Types of Software

① System Software
Collection of programs return to service other programs.
Eg → Operating System like macOS, Linux, Android

② Application Software
Stand alone programs use for a specific purpose. (users & Business needs)
Eg → gaming apps, Banking apps, shoping apps etc

③ Embeded Software
It resides within a product or system and is used to implement and control features and functions for the end user or the system itself. They either perform limited function like in microwave or significant function like a automobiles. Eg → Washing Machine, Satellites, Routers, traffic control

④ Product line Software
It provides a specific capability to different customs

⑤ Web Application
They require internet connection to work {client-server computer program which the client runs on the web browser}
Eg → online forms, Gmail, Yahoo, photo editing etc

⑥ AI Software
They use non numerical algorithm to solve complex problem
Eg pattern recognition software, decision support systems, Google Cloud.

⑦ Scientific Software / Research Software
These are based on numerical algorithm. (↳ change acc to time) (Task take real time)
Eg weather forecasting Eg → MATLAB, AUTOCAD, ORCAD, Stock Market app.

⑧ Simulation Software
It is a program that uses mathematical models to imitate real world processes and products
Eg Car crash modeling, Anylogic

Software Engineering

→ It is the establisment and use of some ethetical sound engineering principal in order to obtain economical, reliable and efficient softwares.

→ It is an application of systematic, decisiplined, quatitifible aproche to the development operation and maintance of softwares (some of)

* Importance of SE

→ It reduces complenity

→ It helps to minimise ~~fast~~ cost

→ It helps in the development of reliable software

19/1

* SDLC (Software Development Lifecycle)

It is a process used to design, develop and test high quality softwares. It is also called software development process.

@ ISO/IEC 12207 is an international standard for software lifecycle process.

- Phases of SDLC

① Requirement analysis & planning →
planning for quality ensurance & identification of all risk.

② Definning requirement →
documenting the requirements in the form of software requirement specification (SRS)

③ Designing the software →

④ Developping the project

⑤ Testing the project

⑥ Deplomentation

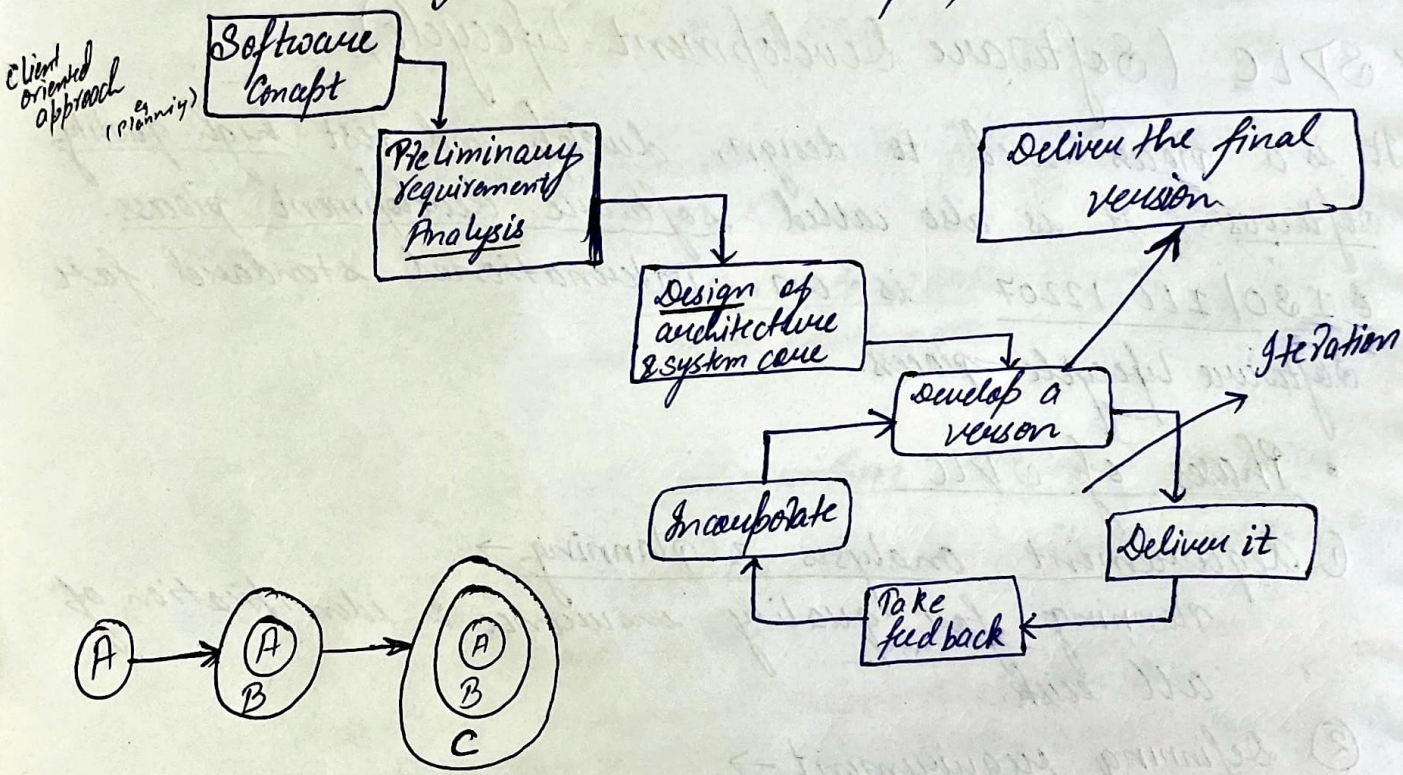
⑦ Maintanance

- Benifits of STLC

① It allows highest level of management skills
② Everyone understands/envolved cost and resources required
③ It improves the application quality & monitoring by being present at every level

- **STLC Models →**

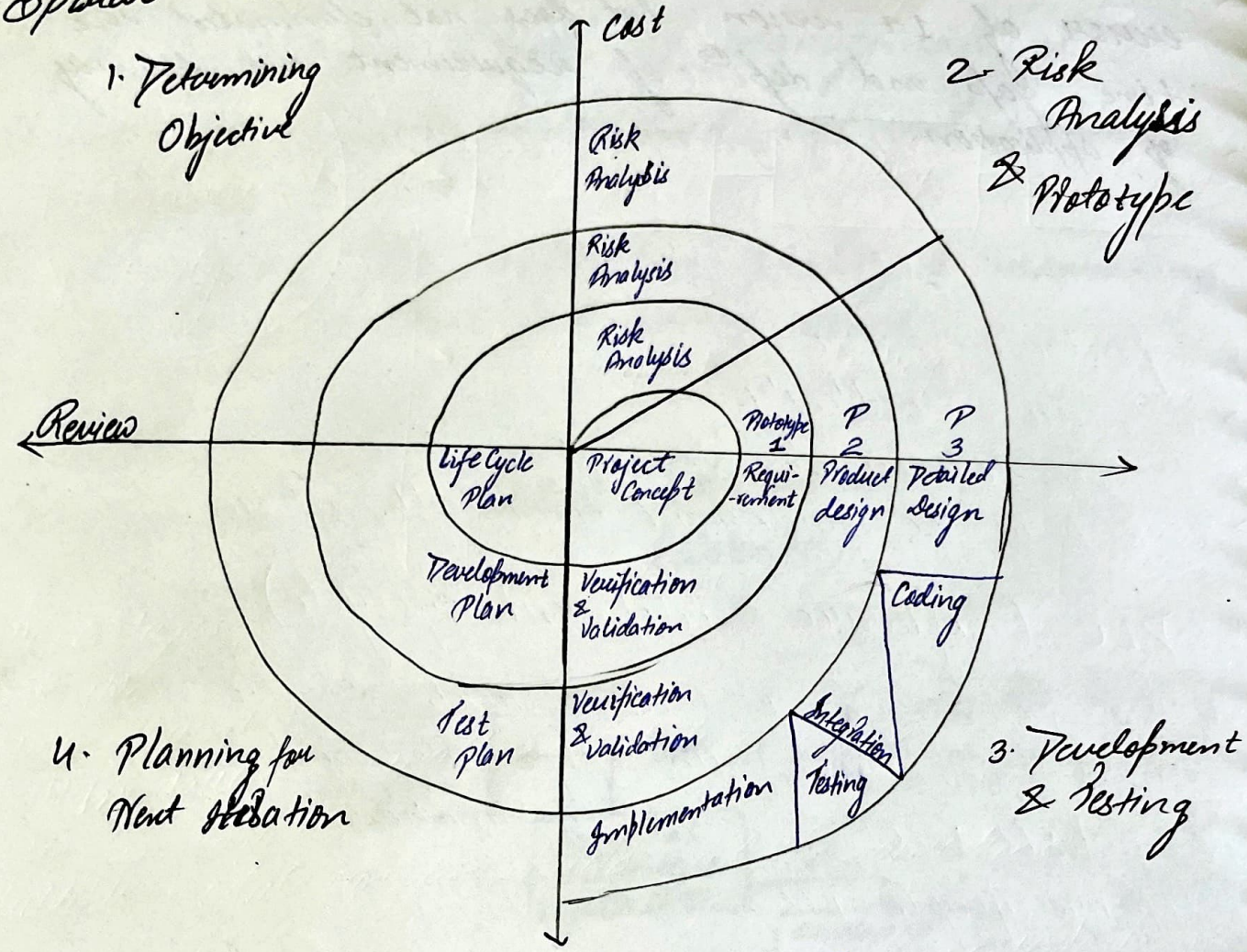
① Evolutionary Models (Incremental model) (Waterfall)



