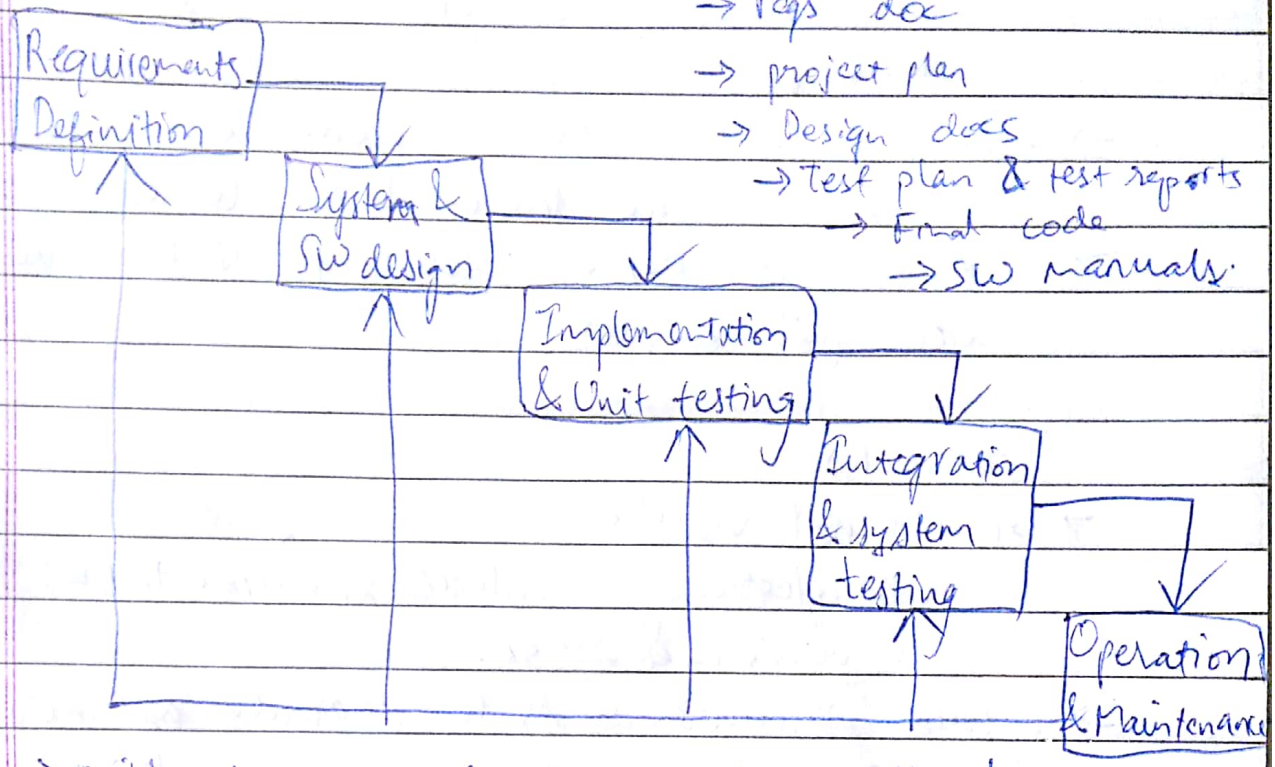
SE INTERNALS

~~Re~~ Process Models :

