Software Estimation

Date

⇒ Software Pricing:

- Hardware
- Software
- Training
- Travel
- Effort
- Business considerations
- Political
- Economic
- Organizational

} influenced on price charged.

⇒ Factors effecting Software Price Estimation:

(1) Contractual Terms: Source Code.

(2) Cost estimate uncertainty: Unsure about estimation.

(3) Financial health: Small profit or breakeven.

(4) Market Opportunity: Place in market area.

(5) Requirements Volatility: Unclear requirements (likely to change)

⇒ Pricing Strategies:

(1) Under pricing: (1) Market (2) Financial.

(2) Over pricing: Buyer wants fixed cost.

⇒ Pricing to win:

i) Buyer's budget is low.

ii) If this is less than the development cost, you reduce some functionalities accordingly and say to add them on new release.

iii) Pay extra for another release and recaves the short fall.

⇒ Estimation Techniques:

① Experience-based Technique:

- Based on manager's past project experience.
- We made an informed judgement of how much effort is required.

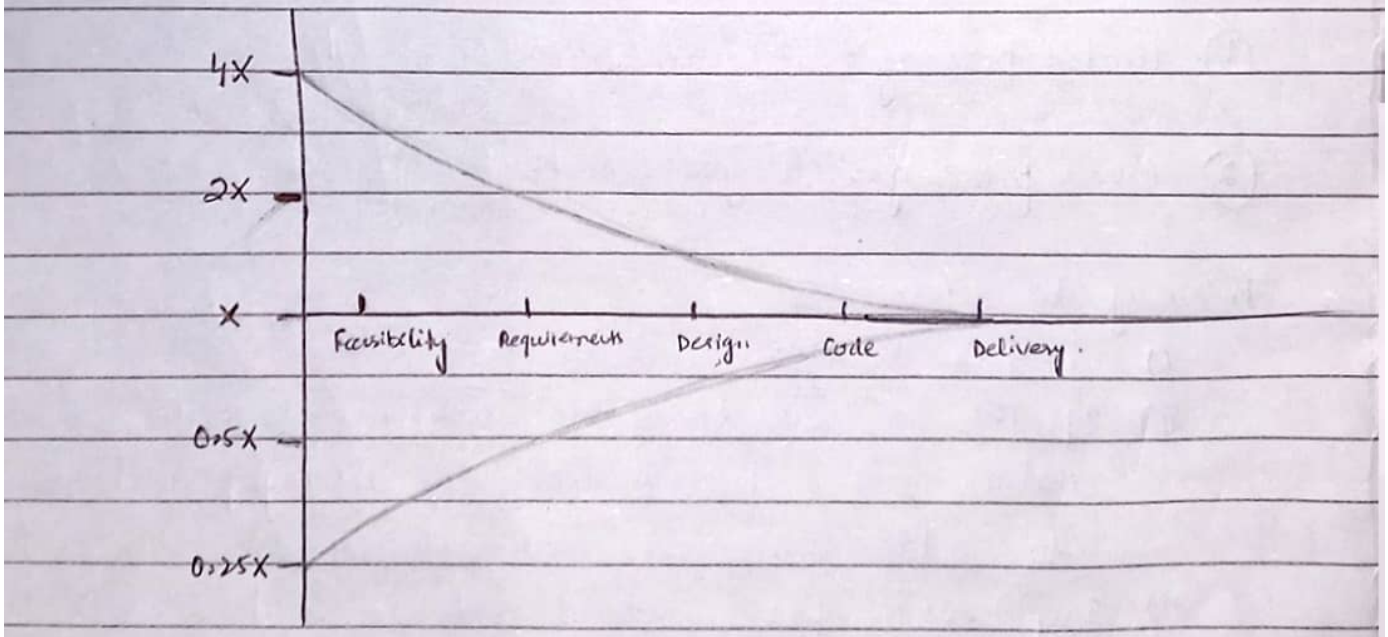
② Algorithmic cost modeling:

In this approach, a formulaic approach is used to compute the project effort based on estimates of product attributes, such as size, and process characteristics, such as experience of staff involved.

⇒ Estimate Uncertainty:

Boehm et al. collected data from large number of projects and concluded that:

If the initial estimate of effort required is X months of effort, then at the time when system is delivered the range may be from 0.25X to 4X.



⇒ Experience Based Approach:

- Rely Based on judgements based on experience of past projects and the effort expended in these projects
- Identify deliverables, or software components to be delivered
- Make a spreadsheet and estimate the amount of effort required for each of them. and compute the total effort required.
- It usually helps to get a group of people involved on the effort estimation and to ask each member of the group to explain their estimate.

Problems:

i) A new software project may not have much in common ~~than~~ with previous projects.

ii) Software development changes very quickly, everyday new technologies are introducing. So if you have to work on a latest technology or domain, then your past experience won't be able to help you for the identification of effort required.

⇒ Algorithmic Cost Modelling:

i, Cost is estimated by using a mathematical function of product, project and process attributes whose values are estimated by project managers:

$$Effort = A \times Size^{B} \times M.$$

- A is an organisation-dependent constant
- B reflects the disproportionate effort for large projects.
- M is a multiplier reflecting product, process and people attributes.

i) The most commonly used product attribute for cost estimation is code size.

ii) Most models are similar but they use different values for A, B and M.

