Date:
P. No:

L) Monday - 11:30
L) Thursday - 11:30
T) Friday - 10:30
Lab) Friday - 02:30

Textbooks: R.S. Pressman (+)
Sommerville (☆)
Pankaj Jalote
Rajiv Mall

Software Engineering

→ Systematic & standardized approach in place of individualized & subjective development. Less arbitrary. More robust.

→ Gives a set of procedures. Makes it repeatable
- What needs to be done
- How it should be done.
- Irrespective of who works on it.

→ Set of standard procedures → Varies with org. customizable but core stages the same.

Date:
P. No:

Software engineering →

Software dev. lifecycle as a set of (series of) steps to be followed.

→ Requirements elicitation → specific to "software service" instead of "software products".

Feasibility analysis comes before elicitation.

→ Rapid prototyping:

→ Brainstorming session:

Generation Phase

Consolidation Phase

- Functional features
- Non-functional features (good-to-have)
- Functional req. must be prioritized.
- SRS - Software Req Specification docs.

(Design is traceable to SRS)

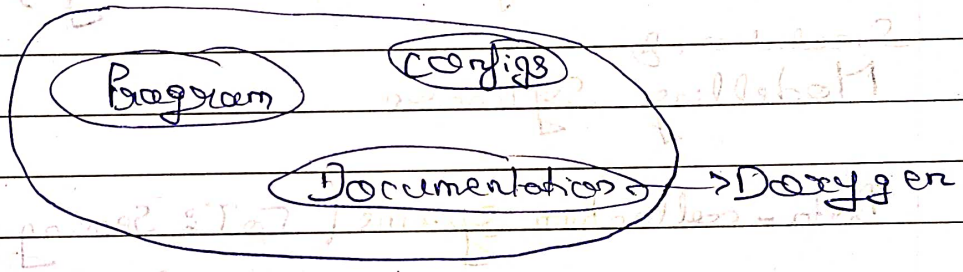
Interviews:-

- In advance

Programs vs Software :-

Programs are written for personal use.
Software → others or community.

Program ⊆ Software



Documentation: Specifies how to use functionalities.

Configuration files: Helps in setup.

Types of software :
- Generic → product
- Customized (bespoke) → service
- Hybrid

Generic: Also analyze the demographic but not a specific client (like customized ones)

Hybrid: ERP, combination of several generic modules is hybrid.

Date:
P. No:

① → Standalone (✕ networking, no dependency, self-sufficient)

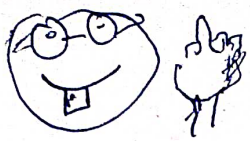
② → Interactive transaction-based (interacts with other people and gets it done. e.g. booking softwares (web-enabled)). Getting boarding passes on airport isn't web-based.

③ → Batch processing software (helps handle big data)
Big data - needs multiple nodes

④ → Entertainment Software (Gaming)

⑤ → ~~Simulation &~~ Modelling Software

Data-collection Systems (IoT: Sensing the environment, involve sensors, cookies analysis)



Date:
P. No:

① → Standalone (× networking, no dependency, self-sufficient)

② → Interactive transaction-based (interacts with other people and gets it done. e.g. booking softwares (web-enabled)). Getting boarding passes on airport isn't web-based.

③ → Batch processing software (helps handle big data) Big data - needs multiple nodes

④ → Entertainment Software (Gaming)

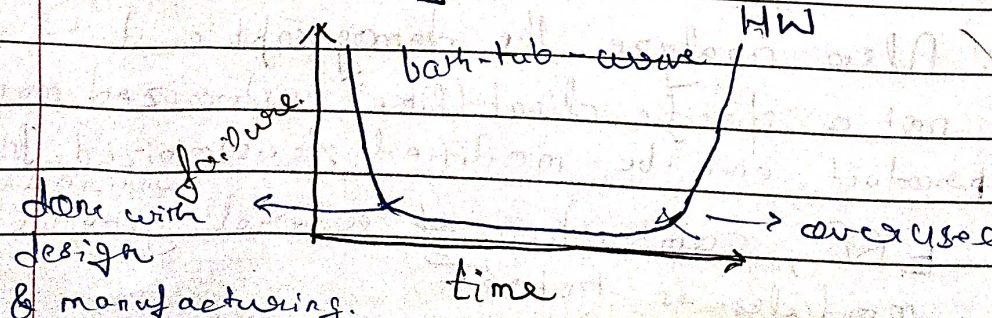
⑤ → ~~Simulation &~~ Modelling Software

Data-collection Systems (IoT: Sensing the environment, involve sensors, cookies analysis)

