SDLC → System Development Life Cycle
(How to develop software?)

Agile Method:

↳ most usable now a days

- scrum → implementation of Agile Method
- include customer is development process
  → take customer's feedback during development period.
- If customer's change his mind during development process, Agile method is easy to modify.

SRS

↳ Software Requirements Specification Documents

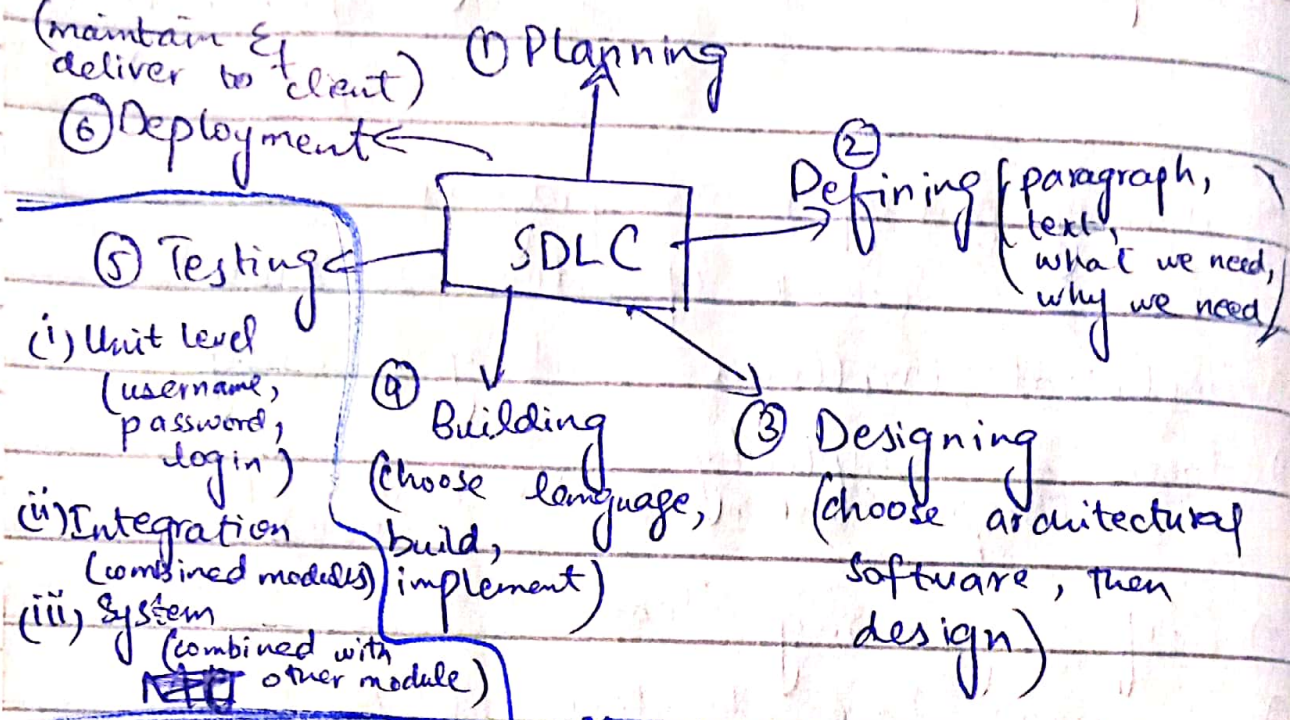
It includes

- UML
- user story
- story-boards
- use cases
- func requirements
- non-func requirements

Principles you follow while Agile Method

1. Early & Continuous Delivery (call clients after 2 weeks & discuss)
2. Working prototypes as progress (chunk of actual software)
3. Technical Excellence and good design
4. Focus on Simplicity (simple solution is the best solution)
5. Self organizing Teams
6. Encourage Face to Face Interaction
7. Deliver Frequently (Gantt Chart)
8. Welcome Changing Requirements
9. Sustainable Development
10. Build Projects around Motivated People
11. Daily collaboration
12. Reflect on Team Behavior.

How to Design software after completion of requirements?