Estimation Accuracy:

i) The size of a software system can only be known accurately when it is finished.

ii) Several factors influence the final size:
- Use of reuse system & components.
- Programming language.
- Distribution of system.

iii) As the development process progresses then the size estimate becomes more accurate.

iv) The values of B & M vary according to the judgement of estimator.

Effectiveness of Algorithmic Model:

i) Algorithmic cost model is a systematic way to estimate the effort required to develop a system.

ii) However, these models are complex and difficult to use.

ii. The practical application of #CM has been limited to a relatively small number of large companies, mostly working in defense & aerospace systems engineering.

⇒ The Cocomo Cost Modelling:

- Constructive Cost Modelling.
- most popularly used
- estimates or predicts the ~~effort~~
  - effort required for the project.
  - total project cost.
  - scheduled time for the project.
- depends on number of lines of code.
- developed by Barry Boehm in 1981.

Levels of Cocomo Model:

→ Basic

→ Intermediate

→ Detailed

→ Basic Cocomo Model:

- It is a single valued, static model that computes software development effort and cost as a function of program size expressed in estimated "thousand delivered source instructions (KDSI)".

per month, man month.

Effort = $a \times (KLOC)^b$ ~~PM~~ or MM

(D) Scheduled Time = $C \times (E)^d$ Months (M)

Total

a, b, c, d = constants, KLOC = Kilo lines of code, E = Effort in MM

D = Total time required.

Signature _____ UNIQUE No. _____

→ Intermediate Cocomo Model:

- The basic cocomo assumes that the effort is only a function of number of lines of code and some constants.

- However, in reality, no system's effort and schedule can be solely calculated on the basis of lines of code.

- ~~It depends upon~~
- ~~For that~~, various other factors ~~such as reliability~~ known as cost drivers.

- Intermediate Cocomodel uses 15 such drivers for the estimation of cost.

- Some of them are: Reliability, Capability, Memory constraints, Experience and so on.

→ Detailed Cocomo Model:

- incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. In detailed cocomo, the whole software is divided into different modules and then we apply cocomo in different modules to estimate effort and then sum the effort.

(1) Planning & requirements (2) System design (3) Detailed design (4) Module code & test (5) Integration & test (6) Cost Constructive model.

Development Modes

Date

	Size	Innovation	Deadline	Dev. Environment.
Organic	Small	Little	Not tight	Stable
Semi-detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex hardware/Customer interface

Basic Cocomo Model

Project Type	a	b	c	d.
Organic.	2.4	1.05	2.5	3.8
Semi-detached	3	1.12	2.5	3.5
Embedded	3.6	1.2	2.5	3.2

Organic (2 - 50K lines of code).
Semidetached (50K - 300K KLOC)
Embedded (Over 300KLOC).

$$\text{Effort} = a \times (\text{KLOC})^b \quad MM$$

$$D = C \times (E)^d \quad M.$$

$$\text{Avg. Resource} = E/D =$$

$$\text{Productivity} = \text{KLOC}/E =$$

SCRUM

⇒ Scrum is the most popular agile methodology.
⇒ lightweight, iterative & incremental framework.
⇒ breaks down the development phases into stages or cycles called 'sprints'.
⇒ The dev. time for each sprint is maximized & dedicated thereby managing only one sprint at a time. 2 week se 1 month tk ka ek sprint hota he, us ke bd user feedback lete hen phir next sprint design krte hen.
⇒ Scrum team has scrum master & product owner with constant communications on the daily basis.

Terminologies:

(1) Backlog: Client, stakeholders, developers sb ne design kiya hua ke kon se chiz kese bnegi.

(2) Daily Scrum: 10 to 15 mins meeting on daily basis stakeholders or developers ke beech bti ke ye ye kam chl rha ese ese, us ke bd dev apni team k sath bhi discuss krta he. (Care members equally important hen)

(3) Scrum Master: Sari chizo ko monetor & handle kr rha he, team ko bhi handle kr rha he or stakeholder k sath bhi communicate krs rha hi client k sath bhi.

Balance le freedom & adaptation

(4) Product Owner: Client.

Adv

- Freedom & Adaptation.
- High quality, low risk product.
- Reduce dev time
- Customer satisfaction
- review current sprint then more next on the new one

Disadv

- More efficient for small team size, because there is no hierarchy.
- No changes on the sprint.

Date _____

Scrum Values:

① Courage: Team me courage aur agr kam mushkl bho h tb bhi us pr work krein or achieve krein. Have the courage to work on tough problems.

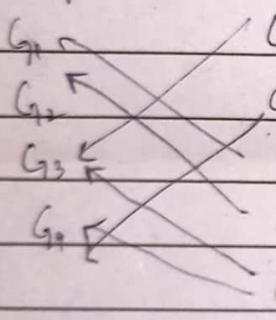
② Focus: Focus on the Sprint and the goals of the Scrum Team.

