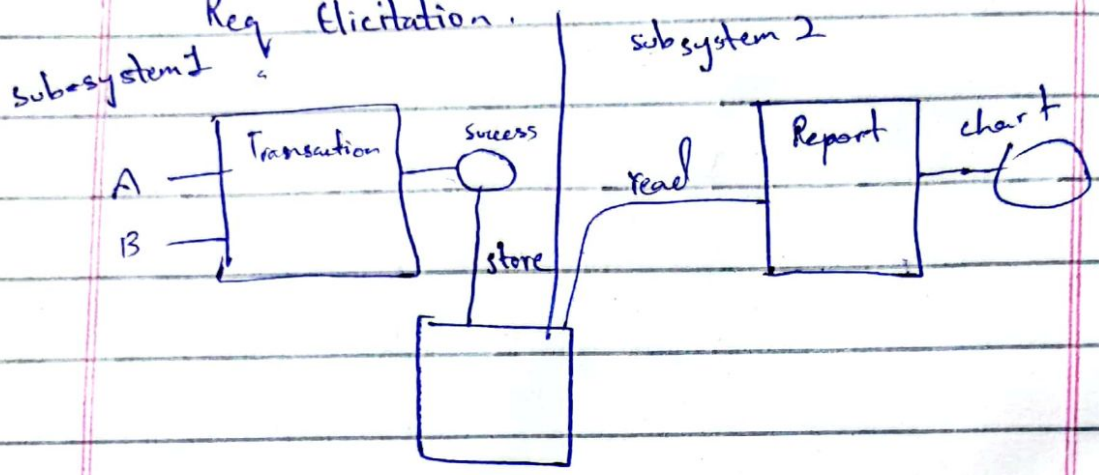
SCD-6 (20/07/23)

Domain Modeling:

Understand the problem

U & E

WRSPM



Software Architecture Models.

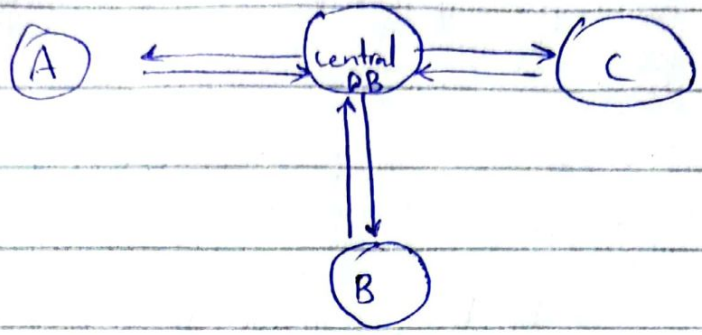
1) Pipe & filter
2) Blackboard.
3) Layered
4) Client server
5) Event Based.

1) Pipe & filter.

Same input & same output everywhere. Eg compiler.

2) Blackboard.

• Central System.



3) Layered Architecture.

⇒ view, controller.

4) Client - Server

Req → Response.

5) Event - based

Auto response on event.

⇒ similar to blackboard.

Concerns:

System Structure (Decompose into components).

Intraction of components.

Modular Decomposition: (software Design - SD)

Modularity:

Breaking down of a component and reassembling / interaction.

Modules:

1) Coupling (low coupled).
2) Cohesion (highly cohesive)
3) Information Hiding.
4) Data Encapsulation.

SCD-7 (26/07/23)

- Data Encapsulation.
- Information Hiding.
- Coupling. — If change in one class, it effects 2nd class ⇒ tightly coupling.
- Cohesion.

(Lose Coupling is recommend)

Tight Coupling:

Contents Coupling — 1) Module A can directly access Module B data members.

Common Coupling — 2) Module A and B are relied on same global data.

External Coupling — 3) Module relying on externally imposed format / protocol / interface).

Medium Coupled: (Acceptable)

Control Coupling — 1) Module A controls the logical flow of module B by passing implementation or by using flags.

data structure coupling — 2) Module A & B rely on some ~~composites~~ composite data structures. Changing data structure directly affects the other module.

Loose Coupling:

Data Coupling: 1) Module A only pass parameters for requesting functionality of Module B.

Message Coupling: 2) Module A sends messages to Module B.

3) (No Coupling)

Weak Cohesion:

coincidential: 1) Different parts of module are together just because they are in a single file.

Temporal: 2) Different parts/code/functions are activated at the same time (move to one file).

Procedural: 3) One part follow the other in time.

4) Similar parts/function are grouped. They are similar but perform different things.

Medium Cohesion:

1) All elements operate on similar inputs produces same output.

2) One part output serves as input to other part.

Strong Cohesion

Object
- 1) Each operation in module can manipulate object attributes.

functional
- 2) Every part of the function is necessary for execution of single well-defined behaviour

Inheritance
- Domain
- Requirement Elicitation.
- Software Architecture.

Construction
- ⑤ Software Design.
- ⑧ Coding & Debugging.
- ⑥ Unit Testing.
- ⑦ Integration Testing.
- ⑧ Integration.

System Testing.

Architecture Maintenance

Construction Phase → 4, 5, 6, 7, 8.

→ Building the software.

→ It takes 30-80% of the time.

Black box Testing

- Done by QA
- whole code is tested.

White Box Testing

We test it on modular approach.

V & V process

Validation & verification

Validation:

We check that in req-elicitation we check the req we gathered have completed all requirements?

Verification.

We hand over the project

to client, we ensure the client is agree on this once got agree it is verified.

Deployment:

- Physical environment.
- Hardware : also he can rollback too.
- Documentation.
- Training.
- DB related concerns.
- 3rd Party software.
- Software.

Failure/Maintenance

Gold Backup.
WARM Standby.
HOT FAILURE

Why construction is formal.

→ All paper work is useless until you don't provide the code in working app, which is done on this phase.

⇒ 30 to 80%.

- Your working code is your actual documentation, So you can tackle most of the by looking at code

⇒ Does your developer know the language.
⇒ What that language provides you.
⇒ low level language ⇒ less performance.

- Key Construction Decision:
  - Choice of programming language.
  - Programming Convention.
  - Current technology wave.

  Selection of Major Construction Practice.

⇒ Your system should be such that if new things are added or changed then the rest should not be disturbed.

⇒ Your code should be reusable.
⇒ What problem is resolved by the software?
⇒ Constraints / Business Rule.