Software Engineering

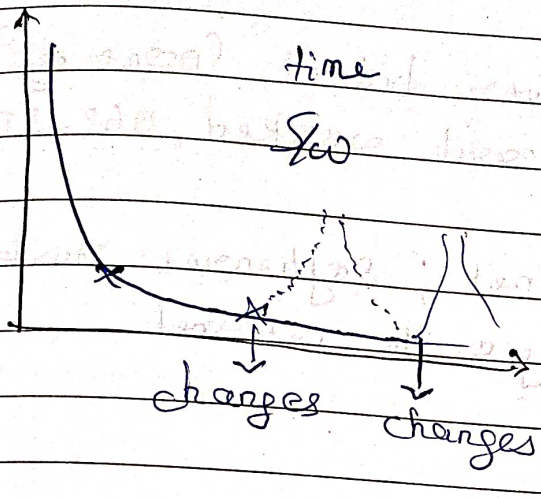
→ Formal & procedural.

→ designing software the same way as traditional hardware design

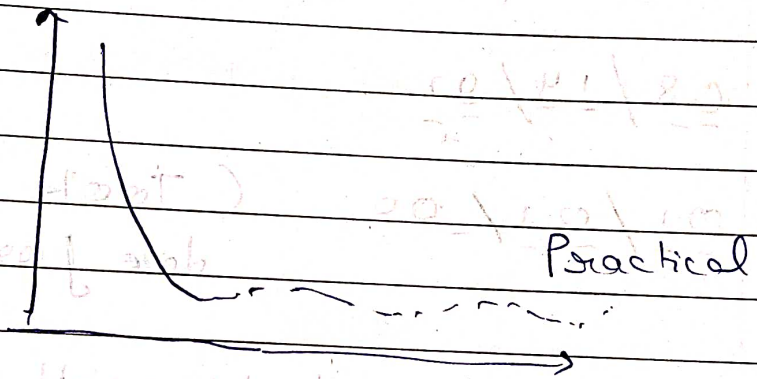


Date:
P. No:

→



Practically:-



Legacy softwares - remain unchanged.

History of Software Engineering :-

1945 - 1965 → origin

1965 - 1985 → S/w crisis
(Hardware evolved faster, unprofessional divided dpvs, not unified)

1968 → 'Software engineering' was coined in NATO Conference on Software Engineering

'International Conference on Software Engineering'
ICSE

1985-1989 → No silver bullet (none of the research worked, 1965-85 a waste)

1990-99 → Internet; softwares started focusing on the Internet.

07/29/91

08/14/97

01/01/00 (Took 5 years to change date format)

2000- → Light weight methodologies

2015 → AI

S/w Engineering - Process of S/w development
⇒ Specification
⇒

(1) Specification: Understanding what's to be developed

(2) Design & development: Endless req. of real world vs limited capacity of prog. world
→ semantic gap.

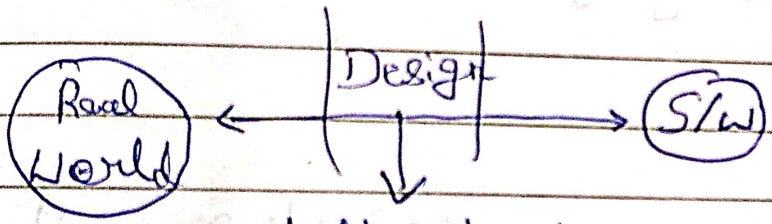
(3) Validation: a) Among the developers (alpha)
b) To the customers (beta)
Unit testing, integration testing

④ Evolution: (Changing) → Preventive
→ Corrective (Bug-fixes)
→ Adaptive
always more

Preventive: improving by anticipating issues in future.

Jan. 17 2024 → Samsung AI

Designing Artifacts



helps bridging the huge gap b/w real world & s/w dev.

Agile dev. → Against the design, jump directly to coding for non-critical softwares.

→ Design is important for real-time & critical softwares Precision.

Design ← 'Model' the requirements in way that bridges the gap.

SCRUM → half-way b/w agile & design.

