## Evolving role of software.

software → as product
→ as vehicle.

as Product

→ transform information
- produces, manages
modifies, display
transmit

→ Deliever computing potential of Hardware networks.

→ ex: Whatsapp

as vehicle

→ controls other program
operating system

→ effects communication
(networking software).

→ help to build other software (tools & environment)

→ ex - windows, ~~mac~~ ios, linux

① Define Software :-

1) instruction - that when execute provide desired feature function & performance.

2) Data Structure - enables program to adequately manipulate information

3) Document that describe the operation and use of the program

## Diffrence between hardware and software

| Hardware | Software |
|---|---|
| → it is manufactured. | → developed or engineered |
| → difficult to modify after manufactured | → easy to modify even after deploy- |
| → cost is concenstrated on production | → cost is concerntrated on design |
| → wears out over time and is component Based. | → deteriorates overtime (and is custom built<br>(progressively worse over time) |

## Problems of software:-

- cost
- schedule — completion in scheduled time
- Quality - maintaining the quality
- Scale and Change.

(Q) What is Software engineering.

this is an engineering branch associated with development of software product using well-defined scientific principle, method and procedure. this is an reliable product.

## (4) Characteristic of software.

1. operational   2. transitional   3. maintenance

1. operational characteristic

- it tells how well a software works on operations.

this can be measured on :

- Budget
- usability
- efficiency
- functionality.
- correctness
- dependability
- security.

2. Transitional characteristic.

this is the most important characteristic when the software is moved from one platform to another.

measured on –

- portability
- interoperability
- reusability
- Adaptability.

3. Maintanenece characteristic

this helps the characteristic of software to update it's features wanted by custome

it can be measured on –

- modularity
- maintainability
- flexibility
- scalability.

# Changing nature of software.

1) system software:-

system software is a collection of programs which are written to service others program.

Some system software are complex, It is having a sophisticated process processes management for scheduling resource sharing.

2) Application software

- this software solves specific business needs. It faciliates business operation or management technical decision making. This is used to manage business function in real time.

3) Engineering and scientific software.

- this software is used to faciliates the engineering function and task. however modern application with in the engineering and scientific area are moving away from conventional numerical methods.

4) Embedded software:-

it resides within the system or product and is used to implement and control feature and function for end-user or for the system itself. It can perform limitted and esoteric function.

5). **Product - line software :-**

Design to porovide a specific capability for use by many different customer. product line software can focus on the limited and esoteric market place oo address the mass consumer market.

6) web apps-

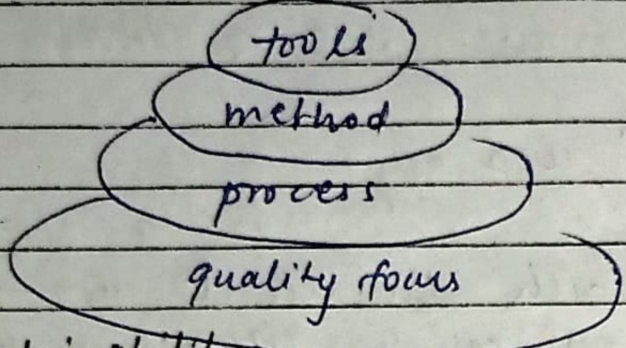
its a client sewer computer program which the clients run on the web server. Web apps can be little more than set of linked hypertext files that prevent info. using text and limited graphics.

7) Artificial Intelligence software -

-it makes use of non-numerical algorithm to solve complex problem that is not amenable to computation.

# Software engineering as layered technology.

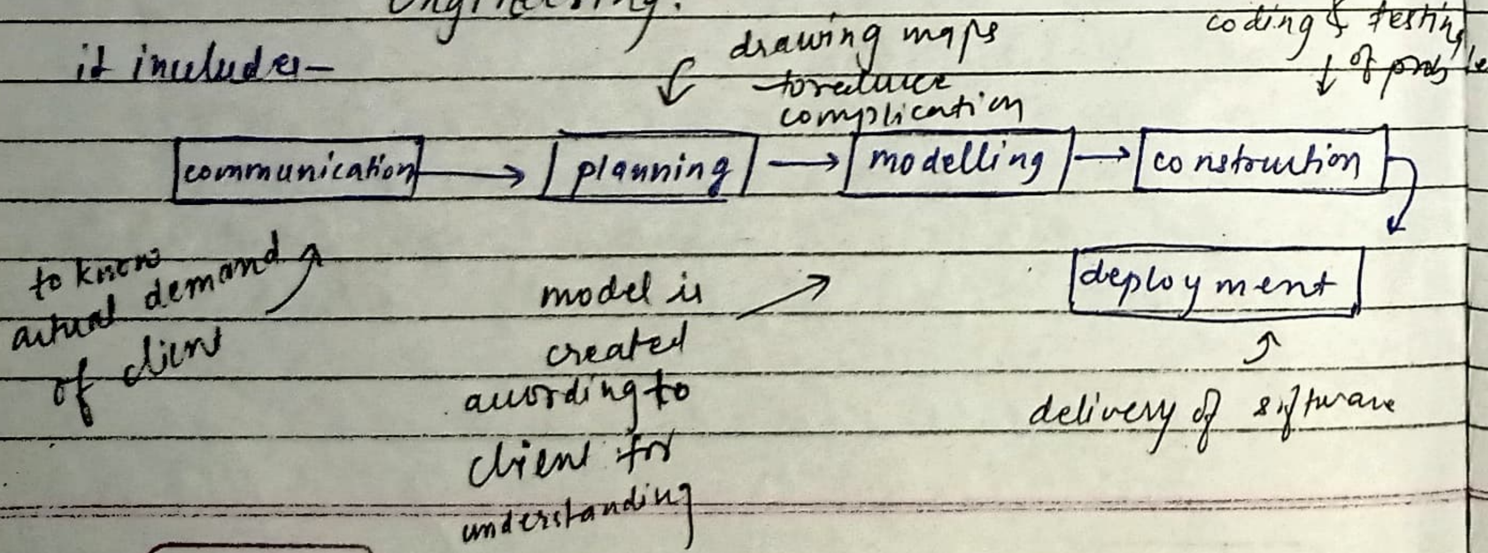
to engineer a software we need to go from one layer to another layer. All layers are connected and demands the fullfilment of previous layer



① quality focus-
- focus on maintainability and usability
- secured (only authorised person's have access).
- continuous process of improvment.

