Software Engineering ①

Software is a set of instructions, data or program used to operate computer and execute specific tasks

Engineering : Kuch Rule and principles followed to develop a software

A systematic, organized approach to providing necessary function in a project at the lowest costs.

(1) Introduction (2) Software development life Cycle
(3) Requirement Analysis (SRS) (4) Software Project Manag (COCOMO)
(5) Software Design (Coupling, Cohesion, UML, DFD class diagram)
(6) Coding & Testing
(7) Maintenance ← (Provide Technical Support Particular Period time)
(8) Quality Magnet, Reuse

What is Software Definition and Evolution

* It is systematic, disciplined, cost effective techniques for software development.

{* Engineering approach to develop a software}

{Puri Approach How to Software develop step by step: Step 1 → Idea [2] → requirement [3] → Physibility [4] → Planing [5] → Implementation & design [6] → Testing [7] → Maintenance [8] → Product deliver}

That all about of Software Engineering.

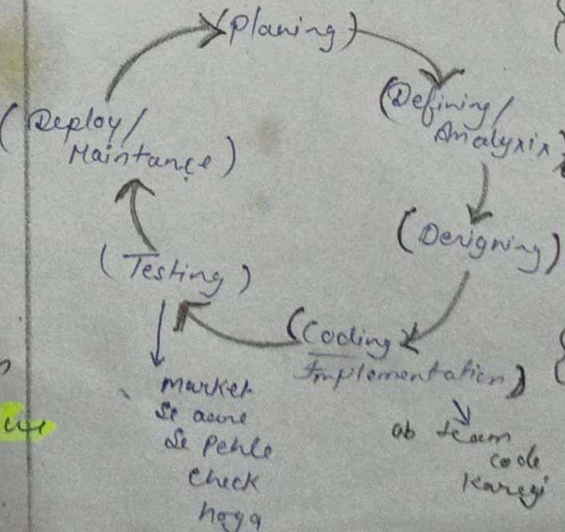
Single step to ek phase. {Probability failure kam hota hai}

Evolution

- 1945 – 65 → Origin (Starting)
- 1965 – 85 → Crisis (Simple build & deliver, Non-brand Approach)
- 1990 – 2000 → Internet (Window) → Search engine
- 2000 – 2010 → Light weight (Mobile par bane software)
- 2010 → Till → AI, ML, DL

actul 2% Soft. use hue sb stretch 98% (OS/360) IBM. Koi design tiyar nahi kiya to fail hogye

Software Development Life Cycle (SDLC)



{Planing}: Mind mein Idea (Initial Step) and developer ne pitch kiya (owner) ye hojayega ap ka kaam.

{Defining/Analysis}: ab wo smjh gya hu (service provider) ye requirement ye cost hogi and team ko bhi smjh diya puri estimation complete (Negotiation complete) then (SRS) in written.

{Design}: Kaun se func hai (Cuplity Cohesion, DFD) frontend, backend blueprint type, coding wale ko help karta hai...

{Classical Waterfall Model} (Regid Model) → fixed hogyi functionality

* **Feasibility Study** : financial term mein budget hai (feasible hai ya nahi)
technical team ho project banne keliye bna skte ya nahi (feasible hai Not feasible)

(2)

↓

* **Requirment Analysis and Specification** : estimation kareng kiya team, kiya resource and (SRS) doc & go to Next Phase

↓

* **Design** : → * **Coding and unit testing** :- Each single unit testing

↓

* **System testing and Integration** :- Each and small unit ko test karne ke bad alpha testing, beta, acceptance testing. Puri module function check karna.

↓

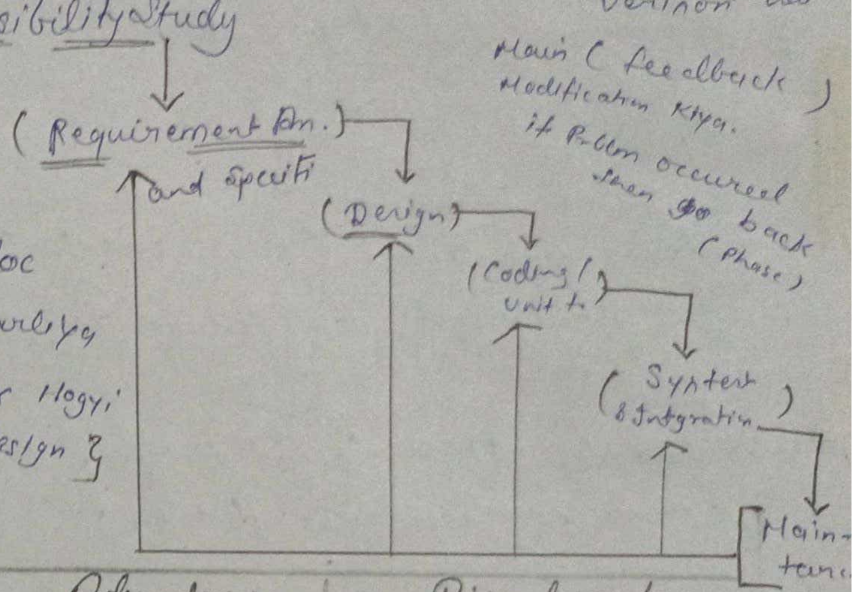
* **Maintenance**
Sare efforts (Problem) achi techniqual & Non technical sab maintenance fix karna hai.

| Back Phase mein nai jayga.

Advantages	Disadvantages
* Basic Model	* No feedback (No backtracking)
* Simple and easy	* No Experiment
* Small Projects	* No Parallelism
	* High Risk (flexibility nahi hai)
	* 60% Efforts Maintenance

{Iterative Waterfall Model}

classical modify version hai (feedback modification kiya. if problem occured then go back (phase))



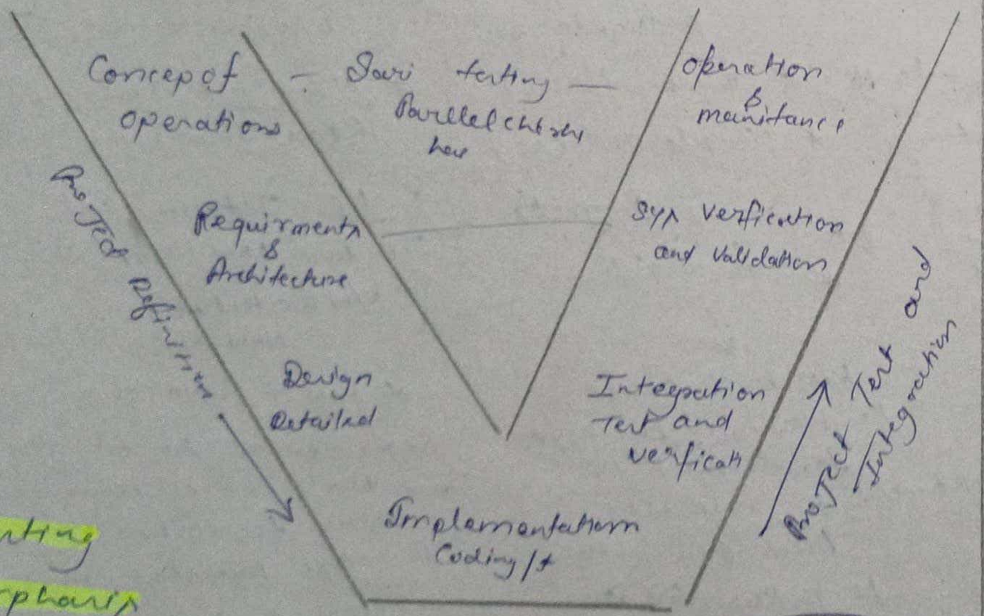
{SRS doc sign karliya to fix hogyi usaki design}

Advantages	Disadvantages
* Base Model	* No Phase overlapping (Parallism work nahi hai)
* Simple and Easy	* No Intermidiate Deliver (Jab pure kam hojyne pe deliver, and customer in b/w partial delivery nahi hai)
* Small Projects	* Rigid (No chang) Requirement ko fix kardiya and next phase mein change karna to nahi hogi (Prototype nahi hai yha)
* feedback ↓ kisi back phase ki wajhse bug aaye to feeback ki help se back phase mein bhej kar fix then continue work kareng.	* less customer interaction

V-Shaded Model

(step by step checking — full checking sys)

* Also known as Verification & Validation Model
* Extension of Waterfall Model
* Testing is associated with every phase of lifecycle
* Verification Phase (Requirement Analysis, System design (Puri building), architecture design (Har floor), Module design (uske fune))
* Validation Phase (unit testing, Integration, System, Acceptance testing)



Testing emphasis hoti har phase mein karta

(reactive tracking banne ke bd testing.)