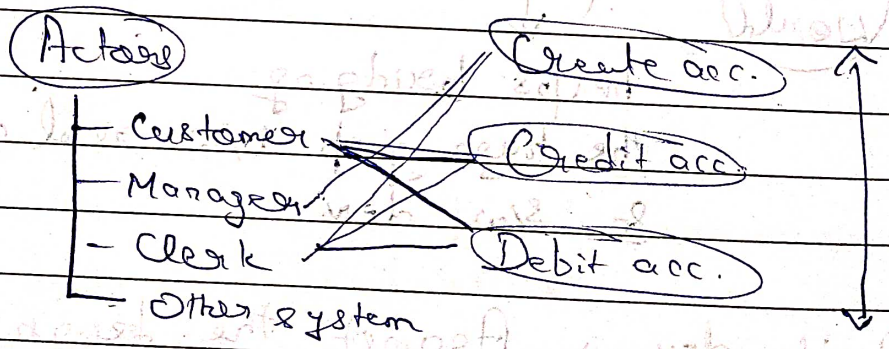
In 1997 → UML (Unified Modelling Language) became almost universally accepted agile technique ↔ extreme programming Aage UML.

UML → 13 diagrams that help modelling a system

(1). Use-Case Diagrams :-

They're made using "functional req".

Eg Banking system.



(2). Class Diagram :-

→ Logical class diagram

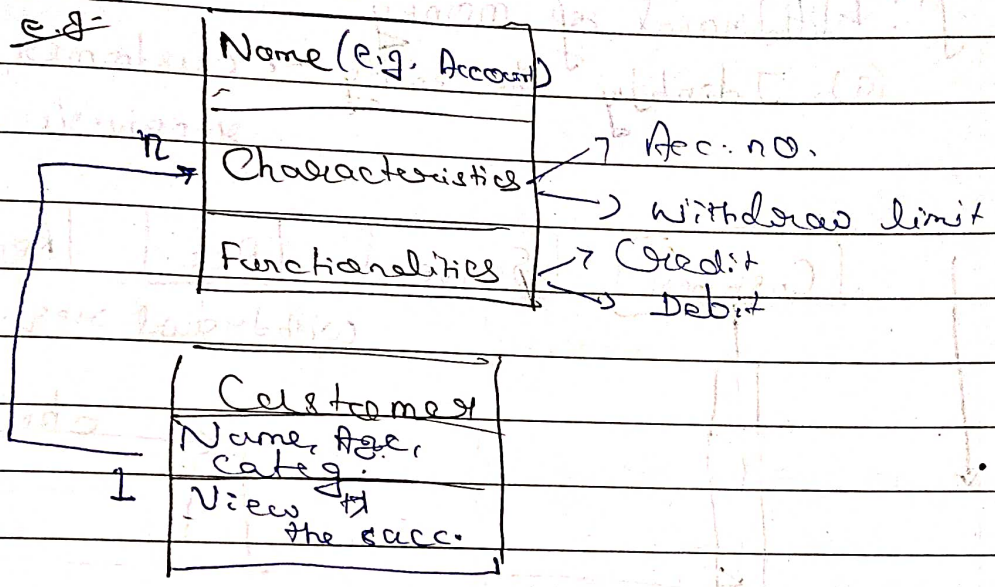
→ Implementation class diagram

Date:
P. No:

Logical Class Diagram

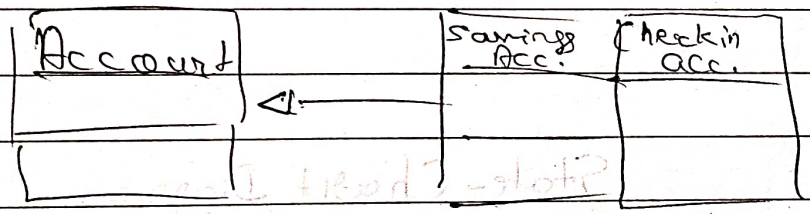
→ Identify entities

e.g.

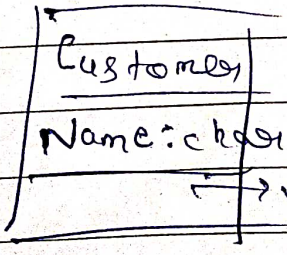


Entities may be connected.

Can create entity using 'inheritance'.