② process -→ covers all the steps to be taken for accomplishment
→ defines that software will deliver on time or not. (effective delivery).
→ foundation or base layer of software engineering.

it includes-


drawing maps
~~to reduce~~
complication
coding & testing of product
communication
planning
modelling
construction
deployment
to know actual demand of client
model is created according to client for understanding
delivery of software

⊛ method-

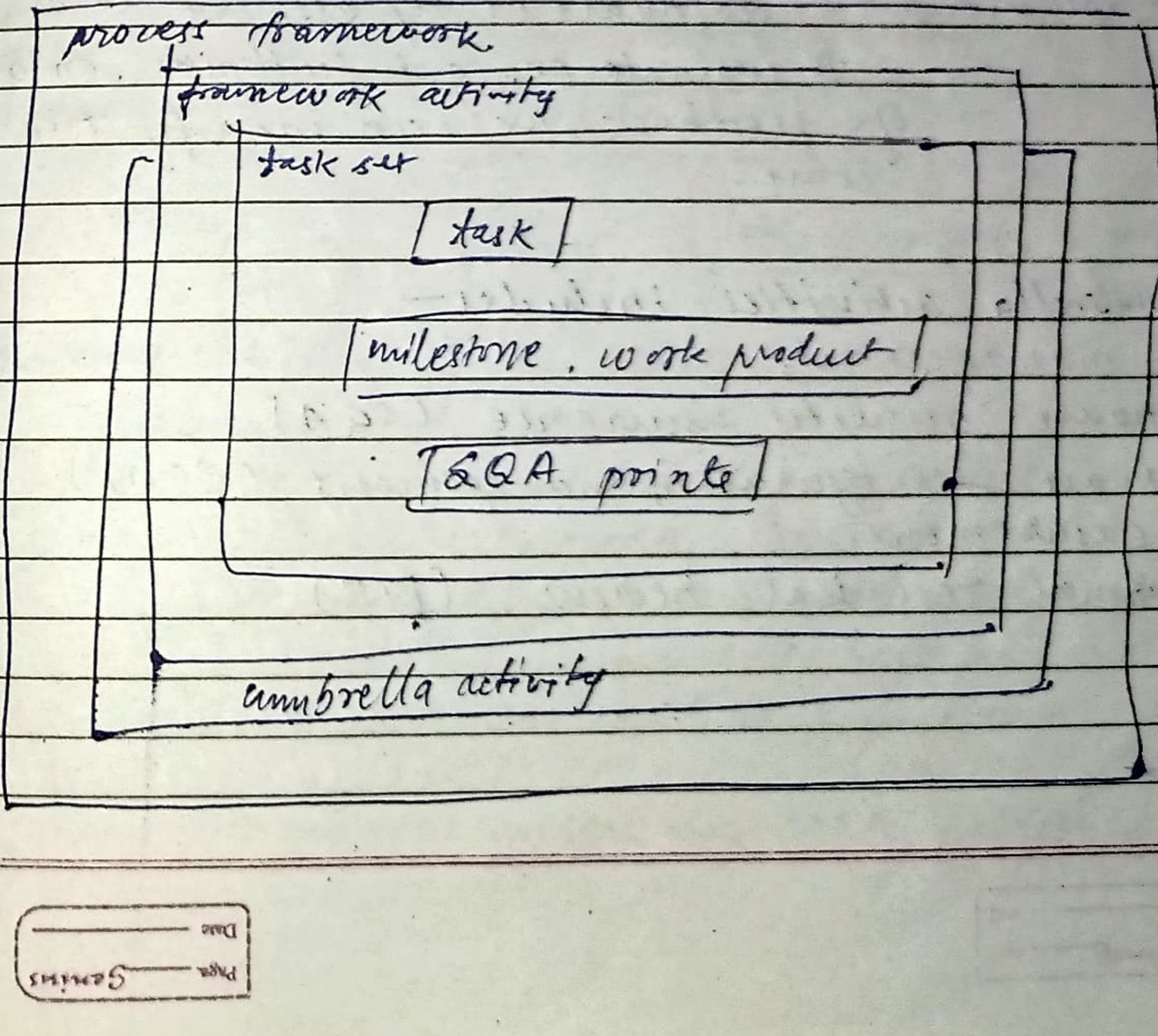
method is having all the information about each steps in process.

⊛ tools -

this tools provides a self operating system for process and ~~models~~ methods which means information created by one tool can be used by another.

## → SOFTWARE PROCESS FRAMEWORK

- is a standard way to build and deploy application it includes all the activities . also includes number of framework activities umbrella that are ~~capable~~ applicable for all software project.



activities—

1) communication:- communication between customer and other stake holders. as well as requirement gathering is done.

2) planning - discussion about techinigal related tasks, work schedule, resources.

3) modelling - building representation of things in real world to get better understanding of requirements.

4) construction - generate the code → test the product and repeat in order to make better product.

5) Deployment - software is represented to customer to evaluate and get feedbacks, on basis of feedback we can modify the soft ware.

Umbrella activities includes—

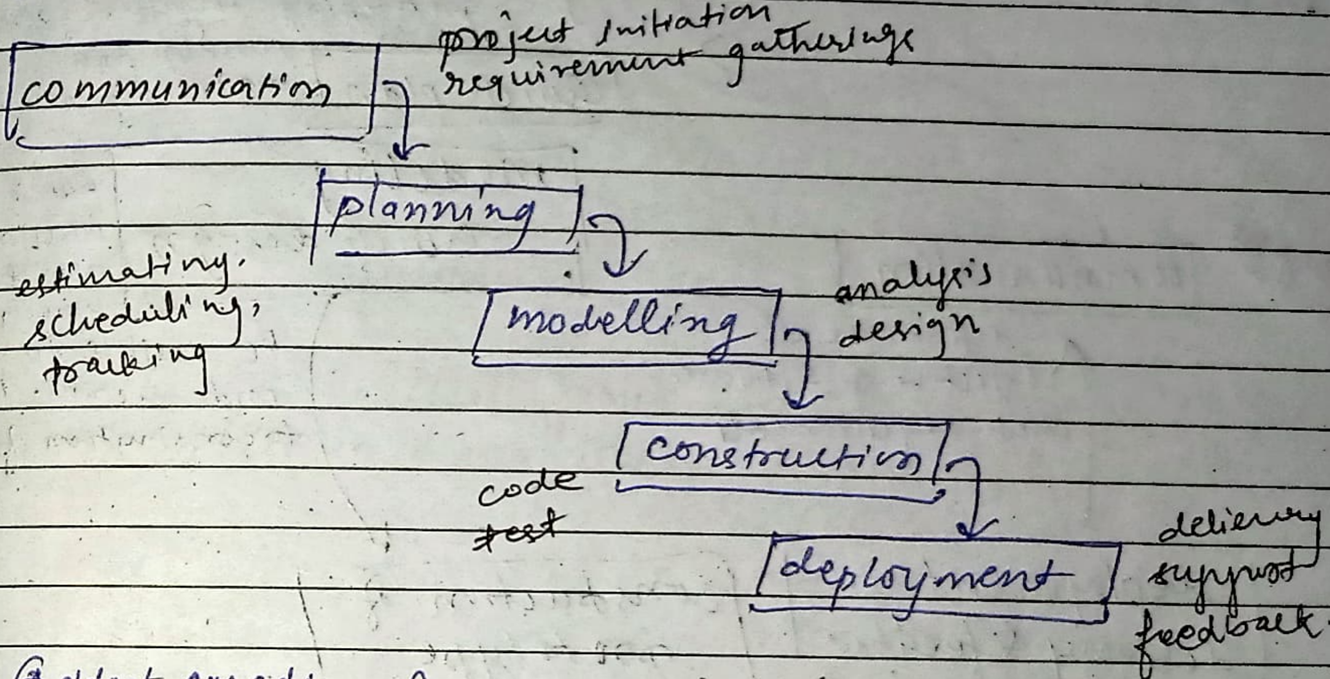
- risk management
- software quality assurance (SQA)
- software configuration management (SCM)
- measurement
- formal techinical review. (FTR)

## Process models:-

prescriptive process models define a predefined set of proces that can predict the process of work flow

### 1.) Waterfall model.

the waterflow model sometime called liners sequential model, that suggest a systematic sequential approach to software development that begins with customers.