Advantages

* Time saving
* Good understanding of project in the beginning
* Every component must be testable
* Progress can be tracked easily
* Proactive defect tracking (cheez banne ke sath checking hoti hai)

Disadvantages

* No feedback so less scope of changes
* Risk Analysis Not done
* Not good for object oriented projects

Chote project kelie theeg (library, Hospital, Bus Management sys par Implement kar skte) V model

(dummy) Prototyping Model

Customer ke point view sunkar ek demo structure prepared (Prototype) bna liya.

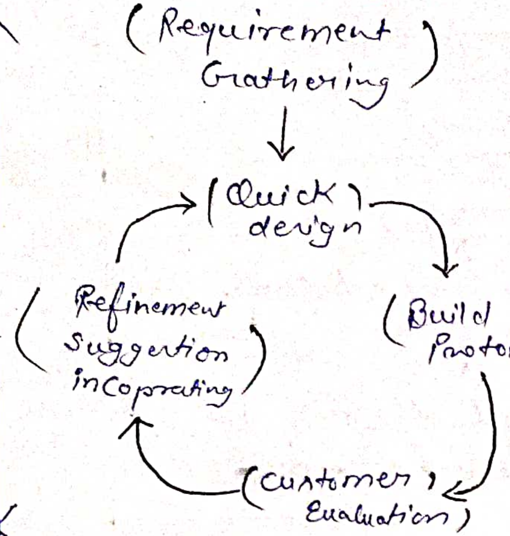
Quick design: Roughly design banya customer ki requirement ke hishab se (Toy) model

Build: fir uski prototype customer ko present kiya

Customer: feedback liya OK & Not OK

Refinement suggestion Incorporating;

Prototype Model



(*) customer Not clear Idea

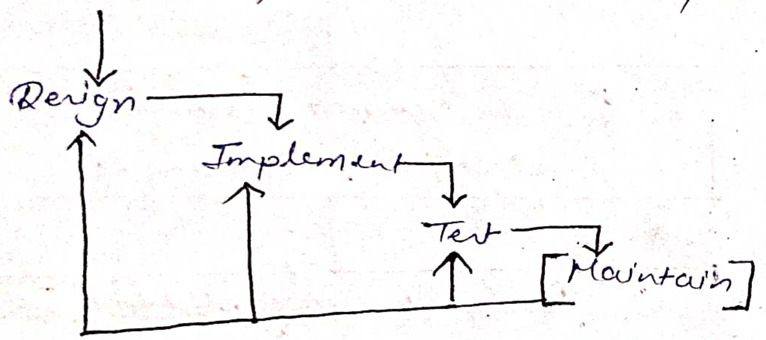
(*) Throwaway Model

(*) good for technical and requirment risks

{customer ko Apne suggestion Point of view dekar ek (Toy) demo model bna kar de skte}

(*) Increase in Cost of development.

Iterative Model



(4) the desired system is fully realized. No long term plans are made

* Iterative model main advantage is its feedback process in every phase.

* Also known as "Design a little, build, test a little, deploy a little model".

Sbse Pehle SRS document bna kar freeze karke fir bnate hai. Isme esa Nahi Hota Kiyun customer ki requirment / technolgy mein changes arhye ho. To plan ko estimate ko change karenge.

* Iterative: har step ke bad feedback lenge.

Advantages	Disadvantages
* Customer requirment are clearly specified.	Not suitable for smaller projects
* Risk Analysis is better	Cost
* Supports changing environment	High skill resources are required
* Initial operating time is less.	
* Better suited for large mission - critical projects	

Evolutionary Model

(A) → (A)B → ((A)B)C
deliver → deliver → customer

* Evolution model is a combination of iterative and Incremental model of software development life cycle.

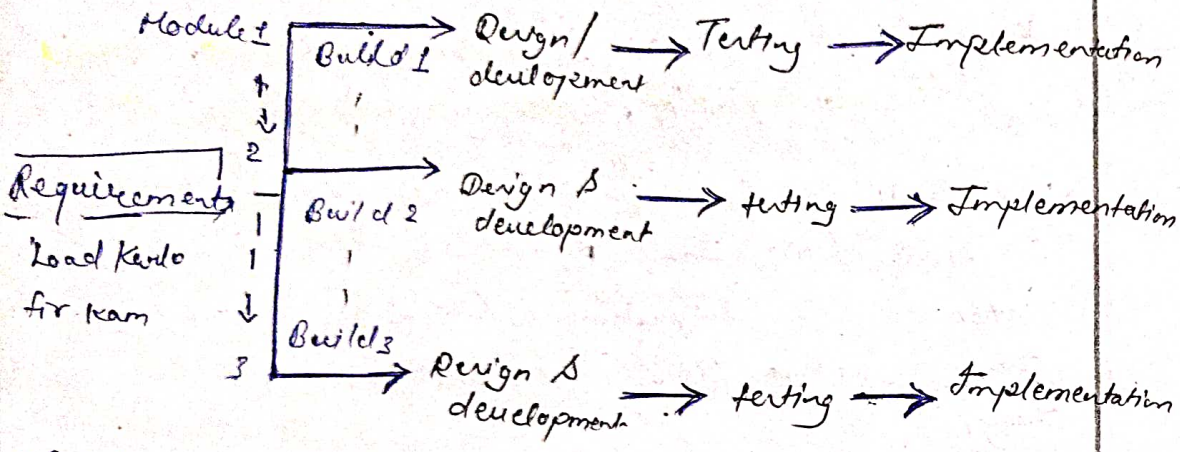
* Incremental model first implement a few basic features and deliver to the customer. Then build the next part and deliver it again and repeat this step until

Incremental Model

module by module by development kar thye hai

(A) (A)(B) (A)(B)(C)

{ Pura Product ek bar m Nahi banate hai todhi todha karke banate.

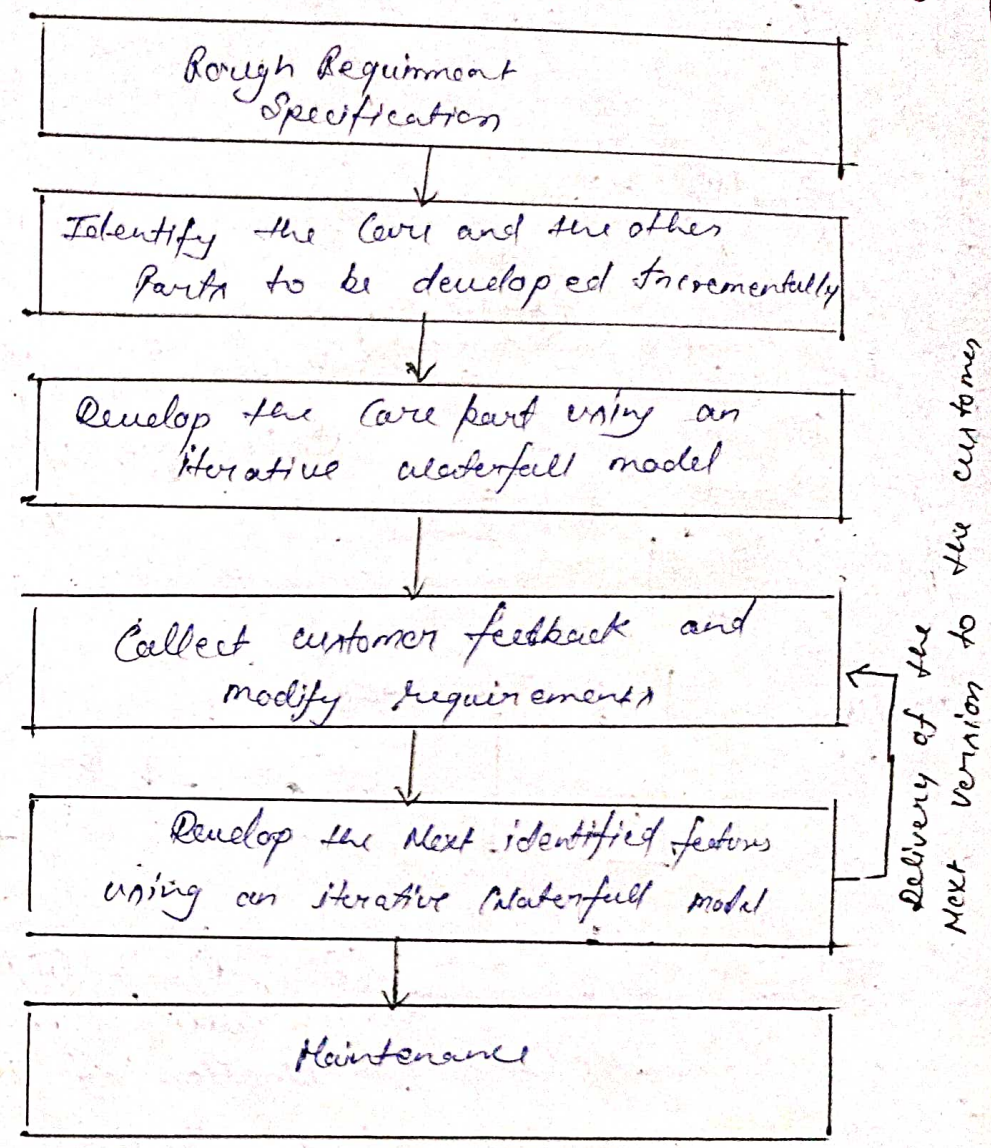


* Customer Interaction maximum (Kya add karni features / remove)
* Large projects
* Early Release Product Demand
* Flexible to changes.

