Try to make repeatable, not boast

Follows a standard procedure which maybe customised by company.

Software Dev Life cycle → series of steps followed
↳ step 1 : requirements / elicitation

Software Service → Dev software for a specific client.
Software Product → generic product like MS word.
standard product accross the world.
Maybe customised by individuals.
Has generic clients.
Feasibility Analysis → check whether projects are feasible before req. phase.
Rapid prototyping: func^n ty of product but omits aspects invisible to client. (frontend)
↳ popular Language: HTML

Katerina Goseva → resources.

Brainstorming → Generation Phase.
→ Consolidation Phase.

func^n ty, non func^n features → consolidation phase
↳ time taken etc.

SRS software requirements specification document during consolidation.

next phase → interviewing.

Software Engineering CS-208

Diff b/w program & software

- program → personal use or small group
- software → others, larger community

Software = Program + Documentation + Config docs

tools = doxygen

CASE → Computer Aided Software Engineering

Types of Software = Generic (Product)
Customised (Bespoke, software services)
Hybrid

Generic → Product software. ie that can be used by many people in general like MS word.

Customised soft. / Services → Software for a specific client. like Infosys, tata consultancy

Hybrid Software → Google Education Apps, ERP packages
↳ customised mails for universities.

* from Soumerville.

Types of Software → ① Stand-alone Software
↳ MS Word, Sufficient in itself, no networking req

② Interactive Transaction based
soft. interacts with other people & gets some work done eg make my trip
Mostly web based

③ Batch Processing Soft.
↳ for processing large amts of data.

first computer → ENIAC

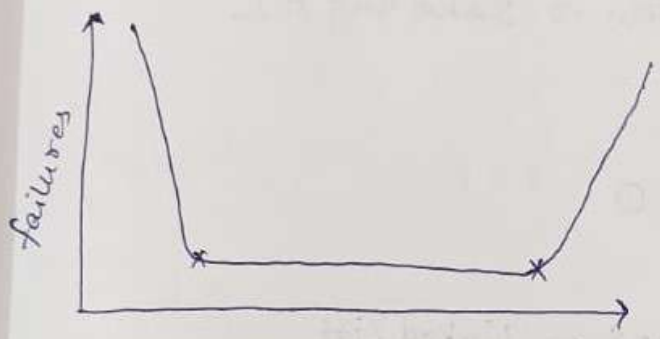
BIG DATA → data that can be handled by multiple nodes (computer)
↳ if a single computer can handle it, then no.

④ Entertainment Software
↳ Social media
↳ Game Industry (Biggest)
↳ music
↳ OTT like Netflix.

⑤ Simulation & Modelling Software FORTRAN
↳ Scientific Applications
↳ Analysis of Data.

⑥ Data Collection systems
↳ COOKIES
↳ IOT Deployments (sensing of env → IOT)

Software Engineering CS-20.

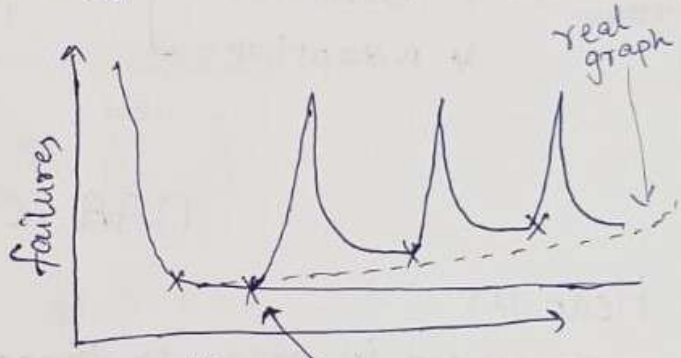


Hardware mapping

Bath tube curve

* don't make changes

* wear and tear occurs



Software mapping

* has updates so failures occurs.

History of S/W Engineering:

first software developed:

1945 - 1965 → origin of S.E.

1965 - 1985 → S/W crisis age (hardware started developing alot. Moore's Law)

1968 → word Soft. Eng. was defined in NATO Conference on S/W Eng.