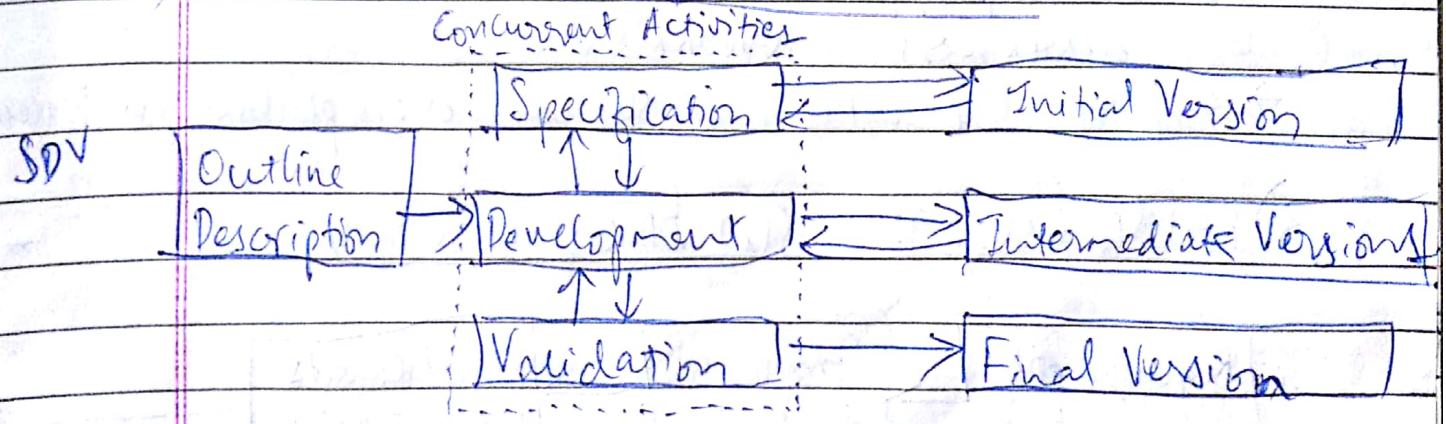
1) WATERFALL MODEL :



Docs :
- → reqs doc
- → project plan
- → Design docs
- → Test plan & test reports
- → Final code
- → SW manuals.

→ visible process, progress can be monitored.
→ inflexible partitioning of project into different stages
→ Commitments must be made at an early stage in the process, which makes it difficult to ~~respond~~ respond to changing customer requirements
Freezing of requirements.

2) INCREMENTAL DEVELOPMENT :



→ series of steps
→ backtracking

Scanned by CamScanner

Benefits compared to Waterfall

→ Cost of accomodating changing reqs ↓.
Amount of analysis & documentation that should be redone ↓.

→ Easier to get customer feedback.
Customer can see demos & implementation.

→ More rapid delivery & deployment possible even before full completion.

Drawbacks:

→ Process not visible.
Not cost effective to produce documentation at every version & stage.

→ System structure tends to degrade as incremented.
Regular change corrupts its structure.

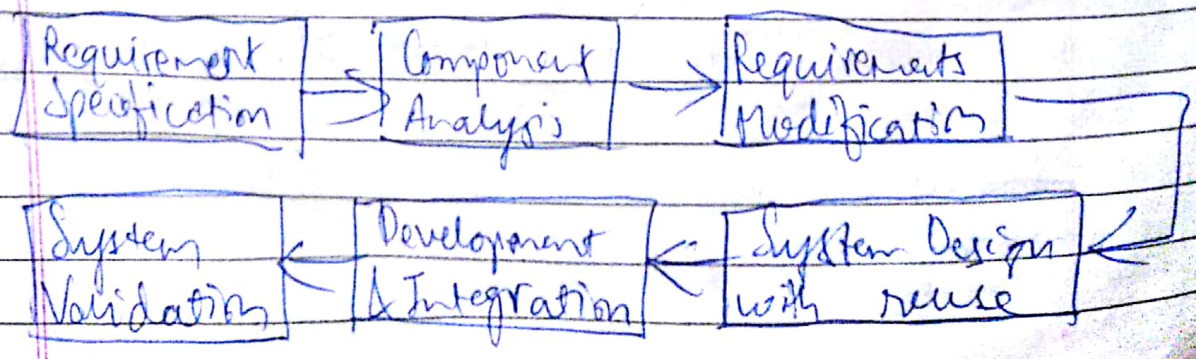
→ Incorporating new changes gets increasingly costly & difficult.

Solution:

→ develop incrementally and just expose it to customers for feedback without delivery/deployment in customer's environment.