Error ki chance bahut kam

* Requirement fix Karli (Locked) todha Paisa Kya ek Module Pe kam Or delivery 1 module then Next.

Eg → mobile Software Tym → To tye Update features ata hai



Evaluting Model

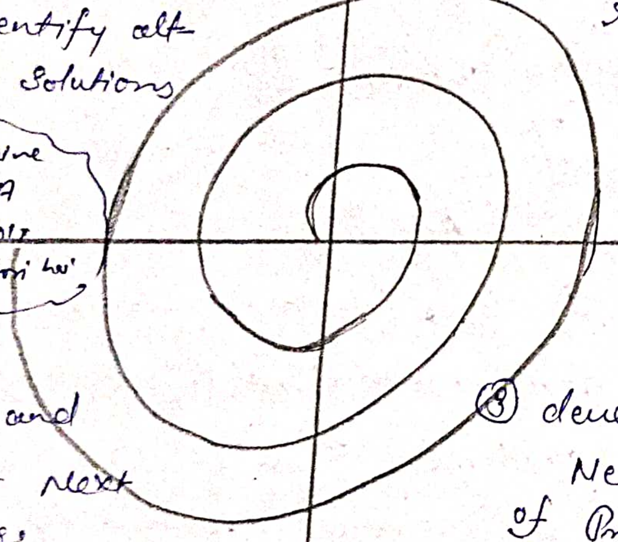
Spiral Model {Risk Analysis Predict Karna Ho}

① objective Determination and identify alternative Solutions

② Identify and resolve Risks

Kaunse risk aaske or unke sol. (Prototype Pe test karlo)

Large Project Mein use karte NASA jaha risk analysis karna bahut jaruri hai



④ Review and Plan for next Phase

③ develop Next Version of Product

refine – Contact – user-customer

mixture use kar sake sare model ke

* Risk Handling
* ↑ Radius of Spiral = Cost ↑
* Angular Dimension = Progress
* (Meta Model) multiple model ko use kar sake hai

Advantages

(1) Risks Handling
(2) Large Projects
(3) Flexible
(4) Customer Satisfaction

Disadvantages

(1) Complex
(2) Expensive
(3) Too much Risk Analysis
(4) Time

"Agile" (Move Quickly) currently use this model

Large Quickly → Small chunks (Iterations) ↓ Release ↓ feedback ↓ Enhance ↓ R-Release

Jo Complex Project use Small chunks Mein divide Kar diye or un Iterations Par Parallel work (develop) Kar diye, and Market Mein release Hoga ab and customer se feedback lenge based on feedback enhance (adapt) ye nahi cheez lani Padegi and fir se release karenge.

{Yha koi heirachal Typ team Nahi hoti sb Independent Hote hai}

Advantage

(1) frequently Delivery
Halke halke ke banye and release.

(2) face to face Communication with client.
(Sahi direction Mein hai ki Nahi)

(3) Changes karne Capability Jada hai

(4) Time kam lga Product aane mein

Disadvantage

(1) less documentation rather than time waste mat karo Jaldi se Jald Implement karo

(2) Maintenance Problem
Ek launch kardi Iterations customer ko dedi market mein run kar rhi and 2nd Iteration aayi to maintain Mein hoti Problem

Scrum :- one of the most popular agile Methodology

(6)

* Scrum is lightweight, iterative and incremental framework
  (iterative ↓ step by step)

* Scrum breaks down the development phases into stages or cycles called "Sprints".

Sprint :- time duration (maximum effort gya hai)

* The development time for each Sprint is maximized and dedicated.

Ek Sprint ko pehle khatm kiya Jata fir review karna stackholder ke sath or ye jo kam hua hai changes / ya Nahi or un review ko New sprint mein lekar aaina and design

* **Backlog** :- form hai jo basic requirement wo hme design karni hai wo mention ki hai (wo sab backlog mein design)

* **Daily Scrum** ; daily basis ki meeting 10-15 min developers and stackholder ki discussion about Product ye chal rha abhi

* (Equivalent sare Team Member) freely de skte Idea koi bhi team mem

Scrum Master :- Sari cheez ko Monitoring kar rha hai team ko handle and sath Stackholder ke sath discussion kar rha hai

Client → (Product owner)

[Advantages]

* freedom & Adaption
* Highly Quality, low risk Product
* Reduce development time up to 40%
* Scrum customer satisfaction is very Imp
* Reviewing the current sprint before moving to new one.

[Disadvantages]

* More efficient for small team size
* No changes in the sprint.

(No hierarchy)

N

Waterfall Classical	Iterative Waterfall	Prototype Model	Incremental Model	Evolutionary Model	RAD Mode	Spiral Model	Agile Model startup mein use
Basic, Same Model Yhi se hi Nikla hai Rigid, Inflexible, Not for Real Project	Basic, feedback dala After every Stage feedback given to the Previous stage Problem in well understood. clear hai → customer ko Apna Idea user ko Pta kiya chiye	user requirement Not clear, Costly ↑, time ↑ No Early lock on requirement → High user Involvement → Reusability demo dikhta hai	Module by Module Delivery Easy to test and debug (A) (A) Requirement Lock	Large Project Yhn Pehle requirment lock Nahi Hoti Pehla Module banya to requirm lock AG Next Module banya fir Requ lock Karna Project Complete, and cost	Time and cost Constant user at all levels (Interaction Har level Par Hoti) → Reusability ↳ Module function 2,3 Month Mein Complete	Risk, What kind of risk estimate Not for Small Project No Early lock on requirment Less Experience on work every Stage pe risk Analysis	Flexible, Advanced Parallel Process divided into Sprints every Person working same level developed team

{ Software Requirements }

(*) It is the description of features and functionality of the target system

client Aaya → usne bola ye system banna hai (target sys)
us sys ki features & functionalites characteristics sb description tha likha jata hai (documentation lekar)

(*) Ye Apne Ap Mein (Engineering hai) matlb art hai lagna.
base of the system.

Proper cheezo se bana, define karo maintain karo kiyunki jo Hmara target system hai wo wesa hi bane jo Maine likhi same wesa hi.

1 (*) Sbse Pehle step Apka system banne lyak hai Ya Nahi (feasibility study)

2) (*) Sb cheezo ko gather karo → client se, stack holder Company beth kar brainstorming karo kiya kaise Hoga or ye kaise banegi (Information Gathering)
(different interviews, meeting, survey, conference)
Sb karliya (requirment gathering / Elicitation)

(3) → Fir hume likha (SRS document)!-
Properly likh liya document bna liya

(4) (Software Requirment Validation):- ab check karna us doc ko finally Jise signature Hogyi and design Phase Mein ya Next Phase Mein kam ho ske

(Software Management bhi bolte kiyunki Manage karna Company and user ko Communication ko b/w user/requirement Gathering

Tool Support for Requirements Engineering

Ye Jo Sari requirement Gather Karoge uske liye Kuch Tool chiye To wo ye hai

* Observation Reports (user observations). user kiya chata use observe
* Question (interviews, surveys and polls)
* Use cases (Puri cheezo se use Karo)
* User Stories
* Requirement workshop
* Mind Mapping → bahut 3rd diagram bna Kar Explain Karna Mtlb theory Pe do
* Role Playing
* Prototyping.

Functional Vs Non-functional Requirements

* Requirement which are related to functional / working aspect of software fall into this categories (functioning Kaise Hogi working Mein)

* Non-functional Requirements are expected characteristics of target Software. (Security, Storage, Configuration, Performance, Cost, Interoperability, flexible, Disaster recovery, Accessibility).

Non-funct Crash testing means real life ex Cars Is cheez se takrai To us case mein kiya hoga kiya feature hogi

Security check lagna Padega.

Storage mtlb ki kitna fast read/write hoga dbms mein response tym

Configuration kis Type ka database use kar rhye hai (server latest) speed.