Intro→
It consist of two phases, in the 1st phase of the product is viewed as a child to assest the fisibility of the pdt & to verify the requirement then the product is through away & the real development start. This version is called throwaway prototype (small project)

In the second phase the product is developed using iterated model it eliminate the flows and error of 1st version but does not eliminated the time gap and def^{n} of requirement and delivery of application.

(2) # Spiral Model



Spiral model is divided into four sections.
This concept was given by Barry Boehm 1980s
He incorporated the idea of risk analysis
This model is an incremental risk oriented life cycle model
The iteration help in regular testing & feedback

Phases →
The spiral model constitute of 4 phases.
Neigher the no. of spiral more will be the cost of developing the project

(1) Determining Objective
It is the gathering of requirement for the software product using the SRS (Software Requirement Specification)

(2) Risk Analysis & Prototype
The possible risk along with their possible solution are generated in this phase. The prototypes helps in understanding the needs & features of the final product.

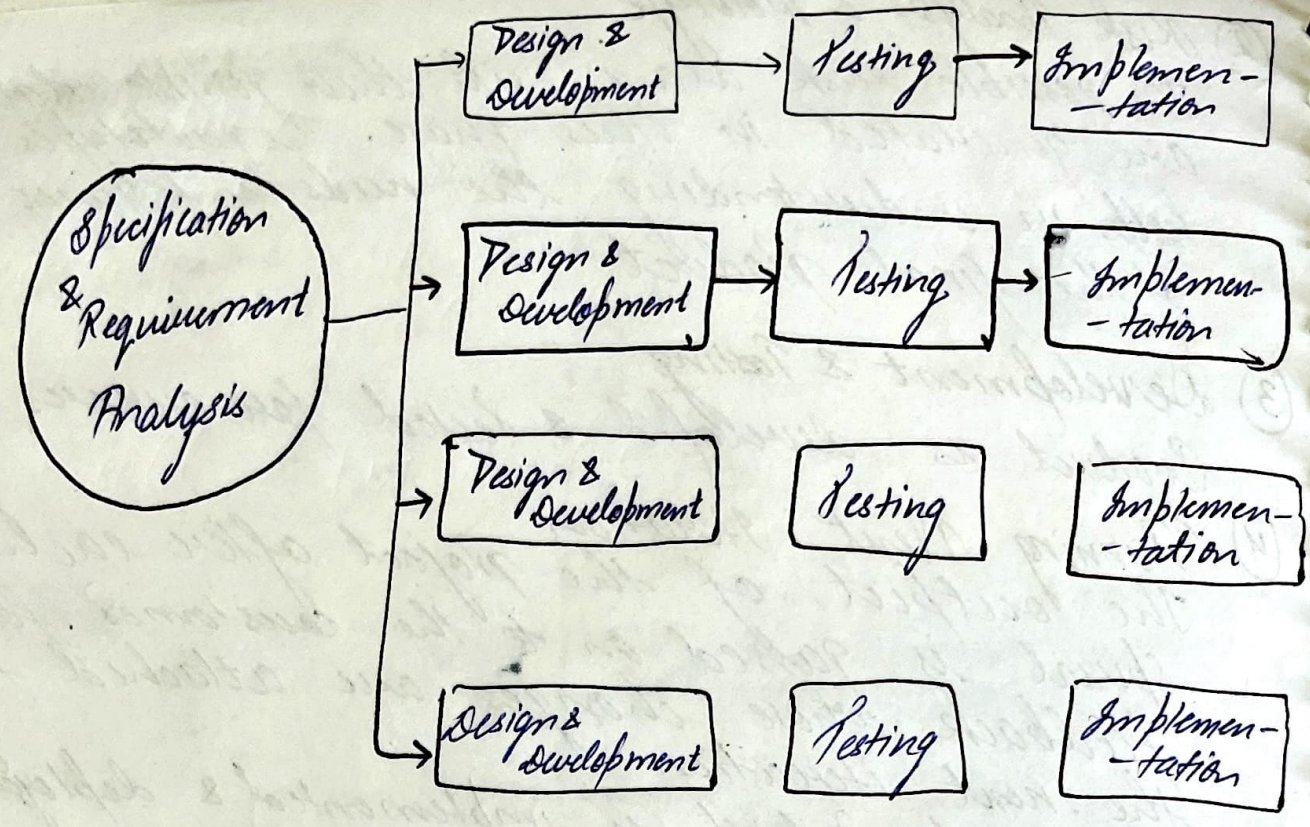
(3) Development & Testing
Product is developed & tested for errors

(4) Planning . Next Iteration
The output of the project after each spiral is passed on to the customer for feedback whose changes are attached to the next iteration.
The final product is implemented & deployed to the customer

* Advantages
→ Highly customizable
→ Risk Analysis oriented
→ Suitable for larger project with heighen risk
→ Lessen chance of failure

* Disadvantages
→ Heighen the no. of spiral more will be the cost
→ Highly experience risk Analysis are needed
→ It is a complex model → not suitable for smaller project
→ Time taking