↳ International Conference on S/W Eng. ICSE

1985-1989 : NO silver bullet (No sol^{n} for everything)
↓
one

1990-1999 : Internet arrives

MM/DD/YY → MM/DD/YYYY problems when 2000 started

2000 → Lightweight Technologies/Methodologies

2015 → AI

S/W Eng
↳ Process of S/W Development → specification
→ Design & Development
→ Validation
→ Evolution (changes)

Phases:

* alpha test → test by dev
* beta test → test by user

anticipate errors ← * preventive
main reason ← * corrective
* Adaptive

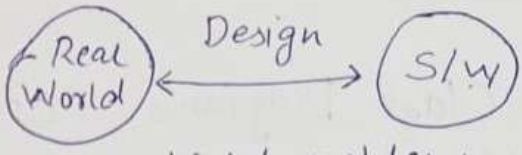
17th Jan → Samsung AI

What is design?

→ Optimisation of product.

→ Easier for prog to understand

→ Bridge b/w Real world & S/W



Agile dev → don't care for design, may have slight problems.
→ straight coding

Real time Software / Systems → where split second delays causes massive problems. Like surgery, & Chandrayaan. Design is very necessary for this.

middleways.

Agile dev → SCRUM method ← Design
↳ don't care for SRS, design implementation, testing.

Design → trying to model what the software will look like based on requirements, in the most optimal way.

model the requirements & bridges the Real World & S/W.

Design → Data flow diag etc.

1997 → UML unified modelling language.

diff schools of thought → Agile and Design.

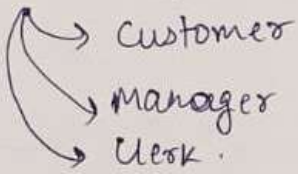
ArgoUML → tool for Lab

UML has diagrams (13 in total) (4-5 should be used)

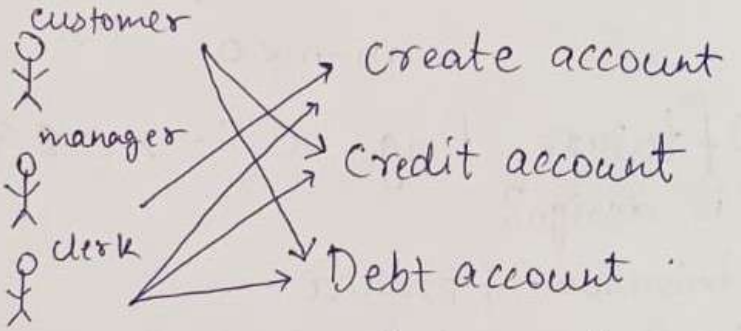
1) use-case diagram

→ functional requirements. (like create account etc) of system.

→ Actors: systems, people who are going to participate in the system can be other systems which interact as well.



We don't consider details
Just superficial



Actors. functionalities.

2) Class Diagrams. (CD)

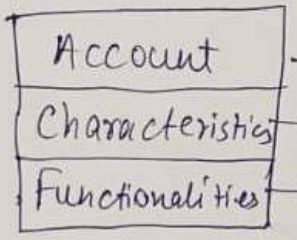
→ Identify diff entities that exist.

→ two parts: Logical CD & Implementation CD

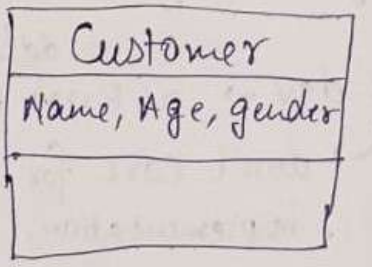
↳ Identify entities like Actors, Account, Customer

Class diag:

3 parts:



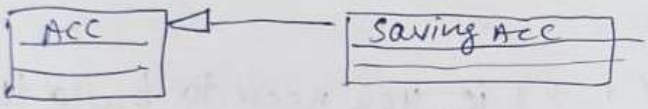
→ name of class
→ Acc No, withdrawl limit.
→ Credit/debit Acc.



After creating the classes, connect them which have a relationship

like 1 account shared by multiple customers.
1 customers having multiple accounts.
inheritance: saving acc inherits from account.

→ child to parent. Signifies inheritance.