Cost effective ho, Interoperability ⇒ mtlb sys humra dusre sys se Communicate ki Nahi kar pa rhye hai to aa-pas mein Comm bahut Imp hai

Flexibility Ho → means agr 100 user hai feature 1k to 1M Ho. (user based) ko support

Disaster recovery :- Outage agya to Na aye (becoz trust and Credibility / ~~[illegible]~~ form loss hai)

Accessibility :- Valid user hi Access kar rhe chai Yo koi or To kaise hum accessibility Provide kar rhye hai

Software Requirement Specifications (SRS)

* SRS is a **description** of a software system to be developed.

(9)

* it lays out **functional** and **Non-functional** requirements of the software to be developed

* It may included a set of **use cases** that describes users **interactions** that the software must provide to the user for perfect interactions.

(team mein consistency banne aye hai kam keliye

Use Cases :- means student password bhool gya to is case mein kiya hoga to kiya framework steps follow karne padheng and kauns kam hoga

Context diagram represent karte SRS doc mein

SRS Structure

(1) **Introduction**

Purpose banne ka purpose kiya hai.

Intended Audience Audience kaun hogi (user kaun hai?)

Scope :- future mein scope kiya hai,

definitions :- mention karne software other than e last-last research

Reference :- Referral do user ki kam (demo typ)

(2) **Overall Descriptions**

User Interfaces :- Means kis public ko kaise user interfaces visible hoga, sirf mention

System Interfaces :- Server kaunse honge wo mention (Apache server) Protocol

Constraints, Assumption and dependencies
↳ Condition (gana bhejoge

User Characteristics different types users ko Diff Types provide karenge facilities.

(3) **System features and Requirements**

functional Requirement :- Apka software kaise kam karega Step1 :- registration Page
Step 2 :- User Id generated then password
Step3 :- Homepage pe aaye to bhi functionality sb btai hai

4 **Uses Cases :-**

5 **External Interface Requirements** :- koi third party ka item ki need hai like payment mode add karna to Paypal ka layout use kiya

6 **Logical database requirement** :- kaunsa database use kar rahe hai kaunsa data kaha save hoga. Kitna storage (max) require hai apko. Kitna ram chahiye hai.

5 NON functional Database :- Kiya data safe hai user name Apka ? Kiya App. secure hai.
Portability Har device pe chal skti hai ya Nahi.
reliable Hona chiye loss Na ho. (data breach Na Ho)
(Main Isme Quality ke bare Mein btna Hota)

(4) Deliver for Approval → user Signature keliye bhejdo. 1 (@)

{USER Requirement}

* Easy & simple to operate
means end user use karenge sys to Profit generate hoga client profit nikal and Company Ne bhi
To end user ki requirement kiya hai.
Simple interface ho. Tabhi use Hoga sys.

* Quick Response :- Performance achi honi chiye sys ki (No system fail) Payment mein response Quick buffering Na Ho or Na failed

* Effectively handling operational Error
Errors :- bahut sare software run horhye backend, live server work kar rhye hai to Problem ayegi hi to use kaise handle kar rha hai humara sys

* Customer Support like Payment ~~failure~~ failed to customer support tha

USER Requirement Specification

(URD) or user req. specification (URS) is a document usually used in software engineering that specifies what the user expects. The software to be able to do.

Contractual Agrement hai. MLG legal document hai Jo is document specified Hogya Wahi kam Hoga upar niche kuch Nahi and also testing, design ke Tym pe bhi guide karega ye doc.

Data Flow Diagram {Bubble chart}

Koi Software/App open ki to Multiple Pages Hote hai like information fill up kiya ab App Next Page pe forward kardegi to hua kiya Ek Module se dusre Module pe forward Horhye [to asal mein data move kar rha hai one point to other Point. to data flow horha hai

(*) A Graphical tool, useful for Communicating with users, managers and other Personnel.

User use kar skye App backend pe team hai To user communicate kar skye hai (teacher) backended se graphical tool.

* Useful for analyzing existing as well as proposed system

* Focus on the movement of data b/w external entities and process, and b/w processes and data stores.

* A relatively simple technique to learn and use

Why DFD

* Provides an overview of

* What data a sys processes

* What transformations are performed

* What data are stored

* What results are produced

* Graphical nature makes it good communication tool b/ween -

user and analyst

Analyst and system designer.

~~testing~~ DFD Elements

rectangle → ▭

* Source / Sink (External Entities) — [user] hua entity external, jo use kar sake.

(11)

* Data flow :- Kis direction move kar rha data — arrow ← →

* Process :- circle se ○ — book issues, book return — ye processes jo kam kar rha hai

* Data stores :- App to data base to use karge, to represent karte inko = parallel lines

External Entity :-

Source :- Entity that supplies data to the system which is called source (input) ko

Sink :- Entity that receives data from the system (output) leta hai

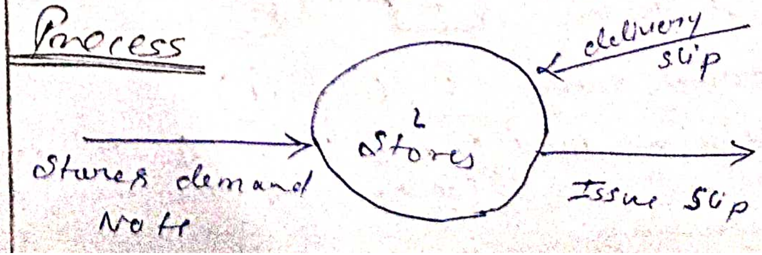
Only supply / recive / Not process

Data flow marks movements of data through the sys - a pipeline to carry data.

* Connects the processes, external entities and data stores

* Generally unidirectional. If same data flow in both directions, double headed arrow can be used

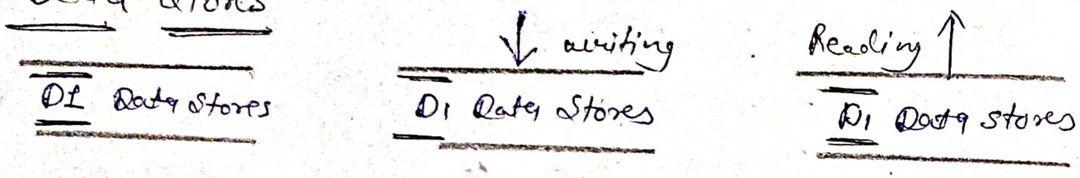
Process



- A circle represent a process.
- Straight line which incoming arrows are input data flows
- Straight lines with outgoing arrow are output data flows
- Labels are assigned to data flow.

(12)

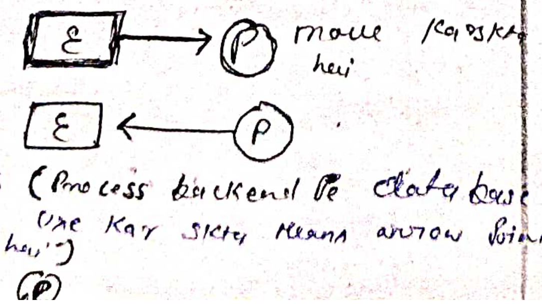
Data Stores



- A Data store in a repository of data.
- Data can be written into the data store: This is depicted by an incoming arrow
- Data can be read from a data store. This is depicted by an outgoing arrow
- External Entity cannot read or write to the data store

Rule of Data flow (Valid condition)

* Data can flow from
- External entity to Process
- Process to external Entity
- Process to Store and back (Process backend pe database use kar skta means arrow point kar skta hai)
- Process to process



- Data cannot flow from
  - External entity to external Entity
  - External entity to Store
  - Store to external Entity
  - Store to Store

(Nahi kar skte ye)

Entity dB ko direct Access nahi kar skte (kisi process ke through Access kar skte)