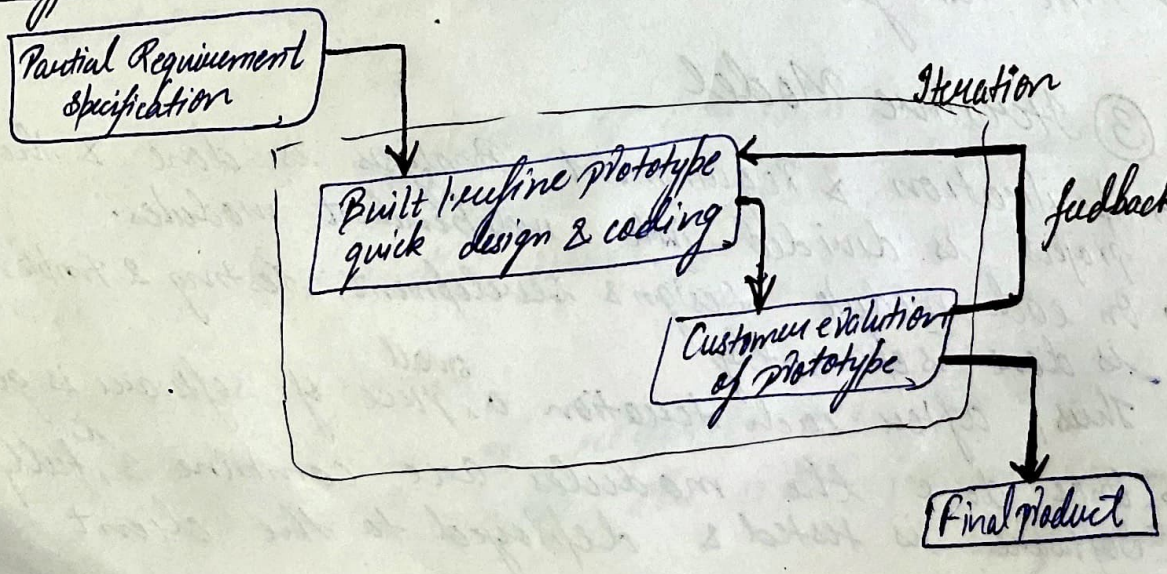
(3) Iterative Model
→ Specification & Requirement Analysis is done & then the project is divided into independent modules.
→ On each module Design & Development, Testing & Implementation is done seperately.
→ Thus, after each iteration a small piece of software is generated
→ Once done the modules are combine & the fully functional Software is tested & deployed to the client



* Advantages
Flexible
Time efficient
Implementation of smaller module is easier
Testing is easier
Multiple module can be build at the same time

* Disadvantages
→ Highly resources requirement → can be costly/expensive
→ Higher the no. of modules required, higher management skills

#④ Prototype Model →

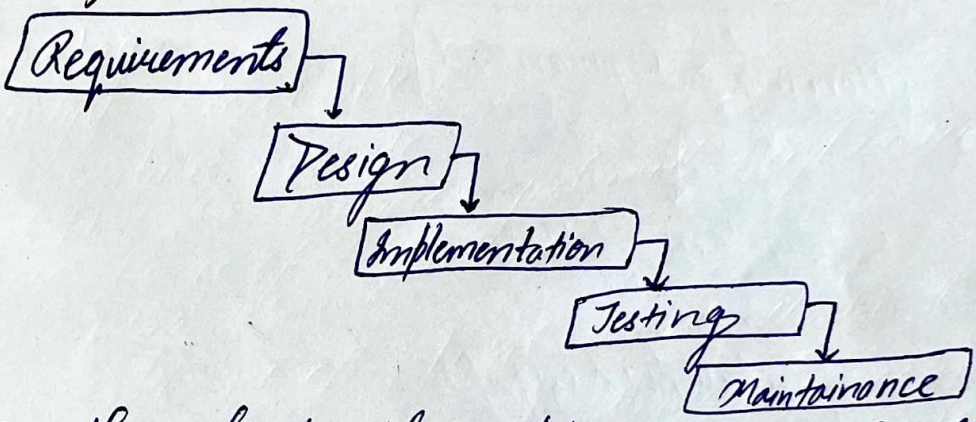


→ A prototype of the end product is developed, tested & refined as per customer feedback till satisfaction is achieved
→ this model is used when the customer donot know the exact requirements beforehand.

• Advantages →
→ reduces risks & errors
→ effective when the requirements are unclear
→ client satisfaction is achieved

• Disadvantages →
→ can be costly & time-consuming
→ a bad prototype may become the final product (unstable)
→ It requires extensive customer colaboration

#⑤ Waterfall Model →



→ It was the first and simplest fully functional model developed using the concepts of SDLC
→ the phases of waterfall model are arranged in such a way that the O/P of the previous stage become the I/P of the next stage.
→ The sequence of the phases are fixed & cannot be reversed

* Advantages →
→ best suitable for small to medium project
→ simple & easy
→ highly structured model

Disadvantages →

→ no flexibility
→ not suitable for large project
→ calculation of cost, time & effort for phase is quite difficult

#⑥ Build & Fix Model →

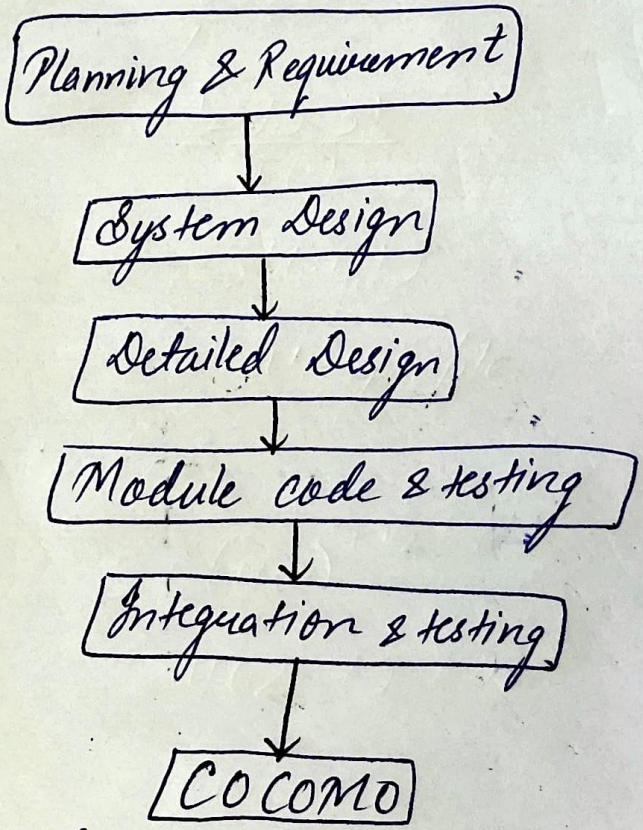
COCOMO

→ Constructive cost model by Boehm was introduced in 1981
It is one of the most used model to estimate the cost of a software product based on the size of the project (KLOC) Kilo lines of codes
(PM : person month)

→ It also helps in estimating other parameters such as size effort, time, quality & schedule

In simple words, it predicts the performance of the software

* Phases of COCOMO →



* Importance of COCOMO →

→ Cost estimation
→ Resource management
→ Project planning
→ It provides a support for decision
→ Risk management

* Types of project A/c to COCOMO

Parameters	Organic	Semi-Detached	Embedded (Eg 3of formula, cal)
Project Size	5 to 50 KLOC	50 to 300 KLOC	300 and above KLOC
Complexity	Low	Medium	High
Team Experience	Highly Experienced	Some experienced & Some inexperienced	Mixed experienced including Experts
Environment	Flexible (Fewer Constraint	Moderate constraint	Strict Environment
Eg	Simple Pay-role System	New system interfacing with existing system	Flight control Software

* Types of COCOMO Model →

① Basic Model

It is used for quick & rough calculation model

(constant values of a, b, c and d for basic COCOMO)

Project Type	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

(Effort) $E = a(KLOC)^b$ PM

(Time) $T = c(E)^d$ months

$People = \frac{E}{T}$ people

Ques The project size is 400 KLOC. Calculate the effort, time & people required for organic, semi-detached & embedded project

① Organic

$$E = a(KLOC)^b$$
$$= 2.4(400)^{1.05}$$
$$= 1295.31 \text{ PM}$$

$$T = c(E)^d$$
$$= 2.5(1295.31)^{0.38}$$
$$T = 38.07 \text{ months}$$

$$\text{People} = \frac{E}{T}$$
$$= \frac{1295.31}{38.07}$$
$$= 34.019$$
$$= 34 \text{ people}$$

② Semi-detached

$$E = 3.0(400)^{1.12}$$
$$= 2462.79 \text{ PM}$$

$$T = c(E)^d$$
$$= 2.5(2462.79)^{0.35}$$
$$= 38.45 \text{ month}$$

$$\text{People} = \frac{E}{T} = \frac{2462.79}{38.45} = 64.04 = 64 \text{ peoples}$$