Logical: functionalities of diff classes. (entities)
CD

Implementation: Logical CD + more details. like name char[100]
↳ if its OOP lang used, then it is reccomended to use this.

UML popularity decreased in 2000's due to agile dev.

Implementation CD → more details on functions with respect to programming point of view.

3) Sequence Diagram (usually included)

→ tells us the sequence to be followed to implement func^{n}s

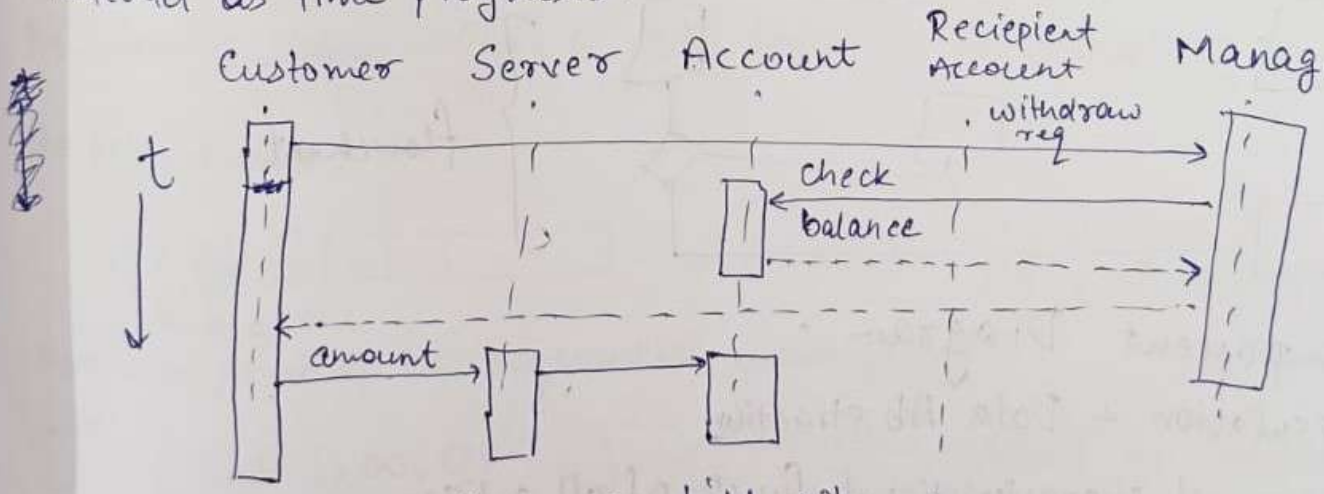
→ All func^{n}s are listed in the use-case diag.

→ ∴ All func^{n}s should have seperate sequence diag.

→ Know which all entities would be involved in each func^n

Like func: withdrawl classes: Customer, Server, Account, Manager, Reciepient.

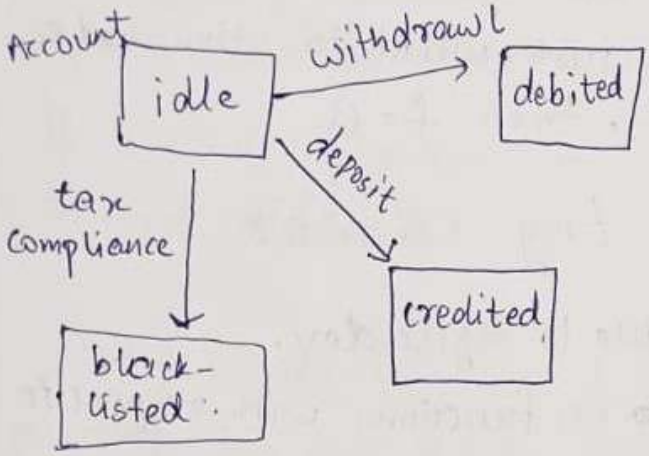
Now know the sequence / timeline in which these entities interact as time progresses.