Ⓐ oldest paradigm of software model.

problems were —

→ real project rarely follow sequential workflow that the model proposes.

→ often difficult for customer to state all the requirementts at beginning

→ customess need to ~~be~~ ~~patience~~ have patience

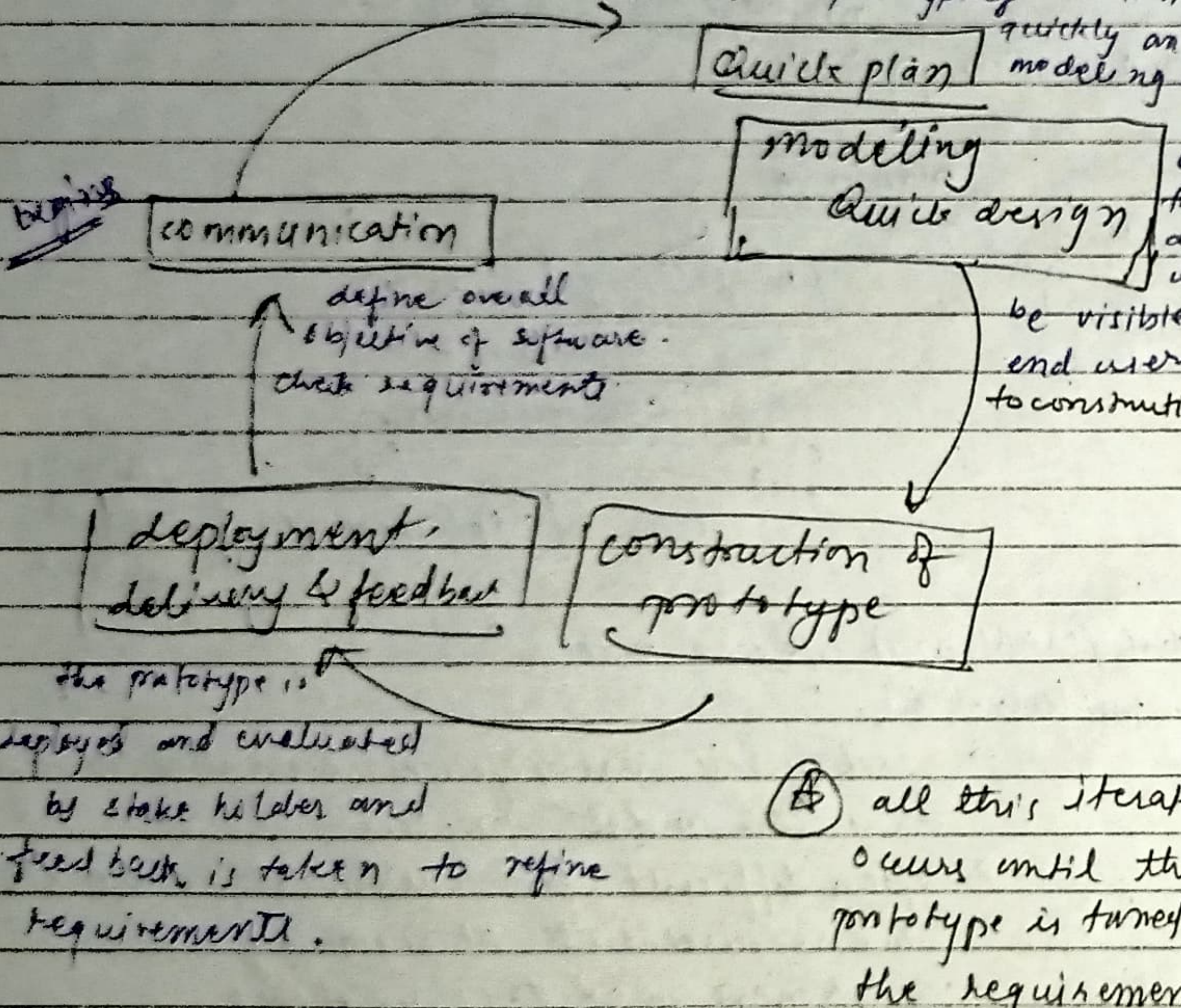
→ major blunders may not be detected

## 3) Prototyping Process Model.

often a customer define a set of general objective in software but doesnot identify the real requirements of function and feature.

By this the developer maybe ensure for the efficiency of algorithm. for this prototyping process model offers best approach.

prototyping paradigm assist you and other stakeholder to better understand what is to be built ~~as~~ when requirements are fuzzy.



prototyping iteration is planned quickly and modeling occurs

Quick design focus on the aspect which will be visible to end user. it lead to construction of proto

define overall objective of software. check requirements

the prototype is deployed and evaluated by stake holder and feed back is taken to refine requirements.

Ⓐ all this iteration occurs until the prototype is tuned satisfy the requirement of various stakeholder.

both stakeholder and engineers like prototyping paradigm. User get feel of actual system. developers get to build something immediately.

