Software Engineering

- It is a systematic, disciplined, cost effective (time) technique for software development.

- Engineering approach to develope a software ~~test, code~~ step by step approach to achieve a goal. (development to maintenance)

Evolution

1945 - 65 → origin
1965 - 85 → crisis
1990 - 2000 → internet
20000 - 2010 → Light weight
2010 - 2020 → AI, ML, DL

Systematic means step by step
disciplined means following predefined rules
cost effective means within limited cost.
time

(¥) In origin time software is just made for working for that time

(¥) In crisis time almost 2% software was running, 56-60% software was not developed at all (was not delivered).

Purpose of SW Eng
- manage large software
- Scalability . better Management
- cost manage

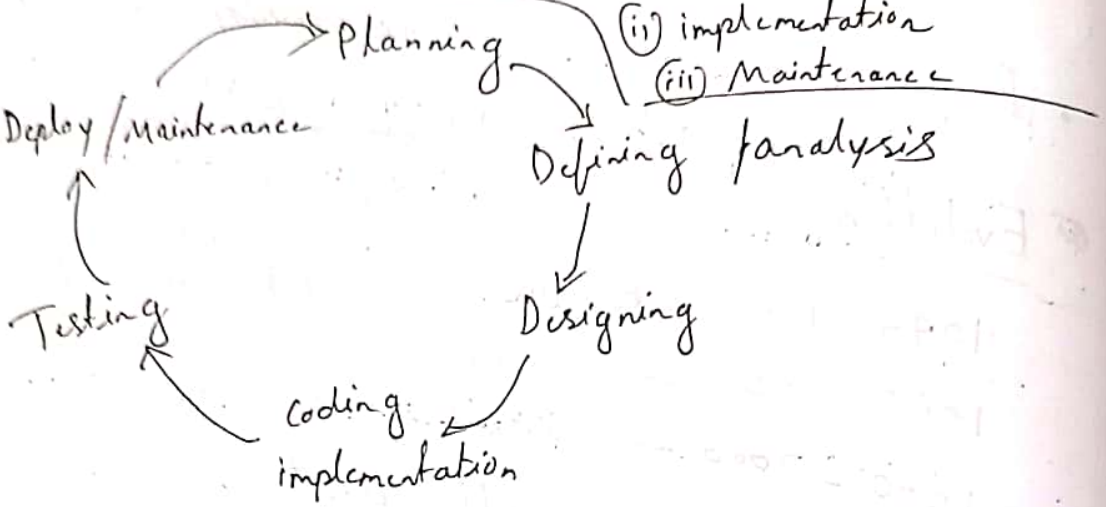
(¥) Watts S Humphery in software engineering

(¥) We should use abstraction (only important things first) & decomposition (divide in different modules)

Software Development Life cycle (SDLC)

Different phases needed to developed a software

3 stage of SDLC
(i) Conception stage
(ii) implementation
(iii) Maintenance



customer → Service provider

SDLC is a process followed for a software project within a software organization.

(i) Planning / Requirement analysis :- Plan the project with senior members with the help of customer. Various techical approaches to that can be used with minimum risks.

(ii) Defining Requirement :- Document the product requirement & get them approved by customer. This is don by SRS (software Requirement specification) document which consist of all the product requirements to be designed and developed during the project life cycle.

(iii) Designing the product architecture :

Based on SRS one Design document specification (DDS) is proposed.

DDS is reviewed by all important stakeholder and based on various parameter like budget, time constraints, design modularity (breaking down separate independent parts), product robustness (strong) best design is selected.

(iv) Coding :→ Product is built using different languages as per DDS.

(v) Testing → Product defects are reported, fixed & retested

(vi) Deploy & maintenance → Released in the market for public.

Sometimes product is deployed for smaller number of people (Beta) then with their feedback it is released for public.

Later Maintenance is done if anything defect is reported.

Popular SDLC models are

- Waterfall model
- V model
- iterative model / incremental
- big bang model
- Spiral model

other related models are

- Agile model
- RAD model
- Prototyping model
- Rapid application development.

classic Waterfall model :=> (linear sequential model)

- Came in 1970s

Feasibility study (whether it can be made with budget or can possible in real life.)