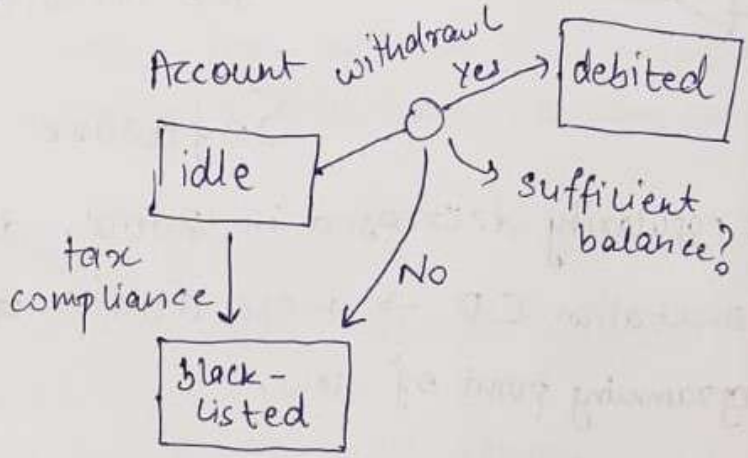


the rectangles length is till when the entity is active in the func^n.

4) State-Chart Diagram

→ Draw the entities & collect all func^n for that entity.
→ the state of the entity will be in a box.
→ can also have cond^{n}s using a ○ for branching.

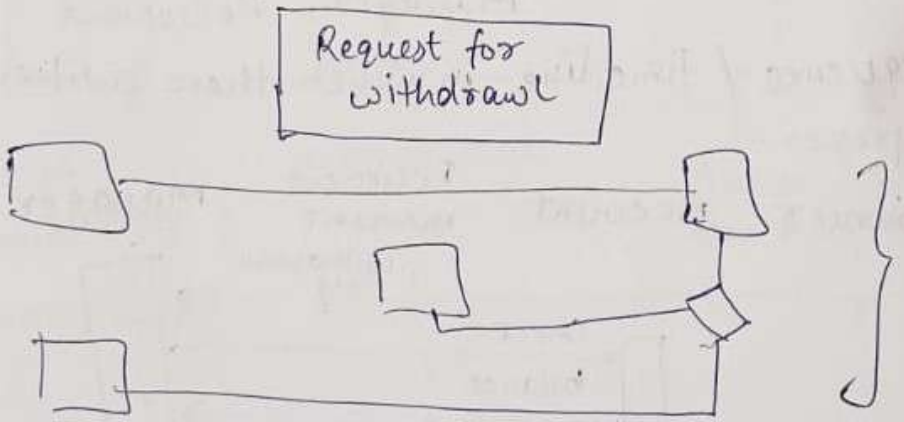




5) Activity Diagram

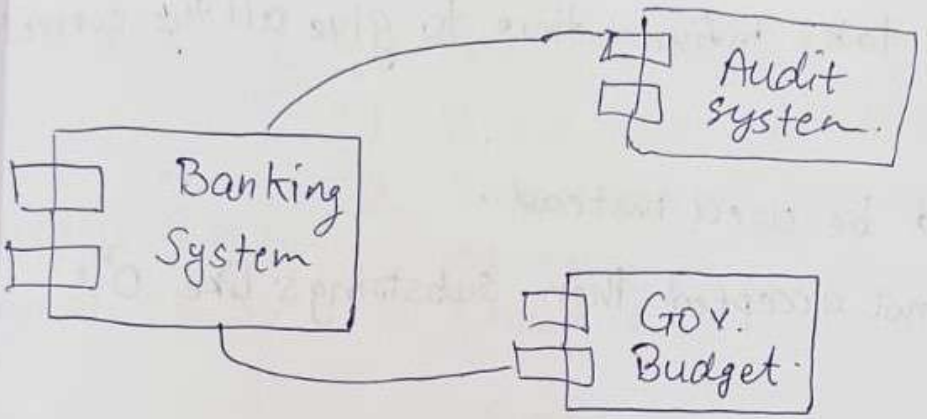
→ kind of flowchart for func^n
→ Can have a single flowchart for multiple func^n. (Diff with seq diag).
→ Has swim-lanes for each entity / class.
→ If multiple entities are involved, then simultaneously draw the block in them.

Account Customer Clerk



6) Component Diagram.

→ Encapsulation + Data Abstraction.
→ Combining all characteristics & func^n of all entities.
→ Connecting all entities with external components without worrying about how they will interact.