Implementation Class Diagram



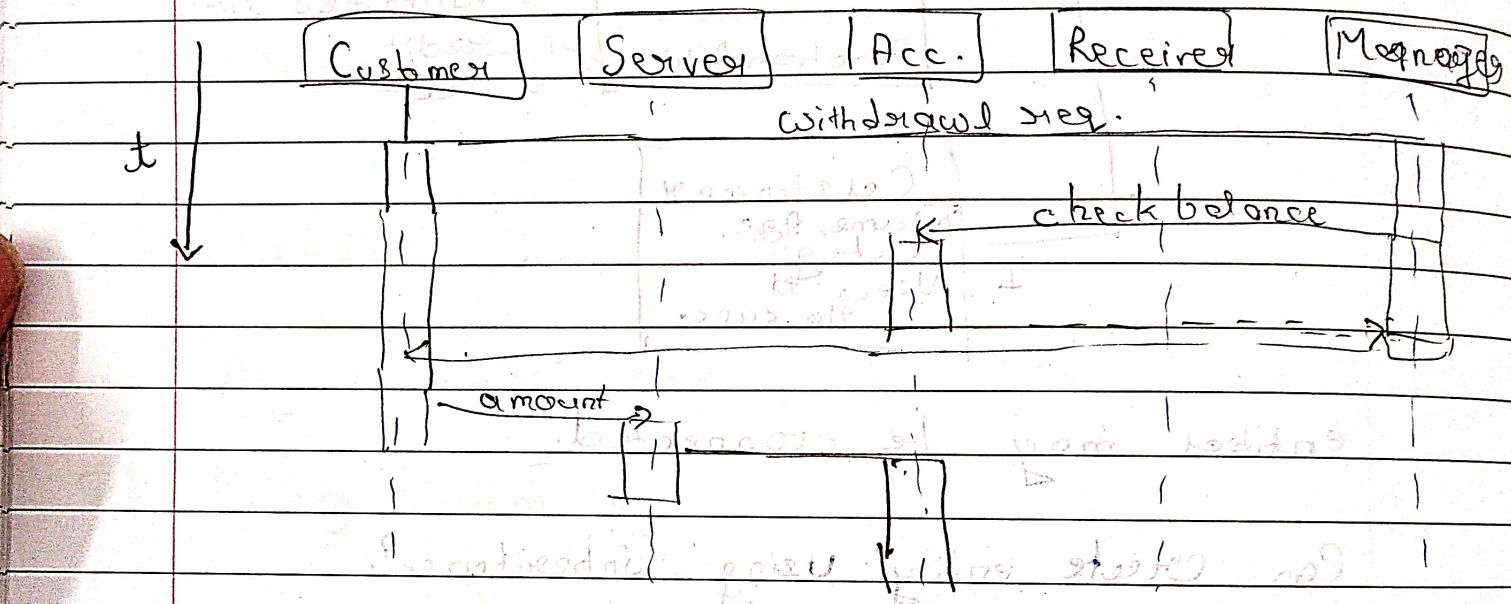
Date:
P. No:

Sequence Diagram

Every f^n listed in use cases needs a seq. diag.

e.g. f^n: Withdrawal of money

@). Identify the entity -> customer, account, receiver's account, ...

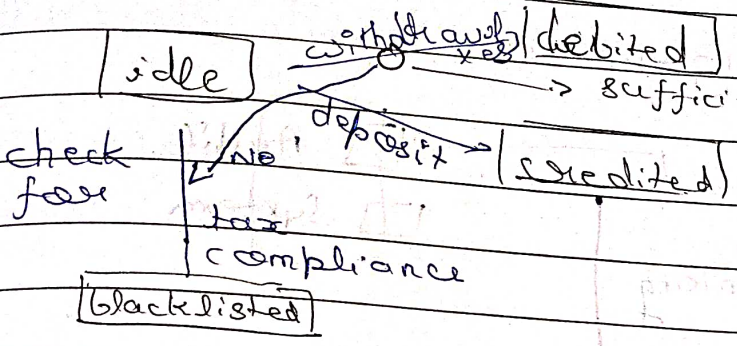


State-Chart Diagram

Every entity in class diag. needs an SCD. Tracks the changes that an entity would go over time.