## (4) Spiral process model.

— Barry Boehm

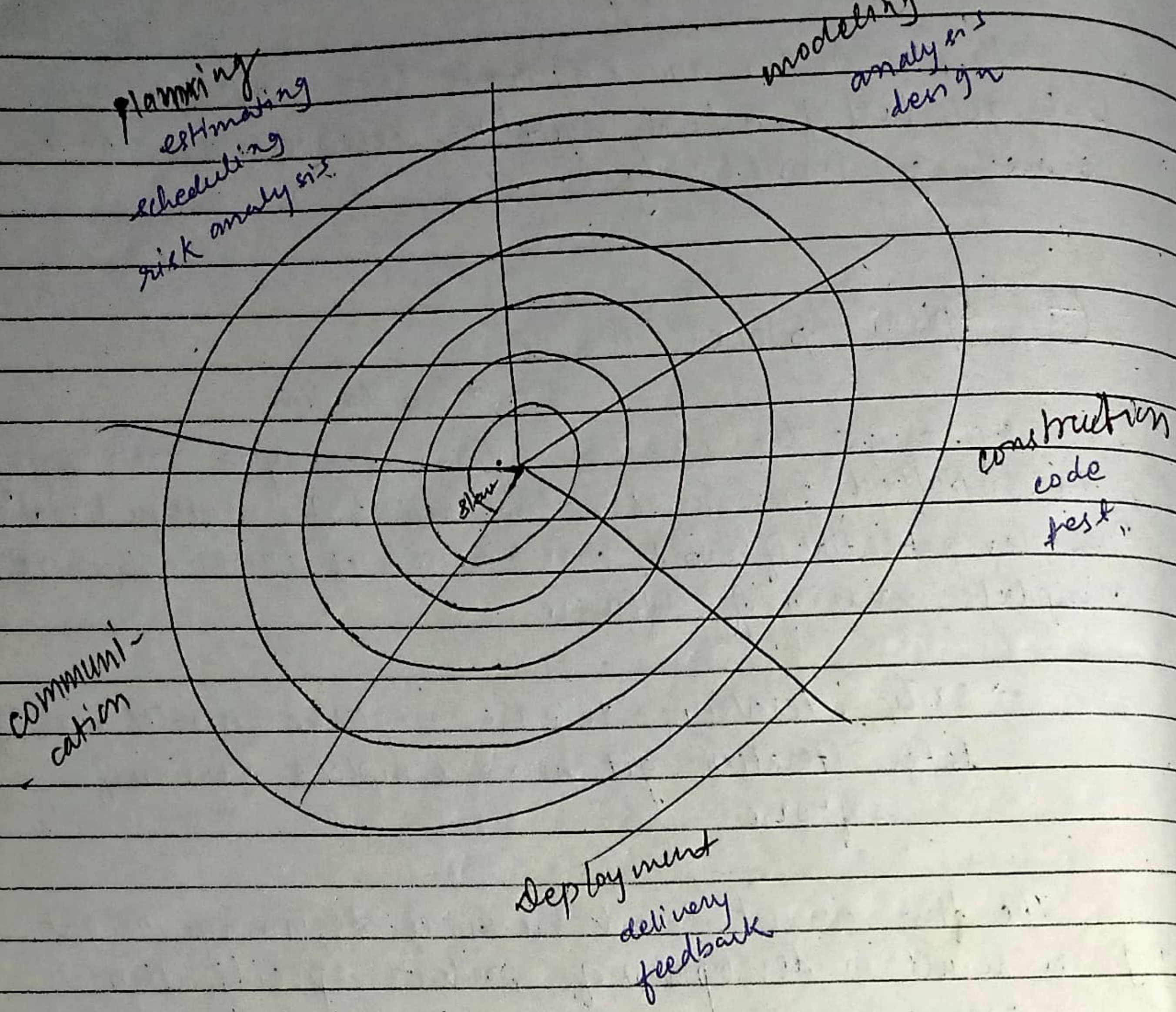
this couple the iterative nature of ~~progra~~ prototyping with controlled and systematic aspect of waterfall model. Provides potential of rapid development of increasingly more complete version of software.

→ early iteration - release might be model/prototype
later iteration - get more complete version of software.

The first circuit round the spiral beginning might be result in development of product specification. subsequent passes around the spiral might be used to develop. a prototype and then progressively more sophisticated version of software.

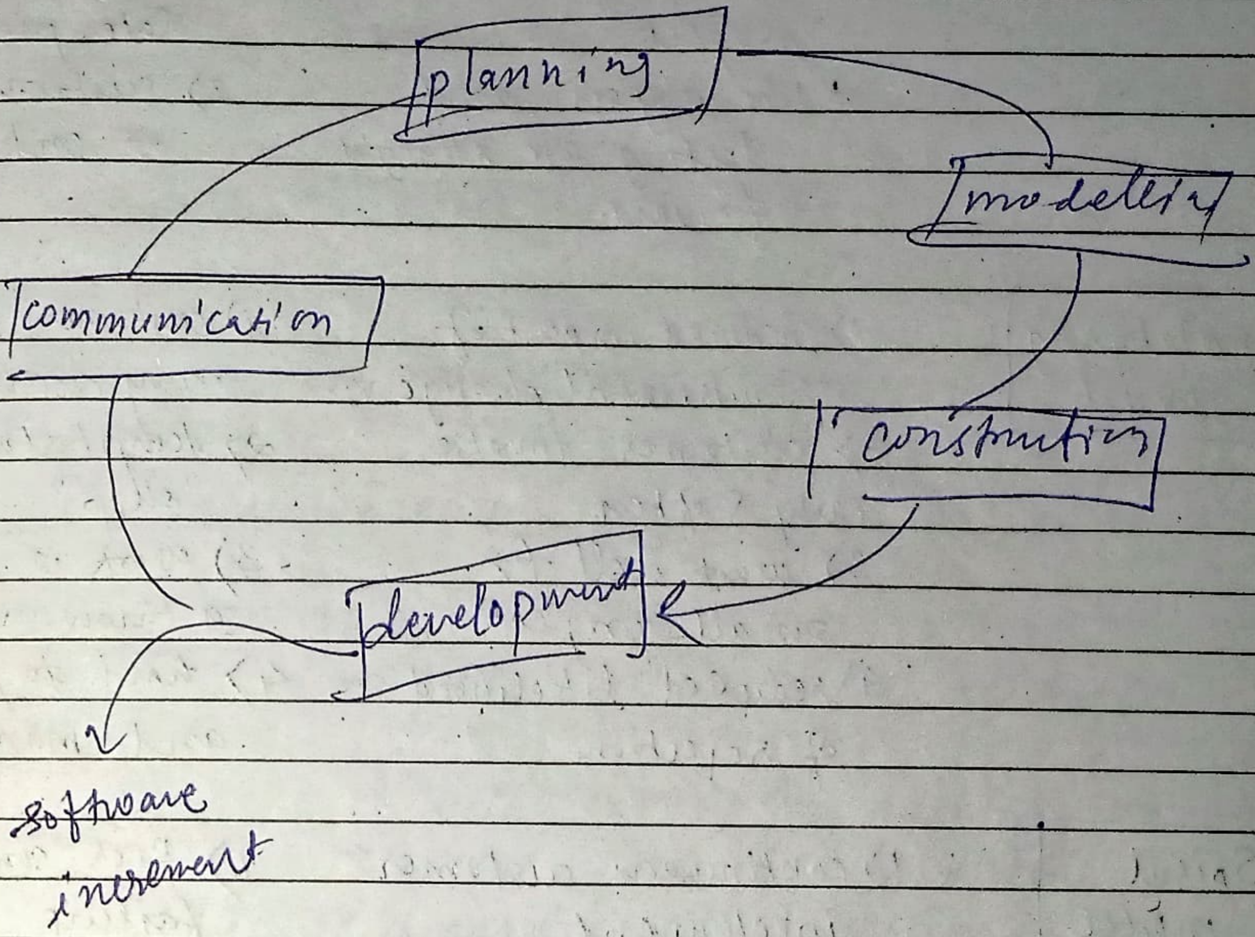
Cost and scheduled are adjusted based on feedback.

Unlike the other process model that end when software is delievered. spiral model can be applied throughout the life of computer software.


planning
estimating
scheduling
risk analysis
modeling
analysis
design
construction
code
test
communi-
cation
Start
Deployment
delivery
feedback

unified process model

it is to draw best feature and characteristic of traditional software process model but characterise them in a way



Pros & Cons.

| | Pros | Cons. |
|---|---|---|
| waterfall model | 1) easy to understand and plan<br>2) works for small project<br>3) analysis and testing are straight forward | 1) doesnot accomodate changes well<br>2) testing occurs in late process<br>3) customer approval at end |
| prototyping model | 1) reduced impact of requirement change<br>2) customer involve early & often<br>3) work well for small project<br>4) reduced likelihood of rejection | 1) customer involvment may cause delays.<br>2) temptation to "ship"<br>3) work is lost in a throwaway<br>4) hard to plan and manage |
| Spiral model | 1) continuous customer involvment<br>2) development risk are managed<br>3) it is suitable for large, complex project<br>4) It works well | 1) Risk analysis failure can doom the project<br>2) The project may be hard to manage<br>3) requires an expert development team |

| | | |
|---|---|---|
| unified process | 1) Quality document is emphesized<br>2) There is continuous process customer involvement<br>3) it accomodate requirement changes<br>4) works well for maintence project | 1) use cases are not always prec,<br>2) It has tricky software increment integration.<br>3) overlapping phase cas cause prob.<br>4) It require an development team |

## 6 capability maturity model integration (cmmi)

this is successor of CMM

objective of CMMI :-

→ fullfilling customers need and expectation.