Audit
system.
Banking
System
GOV.
Budget.

Software Eng → Systematic way of writing software

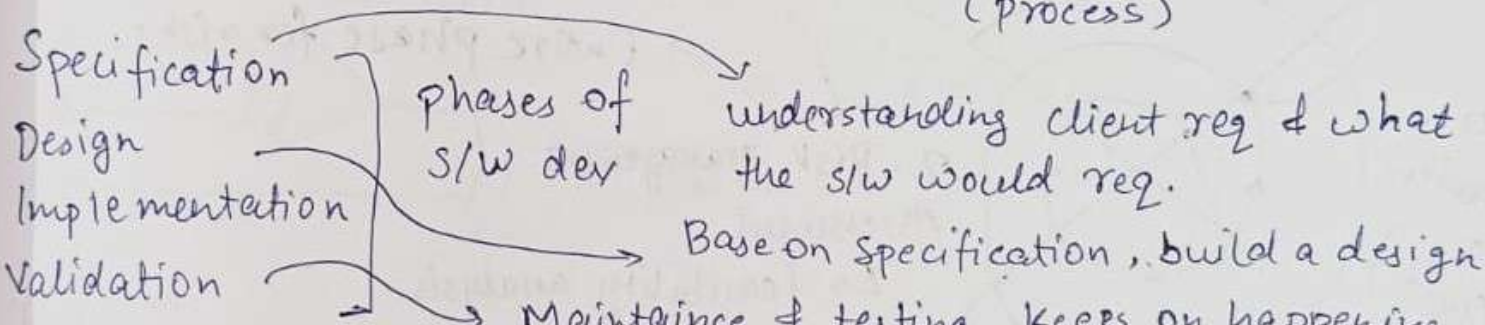
High product Quality → goal.

↳ if we process of making software high quality, then product will be high quality.

Repeatibility → goal.

Winstone Royce

Waterfall Model of S/W development → Widely accepted. (process)



Waterfall Model → Once specification is done, then don't change it. very strict. Can't come come. Then based on specification design and implement the S/W. Freeze each stage as you move.

- → Formal Process
- → Consistent (everyone is on same page)
- → Theoretical Model (but not a practical model) (Nobody now uses it. (But it is a good standard / baseline model)

adv

disadv

- → No feedback
- → Takes alot of time. (So if anything changes with time with client, he still cannot ask the dev to change it
- → Rigid

Iterative Models

→ Don't do it in one go. Repeat all phases.

→ Rapid Prototyping Model & Spiral Model.

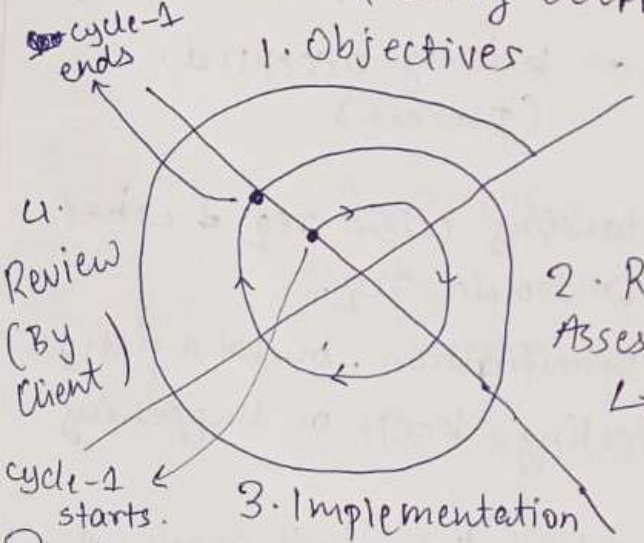
Product Owner: A representative of client involved in dev team so that customer feedback would be included.

Core Product: A quick meet with client & then show a initial model usually a frontend prototype.

first iteration: feedbacks of customer on core product included & then keep on iterating this process.

↳ rapid prototyping

Spiral Model (Barry Boehan)



→ credited alot to the model.

→ Entire phase for risks

↳ feasibility analysis

Radius of spiral → Time spent of each phase. Effort spent.

Adv

→ feedback of the client included

→ Less Time.