MUC → Model you control

- most used architectural software

Database Data
- logic

UI
- cin
- cout

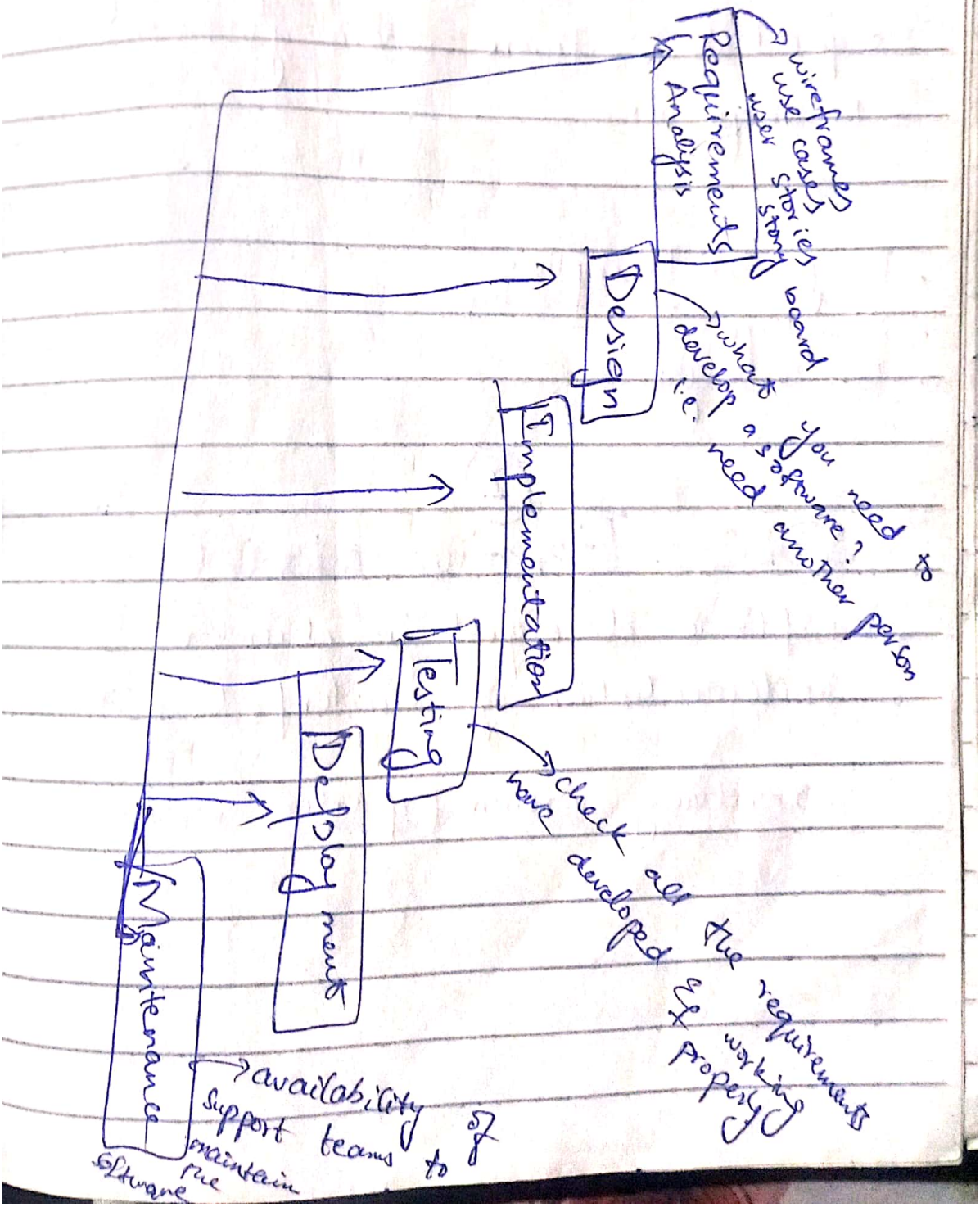
communicate

- Controller (C) takes input
- pass it to Model (M)
- Model processes
- return output to Controller (C)

SDLC → combines all 6 (above mentioned) process.

Waterfall Model

- basic / very first model
- linear sequential life cycle model
- every model has base of waterfall model



(4) Spiral Model

~~Identify~~ . identify risk of software

→ The only model in all models where you (before development) try to discover or identify the risks/errors that may occur during implementation.

① Identification
② Design → Module Design (Different parts of software)
→ System Design (combine all modules)
→ Architectural Design (what database used)

POC → proof of concept (like SE proposal)

③ Construct or Build
④ Evaluation & risk analysis

→ delivery in small modules.

Prototypes: small functionality of actual software

Tables → SQL Database
.json files → noSQL Database

⑤ V-Model

Verification & Validation Model

- Verification: checking if all the requirements has been developed
- Validation: checking if all the requirements are working fine

unit → module → architectural → system

- **Module Design:** low level i.e. CS campus
- **Architectural Design:** High level i.e. CS campus, KSK, VHR
- **System Design:** combination of High level & low level i.e. uni. management system

- **Unit Testing**

Text [Name]
- Alphabet ✓
- Numeric ✗
- Special character ✗

Text [Email]
- Special charcters (. @ -) ✓
- Numeric ✓
- Alphabet ✓

Integration Testing:

Testing all items of one module in-depth at a time.

i.e. sign-in

System Testing:

Testing all modules at a time.

Acceptance Testing:

- Uses use cases & ~~docs~~ test software according to user's wants.
- whether the user accept it or not

- V-Model highly dependent on testers
- not-dynamic → It is static
  - not change at run-time
- requirements can't change

D/f b/w waterfall & V-Model

In waterfall, testing is done after development.

Testing after requirements gathering/ analysis or before coding.

RAD (Rapid Application Development)

-> different teams working on different modules

=> Business Modeling
how and when is the information processed

=> Data Modeling
review & analyze

=> Process Modeling
any changes or enhancements

=> Application Generation
coding

=> Testing and Turnover
↳ (delivery of product)

RAD	~~SDLC~~ Iterative
• can be modified	• rigid, we can't change requirements
• iterative & incremental delivery	
• delivers product quickly	

SDLC → Software Prototype Model

- build functionality in small chunks like (iterative or RAD)
- evaluate developer proposal
  - ⇒ Basic Requirement Identification (all requirements i.e. frontend & backend)
  - ⇒ Developing the initial prototype . not exact function

=> Review of the Prototype
. take user's feedback

=> Revise & Enhance the prototype
. time & budget

Software Prototyping Types

i. Throw away Rapid Prototyping
. rapid or close ending prototyping
. know actual functionality
. SRS already signed

[Throw]

ii. Evaluationary Prototyping (Base Model)
. breadboard prototyping
↳ (main functionality that actually exists)

- create design of actual project
- system built on it

→ Not throw

(iii) Incremental Prototyping

- Make sub-systems through prototypes
- Prototypes integrate to form whole system

(iv) Extreme Prototyping

- UI developed firstly
- backend services layer
  ↳ connect with database to form complete system

Implementation Model of Agile Process

- scrum & XP (extreme programing)

⇒ Scrum

- 3-9 team members

⇒ XP

- Two Members
- XP was first method of Agile Model

Array maped on FIFO becomes queue i.e. printer