→ value creation of investors / stakeholders

→ market growth is increased.

→ improved quality . enhanced reputation.

there are two type of representation in CMMI

- staged representation - uses pre-defined set of process areas to define improvement path.

  - each part in sequence serve as foundation of next.

  - improved path is defined by maturity level.

## continuous representation

- allow selection of specific process areas.
- uses capability levels that measure improvement of each process area.
- allow comparision between different organis-ation process on a process-area-by-process-area basis.

## CMMI Maturity level

1. Maturity level 1: initial.
   - process are poorly managed and controlled.
   - unpredictable outcome of process involved.
   - ad hoc and chaotic approach used.
   - No KPAs defined. (key process area).
   - lowest quality and highest risk.

2. maturity level 2: managed
   - requirement are managed.
   - processess are planned and controlled
   - projects are managed and implemented according to their documented plans.
   - risk involves lower then previous level
   - Quality is better than initial level.

3. Maturity level 3: Defined.
   - process are well characterised and described using standards. proper procedures and methods and tools.
   - focus in process standardization

## 4. Maturity level 4: Quantitavely managed

- process performance and quality are set.
- it is based on customer requirement. organization need.
- higherquality process achieved.
- lower risk.

## 5. Maturity level 5: optimizing.

- continuous improvement in process and their performanc
- Improvement has to be both incremental and innovative
- highest quality of process.
- lowest risk in processes and their performance

## CMMI Capability level

1. capabality level 0: incomplete

- incomplete process or not performed
- one or more specific goals of process area are not met
- no generic goals are specified for this level
- this capability level is same as maturity level 1

2. capability level 1: performed.

- process performance may not be stable
- objective of quality, cost and schedule may not met
- only a start-step for process improvement.

eg.

3. capability level 2: managed

- process is planned, monitored and controlled
- managing the process by ensuring objective are achieved.
- objectives are both model and other cost quality, schedule

4. capability level 3: Defined.

- a defined process is managed and meets the organisation's set of guideline and standard
- focus is process and standardization

5. capability level 4: Quantatively managed.
- process is controlled using statical and quantitative techniques
- process performance and quality is understood in statistical term and metrics
- quantitative objectives for process quality and performance are established.

6. capability level 5: optimizing
- focuses on continuous improving process performance.
- performace is improved in both way— incremental and innovation

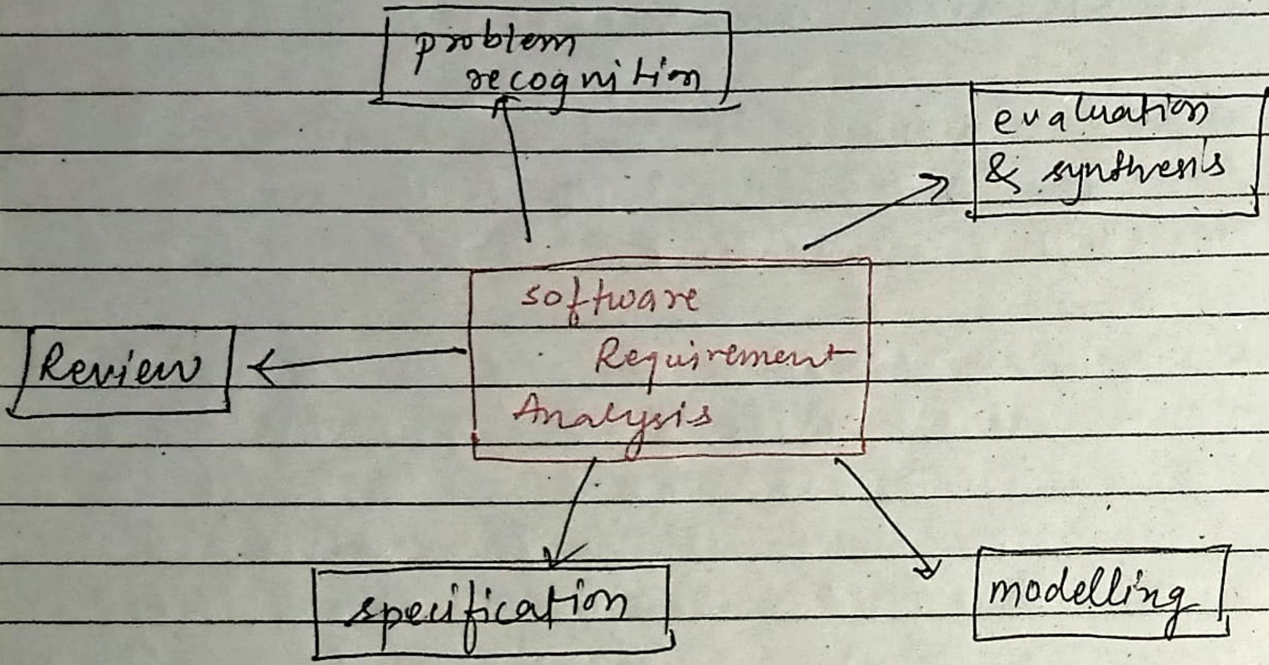
- emphasizes on studying the perfoomance result accross the organisation & ensure that common causes or issues are defined and fixed unit - 2

# Activities involved in software Requirement Analysis

software requirement analysis is necessary to increase the quality of software The requirements are expectation that needs to be fullfilled by software. Analysis here stands for examining all the possible outcomes.

↳ simply means complete study, analyze and describing software requirement that may needed to solve a problem.

⊛ there are several activities some are listed below-



⊛ problem recognition.

all the necessary things suchas. why it is needed, will it add value in software, benificial or not all these things are recognised in problem recognition. so that requirements can be fullfilled.

⊛ evalution & synthesis → to create or form something

↳ judgement ~~or analysis~~

some task that are important to evaluate -

- to define all necessary function
- data objects that are present and observable
- flow of data (worthy or not)
- overall behaviour.
- identify and discover constraints

⊛ modelling

after gathering all the information functional and behaviourial model are established using domain model also known as conceptual model.

⊛ specification-

the software requirement specification (SRS) which means to specify the requirement whether it is functional or non-function should be developed.

5) Review :- After developing the SRS, it must be reviewed to check whether it can be improved or not and must be refined to increase make it better and qualit

## Requirement engineering process

The process of gathering information related to software requirement from client and document them is known as requirement engineering

The goal of requirement engineering is to develop and maintain complex 'System Requirement specification' document.