Ek dusre ko directly store nahi kar skte. → DB1 DB2

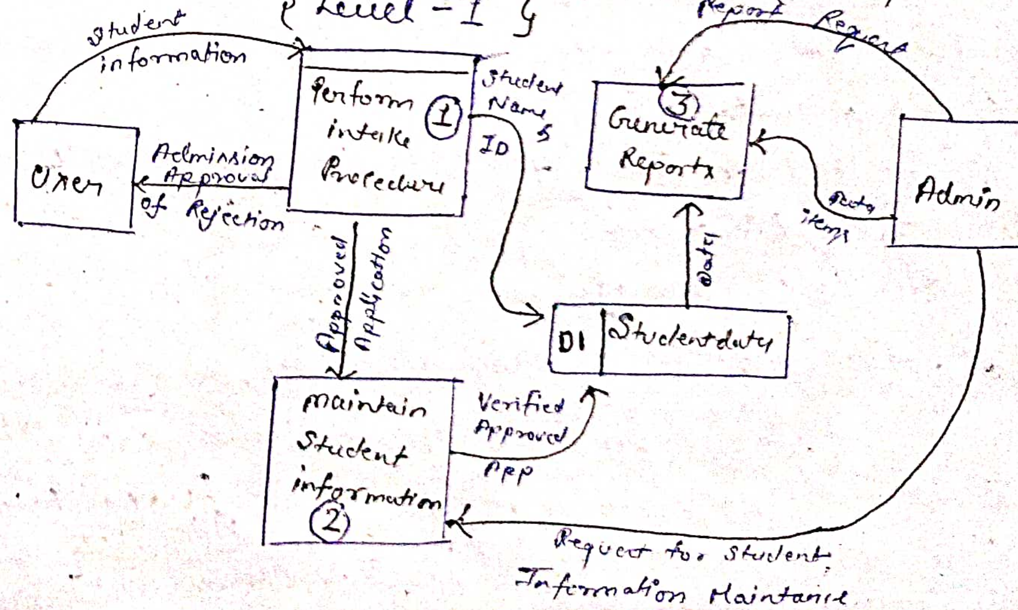
Why Level DFD

{Level – 0} upar but Hoti thi

user ko Admission lena hai

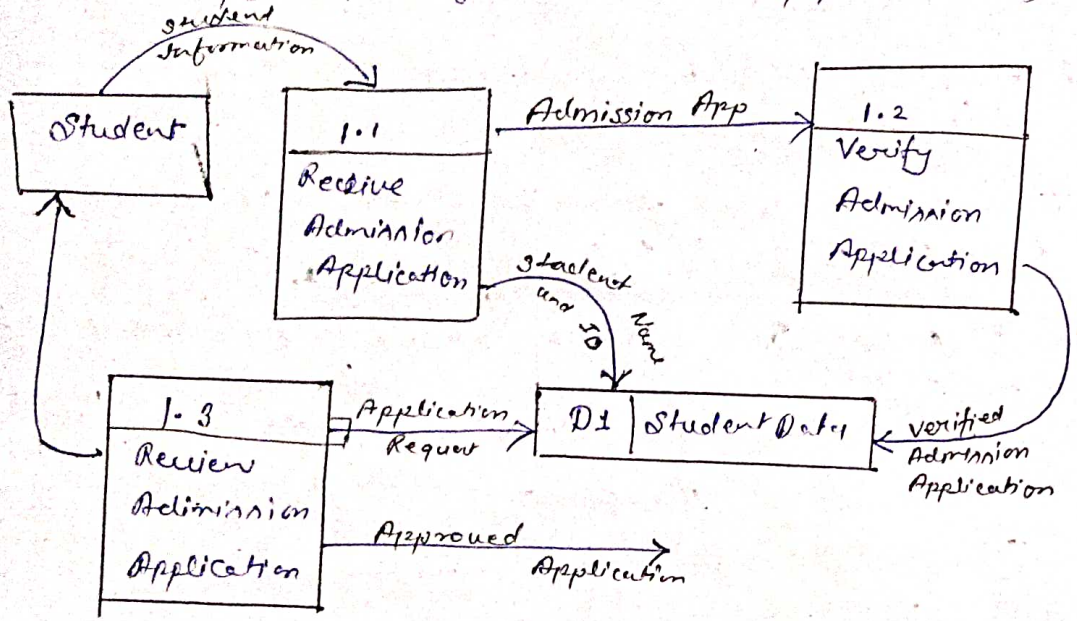
Level-0 ko descripive kardo thoda to Level 1 Hojata user or method se request kar skta

{Level – 1} (in depth.



{ Level-2 } Process 1, Perform Intake Procedure.

Several steps main sub divide karliya (broad)



Decomposition of DfDs

- A sys is too complex to be shown on a single DfD.
- Decomposition is the iterative process of exploding data flow diagrams to create more detail.
- level 0 data flow diagrams may be exploded into successive low levels of detail. The next level of detail would be a level 1 data flow diagram
- The DfD's become linked together in a hierarchy, which would fully document the system.

Why level DfD

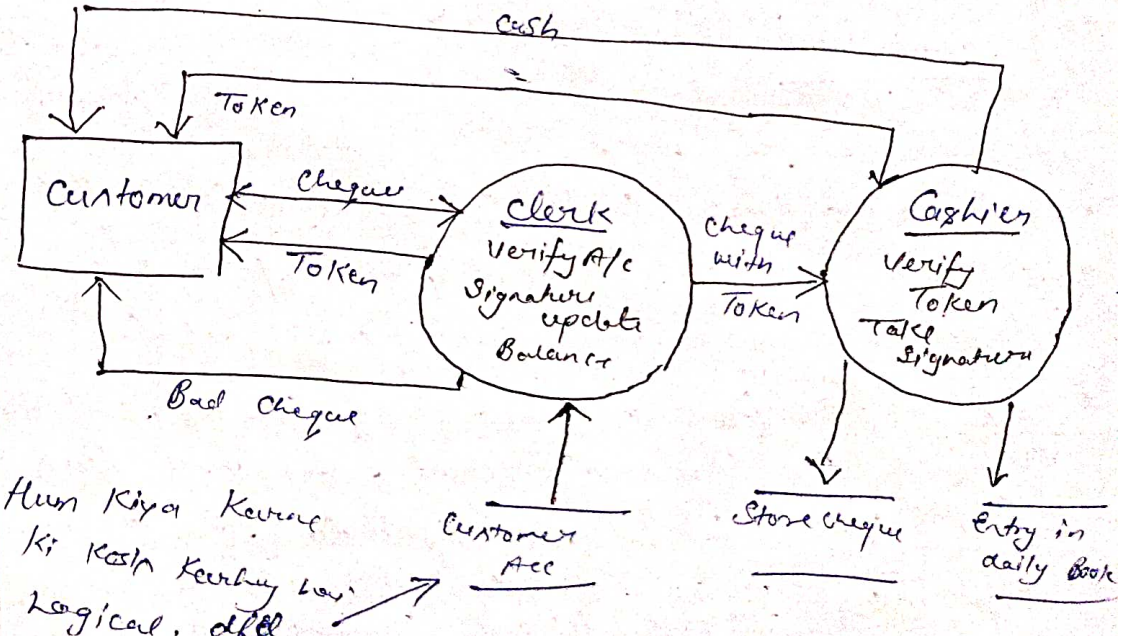
If a DfD is too detailed it will have too many data flow and will be large and diffucult to understand

Start from a broad overview. Expand to details idea simplex

(14) Logical and Physical DfD (Kaun use Karne Wala hai)

Humne Kam Kiya Hoga → Implementation

- DfD Considered so far are called Logical DfDs
- A Physically DfD is similar to a document flow diagram
- It specifies who does the operations specified by the logical DfD
- Physical DfD may depict Physical movement of the goods
- Physical DfD is can be drawn during fact gathering phase of a life cycle

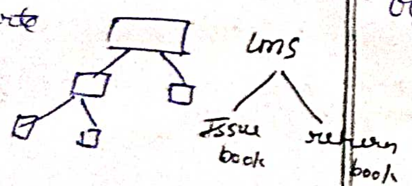


Hum Kiya Karne Ki Kosh Karthy hai Logical. dfd

Physical dfD hume Implementation bhi hai kaise Implement Karenge Logical DfD ko

Software Design Approaches (15)

Function Oriented Design	Object Oriented Design
• System is designed from a functional view point.	• System is viewed as a collection of object (Ex entities)
• Top – down decomposition	
• Divide & Conquer Approach	• Bottom – up approach
• DFD is used	• UML is used
Code Sys ko Top down Break karke module func create karte	Library Management System bana ri Obj ⇒ Student, faculty, Book
DFD Concep use karte The Why graphical ki wajhse	
Old	New Approach



Risk Management

• A risk is "an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives" ← soch Nahi Hoga guarantee Nahi

• A risk management plan is a document that a project manager prepares to foresee risks, estimate impacts, and define response to risk. Kitna damage hosкта Kiya response Hoga agar aya.

requirement gathering dwara information Implement karliya, abhi risk aai Nahi mtlb future Mein aasakti hai Imperative sirf tabhi Hoga Jab risk pahle badi hogi

foresee ⇒ check karna risks

Reactive Vs Proactive Risk Management Strategies

* Reactive risks management are to reduce the damage of potential threats and speed an organization's recovery from them, but assumes that those threats will happen eventually (aane ke bad)

* Proactive risk M. Identifies threats and aims to prevent those event from ever happening in the first place. (aane se pahle)

Types of Risks

- Business risk: Building a product that no one wants or losing budgetary commitments
- Technical Risks: Concerned with quality, design, implementation, interface, maintenance problems
- Project Risks: Concerned with schedule, cost, resources, customer-related issues.