③ Embedded

$$E = a(KLOC)^b$$
$$= 3.6(400)^{1.20}$$
$$E = 4772.81 \ PM$$

$$T = c(E)^d$$
$$= 2.5(4772.81)^{0.32}$$
$$= 37.59 \text{ month}$$

$$People = \frac{E}{T} = \frac{4772.81}{37.59} = 126.94$$
$$= 127 \text{ people}$$

Ques
A project of size 200 KLOC is to be developed. Development team has average experience & scheduled is not very tight. Calculate the effort time & people required & also mentioned the project type taken into consideration

Sol
Semidetached

$$E = a(KLOC)^b$$
$$= 3.0(200)^{1.12}$$
$$E = 1133.11 \ PM$$

$$T = c(E)^d$$
$$= 2.5(1133.11)^{0.35}$$
$$T = 29.30 \text{ month}$$

$$People = \frac{E}{T} = \frac{1133.11}{29.30} = 38.67 = 39 \text{ people}$$

Ques A company needs to develop a software of size 20000 KLOC. The multiplicative factors is 2.2 & Exponentiation factors is 1.5 Calculate the estimated effort (b)

Sol 20000loc → 20 KLOC

$E = 2.2 (20)^{1.5}$

$= 196.77$ PM

~~$T = c(E)^d$~~

~~$= 2.2 (196.77)$~~

(2) Intermediate Model

Project type	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

$E = a (KLOC)^b * EAF$

$T = c (E)^d$

* EAF is effort adjustment factor. It is a multiplier used to refine the effort estimate based on basic COCOMO model

③ Detailed Model

→ It is a combination of both basic & intermediate model

→ It is decomposed into multiple modules and model is applied to them individually.

→ Cost is calculated at each stage separately.

* Staffing level Estimation

⇒ Putnam was the first to study the problem of proper staffing pattern for software projects. He extended the work of 'Norden' who work on R&D.

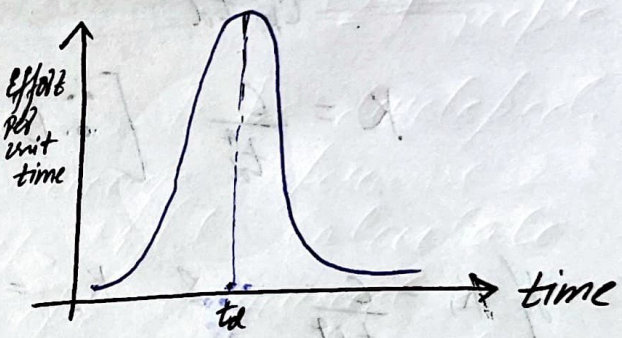
↳ NORDEN WORK'S →

$$E = \frac{K}{t_d^2} * t * e^{-\frac{t}{2t_d^2}}$$

* E = no. of developers required at time t

K = Area under the curve

t_d = Time at which the curve attain the max. value



Rayleigh Curve

↳ PUTNAM'S WORK →

$$L = C_k K^{1/3} t_d^{4/3} \quad \text{——} (1)$$

where t_d = peak time of system & integration techniques

K = Total effort in PM

L = Project size in KLOC

C_k = It is the state of technology (Working environment)

Eg If $C_k = 2$ → Poor methodolgy (Poor development environment)

If $C_k = 8$ → good development environment

If $C_k = 11$ → Best development environment using automated tools.

→ PUTNAM suggested that only small no. of developers are required in the beginning. It reaches the peak during testing after which the staff decreases

→ The ratio of development effort to maintance effort is 40:60.

→ Constant level of staffing would create either an overstaffed situation or an understaffed one

↳ Effect of schedule change on cost

$$K = \frac{L^3}{C_k^3 t_d^4} \quad (\text{From eq } ①)$$

$$K = \frac{C}{t_d^4} \quad \left[\because C = \frac{L^3}{C_k^3}\right]$$

1 year → 2^4
6 month → 16

If $K_1 = \frac{C}{t_{d_1}^4}$ & $K_2 = \frac{C}{t_{d_2}^4}$

$$\frac{K_1}{K_2} = \frac{C}{t_{d_1}^4} \times \frac{t_{d_2}^4}{C} = \frac{t_{d_2}^4}{t_{d_1}^4} = \left(\frac{t_{d_2}}{t_{d_1}}\right)^4$$

This means that when scheduled of a project compresses the required effort increases by 4th power of the degree of compression.

Eg To develop a project 6 month instead of a year the effort required will increased by 16 time

1 year = 2×6 month

$2^4 = 16$

→ Acc@ to PUTNAM it is better to use few people in long span of time rather than more people in a quick span of time.

→ This model is better for very large project & not for small & medium project for that we use Jensen's model

JENSEN'S MODEL

$$L = C_{te}\, t_d\, K^{1/2}$$

C_{te} — working on
C_{te} is effective technology
t_d is time to develop the software
K is the effort needed to develop the software

$$\text{If } K_1 = \frac{C}{t_{d_1}^2} \quad \& \quad K_2 = \frac{C}{t_{d_2}^2} \left[C = \frac{L^2}{C_{te}^2} \right]$$

$$\frac{K_1}{K_2} = \frac{C}{t_{d_1}^2} \times \frac{t_{d_2}^2}{C}$$

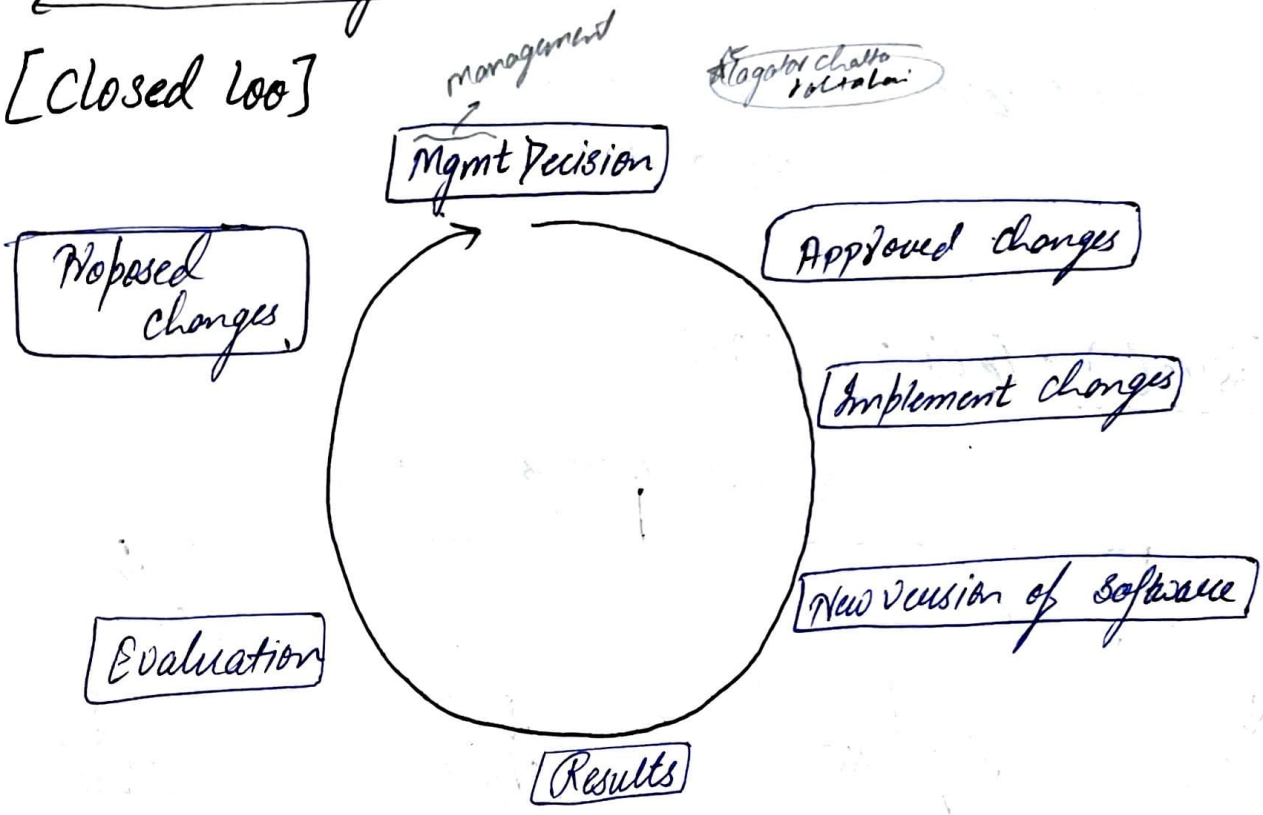
$$= \left(\frac{t_{d_2}}{t_{d_1}}\right)^2$$