→ Experimenting with new SW can disrupt business process

3) RE-USE ORIENTED SW ENGINEERING



Scanned by CamScanner

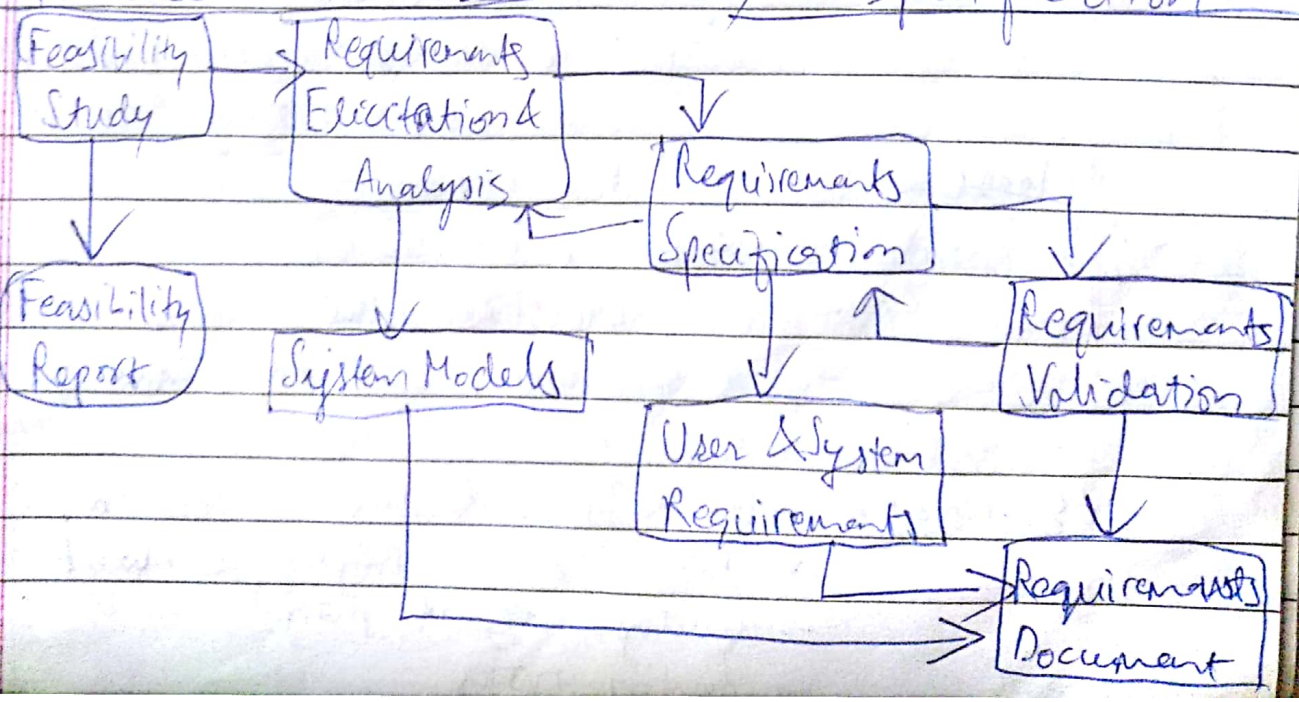
→ when designs/code which are similar to what is required already exist. Modified & integrated into own system.
→ COTS: commercially off-the-shelf systems.

Advantages:
→ reducing amount of SW dev
→ reducing cost, time, risks
→ faster delivery

Drawbacks
→ May not meet reqs when reqs specs change
→ Control over system evolution is lost as reusable components may not be under developer's control.

3 types of SW components used:
1) Web services
2) Collections of objects that are ~~given~~ developed as a package
ex: .NET, J2EE
3) Stand Alone ~~syst~~ SW systems.