→ Easy to incorporate changes.

Disadv

→ Process is not visible.
(Not clear what phase is occuring. Overlaps usually occur.)

→ Degradation of Quality
(as changes occur alot, adapting those changes reduces quality)

→ Scope Creep.
(with time, the req. change alot & we have new func^n to implement)

Refactoring → fixing the changes & improving quality

Biggest weakness of iterative model

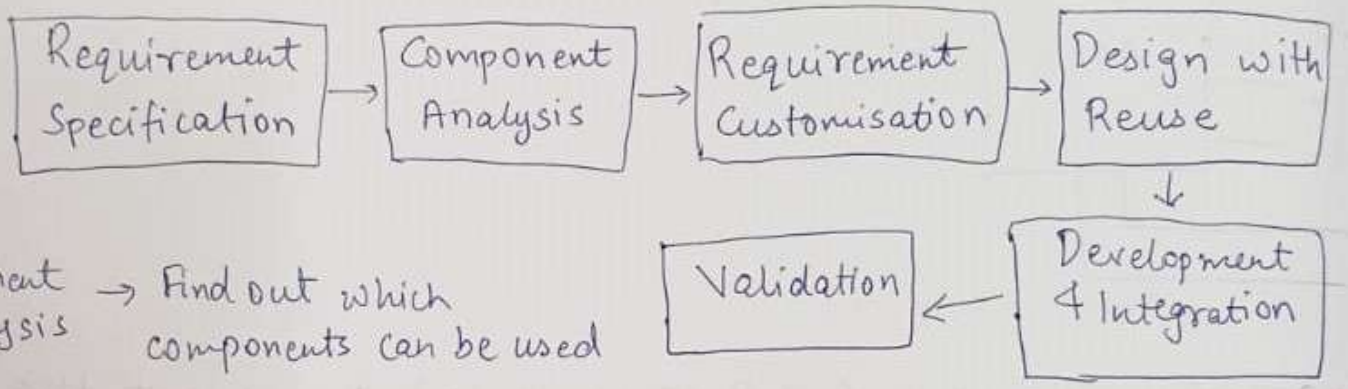
add^n room created

↳ house example

If project is not critical & needs dev quickly → Incremental model
Waterfall Model → Critical project which can take years.

Re-use Oriented Model

→ Package components made available to public (open source like)
→ Connect these components to build up your app.



Component Analysis → Find out which components can be used

Requirement Customisation → Customise the components according to your needs.

Diff b/w web application & web service?
- web application: meant for human consumption like Times of India
- web service: Built to use for some other application (API) Technology Egnostic → independent of device

Agile : termed around ~~1990~~ 2001
↳ Agile Manifesto → Utah

1991 → World Wide Web introduced

Diff b/w WWW & Internet?
- WWW: Application layers built on top of the connection by internet. (hosted)
- Internet: Backbone. Physical connect^n Network of Networks.

Network → Connection b/w computers.

The Internet → Network of Networks. (all computers connection)
internet → Network (connection b/w few computers)

Sir Inventor of Internet →

Developer of Linux →

Individuals & Interactions over processes & tools

→ Main focus uptill now Process Quality. But Agile neglected it.

→ Getting qualified individuals who make quality s/w.

→ Informal processes.

Working S/W over comprehensive documentation

→ Doc^{m} is not imp.

→ Develop S/W which doesn't require documentation.

int x int age

→ Write comments, meaningful names etc so that documentation is in the code itself. Developers gets content in code itself.

Customer Collaboration over Contract Negotiation.

→ Req may change over time, so freezing them isn't beneficial as done it processes.

→ But payments become issue. Hence dev's payed with time spent. Earlier dev's were payed with the req's that were fullfilled.

Responding to change over following a plan.

→ Don't have any plan. Make a ~~prdtg~~ prototype and show to client. Then incorporate changes. Repeat it. By repeating it, the S/W comes more close to the client reqs. Welcome change (processes make changes in acceptable)

Most Imp factor in Agile → Change.