Requirement engineering process-

1) feasibility study
2) requirement gathering
3) software requirement specification
4) software requirement validation

### 1) feasibility study

the process of analyze and detailed study about all the function and feature that is expected from software is studied in 'feasible study' process. feasible basically means "things that can be easily done":

### 2) Requisement gathering

If feasible study is positive then previous information and feature (function) are discussed in detail.

## 3) Software Requirement Speification (SRS)

-- SRS is a document that is created by system analyst after the requirement are collected by various stake holders.

- SRS defines , speed of operation, response times, software interaction with hardware, external-internal interface speed of operation, maintainnability, security, quality, speedy recovery from crashes.

- requirements are recieved in neutral language

↳ ~~system analyst~~ responsibility → to document in technical language

sothat they can be used and comprehened by software development team

Features :- → technical requirements are expressed in structured language

→ design description should be written in Psuedo code

→ formats of forms and GUI screen prints

→ conditional and mathematical notation for DFD's (Data flow Diagram)

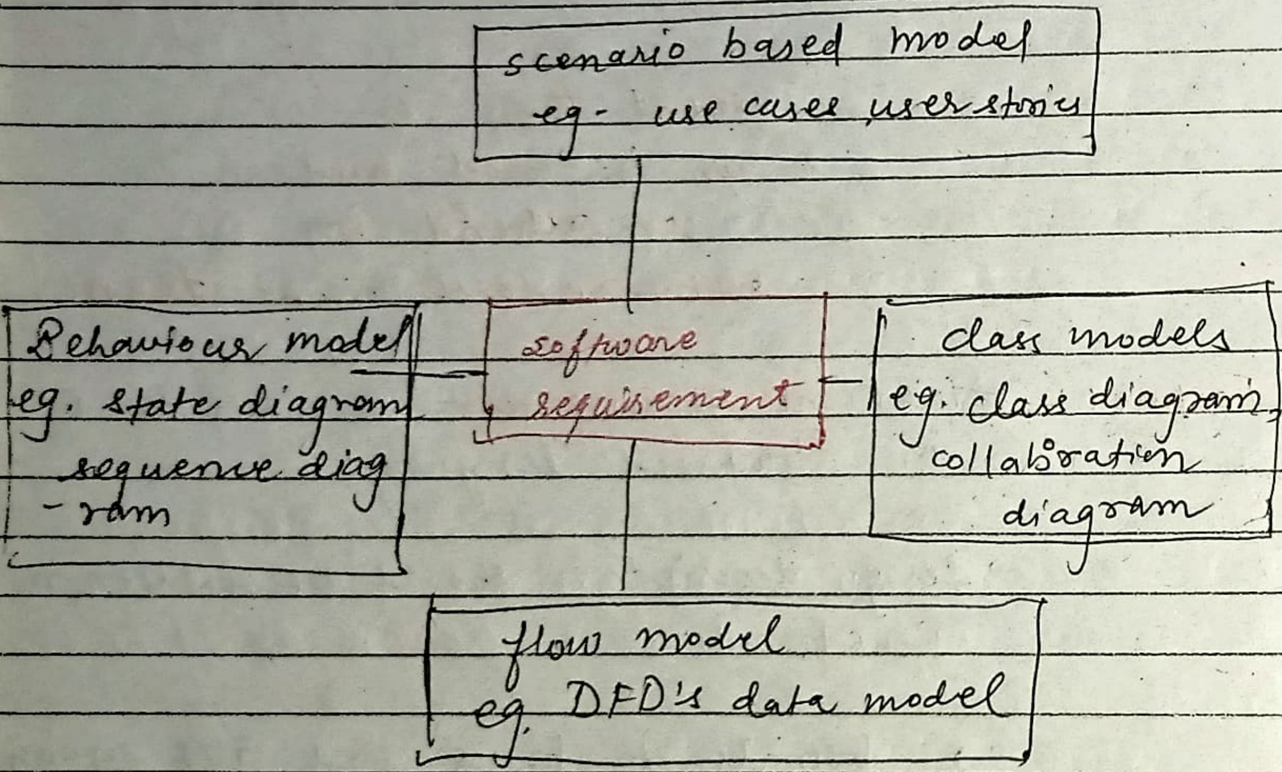
## 4) Software Requirement Validation

requirement mentioned in SRS needs to be validate to avoid any kind of problems.

points to be check. -

1) if they can be practically implemented
2) if valid according to functionality & domain of software
3) if there are any ambiguities
4) if they are complete and can be demonstrated.

### Element of Requirement Analysis.



UML

unified modelling language

as name specifies this the language of documenting models. provides set of basic graphical notation
UML is not a system design or development methodology it self.

UML was developed to standardise the large number of object oriented modelling notations that are existed previously.
UML was developed by object management group in 1997.

What is a model?

A model is an abstraction of a real problem (or situation) and is constructed by leaving out unnecessary detail and makes it easy to understand the problem or situation.
This is a simplified version of real system

Why construct a model?

Because it helps to manage the complexity.

Why we need for SRS;

An SRS forms the basis of an organisation's entire project. It set's out the framework that all the development will ~~flow~~ follow. It provides critical information to all teams, including development, operation quality assurance & maintainence. ensuring the teams are in agreement.

Characteristics and components of good SRS.

1) correctness :- user review is used to ensure that the correctness of requirement stated in SRS, SRS is said to be correct if it covers all the requirement that are actually expected from the system.

2) completeness :- covering all functional and non functional requirements of an software indicates completion of SRS.

3) consistency - it'll be only consistent if there to is no clashing between two requirements.

4) unambigousness - SRS is said to be unambiguous if all the requirement stated have only 1 interpretation. some ways to prevent it is using ER model, reviews, etc

5) Ranking for importance and stability - there should be some criteria, so that we can classify which requirement is more important than others as desirable or essential.

6) modifiability - SRS should be easily modifiable and should i capable of easily accepting any changes to

to system In some extent.

7) verifiability → A SRS should be verifiable if there exist special technique to measure every requirement is met by the system or not.

8) Design Independency → there should be option for choosing multiple design alternative for final system

## unit - 3) Software Project Management

Project Estimation Techniques.

- Different parameters that are estimated includes-
  - project size
  - effort required to complete project
  - project duration
  - cost

these helps in basis for resource planning and scheduling

there are 3 catagories —

- Empirical Estimation Technique
- Heuristic technique
- Analytical estimation technique.

1) Emperical Estimation Technique—

→ This is based on educated guess of project parameters, the prior experience with development of similar product is quiet helpful.

→ EET is based on common sense & subjective decision there two technique for this,

1) expert judgement.

2) Delphi technique.

2) Heuristic Techniques—

assumes the relation among the different project parameter can be satisfactorially modelled with suitable mathematical expression.

once the basic parameters (independent) found other dependent also be found.

heuristic model divided into two categories

1) single variable.

2) multi variable.

i) Singe variable — assumes that various project can be predicted based on a single previously estimated characteristic basic (independent) characteristic of software such as size.

dependent para.

exp → estimated parameter $= c_1 \times e^{d_1}$

where e represent characteristic of the software that has already been estimated (independent variable). $c_1$ and $d_1$ are constant they are determined from past data.

estimated parameter can be effort, project duration, staff size etc.