PROCESS ACTIVITIES: 1) SW specification:



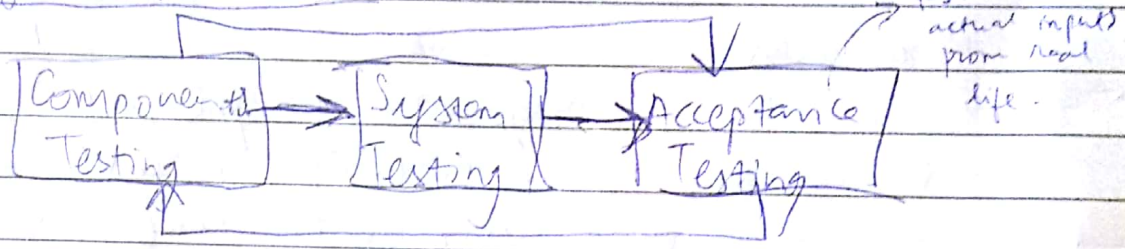
Scanned by CamScanner

2) SW design and implementation

[Design Inputs] → [Design Activities] → [Design Outputs]

a) Architectural Design
AICT b) Interface Design
c) Components Design
d) Database design

3) Software validation



4) Software evolution

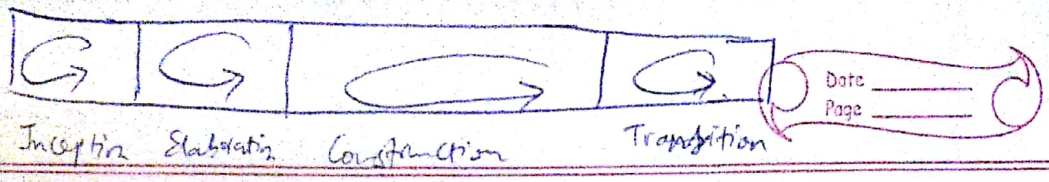
IECT RUP- Rational Unified Process:
Hybrid Process Model.
3 perspectives: → dynamic: phases of the model over time
→ static: shows process activities that are enacted
→ practice: suggests good practices to be used during process

~~Phases:~~

1) Inception: → business case establishing
→ identify users & entities
→ what contribution SW will make to business

2) Elaboration: → develop understanding of problem.
→ establish architectural framework.
→ develop project plan
→ identify risks

Scanned by CamScanner



→ develop requirements model

3) Construction → system design, programming, testing
→ working SW system
→ complete documentation.

4) Transition : → Moving the system from developer → user community.
→ make it work in real time environment.
→ documented software system ✓.

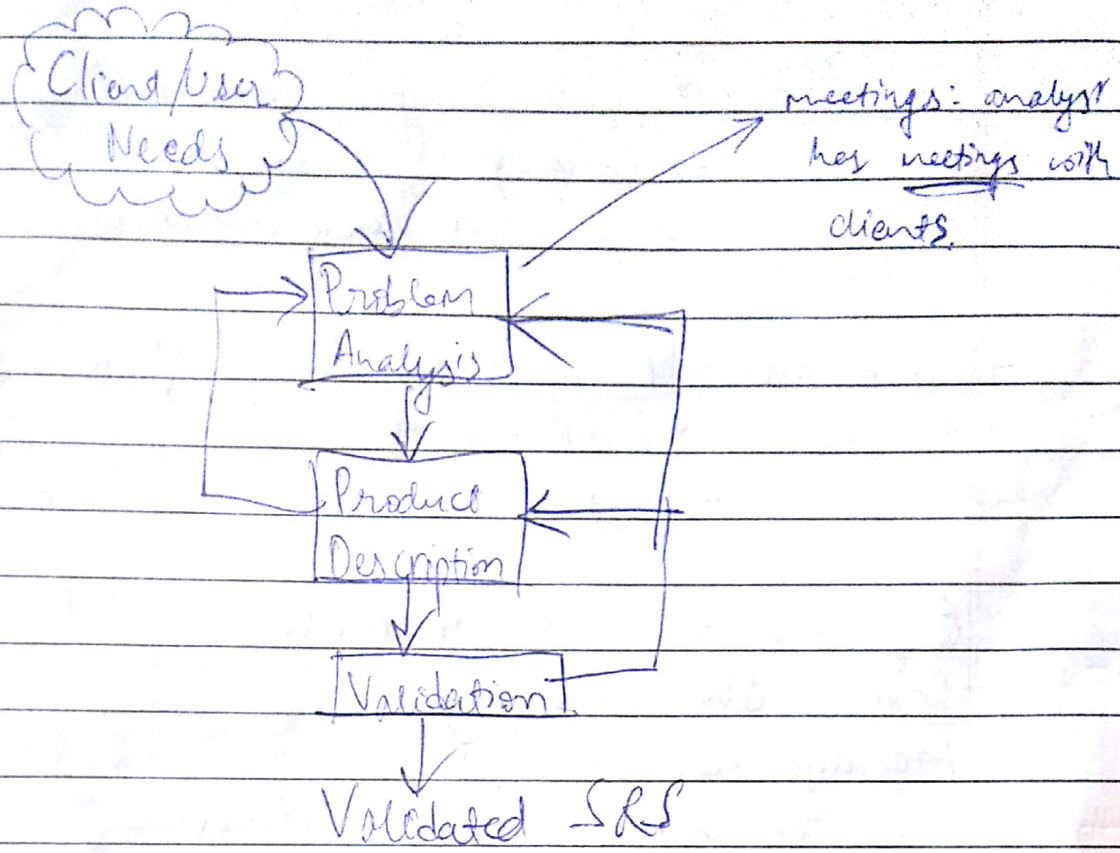
6 fundamental best practices:
1) Develop SW iteratively
2) Manage requirements
3) Use component-based architectures
4) Visually model SW → to represent
5) Verify SW quality
6) Control changes to ~~so~~ SW.