→ When the scheduled of a project compresses the required effort increases by 2th power of the degree of compression

* Estimating Software maintainance cost

Boehm's Software Maintainance Model —

[Closed loop]



Software maintenance cost

$$ACT = \frac{(KLOC_{added} + KLOC_{Deleted})}{KLOC_{Total}}$$

$$AME = ACT * SDE$$

$$Total\ Effort = AME + SDE$$

If ACT = Annual change traffic (1 year me kitna change hua hai)
↳ changes in a sourced code in 1 year.

AME = Annual Maintainance Effort

SDE = Software Development Effort

Ques The ACT of a software is 20% per year. The development effort is 700 PM. Computer AME. If the life time of the project is 15 years what is the total effort of the project

SDE = 700 PM

AME = 20% of 700

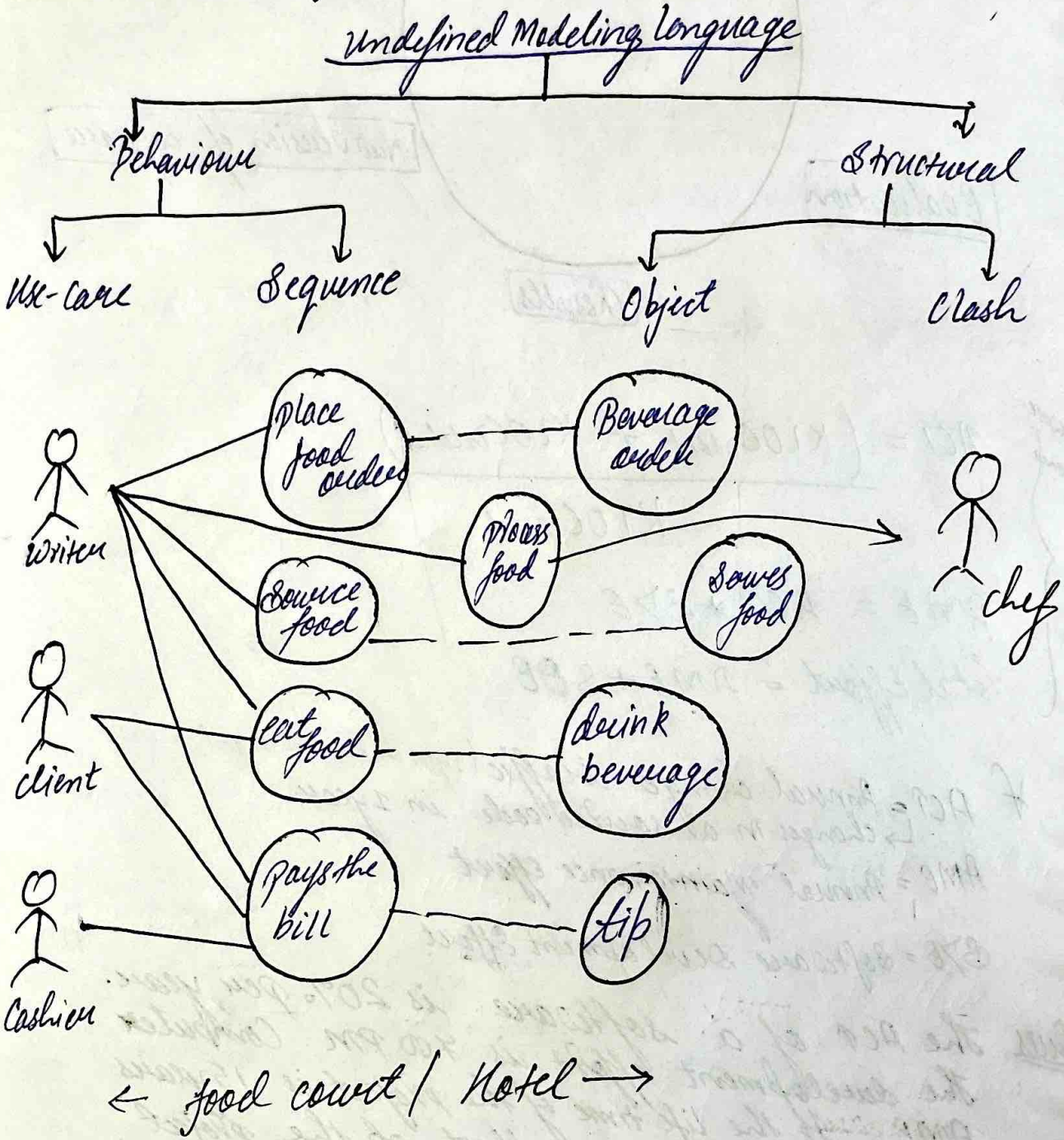
$= \frac{20}{100} \times 700 = 140$ for 1 year