(Planning chhod kar chalo yrr)

Risk Assessment

- It is process to rank the risks in terms of their damage
- Determine the average probability of occurrence value for each risk
- Determine the impact for each component based on impact assessment matrix.
- Risk Assesment Values are determined by multiplying the scores for the probability and severity values together

RM ← RA → Identify, Prioritise, Assess karna
RC

Pehle kaunsa risk handle karna

RA = Probability × Severity

Calculate risk. Impact Assessment Matrix

Risk Control Vs risk Mitigation → harmness kam krna.

* Risk Control is basically the specific actions to reduce a risk event's Probability of happening. Ex - Correct inspection and Maintenance of the car reduce the likelihood of mechanical failures

* Whereas Risk Mitigation is the set of actions to reduce the impact of a risk Event. Ex, Airbags are used to reduce the impact of an accident on the passenger and

risk ki happen hone ki probability ko kam karke to ye hai Control (hone se pehle ki bat karta)

Mitigation: kisi cheez ki harmness ko kam karna (damage reduce) (Hone ke bad)

Ex risk kab occur honga → severity value
like, fire, earthquake → to uska damage.

Risk Mitigation & Control Strategies

- **Risk Avoidance:** (Schedule bahut Tight ho Job Worker ka) To unke worker ko / ko Khush karne ki Minija techniques lagao
- **Risk Transfer:** like Amazon ke third party hire kar liya, Privide delivery. ko means transfer karde delivery kisi private company ko.
- **Risk Reduction:** risk ko reduce for future risk aaskta worker Job chhod ske to pehle se new bande recruit karke rkh,
- **Risk Monitoring:** open network to galat data aaskta To monitor karna. (unseen activity ko monitor karna,)

Project Estimation Techniques

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

- Project size
- Effect required to Complete the Project
- Project duration and
- Cost

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

* 1 Empirical E.t
* Heuristic
* Analytical

Emphirical Estimation Techniques (guess karna)

* Empirical E. t. are based on making an educated guess of the project parameter and common sense
* This technique is based on prior experience of development of similar product and project.
* an educated guess based on past experience
* Two popular empirical estimation t are:
  - (i) Expert Judgment Technique
  - (ii) Delphi Cost Estimation Yahn solo guessing more (Isme team of expert guess kar rhye)

• Expert Judgment Techniques -(i) (Single)

In this an expert makes an educated guess
* of the problem size after analyzing the Problem thoroughly.

* The expert estimates the cost of the ~~expert~~ different components of the System:

Ex ⇒ GUI, database module, Communication Module billing module, etc.

* Combination them to arrive at the overall estimate

• Delphi Cost Estimation -(ii) (grouping)

In Carried out by a team Composing of a group of experts and a Coordinator

The Coordinator provides each estimator with a copy of the SRS document and a form for recording his cost estimate.

Estimators Complete their individual estimates anonymously and Submit to the Coordinator

(2) Heuristic Techniques.

In Heuristic Techniques the relationship that exist among the different project
* Parameters can modeled using suitable mathematical expressions

* Once the independent parameters are known the dependent parameter can be easily determined by Substituting the values of the Corresponding mathematical expression.

* Assumes that the characteristics to be estimated can be expressed in term of

* Can be classified as Single Variable and multivariable models

(*) Single Variable Model :-

Estimated Parameter = $C_1^{e_2} * e^{d_1}$

In the above expression, e the characteristics of the Software which has already been estimated (Independent variable).

Estimated parameter is the dependent parameter to be estimated.

The dependent parameter to be estimated
* Could be effort, project duration, staff size, etc.

C1 and d1 are Constants and are usually
* determined using data collected from historical data

The basic COCOMO Model is an example
* of Single Variable Cost Estimation Model

(1) A Multivariable Cost Estimation Model takes the following form:

Estimated Resource = $C_1 * e_1^{d_1} + C_2 * e_2^{d_2} + \cdots$.

Where $e_1, e_2 \ldots$ are the basic (independent)
* characteristics of the software already estimated

* $C_1, C_2, d_1, d_2 \ldots$ are Constants

* The intermediate COCOMO model can be considered to be an example of a Multivariable estimation model.

COCOMO (Constructive Cost Model)

- was first proposed by Dr. Barry Boehm in 1981.
- Is a heuristic estimation technique - this technique assumes that relationship among different parameter can be modeled using some mathematical expression.
- This approach implies that size is primary factor for cost, other factors have lesser effect.
- "Constructive" implies that the complexity.
- COCOMO prescribes a three stage process for Project estimation.
- An initial estimate is obtained and over next two stages the initial estimate is refined to arrive at a more accurate estimate
- Project used in this model have following attributes;-

1. ranging in size from 2000 to 1,00,000 lines of code

2. Programming languages ranging from assembly to PL/I.

3- These projects were based on the waterfall model of software development.

- Project used in this model have following attributes :-

Ranging in size from 2,000 to 100,000 lines of code

Programming languages ranging from assembly to PL/I.

- Bohem stated that any software development Project can be classified into three Categories.

1 Organic : if the Project deals with developing a well understood application program.

- The size of development is reasonably small and experienced.

- The team members are experienced in developing similar kind of Project.

2. Semidetached :- if the development team consists of a combination of both experienced and inexperienced staff.

Team members have limited experience * about some aspects but are totally unfamiliar with some aspects of the system being developed.

* Mixed experience.

3. Embedded :- if the software being developed is strongly coupled to complex hardware.

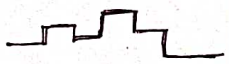
Software project that must be developed within a set of tight software, hardware and operational constraints.

Mode	Project size	Nature of Project	Innovation	Deadline of the Project	Development Environment
Organic	2-50 KLOC	Small size Project, Experienced developers in the familiar environment. Ex → Pay roll, inventory Project etc	Little	Not tight	familiar & in house
Semi detached	50 - 300 Kloc	medium size Project, Medium Size team, Any previous Experience on Similar Project. Ex: utility Systems like Compilers, dB System, editors etc	Medium	Medium	Medium
Embedded	Over 300KLOC	Large project, Real time System, Complex Interfaces, Very little, Previous experience Ex: ATMs, Air Traffic Control etc	Significance	Tight	Complex Hardware / customer interfaces required

Person Month (PM)

- The effort estimation is expressed in units of Person Months.
- An effort of 100 PM does not imply that 100 Persons should work for 1 month nor does it imply that 1 person should be employed for 100 months, but it denotes the area under the Person-month curve.
- It is the area under the Person-month Plot

Ye graph Hota hmrd requirement ko hishab se kitne employes ki front specific tym pe



3D work 4 month = 12 PM

1 Basic COCOMO Model (SLOC, KLOC)

- Basic COCOMO model computes software development effort, time and cost as a function of Program size.

$$\text{Effort} = a_1 \times (\text{KLOC})^{a2} \text{ PM}$$

$$T_{dev} = b_1 \times (\text{effort})^{b2} \text{ Months}$$

↓ Constant values

(Estimation of dev effort)

Organic = Effort = $2.4 (\text{KLOC})^{1.05}$ PM

Semi-detached : Effort = $3.0 (\text{KLOC})^{1.12}$ PM

Embedded : Effort = $3.6 (\text{KLOC})^{1.20}$ PM

(Estimation of develop Time)

Organic : $T_{dev} = 2.5 (\text{effort})^{0.38}$ Months

Semi-detached : $T_{dev} = 2.5 (\text{effort})^{0.35}$,,

Embedded : $T_{dev} = 2.5 (\text{effort})^{0.32}$ months

Calculating Effort and Productivity

$$\text{Avg staff size (SS)} = \frac{E}{D} \quad \begin{matrix} \to \text{Effort} \\ \to \text{Person} \\ \to \text{development} \end{matrix}$$

$$\text{Productivity (P)} = \frac{\text{KLOC}}{E} \text{ KLOC/PM} \quad \to \text{Estimation}$$

Merits: Basic cocomo is good for quick, rough and early estimate of software costs

Demerits: It does not account for differences in hardware constraints, personal quality and experience, use of modern tools and techniques and so on.

The accuracy of this model is limited because it does not consider certain factors for costs estimation of software.

Intermediate COCOMO MODEL

Basic Cocomo + 15 Cost driver bhi hai unke bases pe value ko use karte hai.

- **Product Attributes** * Required Software reliability * Size of app db * Complexity of the product.
- **Hardware Attributes** * Runtime Performance Constraint * Memory Constraint * Volatile of the Virtual Mach environment * Required turnabout time
- **Person Attributes** * Analyst Capability * Software engineering Capability * Application experience * Virtual Machine Experience * Programming lang exp
- **Project attributes** * Use of Software tools * Application of Software Engg methods * Required development schedule.

$$E = a_1 (KLOC)^{b_1} \times EAf \quad (\text{Effort adjustment factor})$$

Software	a_1	b_1
organic	3.2	1.05
Semi-Detached	3.0	1.12
Embedded	2.8	1.20

Merits • This model can be applied to almost to entire software product for easy and rough cost estimation during early stage
• It can be applied at the software product component level for obtaining more accurate cost estimation.