VALUE OF A GOOD SRS:

→ concerns of user to be communicated to developer.
→ bridge gap b/w parties involved → shared vision
→ establishes agreement b/w client & supplier on what product will do
→ Developer understands what client expects.
→ Provides reference for validation of final product.
→ helps client determine if all reqs have been met
→ high quality SRS → high quality SW
→ high quality SRS reduces development cost.

Scanned by CamScanner

Requirement Process :



ADV

Requirements Specification :

Desirable Characteristics of an SRS

CUVCR

- → Correct.
- → Complete → sufficient detail
- → Unambiguous → only one interpretation
- → Verifiable
- → Consistent → no conflicts.
- → Ranked for importance and/or stability.

Components of an SRS :

FPDE

- → Functionality
- → Performance → Static / Dynamic
- → Design constraints
- → External Interfaces
- → Standards Compliance
- → Hardware Limitations
- → Reliability & Fault Tolerance → recovery requirements.
- → Security

Scanned by CamScanner

Structure of a Requirements Document

PSDRO

1. Introduction
   - 1.1 Purpose ✓
   - 1.2 Scope ✓
   - 1.3 Definitions Acronyms and Abbreviations
   - 1.4 References ✓
   - 1.5 Overview

PRUGA

2. Overall Description
   - 2.1 Product perspective ✓
   - 2.2 Product functions ✓
   - 2.3 User characteristics —
   - 2.4 General Constraints ✓
   - 2.5 Assumptions & Dependencies ✓
3. Specific Requirements
   - 3.1 External Interface Requirements.
     - 3.1.1 User Interfaces ✓
     - 3.1.2 HW interfaces ✓
     - 3.1.3 SW interfaces ✓
     - 3.1.4 Communication interfaces ✓
   - 3.2 Functional Requirements ✓
   - 3.3 Performance requirements ✓
   - 3.4 Design Constraints
   - 3.5 Attributes
   - 3.6 Other requirements

(*) When use cases are employed, functional reqs section of SRS is replaced by use-case descriptions

Scanned by CamScanner

Functional Specification with Use case:

Use cases specify the functionality of a system by specifying the behaviour of the system captured as interactions of the users with the system.

Basics:

→ Primary Actor
→ Preconditions
→ Main success scenario
→ Exception/extension scenario

Use cases:

→ system behaviour can be specified.
→ functionality can be specified
→ textual descriptions
→ Both developers & client can understand.