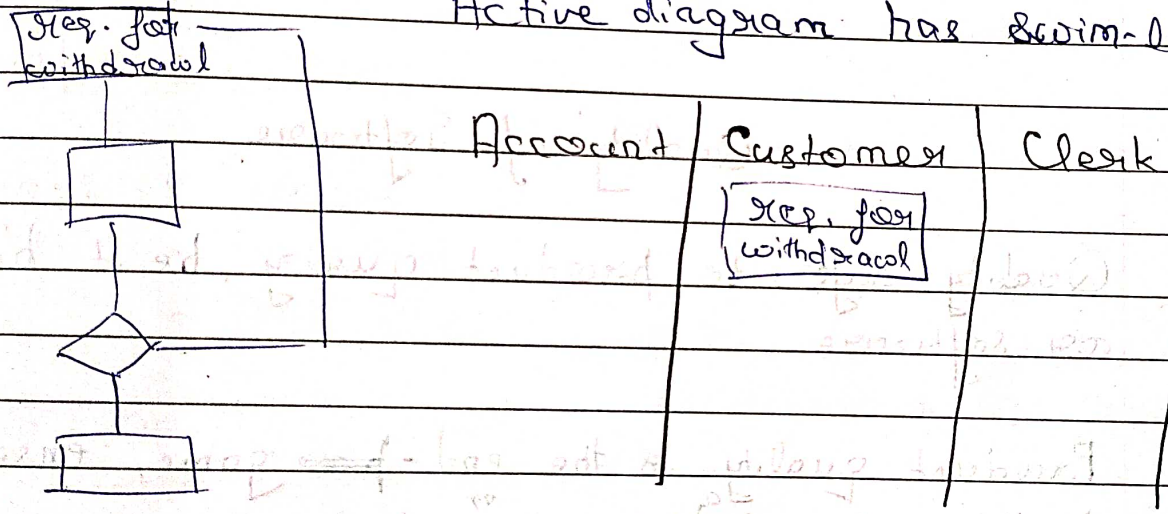
e.g. Account

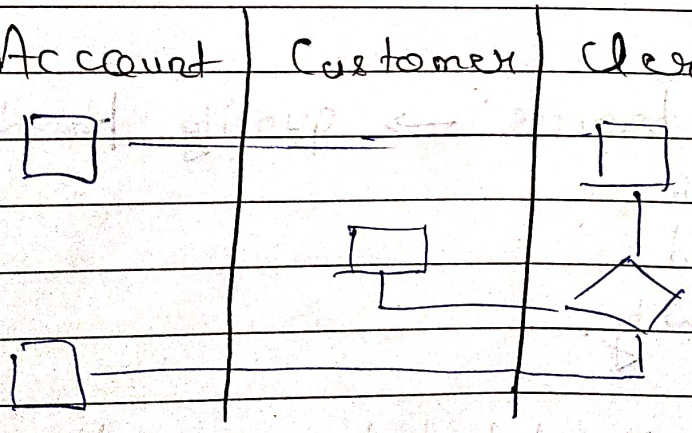
Date:
P. No:



Activity Diagram

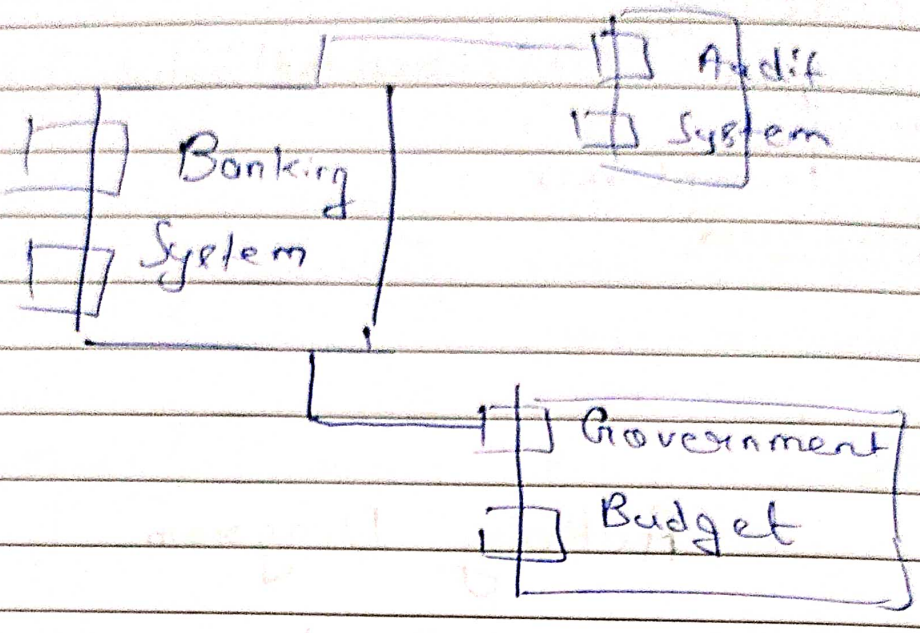
Active diagram has swim-lanes.