Avg 15 year = 140 × 15 = 2100

Total Effort = 2100 + 700

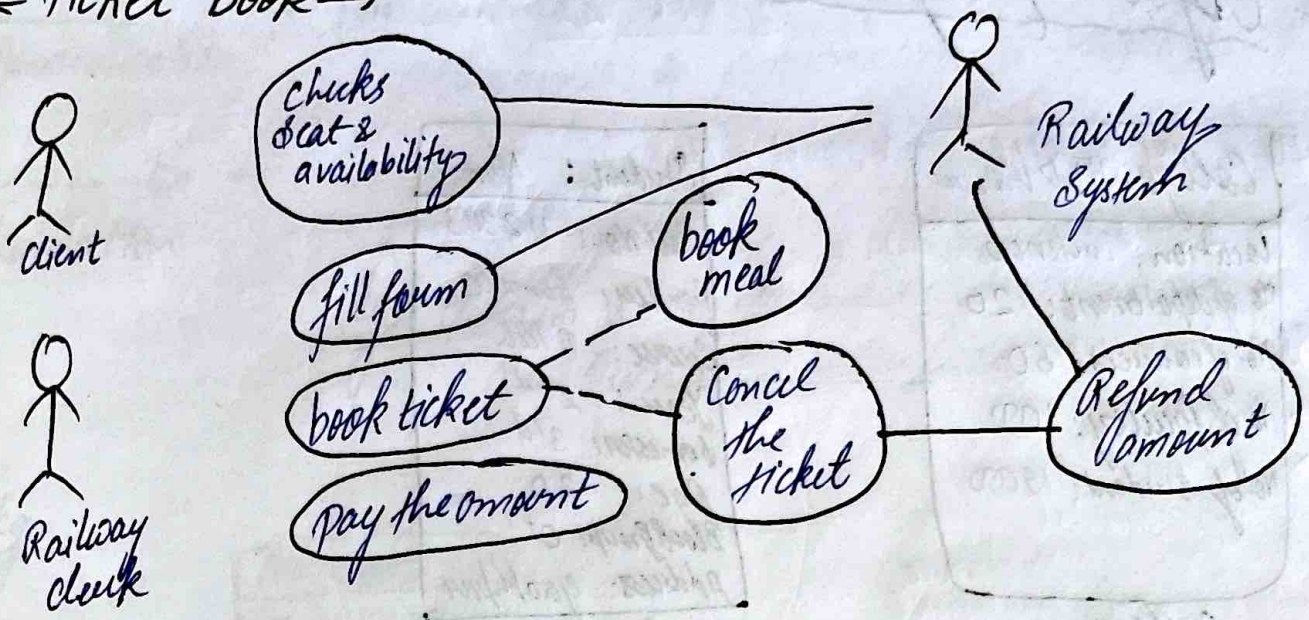
= 2800 PM

Visual Modeling - UML



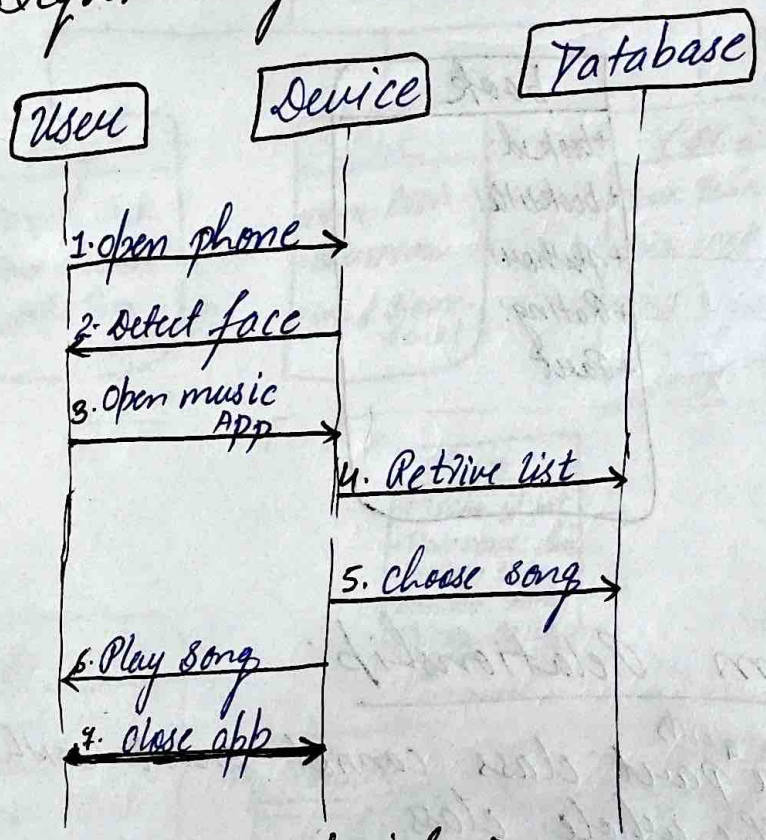
← food court / Hotel →

← Ticket book →

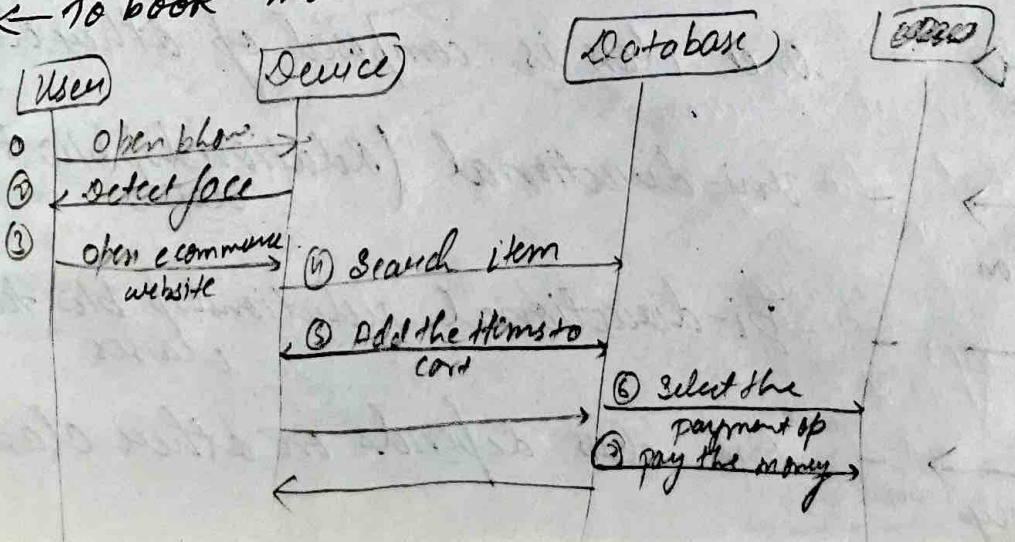


Sequence Diagram

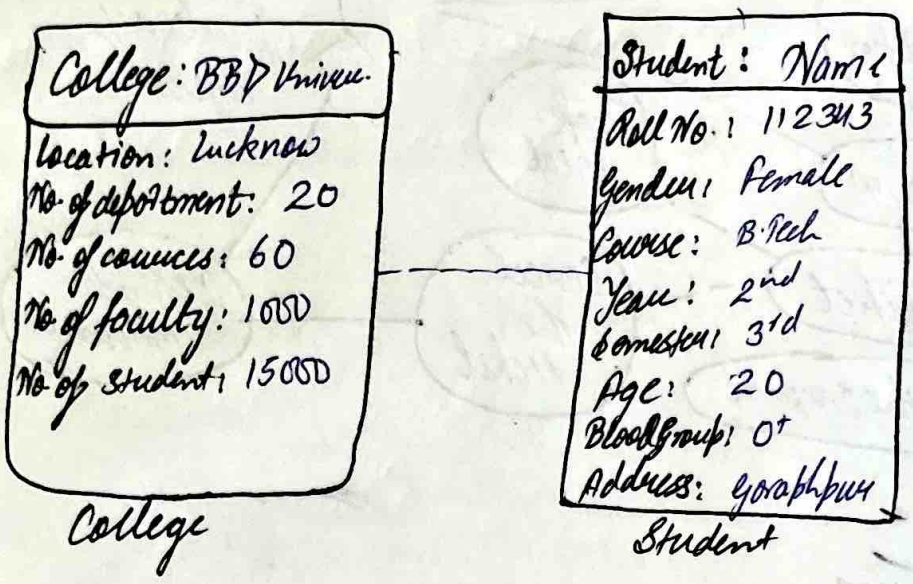
①

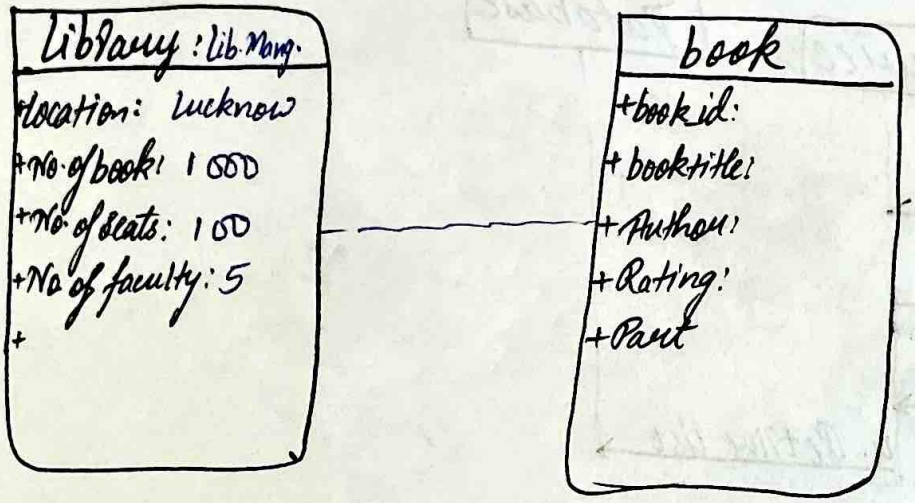


② ← To book material →



Object Diagram





Class Diagram Relationship :

Composition — The part class (or sub) cannot exist without the whole class

Aggregation — One class is composed of other classes

Directed Association — Uni-directional (Relationship b/w two classes)