Requirement analysis & specification

Design

coding & unit testing

System testing & integration

Maintenance

Each & every stage should complete before moving next stage

Adv :-

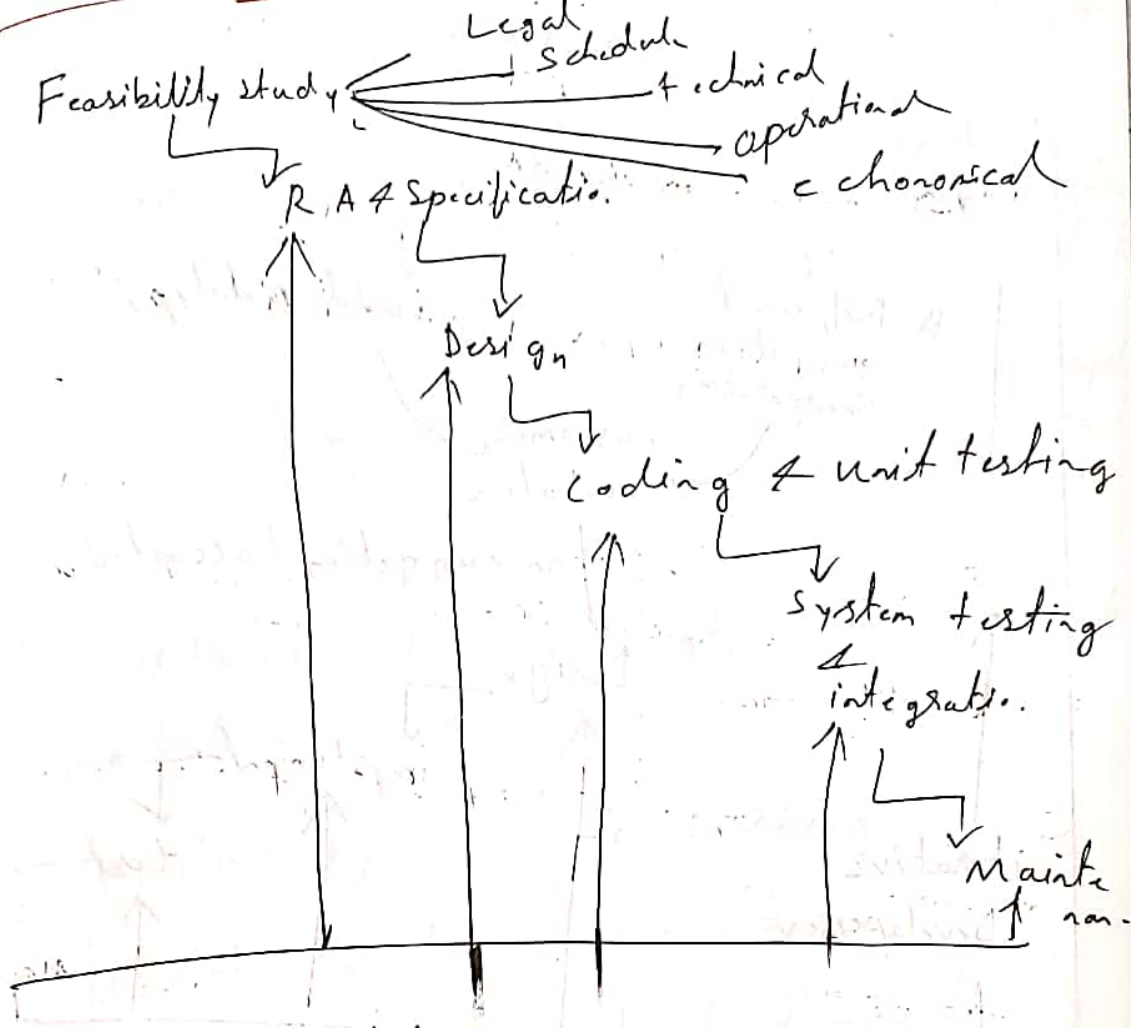
- Basic model
- Simple & easy model
- Small projects can be made (as customer has fixed requirement)

Dis :-

- No feedback (customer can not change anything in the phases. it can only done in maintenance)
- No experiment
- No parallelism (more than one team cannot work at same time)
- High risk
- Not flexible

- 60% effort in maintenance.

iterative waterfall model



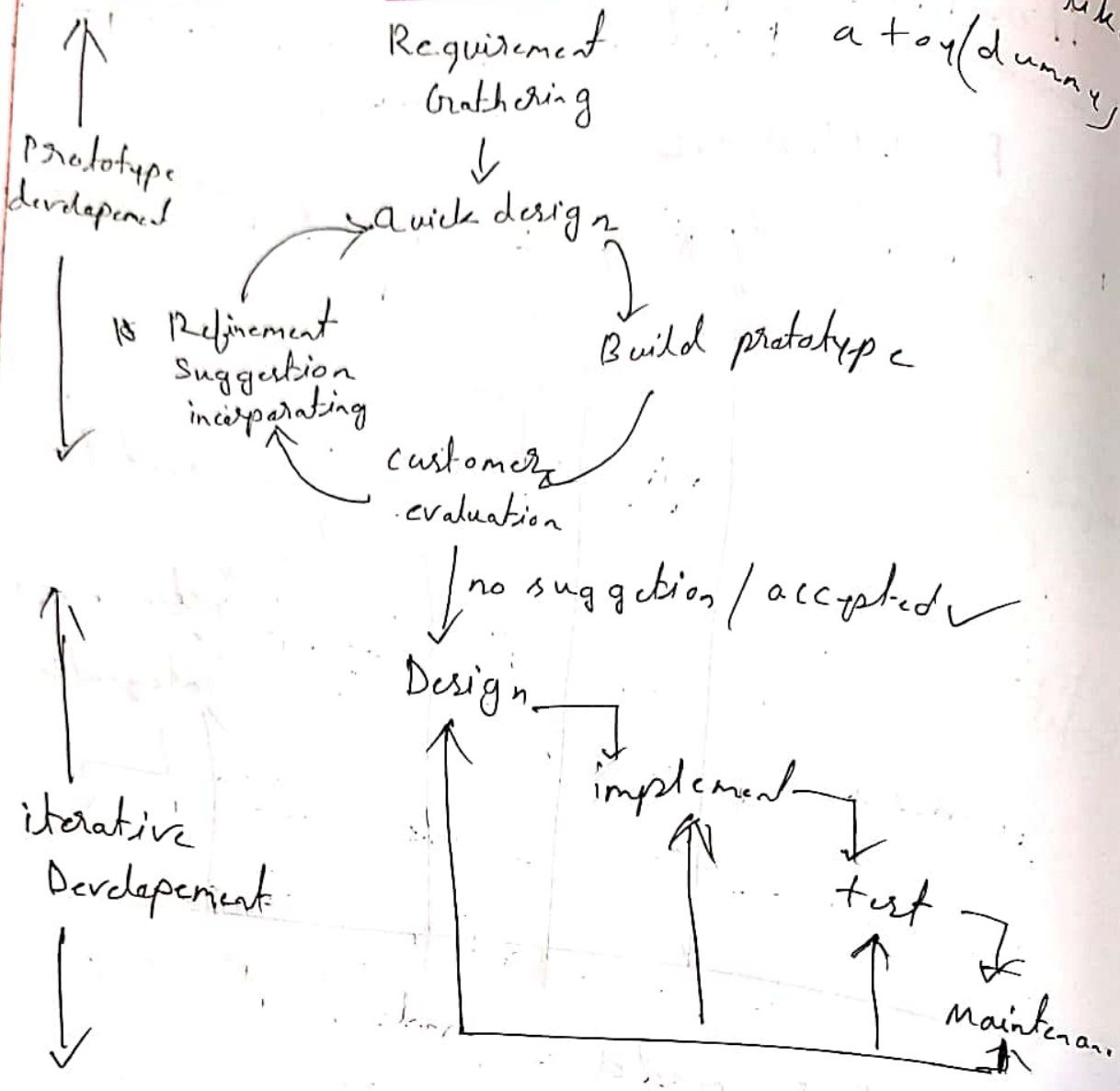
Adv
- Base model
- Simple & easy
- Small project can be made
- Feedback can be given by different phase

Dis
- No experiment
- No phase overlapping (parrassel)
- No intermediate delives,
- Rigid (no changes in requirement only mistake can be changed with feedback.