Agile S/W Dev:

→ Waste of time in process. Documentation wasn't really referred.
→ Waste of resources in process.
→ Agile dev was fast with decent quality S/W.
→ Used for S/W ~~for~~ which is less critical.
→ earlier paid for features. Now in agile paid for time.
→ Main feature : Change

Client is understanding the S/W if he asked for change. With every change, the S/W comes closer to the req.

But if no. of clients are more, then there will be chaos.

→ Working S/W should be delivered frequently so that the client can "play" with it and suggest changes.
→ Balance risk reduction with feature accomplishment.

Two week iterations. 3 iterations ≡ 1 release.

→ Time-boned development

Knowledge about velocity. fix a time period & develop in that.

Improvement of overall project plan.

Velocity: rate at which work has been done / team can work. So if we know about our velocity, then time-boxed dev is easy.

→ Valuable S/W should be delivered. Deliver most valuable ie higher priority features first & deliver it. Priority could be understood when meetings with client are regular.

→ Divide the team along features

→ Integration + test + doc = completion

→ focusing on completing features

→ Customer Involvement

typical S/W standard ↑

- Customers are usually invisible. Single defined customer are rare.
- Generic S/W like MS word have customers all over the world.
- Hire a few people of diff classes to act as customers for dev help like Beta testing ↳ customer surrogate
- Customers are heterogeneous not homogeneous.
- Product owner: somebody part of dev team & contributing to the dev by giving feedback, not coding. Representative of customer Usually there are diff product owner for diff features.

→ Change

- It signifies that the client is understanding the system
- Disadv ⇒ time delay, elimination of features.

→ Sustainable

- Have good idea of capabalities of team.
- Determine each teams velocity (not comparable)
- Realistic planning
- Don't burden the team
- Review & Code Inspection
  - External: higher level people not involved in dev, like boss, review the code.
  - Internal
    - ↳ pair programming: program with 2 people: Navigator & Pilot. Navigator: suggests the pilot with features & refinement
    - ↳ Review team: Some people in the dev team review other team code.

→ Trust
. Between developers internally & b/w developers & client.
. Communication, transparency, honesty, touch.
. Developers, Managers, Customers.
. Don't do things without telling each other
→ Face to Face (F2F)
F2F comm^n is preferred.
. Re-did furniture for everyone to dev F2F. Team worked together
↳ was costly (disadv)
. Keep furniture which was interactive. Like ping-pong games etc.
Daily Synchronization is a must.
. Have common understanding
. In scrum have Stand-up meetings: while sitting people relax & talk about useless things. ∴ Don't let people sit. People ∴ say things in short precise manner.
. get feedback, deal with problems & roles.
. Each team knew about features of other team too.

→ Attention to Technical Excecellence
. The first time is never perfect design. Iterative process makes it.
. Refactoring: (Kent Beck) Identify bad smells of code. Break this code into more segments to make it better.
. Automated & Synchronized with dev
. Test & validate the code.
. Have time for refactoring. (safety net)
→ Integration
. Integrate changes as often as possible.
. Makes progress visible & measurable.
. Conflicts easier to solve.
. Each integration results in a running system & provides feedback from the client.

Software Eng. CS-208

General know. Ques = ~~5 marks~~ 5 marks.] 40 marks
7 questions = 5marks

1 UML question. Draw Diag.

2001 → Coined termed Agile

What is velocity? → rate of work done
↳ Each team's velocity differs.

What is Product Owner? → related questions

What is Change?
↳ Sometimes to add clients features, you have to discard other features.

Review → Product
Team burnout

External → some outside people
Internal → pair prog.

F2F communication drawback: 93% comm are non verbal. So F2F won't help much.

Refactoring → bad smells in code. Code could improve more either structurally or logically.

→ Integration
. Maybe a feature works well individually but after integration it may cause problems.

→ Self Organising Team
- Each team defines its own process. Don't over-centralise. Let the team do what it wants. Don't over specify & over rule.

Agile Alliance
- Co-operated movement. Not owned by some company.
- It someone suggested something good, they would propose to it & they would spread it around.
- Cerificates & courses.
- Online blogs & publications.
- Hosts annual user conference.
- Guides people with respect to agile dev.

Pair Prog: Driver → programmer
Navigator → suggestions
] switch roles periodically ↳ helps prevent error & burnout.

Talk each min & even Listen.