Association — Bi-directional relationship b/w two classes

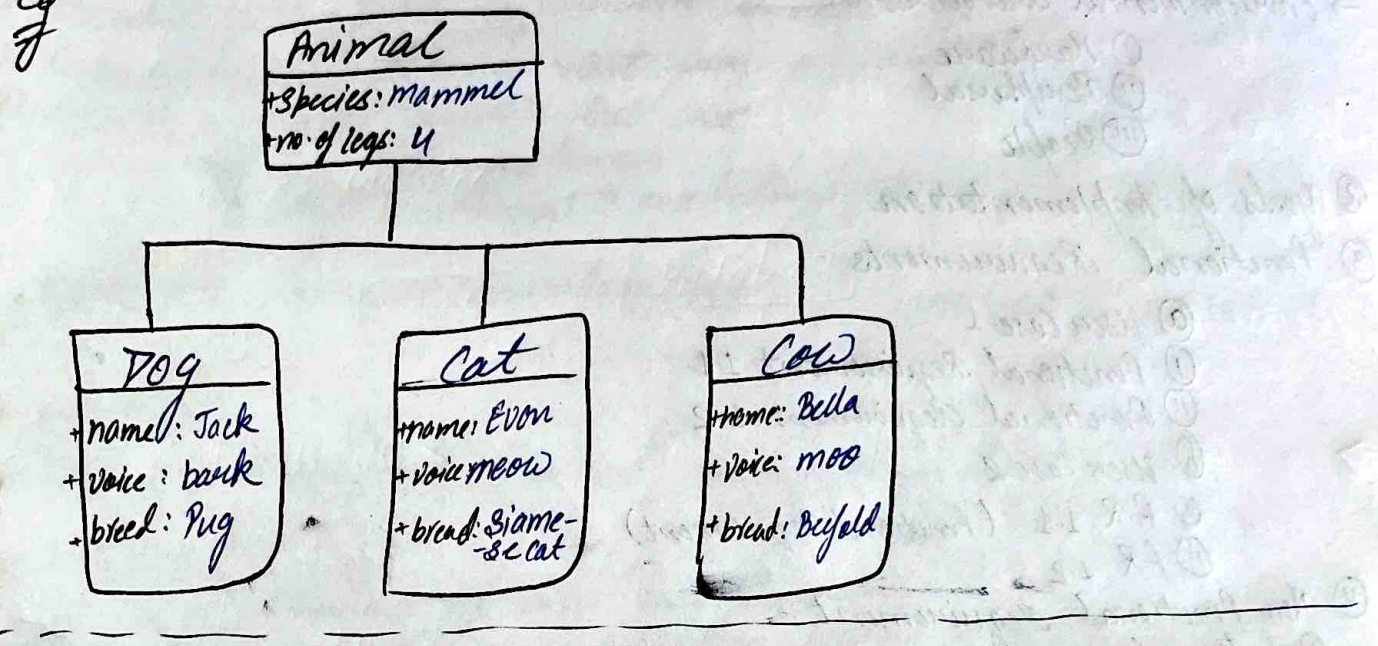
Dependency — One class depends on other class.

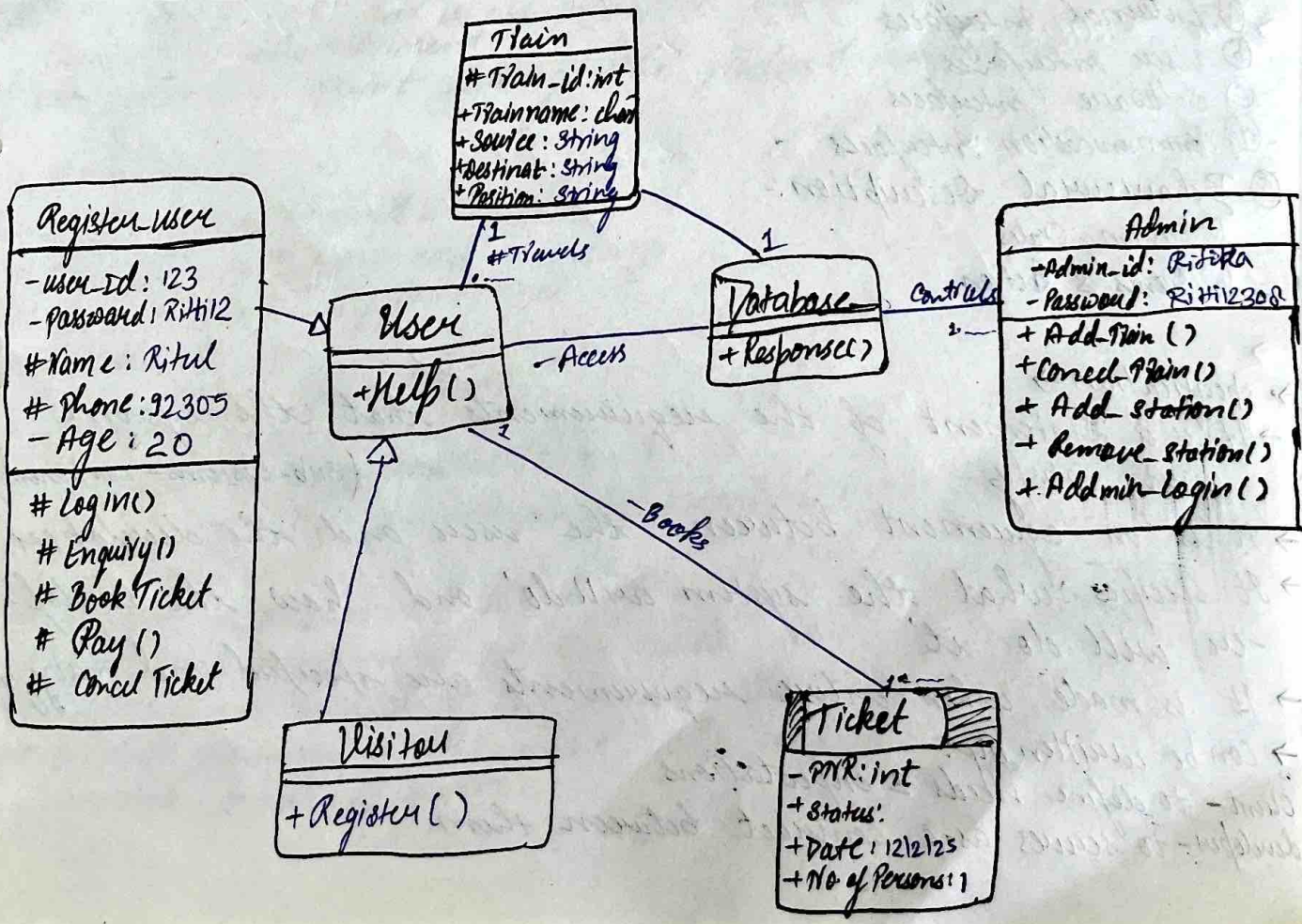
Generalisation – (Inheritance) once class inherits the attributes & behaviour of parent class

Realisation – A class implements the feature of an interface

Eg Person & corporation both realizing on 'Owner' interface

Eg





Unit-2

Software Requirement Specification (SRS)

→ IEEE830 (Institute of Electrical & Electronics Engineers)

① Introduction :-
→ Purpose
→ Overview
→ Environmental Characteristics
  (i) Hardware
  (ii) Peripheral
  (iii) People
② Goals of Implementation
③ Functional Requirements :-
  (a) User Case 1
  (i) Functional Requirement 1.1
  (ii) Functional Requirement 1.2
  (b) User Case 2
  (i) F.R. 1.1 (Functional Requirement)
  (ii) FR 1.2
④ Non-Functional Requirement :-
  (a) External Interfaces
  (b) User Interfaces
  (c) Software Interfaces
  (d) Communication Interfaces
⑤ Behavioural Description :-
  (a) System States
  (b) Events & Action

→ Specification →
→ It is a statement of the requirements that the system (sub-systems + components) must satisfy
→ It is an agreement between the user and the developer
→ It specifies 'what the system will do' and 'how the develop-er will do it'
→ It is made when entire requirements are specified and analyzed
→ Can be written by :
Client - to define needs & expectations
developer - to serves as a contract between them.