SEE EXAMPLE

thing (use case)	its Scope
business use case	enterprise or organisation
system use case	system.
component use case	subsystems

Develop Use cases:

Step 1) Identify actors and their goals
Step 2) Specify main success scenario
Step 3) Enumerating failure conditions
Step 4) Handle failure scenarios

Scanned by CamScanner

Other use cases:

1) Data Flow Diagrams:

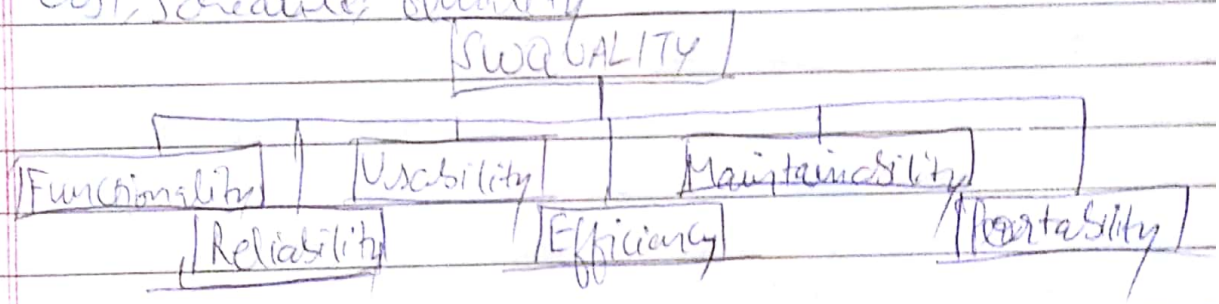
→ shows the movement of data through processes
→ Process shown by circles
→ data flows → arrows
→ rectangle → source or sink, consumer of data.
→ All external files → straight labeled line

2) ER diagrams

SOFTWARE PROBLEM & PROCESS:

Cost, Schedule, Quality

FRUEMP

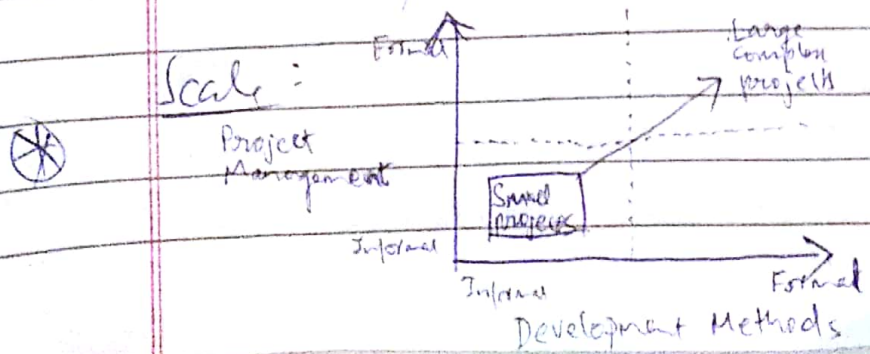


SW quality standards:

→ Functionality: performs ~~sq~~ spec. reqs.
→ Reliability: failure free service
→ Usability: capability to be understood and used.
→ Efficiency: meets performance standards.
→ Maintainability: capability to be modified for correction
→ Portability: capability to be adapted into new env.

Cost of dev. SW: person-months of effort spent in dev
Productivity measure: LOC per person per month

Scale:



Scanned by CamScanner

(1) Component SW Processes: Software Process

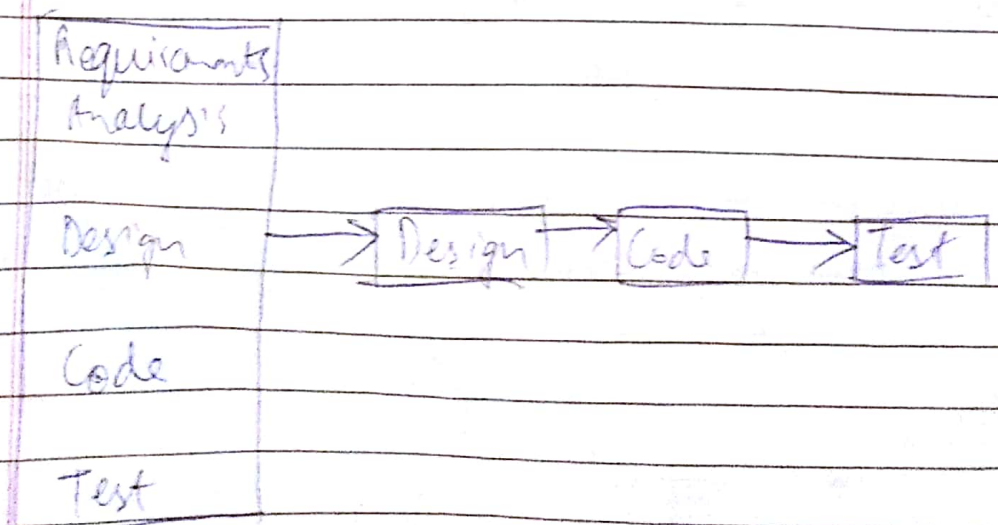
- Product Engineering Process
  - Development Process
  - Project Management Process
  - Software Configuration Management Process
- Process Management Process (to analyse & improve overall process)

Other models:

5) Prototyping Model:

→ throwaway prototype built based on currently known requirements.
→ client gets actual feel of SW. → inputs & feedback
→ demonstrate feasibility.
→ Risks reduce
→ Focus is to include features that are not properly understood.
→ "Quick & Dirty" approach
→ Minimal documentation.

DCT

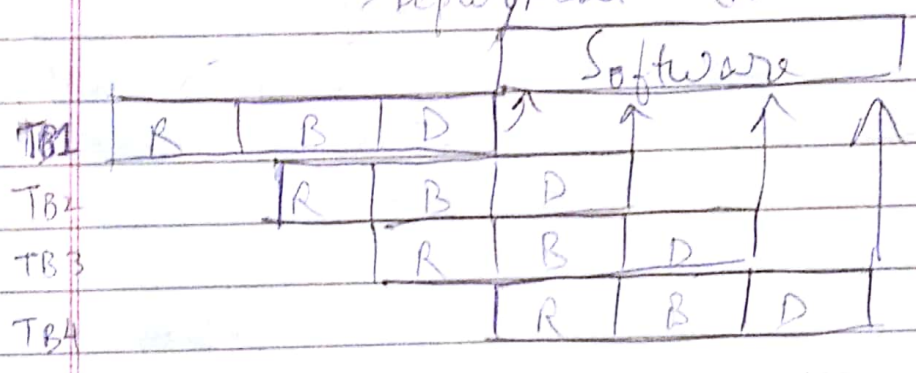


Scanned by CamScanner

6) Time Boxing Model

- → parallelism b/w iterations
- → new iteration commences before current iteration release
- → reduced avg. delivery time
- → Time box → fixed duration of ~~iteration~~ each stage / iteration.
- → dedicated team for each stage
- → pipelining of iterations
- → Stages → requirement spec (R)
  - → Build (B)
  - → Deployment (D)

RBD

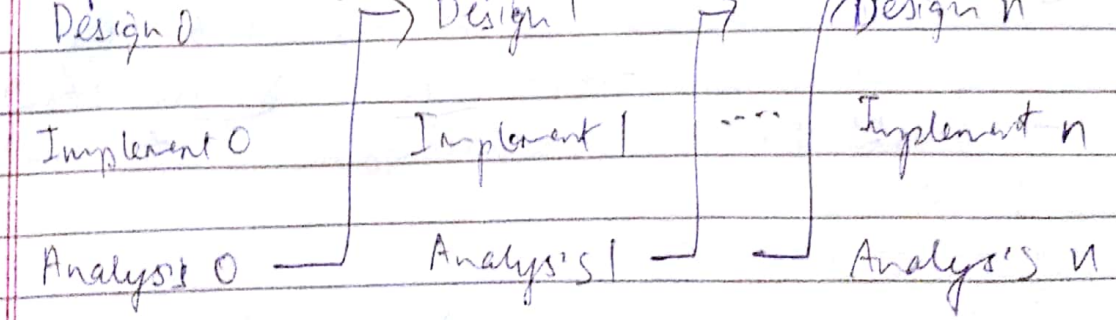


R team	Req. Anal for TB1	Req Anal for TB2	Req Anal for TB3
B team		Build for TB1	Build for TB2
D team			Deployment for TB1