- less customer interaction.

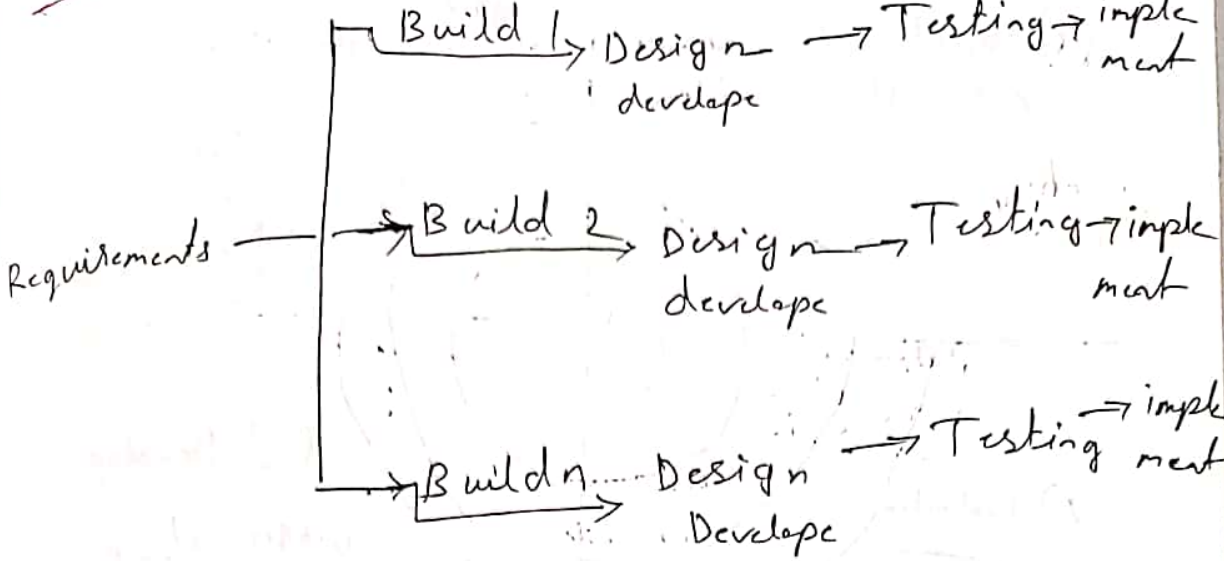
Prototyping Model:

Prototype is like a toy (dummy)



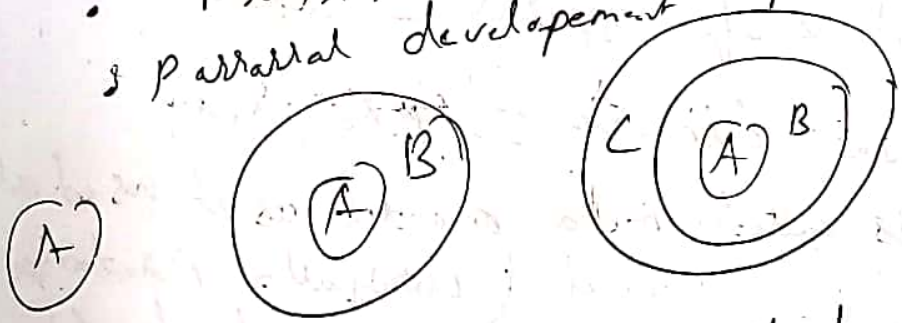
- customer Not clear with idea the this model is ~~waste~~ used
- Throwaway model (if not liked the prototype then it is thrown)
- good for technical & requirement risk
- client interaction is there in time of prototype development
- Error can be detected earlier
- increase cost for development.
- Total dependency of prototype

incremental Model



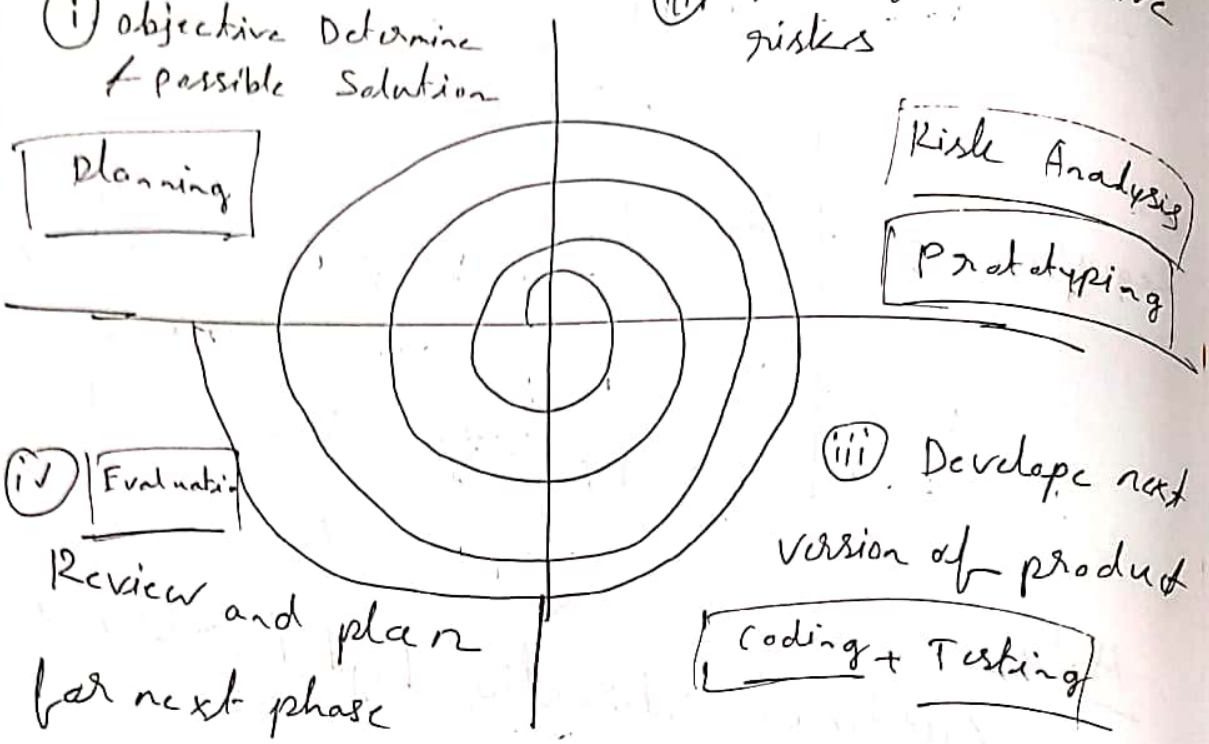
Adv:

- customer interaction high (customer check each module personally
- Module by module working
- used in large project
- early release product demand
- Flexible to changes possible
- parallal development



Dis - more resource needed
- not for small project
- more management attention required

Spiral Model ⇒ (Reduce risk) Less experience



(if any modification needed by project manager or customer, it is again do the same thing.

(*) It handle the Risk in every Step

- Radius of spiral = cost
- Angular dimension = progress
- It is called meta model as it used all model (waterfall + Prototype + feedback)

Adv :-

- Risk handling at early stage
- Large project & complex project useful
- Flexible
- Customer satisfaction

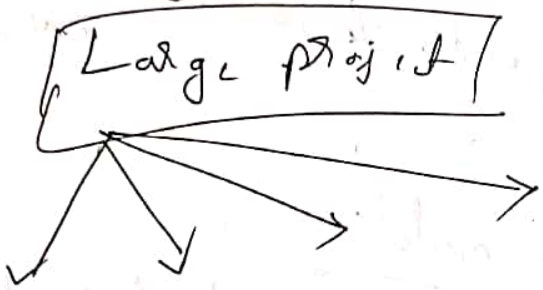
Dis :- (i) complex • Expensive • Time

• Too much Risk analysis,
• not suitable for small project

~~Comparison~~

Agile model :- It is latest model used by most company now a days

Agile means move quickly



Small chunk (iteration) / partially a lot of chunk is being done)

↓ Release

↓ feedback

↓ enhance

↓ Re release

Adv

• Frequent delivery
• Face to face communication with client
• changes possible
• Time less

Dis :-

• Less documentation
• Maintenance problem.

Comparison :

(i) Classical waterfall : • Basic model
• Rigid
• Not flexible
• Not for Real project
• No feedback in between phases

(ii) iterative waterfall :
• Basic model
• Only feedback is there till Requirement analysis

(iii) Prototype model :
• User Requirement not clear then it is best
• costly
• No early lock on requirement
• High user involvement
• Reusability

(iv) Incremental model
• Module by module delivery
• Easy to test & debug
• Requirement lock

(v) Evolutionary model

- Large project
- Requirement not lock

(vi) RAD model

- Time & cost constraint
- user at all level
- Reusability

(vii) Spiral :=

- Risk evaluation
- Not for small project
- No early lock on requirements
- Less Experience can work

(viii) Agile :=

- Flexible
- Advanced
- parrarel
- possible divided

to pieces

Project Estimation Technique :⇒

Estimation of various projects parameters is an important project planning activity. The different parameters of a project that need to be estimated include –

- Project size
- Effort required to complete the project
- Project Duration
- cost

Accurate estimation of these parameter is important for resource planning and scheduling. Estimation Technique can be classified as

- Emphirical Estimation Techniques
- Heuristic Estimation Techniques
- Analytical Estimation Techniques

* Its part of planning (after SRS it is done)

Emphorical Estimation :⇒ guessed estimation with the help of past experience. educated guess.

Two popular Technique is

- Expert judgement: Expert give educated guess of different module (component) and combine them.

⊙ Delphi cost : carried out by a team and coordinater. every team give estimate to coordinater anonmously.

Heuristic Estimation :

• different project parameter can modeled using suitable mathematical expression

type is

⊙ single variable - The basic cocomo model is an example of single variable cost estimation model. CoCoMo Model is an example of it

CoCoMo :⇒ constructive cost Model

• Was first proposed by Dr. Barry Boehm in 1981

• is a heuristic estimation technique

• This approach implies that size is primary Factor for cost, other factors have less or effect

• constructive means complexity

• cocomo prescribe a three stage process for project estimation

(i) initial estimate

(ii) + (iii) - Refined to arrive more accurate.

• Projects used in this model have following attributes -

- Line of code (LOC) 2000 to 1,00,000
- language ranging from assembly to PL/I.
- These projects were based on Waterfall model of software development
- Boehm state that software development project can be classified into 3 category

(i) organic: • Typically 2-50 KLOC
Line of code

- Small size project (Pay roll, inventory project etc)
- Little innovation
- Deadline not tight
- can be done in office.

(ii) Semidetached: ⇒ 50-300 KLOC

- Medium size project (Database, compiler, editor)
- innovation medium
- Deadline Medium
- can be done in office

(iii) Embedded: ⇒ over 300 KLOC

- Large size project (ATMS, air traffic control etc)
- complex interface
- innovation significant

- Deadline Tight
- can be go to customer place on sight
- complex hardware.