The COCOMO model is an example of single variable cost estimation model.

ii) multi-variable cost estimation model

—assume that parameters can be predicted based on more than one inde-pendent variable.

expression—

estimated process

$= c_1 \times p^{d_1} + c_2 p^{d_2} \ldots$

where $P_1, P_2$ are basic independent characteristic and $c_1, c_2, d_1, d_2$ are constants.

multivariable estimation model are expected to give more accurate estimates compared to single variable estimation model.

· The intermediate COCOMO model is an example of multi variable cost estimation model.

3) Analytical Estimation Technique -
Derive the required result starting with basic assumption (certain) regarding a project. It does have scientific basis. Healstead's software science is an example of Analytical Estimation Technique.

## COCOMO MODEL

Constructive Cost estimation Model was proposed by Boehm in 1981.

- cocomo prescribe a 3 stage process for project estimation
- in first stage a initial estimate is arrived.
- in other two stages initial estimate is refined to obtain more accurate estimation.
- cocomo uses both single and multi-variable estimation model at different age

Three stages are 1) basic cocomo
2) intermediate cocomo
3) complete COCOMO.

# Basic COCOMO model.

any software development can be classified into 3 categaries based on their development complexity.

1. organic
2. semi detached
3. embedded:

3 basic classes of software development project -

1. data processing programs → application program
2. compilers, linkers etc. → utility program
3. operating system & real time system program } → system programs.

Brooks [1975] states

difficulty level of program

utility = 3 * application & system = 3 * utility.

# Definations →

1) Organic - if the project deals with developing a well understood application program. The team members are experienced in developing similar projects.

2) Semi detached - if development team consist of mixture of well experienced and inexperienced staff.

3) embedded - if software being developed is highly coupled with ~~sof.~~ hardware. or if stringent regulation on procedures exist.

## Person Month -

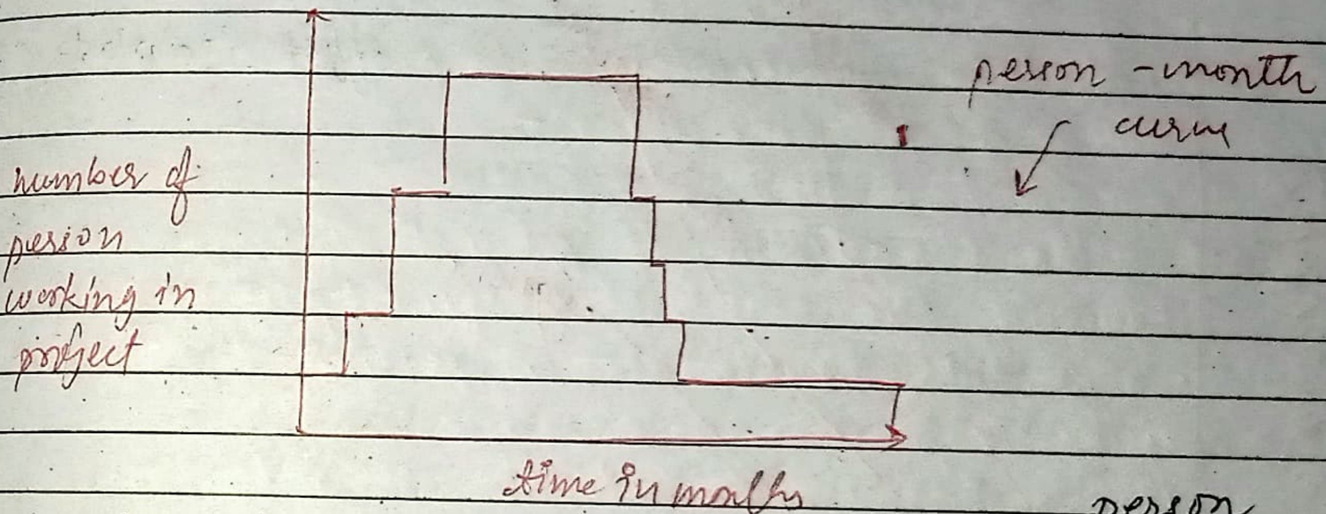
person-month (PM) is unit for measuring effort.

person month is considered to be an appropriate measurement unit for measuring effort, because developers are typically assigned to a project for a certain number of month.

doesn't mean
100 pm x → 100 person should work for 1 month

neither. it mean 1 person should work for 100 month.

The effort estimation simply denotes the area under the person-month curve for project.



this shows that different number of ~~personal~~ person can work at different points in project development.

#general form of COCOMO expression →

$$\text{efforts} = a_1 \times (KLOC)^{a_2}\ PM$$

$$T_{dev} = b_1 \times (\text{efforts})^{b_2}\ \text{months.}$$

here KLOC → kilo lines of code ← (every line of [illegible] text considered as 1LOC)

$a_1 a_2\ b_1 b_2$ → constants for each category of software

Tdev → is estimated time to develope s/w

Estimation of Development effort :-

organic - effort = $2.4(KLOC)^{1.05}$ PM

semi-detached : effort = $3.0(KLOC)^{1.12}$ PM

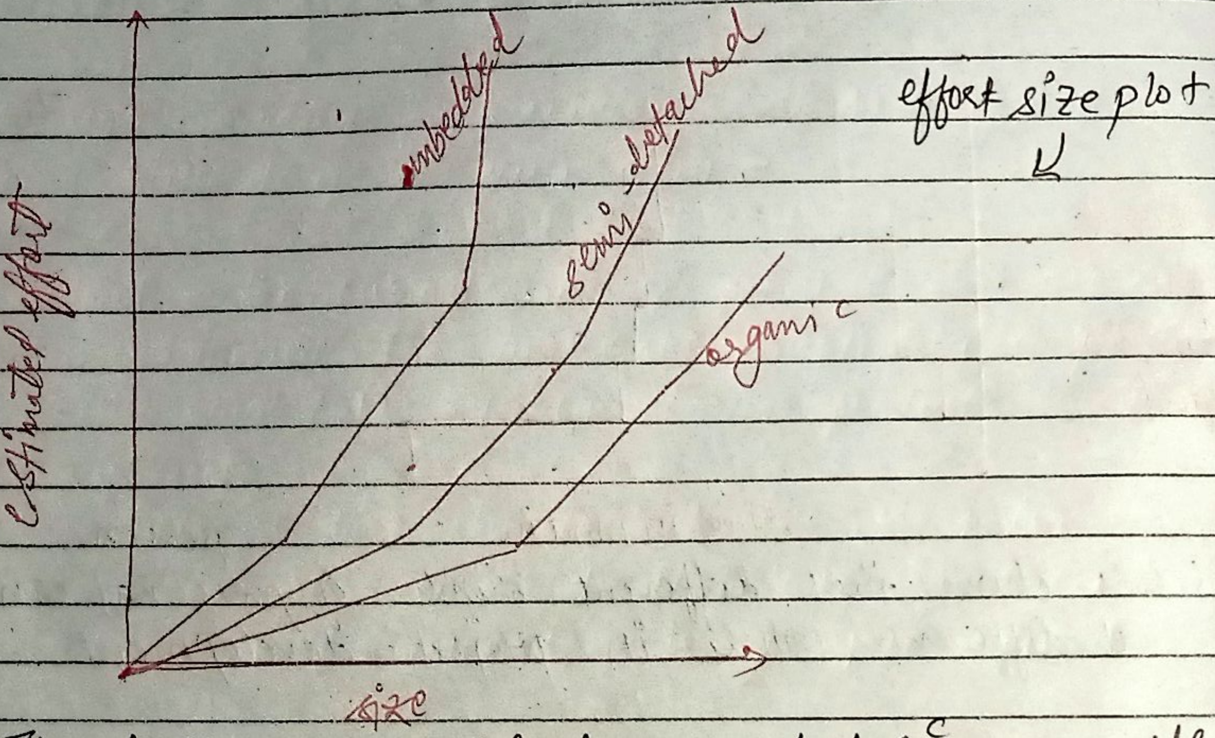
embedded : effort = $3.6(KLOC)^{1.20}$ PM