7) Iterative Model : Increments

3 phases: design, implementation, analysis

DIA



OR :

Requirements & Architecture Design → waterfall

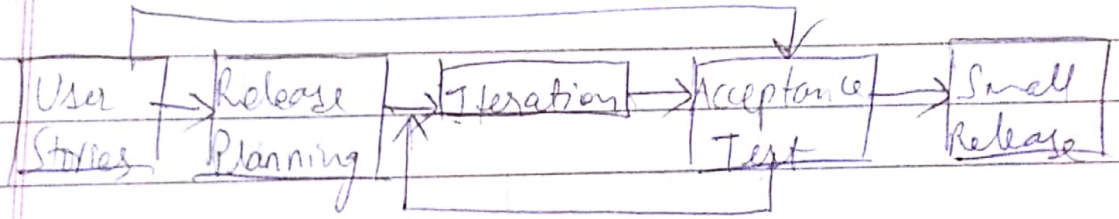
Delivery & development of SW → iterative

Scanned by CamScanner

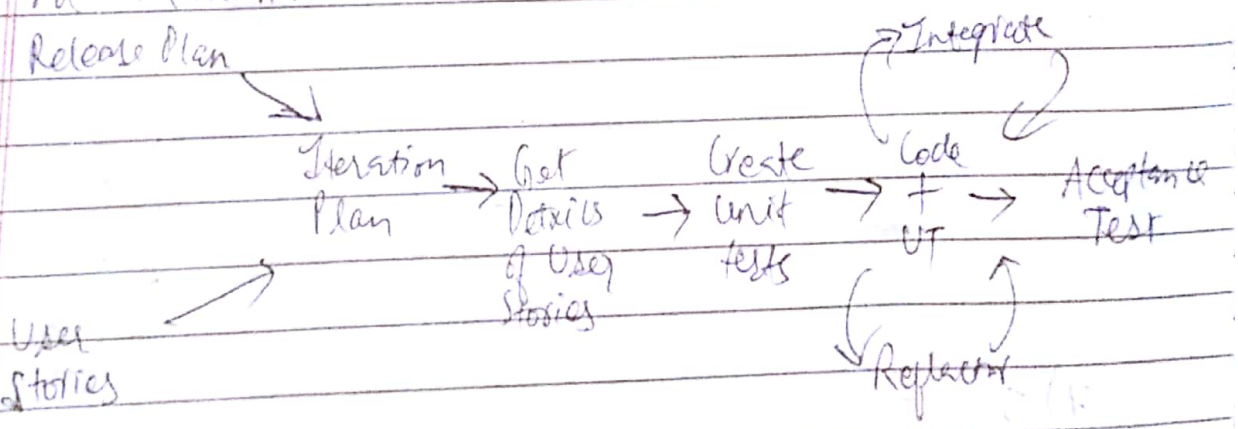
8) Extreme Programming & Agile Process:

→ Working SW is measure of progress
→ Even late changes in req should be entertained
→ SW should be developed & delivered rapidly in small increments
→ Face to face communication is preferred over documentation
→ Continuous feedback & customer involvement
→ Simple but evolving system design (rather than elaborate)
→ starts with user stories

URIAS



An iteration:



PROJECT MANAGEMENT PROCESS:

→ necessary to ensure that the engineering process ends up meeting objectives of cost, schedule & quality.
→ 3 phases: planning, monitoring & control, termination analysis.

1) Planning
→ cost estimation
→ schedule & milestone determination
→ project staffing
→ quality control plans
→ controlling & monitoring plans

Scanned by CamScanner

2) Project Monitoring & control phase
→ longest duration
→ all activities of development
→ monitoring potential risks
→ info provided after each step

3) Termination analysis :
→ after development is over
→ provide info about dev process to learn from it.
→ post mortem analysis

Scanned by CamScanner