Person Month (PM) :⇒

- The effort estimation is expressed in units of person months (PM)
- An effort of 100 PM does not imply that 100 persons should work for 1 month or 1 person 100 month.
- It is calculated from the graph
- 3 Developer works for 4 month then 12 PM

) Software cost estimation should be done through three stages

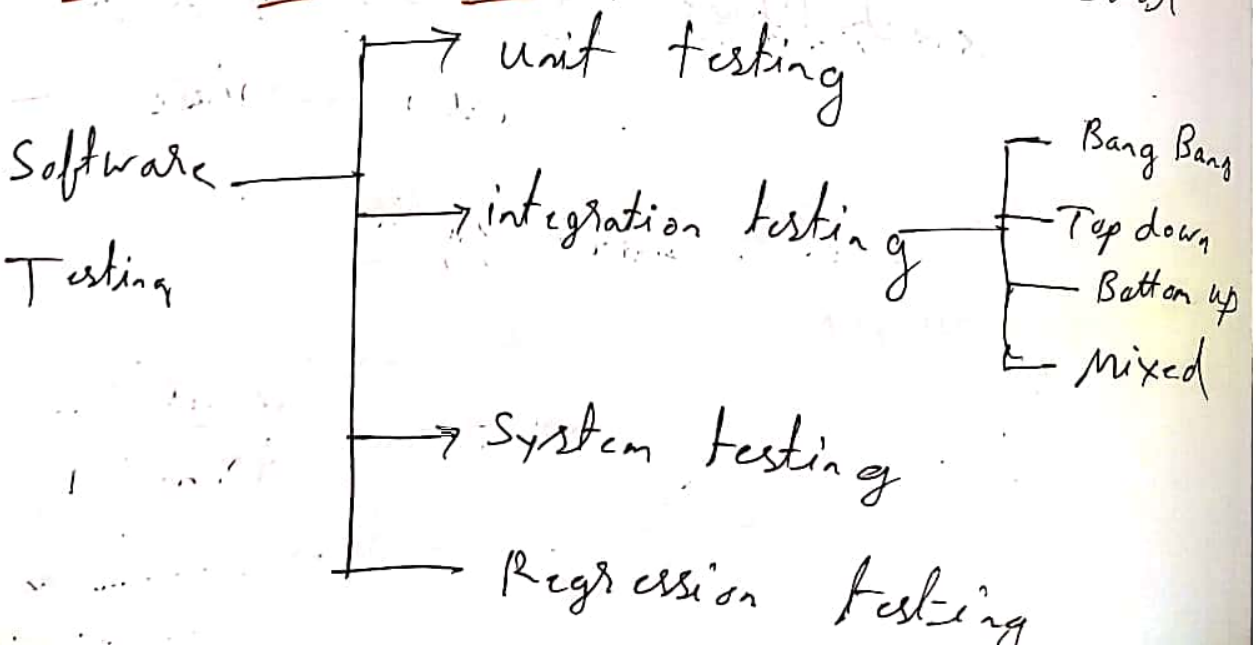
(i) Basic COCOMO – (approximate estimate cost & people)

(ii) Intermediate COCOMO (added 15 additional predictors)

(iii) Complete COCOMO (amount of effort in each phase) – there is 6 phase.

Verification	Validation
Are u building it right?	Have you build the right thing
Done by developer	Done by Tester
Can whether artifact confirms its previous artifact	check final product against specification
Concern with error in phases	make final product error free
Method invalve inspection, unit testing, integration testing	invalve System testing
Static & dynamic activity	Only dynamic