Demerits • The effort multipliers are not dependent on phases (Single homogenous entity hai consider karte)
• A product with many components is difficult to estimate

Complete COCOMO Model

Basic + Intermediate language or Project ke hisab kar type

The cost driver's impact on each step (analysis, design, etc) of the software engineering process.

Example • A management Information System (MIS) for an organization having offices at several places across the country:
- Database Part (Semi-detached)
- Graphical USER Interface (organic)
- Communication Part (embedded)

• Costs of the components are estimated separately:
- Summed up to give the overall cost of the system

Verification	Validation
1) Are you building it Right? (In Process) pe puch te	Have you build the right thing
2) check wheather an artifical conforms its previous	Check the final Product against specification
3) Done by Developer	Done by Tester
4) Concerd with phase containment of error	Aim is to make final Product error free
5) Methods involves Review (Test Part by Part) Inspection, Unit testing and integration testing	Involves System testing
6) Static and dynamic mosty Activities (bina execute karke check nahi krrtye)	Only dynamic (jab execute karke dekhte) Car drive karke check

Software Testing
- ① Level → Unit Testing
- ② Integration Testing
  - → Bang Bang
  - → Top Down
  - → Bottom up
  - → mixed
- ③ Sys testing
  - func — Based on who is doing Testing
    - → Alpha (company)
    - → Beta (friendly) (Kaun Krta Test)
    - → Acceptance (User check)
  - → Performance / Non functional
    - → Volume → Install
    - Load
    - Stress
    - Security
    - Configuration
    - Compatibility
    - Recovery
- ④ Regression Test (maintanance) (upgradation) Useful / or Not

* Coding/development Phase # Unit testing Main ata
* during dev phase checking
  - → white box (test case design Approach hai)
  - → Black box (Test desiging hai) Input → output dekr check krte
* Integration Testing Proper testing (full unit ko compile) test compatible hai
* Sys testing :- Pure SYS (car → engine, tyre, body etc) proper driving test fully inspection, senerios alag

Introduction

Design Phase transforms SRS Document: into a from easily implements in some Programming language.

SRS Document —Design Activity→ Design Doc

A module consist of
(1) several functions
(2) associated data structure

class → method → Attributes
↳ Attributes

Data	D1 D2 D3
F1.. F2.. F3.. F4..	functions Module

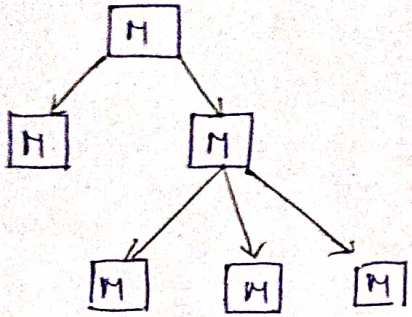
Modularity (S. modules divide kiya diff methods depending kam se kam Ho)

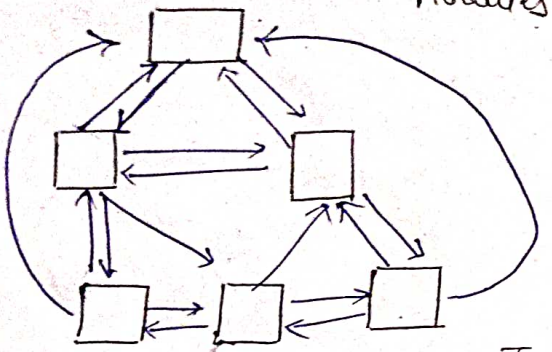
modularity is a fundament Attribute of any good design.

Decomposition of a Problem cleanly into Module
(1) modules are almost independent of each other
(2) divide and Conquer Principal.

If modules are independent * modules can be understood seperately. * reduces the complexity greatly.

Example of Clearly and Non-Clearly Decomposed Modules





dependency bahut Jada kam Kma hai

In technical terms, Modules should display

(1) High Cohesion: (togetherness (grouping) within the Module/ func / Attributes Kaise group krte Ek team Ka Kam Kaise ye.

(2) Low Coupling: within module Do yy use Jada. module module ke andar Ke beech dependency unki bat krte hai (inter)

* Cohesion is a measure of: functional Strength of a module.

A Cohesive module performs a single task or function

* Coupling b/w two modules: a measure of the degree of interdependence or interaction b/w the two modules (inter) within the module.

Classification of Cohesiveness

function	sequential	Communication	Procedural	temporal	logical	Coincidental

← Degree of cohesion

Logical Cohesion: (within the Module)

All Elements of the module Perform Similar Operations

Ex- error handling, data input, data output etc.

When logically categorized elements are put together into a module, it is called logical Cohesion.

(Attendance system → create, delete, view Ek module data ki Kosis karo)

Temporal Cohesion (Same tym pe occur hotey - unhe Ek data kosis kro)

The module Contains tasks that are related by the fact that all the tasks must be excuted in the same time span.

When elements of module are organized such that they are processed at a Similar Point in time, it is called temporal Cohesion.

Procedural Cohesion (Koyne algo kam krtey ek mein)

if the set of functions of the module all Part of a Procedure (algorithm) in which certain sequence of steps have to be carried out in a certain order for achieving an objective.

E.g the algorithm for decoding the msg.

Communicational Cohesion

(Jahn data process/Information Ek vari ki values koi otherwise kr rha to Un module ko Ek mein dete)

If all functions of the module Refer to the same data structure.

Ex: The set of functions defined on an array or a stack

Sequential Cohesion

(Ex input → output → output / Sequential input → output)

If the elements of a module forms different parts of a sequence, output from one element of the sequence is input to the next

Sort → Search → display

Functional Cohesion

If the different elements of a module cooperate to achieve a single function.

It is considered to be the highest degree of cohesion, and it is highly expected.

Coincidental Cohesion

The module performs a set of tasks which related to each other very loosely, if at all.

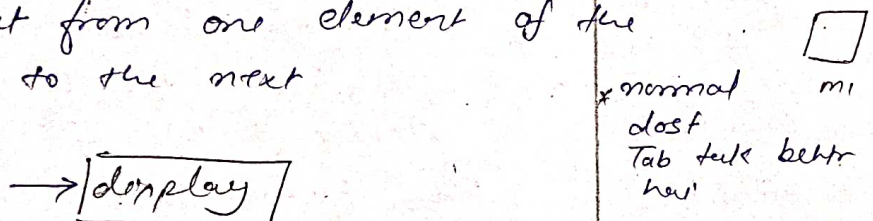
* The module contain a random collections of funct

* Cohesion measure the strength of relationship b/w pieces of functionality within a given module.

Coupling indicates:

How closely two modules interact or how interdependent they are.

The degree of coupling b/w two module depends on their interface complexity.

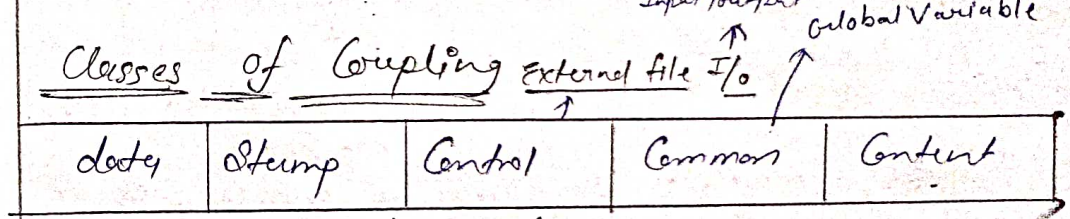


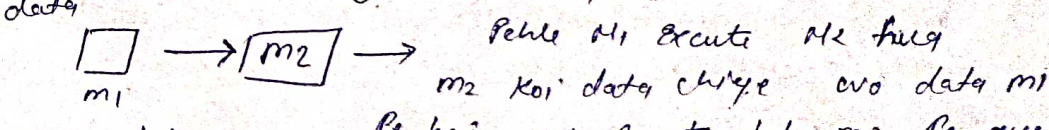
* normal dost Tab talk better hai

* Ye done couples hai to like each dependency km se kam ho freely move kr ske

* Coupling mein restriction lgne lgti

Classes of Coupling





Pehle m1 excute ho tabhi m2 koi data chiye ayga data m1 Pe hai m1 excute tab m2 Pe ayga yhn data dependency.

Stamp :- Complete data structure yhn se move hota To dependency aaskti.

Eg: Linkedlist, if else, Control execution

Same module mein use karnge

Unit Testing (Tier 1 testing) before Integration

- Unit testing is the process of checking small pieces of code to ensure that the individual parts of a program work properly on their own.
- Unit tests are used to test individual blocks (units) of functionality
- Unit Testing is done by developers

Integration testing (Sare unit ko combine then testing)

- Integration tests is conducted to evaluate the compliance of a system or components with specified functional requirements

It occurs after unit testing and before sys testing

Types of Integration Test → 4 types

Big Bang :- Kam se 2,3 combine then test

Mixed :- Top-down + Bottom up

Top-down :- Top priority wala use test for bad ke test.