Estimation of Development time :-

organic : Tdev = $2.5(efforts)^{0.38}$ months

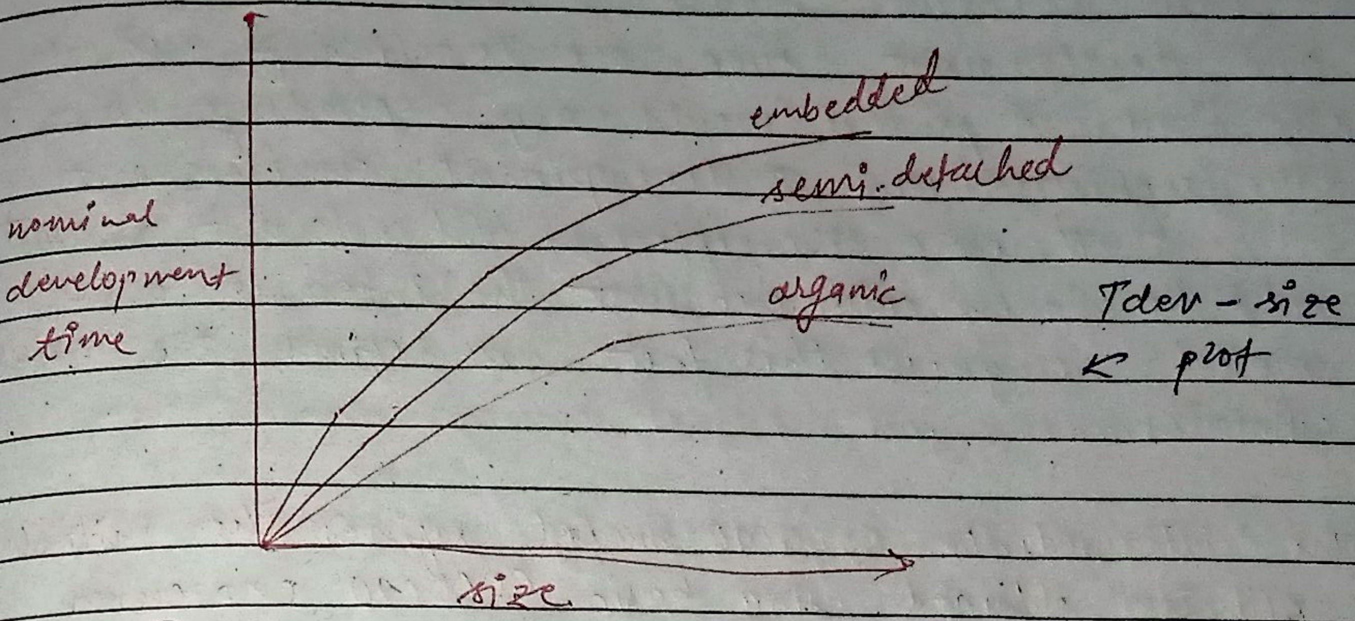
semi-detached : Tdev = $2.5(effort)^{0.35}$ month

embedded : Tdev = $2.5(effort)^{0.32}$ month



The effort sequired to develop a product increases rapidly with project size. However increase in effort with size is not bad this is due to COCOMO model.

Observation from development time - size plot



⊛ the size of product increase by two times. the time to develop the product does not double but it sises moderately.

⊛ The development time is roughly the same for all three catagories.

Problem 3.2 page No. 138.

estimated formula for organic software

$$effort = 2.4 \times (32)^{1.05} = 91 \text{ PM}$$

$$Tdev = 2.5 (91)^{0.38} = 14 \text{ month}$$

$$cost = 91 \times 15000 = 1,465,000$$

# Intermediate COCOMO MODEL

The basic COCOMO model assume that effort and development time are function of product size alone. A product would vary depending up on the sophistication of development environment. Therefore the effect of all relevant parameter must be taken into account. the Intermediate COCOMO model recognises this fact and refines the initial estimates.

The intermediate COCOMO model refines the initial estimates obtained using basic COCOMO expression by scaling the estimate up or down based on the ~~ve~~ evaluation of set of attributes of software development.

For each grading of parameter Boehm suggested as being attribute of following items:-

1) product:- The characteristic of product that are considered include the inherent complexity of product, reliability requirement of product

2) computer:- characteristic of the computer that are considered to include the execution speed required, storage space required, etc.

3) Personnel :- The attribute of development personnel that are considered include the experience level of personnel, their programming capability, analysis capability etc.

people employed in an organization

4) Development Environment·

this attibute capture the develepoient facilities available to the developers.

# complete COCOMO model

~~A~~ major short coming of both basic and intermediate COCOMO model is that they consider a software product as a single homogenous entity. But most large system are made up of several smaller sub system.

Some subsystem may be considered as organic type. some semi-detached, some embedded some subsysteme reliability requirements may high, for some development team might have no prior experience.

Let's take a example of distributed MIS ( management information system) product and They have offices at several places in country. and have following components —

1. Database part ← semi-detached s/w.
2. Graphical user interface (GUI) part ← Organic s/w
3. communication part. ← embedded s/w

all these 3 component can be estimated separately ~~fo~~ & summed up to give the overall cost of the system.

# # SCHEDULING

scheduling the project is an important project planning activity. after this Project manager monitors the timely completion of task and takes any corrective action that may be necessary whenever there is a chance of schedule slippage.

To schedule project, project manager needs to do –

1. Identify all the major activity that need to be carried out for project
2. Break down each activity into task.
3. Determine dependency among the different task.
4. Establish the estimates for the time duration necessary to complete the task.
5. represent the information in form of an activity network.
6. Determine Task start and end dates from information presented in activity network.
7. Determine the critical path. A critical path is chain of task that determines the duration of project.
8. Allocates resources to task. (the activities are broken down into smaller logical set of

A project manager break down the task systematically by using the work down structure technique.

After the task has been broken, Project manager needs to find dependency among task this determines the order in which the different task would be carried out. for example if task A requise result of task B, then task B should be ~~sho~~ scheduled first. then we can say that A is dependent on B.

The task dependency define a partial ordering among task.

smaller activity).