Type of Testing :- ⇒ Finding the bug/ error

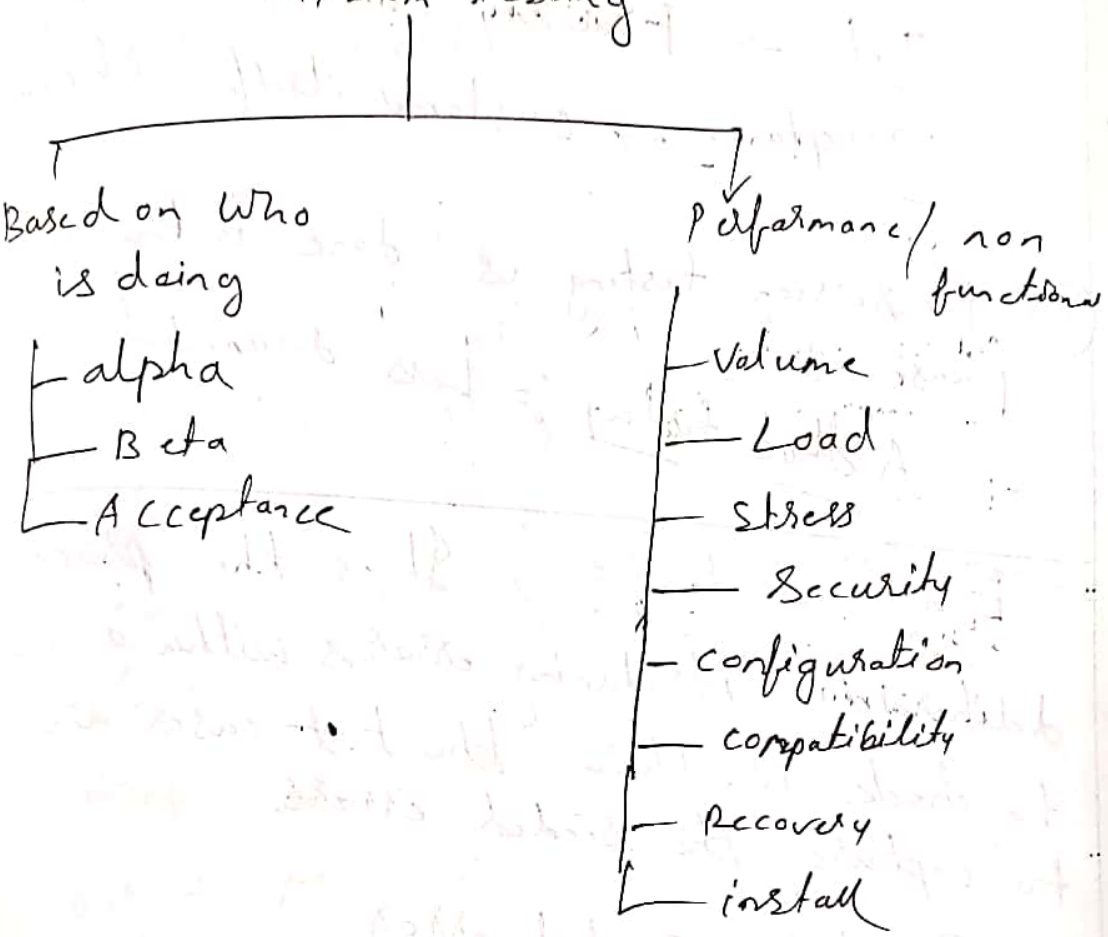


Types are categorized based on level

level 1 – unit → done in coding stage

level 2 – integration
level 3 – system
} – done in testing stage

level 4 – Regression – done in Maintenance stage



(*) Unit testing means task Smallest unit of code in coding/development phase

(*) Test case design approach is of two type

(i) black box testing – (Functional) – Without knowing internal structure

(ii) White box test – (Structural) → knowing the internal structure

(*) integration testing can be done after integrating the modules

(*) System testing is done in different Scenario. Whole product is tested

alpha → developer is testing

Beta → Friendly customer is testing

acceptance → customer itself check

(*) Regression testing is done in the maintenance phase

Adhoc testing := Less documentation testing

Error Seeding :=> It is the process of deliberately introducing errors within a program to check wheather the test cases are able to capture the seeded errors.

S = total seeded error

s = total seeded error detected

N = total error

n = total non seeded error

$$\frac{n}{N} = \frac{s}{S}$$

$$\Rightarrow N = \frac{nS}{s}$$

total errors

total no of undetected errors

$$= \frac{n \times (S-s)}{s}$$

4. Project risk factor is considered in
(a) spiral model (b) waterfall model
(c) prototyping model (d) iterative model

=> (a) ✓

5. Path coverage technique is used in white box testing

Equivalence
Cause effect
State based
} is used in black box testing

group of tests
+ve/-ve inputs

user id + password = login
user id + password = not login
✓ ✓ ✗

Cyclomatic complexity => Region bounded-e/

number of

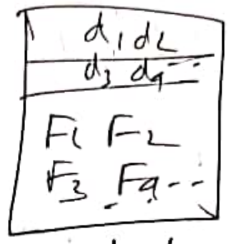
3+1
= 4 ✓

Design phase:

SRS Document $\xrightarrow[\text{activities}]{\text{Design}}$ Design Documents

A module is consists of

- Several functions
- associated data structures



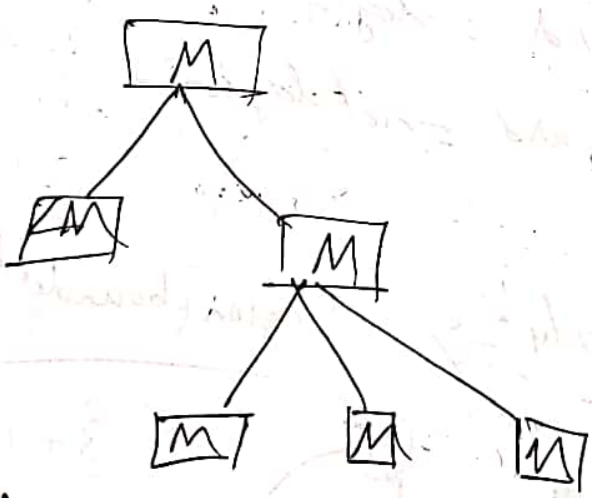
Module

independence

Modularity is a fundamental attributes of any good design

Decomposition of a problem clearly into module ~~has minimum~~

- Modules are almost independent of each other
- divide & conquere approach



Dependency low

(*) Module should display
- high cohesion
- low coupling

} for better software

- Cohesion is a measure of functional strength of module
  It is within the module (intramodule)
- coupling is between modules
  (i) interdependent of modules
  (ii) a measure of the degree of interdependence or interaction between two modules.

Classification of cohesion: represent detail design

High
- functional
- Sequential
- Communicational
- Procedural (Same algo)
- temporal - all task execute in same time span
- logical ← (all elements perform similar operations)
- Coincidental - (instruction no relation with one another)

Low

Classes of coupling: -> represent High level design

Low
- data coupling
- stamp coupling
- control coupling
- common coupling
- content coupling

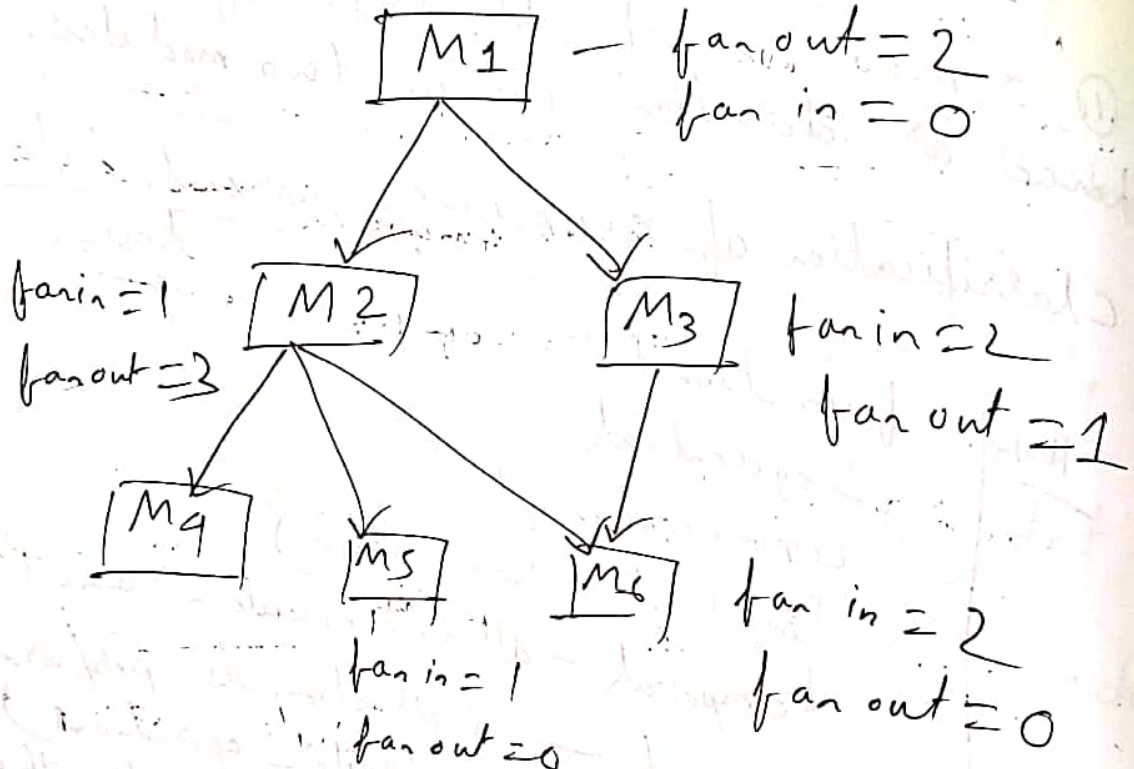
High (bad)