Bottom up :- Low priority pehle test for upar ki trah test.

Yhn white Box → Internal structure.

System Testing (unit → Integration → Sys test)

System testing is a level of testing that validates the complete and fully integrated software product.

* The purpose of a system test is to evaluate the end to end system specifications

* System Testing is black-box testing   input diya I → [ ] → O Output desirable output

* Sys testing categories based on: who is doing the Testing?

* Sys testing Cat : functional / Non-functional requirements
  - functional ↓ Brake / Lock etc
  - Non-functional ↓ Load 4 seater but 5, 7 test people.

director / writer → Premier / Actress / testing → user client

Eg: alpha — Beta — Acceptance

Independent team test karegi. Isko test yhn funct / Non func Testing Hogi

Black box → External testing. (Non-func

Types of sys testing — user / lakh, kroror

- Performance → speed, scalability, stability, reliability
- Load Test → Jmb dB kam krlo ab extra load dal ke dekh
- Stress test → Jaan kar error dalte for test karte. to check roboustness
- Scalability Test :→ User bases badh to fr bhi Makhan
- Security Test → Active / Passive / denied / sql injection
- Configuration Test

White Box

* The developer can perform white box testing
* What the software is supposed to do, also aware of how it does it. (Kiya Kare Software / or Kaise Kam)
* To perform WBT, we should have an understanding of the programming lang.
* In this, we will look into the Source Code and test the logic of the code
* In this, the developer should know about the internal design of the code
* Test design techniques Control flow testing, Data flow testing, Branch testing, Statement Coverage, Decision Coverage, Path testing
* Can be applied mainly at unit testing level but now in Integration, System level also

---

* White box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) is a method of software testing that tests internal structures or working of an application.
* White box testing can be applied at the unit, integration and system levels of the software testing process
* Some of the top White box testing tools to use: Veracode, Cppunit, Nunit, RCUNIT

Black box testing

* The test engineers perform the perform the BBT.

* What the software is supposed to do but is not aware of how it does it.
  Kya Karega Pta hai / Kaise Karta hai Nahi Pta

* To perform BBT, there is no need to have any understanding of the programming lang

* In this, we will verify the functionality of the app based on the requirement specification
  Test drive car engine, exhaust, calling sys ~~…~~ No focus

* In this, there is no need to know about the internal design of the code

* Test design techniques; Decision table testing, all pairs, testing, Equivalence partitioning, Boundary value analysis, Cause effect graph.

* Can be applied virtually to every level of software testing unit, integration, system and acceptance.

1.

Statement Coverage technique

- Statement Coverage technique is used to design White box Test cases
- This techniques involves execution of all statements of the source code at least once
- It is used to calculate the total number of executed statement in the source code out of total statements present in the source code
- This technique Cover dead code, unused code and branches

Total lines of Test hoti chaye — To lines of code To sari execute karke check

Condition Coverage Technique

* It is used to cover all conditions
* Conditions Coverage is also known as predicate Coverage in which each one of the Boolean expression have been evaluated to both True and false.

Test case humei sari condition ko evaluate kare

Page :
Date : / /

Data flow Testing

- It is one of the white box testing techniques.
- Data flow testing focuses on two points :-
  (1) In which statements the variables are defined.
  (2) In which statement the variable are used
- It design the test cases that covers control flow paths around variable definitions and their uses in the modules

30, 20, 10

```
1. Read a, b c
2. If (a > b
3.     x = a + 1
4.     Print(x);
5. else
       x = b - 1
6      Print(z)
```

	Define	use
a	1	2, 3
b	1	2, 5
c	1	NA
x	3, 5	4
z	Not defined	6
	available	

Advantage of D F :

* A variable that is declared but never used within the program
* A variable that is used but never declared.
* A variable that is defined multiple times before it is used
* Deallocating a variable before it is used

Date : / /
Page :

Black Box :- Boundary Value Testing

- The black box testing techniques are helpful for detecting any errors or threats that happened at the boundary values of valid or invalid partitions rather than focusing on the centers of the input data.

min	Max (~~range giving~~)
Test Case Let us Speed per	Number # 1 assume a test case that takes the of a car from 40 to 80 get best fuel efficiency

Boundary value test cases

Invalid Test Case (min − 1)	~~Test~~ Valid Test (min, +min, −max, •max)	Invalid Test Case (max + 1)
39	40, 41, 79, 80	81

Software Maintenance

- The primary goal is to modify and update software applications after delivery to correct errors and improve performance.
- Software maintenance is an inclusive activity that include :
  - Error Corrections
  - Deletion of obsolete Capabilities
  - Enhancement of Capabilities like user to (new technology) 1 million
  - Optimization. → Speed Performance responsiveness fast No.

Page :
Date : / /

Maintenance = 67%   implementation = 7%

Testing = 15%   Designing = 8%

Requirement = 3%

Kam % hai

Types reactive

1. Corrective Maintenance — 20% Bugs fixes
2. Adaptive Maintenance — 25% Platform Ind.
3. Preventive ,, → 8% aane se pehle Problem.
4. Perfective ,, → 50% → Performance, enhancement obsolete Ho Jye

Proactive

Maintenance Metrics

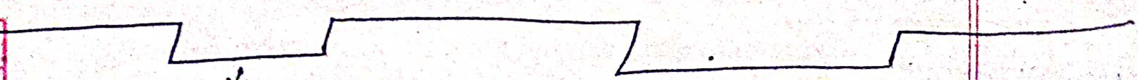
(1) Mean Time B/w failure (MTBF) : It is the avg time available for a system or components to perform its normal operations b/w failures. <kitna chalta tha > for failure >

• MTBF = (SUM of Operational time / Total number of failures).
dono failure ke beech kitna up tha.

(2) Mean Time to Repair (MTTR) : It is the avg time required to repair a failed component.

• MTTR = (Sum of downtime Periods / total number of failures)

availability = MTBF/(MTBF + MTTR)



repairing tym kitna lga sahi hone M.

Reverse Engineering / Backward Engineering

* Software reverse engineering can help to improve the understanding the underlying source code for the maintenance and improvement of the software.

* In some case the goal of the reverse engineering process can simply be a redocumentation of legacy systems.

Code → Module specification → Design → Requirement specification

Usage of Reverse Engineering

Malware developers often use reverse engineering techniques to find vulnerabilities in an operating system to build a Computer virus that can exploit the system vulnerabilities.

Reverse engineering is also being used in cryptanalysis to find vulnerabilities in substitution ciphers, symmetric-key algorithms or public-key cryptography.

Date : / /

Computer aided Software engineering (CASE)

- It is the domain of software tools used to design and implement applications
- CASE Tools were used for developing high-quality, defect-free, and maintainable software.
- It helps to improve Software development as well as maintenance
- Improves quality at lower cost & increases Productivity.

Case Examples

* Planning & Requirement Phase (Flow Chart Maker, Creately, Pro office, Basecamp
* Design phase (Animated Software Design, UML)
* Coding (Scope, Eclipse)
* Testing (Selenium, Cucumber)
* Web Development tools (Fontello, Adobe Edge Inspect, Foundation 3).
* Quality Assurance tools (Soapui Test, Apps Watch)
* Maintenance (Bugzilla for defect tracking, Hp Quality Center).

Date : / /
Page :

Performance / Non functional testing can determine how a system performs in term of responsiveness and stability under a particular workload.

A performance test measure Stability, Speed, Scalability, and throughput of your product

Load testing :- It evaluates the product's ability to perform under increased load.
for Ex -> maximum capacity of a web server.

Stress testing :- It is normally used to understand the upper limits of capacity within the system.

Ex⇒ Capacity of a website or an e-Commerce platform during peak traffic hours such as during a major sale event.

Volume testing :- It basically testing the ability of a db management sys handle a large amount of data.

Endurance testing :- This type of testing evaluates the system's performance over a long period of time, usually under normal or expected load conditions, to identify any memory leaks or performance degradation over time.

Response time testing :- This type of testing evaluates the time it takes for the system to respond to a request, with the goal of determining the acceptable

Page :
Date : / /

response time for end-users.

Spike testing :- Tests the product reaction to sudden increases & decrease in the load.

Regression Testing

Regression testing is a type of software testing that verifies that changes made to the system such as bug fixes or new features do no impact previous working functionality.

There are several types of regression testing, including

Full Regression Testing :- Testing the entire application from start to finish after changes have been made.

Partial Regression Testing Testing only those parts of the application that were affected by the changes.

Date : / /