Concesus is for somewhere in b/w agile dev. & design.

Component Diagram :-



Quality of Software

Quality refers to product quality, be it hardware or software.

Product quality is the end ~~game~~ game, ensured by high quality process.

Quality process → Quality product
→ systematic.
→ scheduled.
→ repeatability.

Set of well-established, non-arbitrary processes → quality

Historically, we had no silver bullet.

Date:
P. No:

Waterfall Model of S/w development (Winston Royce)

Specification → Design → Implementation → Validation

- Specification
  - req. elicitation
  - analysing / consolidating requirements
- Design
  - low / high level design
  - bridges gap
- Implementation
  - coding
- Validation
  - maintenance
  - evolution
  - testing

Waterfall model is strict, ~~that~~ in a way we can't return to prev. stage (no matter how long it takes, perfect a stage comprehensively).

Pros:
- Formal
- Consistent
- Theoretical - (not so practical, but educational, sets standards for others models)

Cons:
- No feedback from client (can't go back, or adapt with technological changes)
- rigid
- time consuming

Date:
P. No:

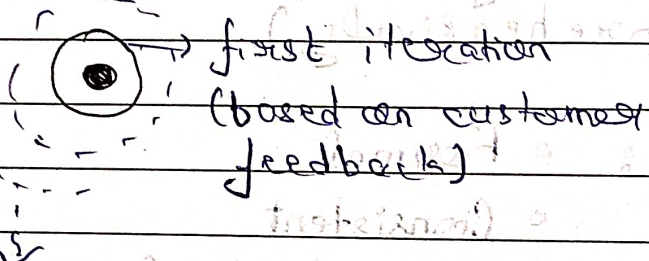
Iterative models:-

1). Rapid prototyping model
2). Spiral model

Get the customer involved (PRODUCT OWNER)
↓ representative of client in der team.

Rapid prototyping model →

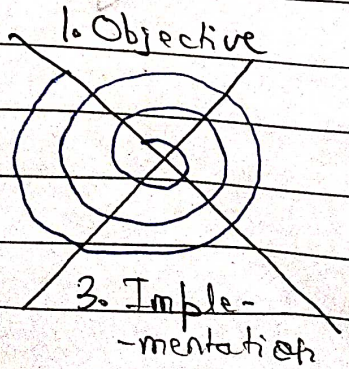
- Core product



It's important not to waste time in a single iteration

Spiral model →

By "Barry Boehm"



Radius of a spiral in a particular cycle gives time and effort.

- International Conference on Software Engineering.

pros:
- has feedback
- time efficient
- can incorporate changes

cons: negati
- process is not visible (in waterfall model, each phase was clear, iter. models may have overlap)
- less repeatable & many changes lead to degradation.
- scope creepin

Scope creep → goes beyond necessary

DOD-STD-2167

Sys. Req

Soft war req.

Analy.