data shared
Sharing same data structure

can change data of other module

Characteristic of Module Structure

- **Fan in** → indicates how many module directly invoke a given module
  - high fan in represent code reuse and disin general encouraged
- **Fan out** → a measure of the number of directly controlled by given module



(*) fan in should be high & fan out low

SRS :

- SRS stands for Software Requirement specification
- document prepared by business analyst or system analyst
- Describe the features & behaviour of Software
- Detail description of all functional & non functional Requirement
- SRS is an actually agreement between client & developer

parts of SRS :- (i) Functional Requirement
(ii) Non functional Requirement
(iii) Goal of implementation

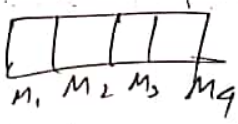
characteristics :-
(i) complete
(ii) consistency (easily understand from begin to end)
(iii) Feasible (implementation possible)
(iv) Modifiable
(v) Testable
(vi) correct

Software Designing :=> - meaning representation of Software

- SRS —Design→ blueprint of software
  DDSC Design document specification
- Reduce risk & effort
- Entire Software breaking into smaller parts (top level design)

Design is divided into two parts

(i) Top or High level design (breaking in modules)



(ii) Detail / internal design (design of module each)

(X) Cohesion	Coupling
• describe relationship between two or more elements of particular module	• describe relation between two or more module
• focus on functional strength of module	• focus on dependency between module.
• Represent ~~top level~~ detail design	• Represent top level design
A - B	A / B

Different Approaches of software design :=>

(i) Function oriented design (FOD)

(i) Object oriented design (OOD)

FOD :=> top level module is decomposed in more sub modules where each module has well defined task..

OOD :=>

- Entire system is viewed as an object
- Each object has its own state & behavior

Structural Analysis :=>

Systematic development method that allows analyst to understand overall functionality of the system with the help of DFD (Data flow diagram)

- we decomposed high level function into subfunctions

it has 3 measure principle

(i) Top down decomposition
(ii) Divide & Conquere principle
(iii) Graphical representation of analysis — DFD

DFD :=>

- stands for Data Flow diagram
- Also known as Bubble chart
- Graphical Representation of data flow

 Entity → Data flow

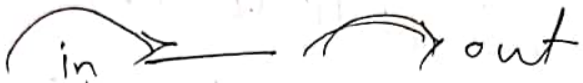
 - process = Datastore

Type of DFD =>
- logical DFD
- Physical DFD

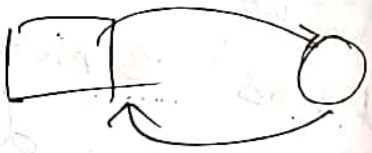
Rules of DFD

- each process should have atleast one input & output

- Each data store has atleast one data flow in & data flow out.

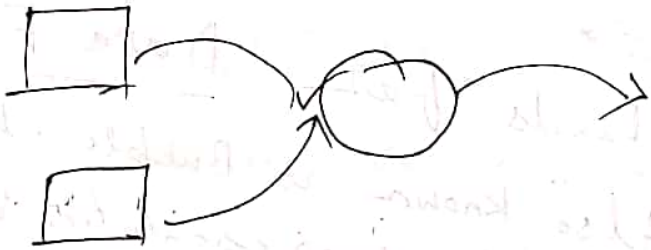


- All external entity must be connected through a process & entity can provide data to process or take data from process.



Levels of DFD :=>

(i) 0th level DFD (context level DFD) : Provide entire system data flow and processing with a single process is called as context



(ii) 1st level DFD :- include database and various important sub unit

Coding :⇒

- We can transform design to a real implementation with the help of programming language
- Should follow coding standard so that code is easily understandable, easy debugging & error free

coding Standards are:-

(i) Less use of global variable
(ii) Exception handling mechanism
(iii) different naming convention for global, local, constant
(iv) use of comment
(v) module should contain 1-20 line of code
(vi) Don't use short form for name of variable & method

Total Quality Management (TQM) :⇒

TQM means -

(i) Total → involvement and input of everyone
(ii) Quality → Fully meets the customer requirements and customer expectation
(iii) Management → The way how we manage operate, control & handle our organization.

Key Principles are:

- commitment from management
- Employee environment
- Continuous improvement
- Customer Requirement

Effect of TQM

improve quality
↓
increase productivity
↓
Lower cost & higher profit
↓
organization growth

Software quality assurance (SQA):=>

- SQA says whether the software is quality or not..
- SQA is checked in all the phases along with development.
- The quality is verified by the third party organization like ISO (international standard organizatio.)
- Several quality factors are

(i) portability (iii) Functionality (v) maintainability
(ii) Reliability (iv) Error free
(v) usability

ISO :=>

- stands for "international organization for standardization
- organization to verify the quality of the product.
- ISO 9001 is used to check quality

SEICMM :=>

- Stands for Software engineering institute capability maturity model.

it give total 5 level

(i) initial (ii) Repeatable (iii) Defined (iv) Managed (v) optimizing

Software Maintenance :=>

- To enhance the performance
- To add / remove features
- To correct errors in current Product

(i) Adaptive Maintenance : For running on different platform, change interface. (By developer

(ii) Perfective Maintenance : Maintenance so that we can use for a very long time.

(iii) Corrective Maintenance: Problems reported by user is resolved.