* Characteristics: of a good SRS document

(I) Correctness → provides accuracy of requirements stated
(II) completeness → All info. relating to functionality, performance. design, constraints, queries & responses, labels, references, tables, diagrams, definition of terms used, index
(III) consistency → if there is no conflict
(IV) unambiguousness → elements should be uniquely interpreted
(V) Stable → the product can be modified
(VI) modifiable → ↑ to some extent, modifications should be indexed to.
(VII) verifiable
(VIII) traceable → if origin of requirement is clear
(IX) design independence → option to select from multiple design
(X) testable → should generate test cases
(XI) understood by the customer
(XII) right level of abstraction → data hiding

* Properties (Same as characteristics)

(I) Concise
(II) structured
(III) Black box view - focus on what rather than how

* Uses (need)

(I) It is a statement of user requirements.
(II) A statement of the user interface b/w the machine & the controlled environment
(III) A reference point during the product maintainance

* SRS process

→ Define your product's purpose
→ Describe what you are building
→ detail the requirements
→ deliver it

* Components of SRS →

(I) Functional Requirements →
It discuss the functionalities needed from the system such as:
→ detailed description of the inputs
→ Operations performed on the inputs
→ algorithms needed to implement the system
→ It must clearly state what the system must do if invalid input is given.

② Performance Requirement—
It discuss the performance constraints of the system such as:
→ problem analysis using UML & Data dictionary
→ IEEE standard for SRS

③ Design Constraints—
→ such as standard (formats & procedures) compliance, resource limits, operating environment, reliability and security requirements

④ External interface requirements→
Includes specification of all interactions of the software with people, hardware & other software.

* Axiomatic specification:—

① Axiom :- These are statement that is excepted as true without proof.

⇒ In this first order logic is used to write the pre and post conditions to specifies the operation of the system in the form of axioms.

⇒ Pre-condition → conditions to be satisfied before envoking a system successfully.

⇒ Post-Condition → condition to be satisfied after the system executing successfully.

② Develop Steps :-
→ specify the input range and the constraints
→ specify the condition which holds the output
→ stablish changes that made to the input parameter.
→ Combine all the pre & post conditions of the function.

Ques Specify the pre & post condition of a function that take a real number as argument & returns half the input value. if input is less than or equal to 100. else it returns Double the value

Sol@ $f(x: real): real$

Pre : $x \in R$

Post : $\{(x \leq 100) \wedge (f(x) = x/2)\} \vee \{(x > 100) \wedge (f(x) = 2 * x)\}$

Ques Specify of the function named search which takes an integer array & an integer key value as x arguments & returns the index in the array where the key value is Present

search (x: Array, key: integer): integer.

Pre : $\exists i \in [x.first .. x.last], Y[i] = key$

Post : $\{(x'[search(x, key)] = key) \wedge (x = x')\}$

* Complement :- It is a parameter named after the function executed successfully

features / def / Benifit / Case Emp → Dig

Computer Aided Software Engineering (CASE)

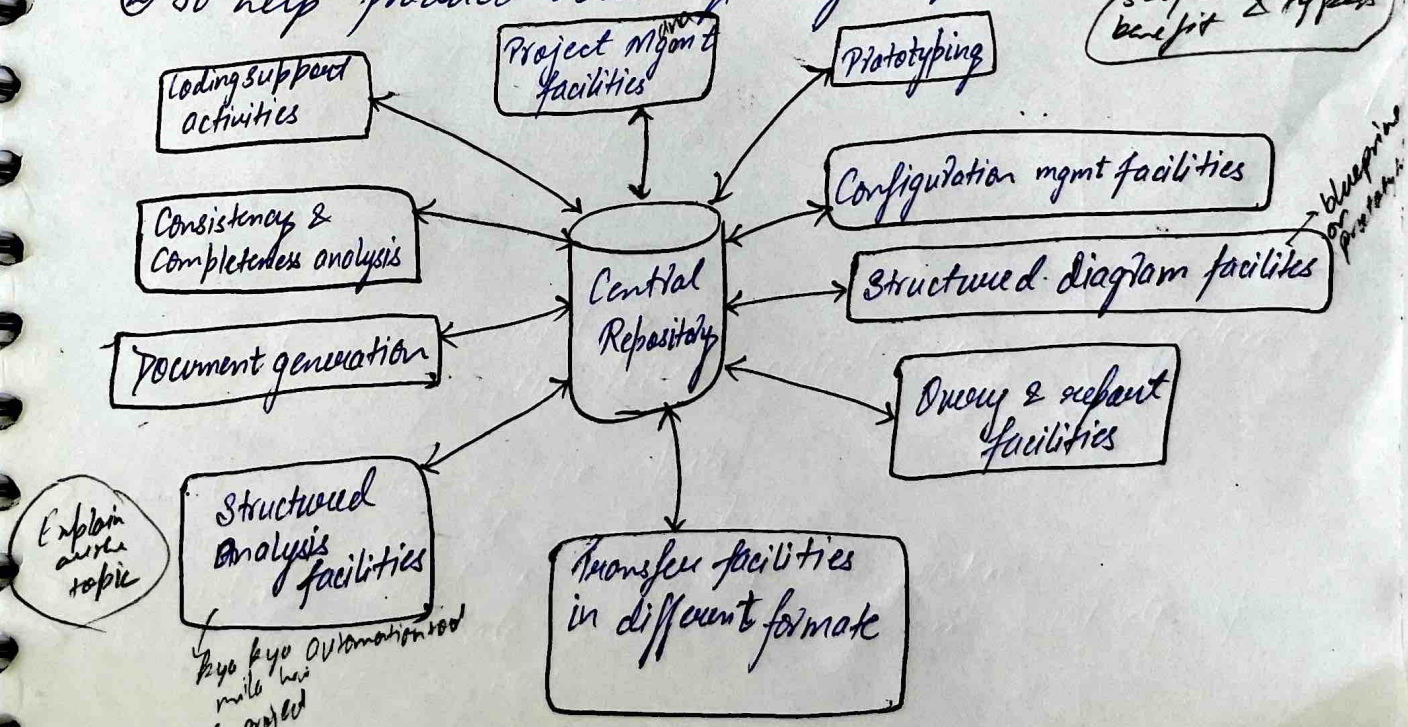
Software revo

→ A CASE tool is a generic term used to denote any form of automated support for SE.

→ The primary objective of using case tools are

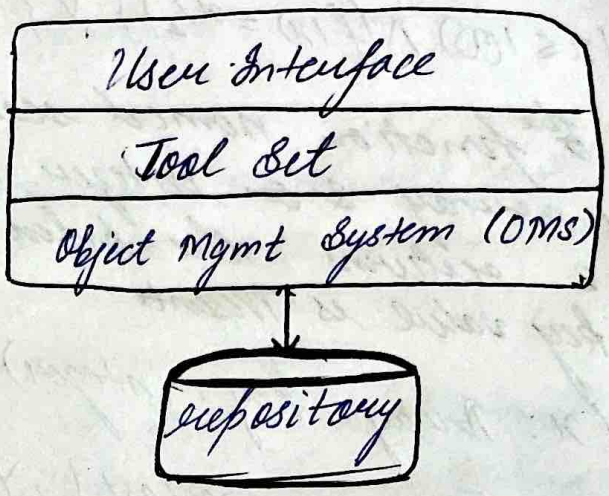
① To increase productivity

② To help produce better quality software at lower cost



* CASE Environment

* Architecture of CASE Environment



Explanation

OMS (Object Mgmt System) →
different case tools, we present the software product as a set of entities such as design, data, project plan etc. The OMS maps these entities into the underline storage management system (repository)