Design

coding

Testing

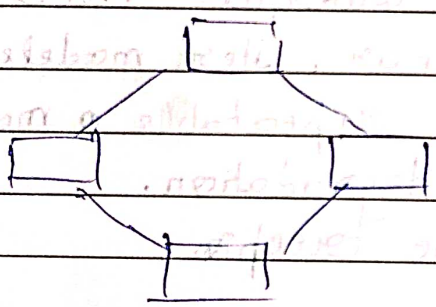
RUP

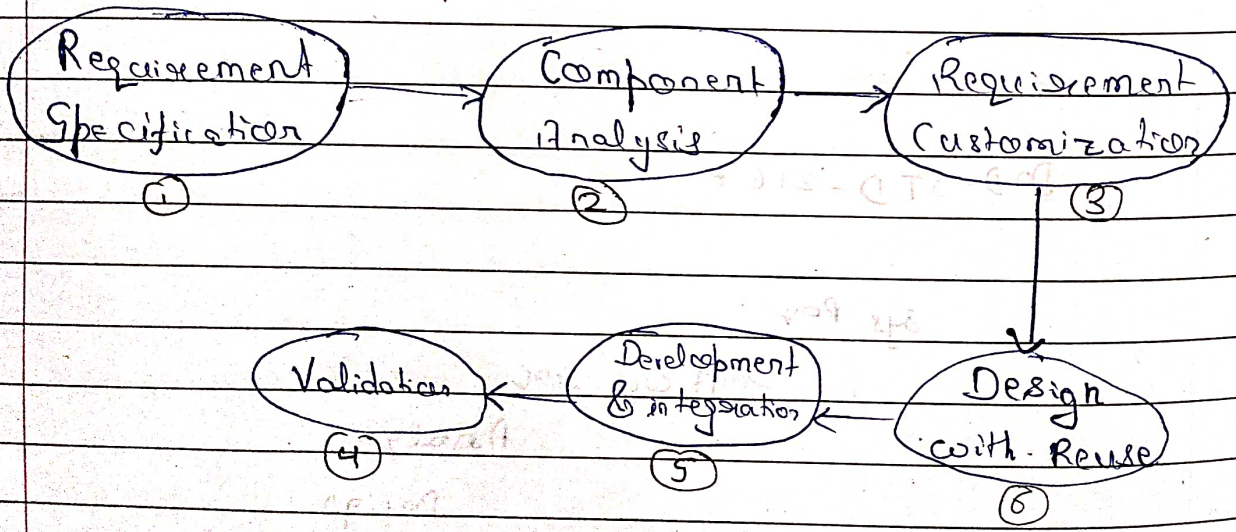
Date:
P. No:

* Planned, strategic → waterfall } although Big-Tech has their own proprietary methods
Short, quick → iterative }

Reuse-Oriented Models

Components → Connect them
(web services → unlike web apps for human consumption, API, etc.) → they are for consumption by other apps e.g. weather app → web service





2. Look for diff. components based on req.
3. Modify the req. based on the availability of components.

Date:
P. No:

Web services eased the process of reuse.

Agile Development :-

Coined in 2001, Utah. (due to discontent with formal models). ↓ Snowbird

an internet vs The Internet → the big one

~~Interconnne~~ Network of networks → Internet
Application layer on top of the backbone i.e. Internet → WWW

- Invention of Internet (Inventor)
- Created of Linux.

① Individuals & interactions : over processes and tools
↓
everyone introduces their own creative input
(less standardization)

② Working software over comprehensive documentation.
(int age vs int x).

Agile won't work in critical developments.

Date:
P. No:

③. Customer collaboration over contract negotiation
→ Difficult to charge the client. Time-frame based payment system instead of featurewise payment.

④. Responding to change over following a plan.

Start with clean slate → Improvise → Show prototype to client → Change. → Better product

(These were 4 agile manifesto)

Change is the most important factor in agile development

May ~~not~~ create conflict.

In formal SDE, many documentations were rarely used → waste of resources.

Agile → no formality

Change → Client has to understand ~~for~~ what's going on in order to suggest changes

Date:
P. No:

working → valuable, del. → cust. involvement.

Product Owner → Part of the dev. team that contribute through feedback.

Change :-

Sustainable :-

VELOCITY → Change at which ~~developer~~ team works

Scrum :

face-to-face: Daily sync. group by features but everyone has big picture.

→ Stand-up Meetings → SCRUM

→ Cont. attention to techical excellence.

Kent Beck → bad smells in the code
→ Reduce function parameters (<3).
→ Short f's are preferred.

8 | 11 (12 - 15)

Date:
P. No:

integration...

→ Change drives the development process
→ Quality of deliverables depends on velocity
→ It's important to know about velocity for sustainable development. → Keep reviewing the proc in terms of product development, burnout, proper management, etc.

Pair programming → Programmer & Driver
May impede creativity but final product is better

Trust → devs, customers, managers

→ may disregard customers for 'technical excellence' Otherwise quality of software may take a beating.

integration...
→ A decent feature may hamper larger application.

Date:
P. No:

Self organizing teams...

→

Pair Programming

Driver → Activity types codes, receives design

Navigator → Suggestions, inputs

Do's →

→ Talk
Don't insult / joke
Listen
Passing keyboard ✓
Switching roles to prevent errors
Patient
Explain yourself
Don't get frustrated
To teach is to learn
Respect
Give them time to correct mistakes
Take breaks
Be prepared, don't be late
Don't stink

Date:
P. No:

Don't ->
Hog the keyboard
Be bossy
Be intimidated / imposter syndrome
Be quiet (speak up if you do not agree)
Suffer in silence (contact professors).

Cons → Timing