Reverse Engineering :=>

- it is also called back engineering
- it is the process of recovering and understanding the basic method, design & required specification of legacy S/W product.
  (already available)
- It is a process to rethink, restructure & rebuilt legacy Software

---

(X) incremental Model = Linear Model + Prototyping model

- Win Win spiral has extra feature like → set of negotiation activities at beginning

(✓) Client Server application - Concurrent Model

(X) less experience on similar → Spiral Model

(X) Requirement not need in early → Spiral + Prototype

(X) no cost barrier & within time → RAD

(X) Software engineer shall act in a manner that is best interests for their client & employer consistent with public interest & ensure that their product & modification meet the highest professional standard possible.

(X) Software bug & failure - Responsibility of developer + company

(X) Software is set of programs documentation & configuration of data.

(X) Software ~~Validation~~ Verification not the fundamental notion for engineering

(X) Build & Fix model suitable for Small line of code (100-200)

(X) RAD → RAPID application development

(X) Types of Prototyping models are
- Horizontal
- domain
- vertical

(X) Prototyping model not include coding.

(X) RAD Model has 5 phases
- Business modelling
- Process M
- Application generation
- Testing
- Turnover

User is involved in all phases

Model

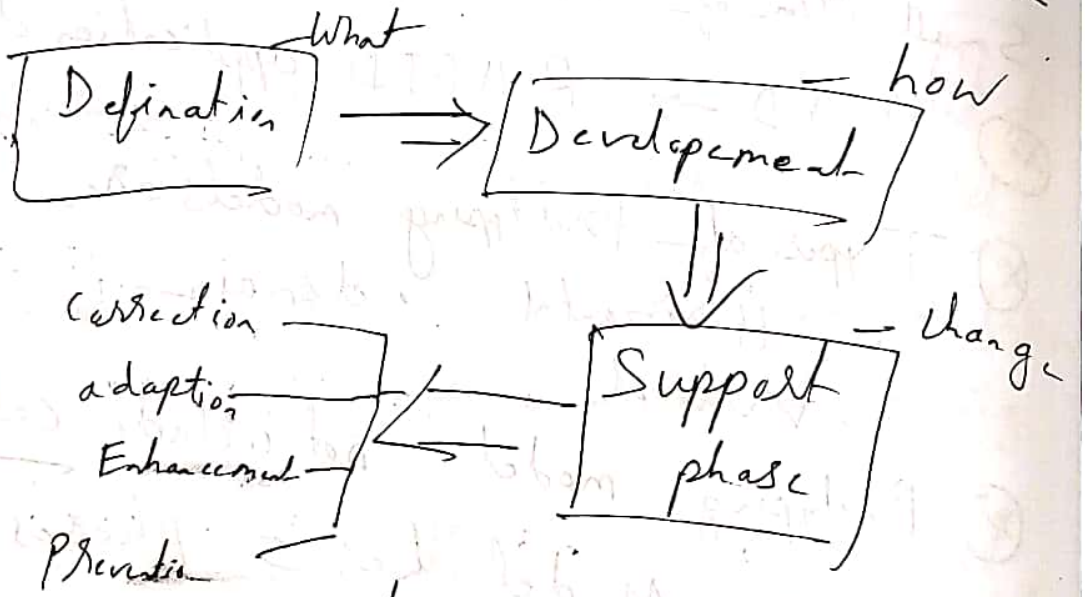
- Both RAD & Prototyping facilitate reusability of components.
- Spiral has high reliability Requirement RAD has not
- Software process quality ← Productivity, timeliness, visibility
  product quality — portability

(x) Software product ← Generic, customised

(x) maintainance cost >> develope cost

(x) After deployment customer give feedback

(x)



phases in SWE

(x) Software quality ← External ← Salibility, usability, reliability, usability, Sec.
internal — reusability

- RUP — Rational unified Process (IBM)
- Agile = incremental + iterative
- Scrum has 3 phases
- In agile development it is more important to build software that meets customer need today not worrying about features of future

---

- FAST → Facilitated Application specification Technique
- QFD — quality function deployment
- users are always most important stakeholder
- Functional Requirement — what the software has to do

non functional — quality of software.

Portability, Robustness, maintainibility

- Non function Requirement classified in

F U R P S

- F — functionality
- U — usability
- R — Reliability
- P — Performance (Speed, response time, throughput)
- S — Supportability (adapt, service test)

(X) 3 feasibility study is conducted in Req

- economical (cost)
- technical (hardware/ software)
- Legal + Schedule + operation

(X) Req analysis has 5 Phase

- Problem Recognization
- Evaluation
- Synthesis
- Modeling
- specification
- Review

- SRS is Consistent → no subset of individual requirement described in it conflict with eachother
- SRS is a contract. It is written by developer & in some case it can be written by customer too
- SRS is also called black box document (external only)
- SRS doesn't include design solution

- Structure chart is a chart which used for structured design
- Cohesion — module focus on only one thing — It should high
- coupling — module is connected to other module
  It should low.

- Rectangle represent ~~data store~~ entity in DFD
- — — data store
- ◯ — user

- Structural analysis is a software engineering technique that uses graphical diagram to develope or portray system specification that are easily understood by user
  - this view
    - Functional view (DFD)
    - data view (ER diagram)
    - Dynamic view (State transition diagram)
  - It is easily understand by ordinary customer

- Context diagram is also called level 0 DFD
- ↷ — data flow
- DFD always accompanied with data dictionary
- Data store (=) represent physical file / logical file or data structure.

Project Metrics is a quantitative measure to minimize of development schedule & accessing project quality frequently.

- Direct measure of SE — cost, Effort
  indirect " " → Efficiency, quality, complexity, Reliability
- Function point analysis (FPA) has five domains
  - input, output, inquiries, interfaces, logical files
- Defect Removal Efficiency = $\frac{E}{E+D}$ (E – error before delivery; D – Defects after delivery)

- Maintenance testing is done using breadth test & depth test
  - White box — structured testing
    black box — behavioural "
  - Exhaustive (same test cases different product) testing is impractical but possible
  - Adhoc testing — done without planning & documentation.
- Boundary Value analysis is a black box technique
- kLoc & function point is used in measuring size of a software.
- Albrecht — Function point