③ Commitment: Very important

④ Respect: Extremely Important

⑤ Openness:

Groups bids Evaluation



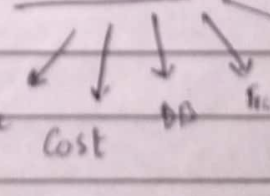
	b_1 (50K)	b_2 (40K)
Lowest	2	1

Requirements Engineering:

- The process of defining, documenting & maintaining requirements in the engineering design process

OR

The process of establishing the services that a customer requires for a system & the constraints under which it operates and is developed.



When do we do it!

— Waterfall Model (First phase me usna hoga req. eng.)

— Others (Continue life time of projects)

Date

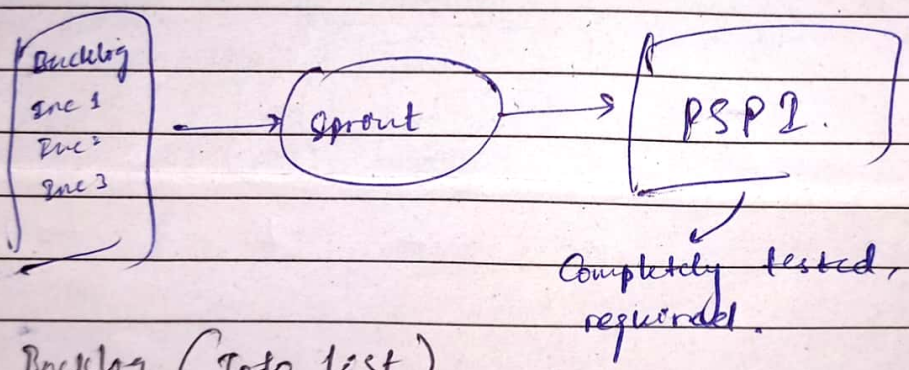
↑ important & difficult job

Scrum Master: Responsible to ensure the execution of smooth scrum process. Guide the teams involved.

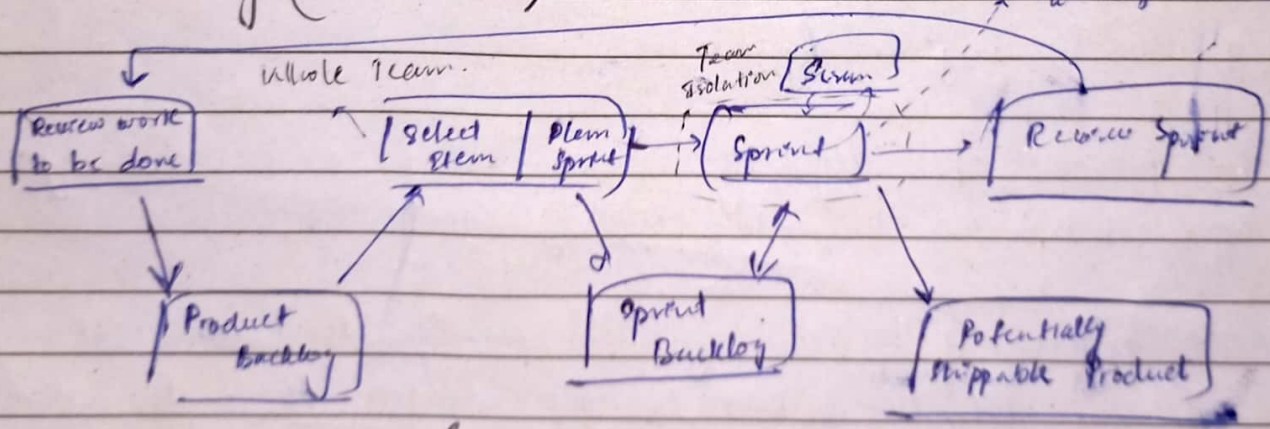
Development Team: Group of around 7 software developers (self organising).

Product Owner: May be customer, may be scrum master or head of project management (project manager).

Potentially Shippable Product Increment:



Product Backlog (Todo list).

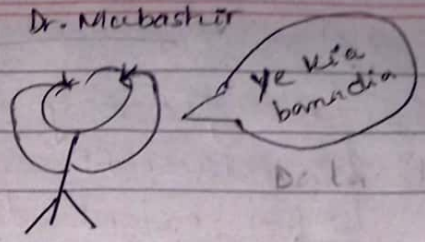


Scrum Sprint Cycle.

Task Boards:

Stories	ToDo	In progress	Testing	Sprint End

Software Engineering



Types of Software

1. Application Software.
2. System Software.
3. Malicious Software. (Pegagus).

What is a good Software? Engineer

⇒ Maintainability (Modification & Updation).
⇒ Dependability and Security — Reliability
— security
— safety.
⇒ Efficiency. (fast, optimal, storage, accuracy)
⇒ ~~Accuracy~~ Acceptability.

Why a software projects fails?

Challenges in Software Project.

- Effort Intensive
  - Cost Management
  - long development time.
  - Changing needs of Customer.
  - High risk of failure — user acceptance
    — Performance
    — Maintainability

Time Slippage.

Adhoc Approach ↓ Make a project without any documentation.

Adhoc app. use krne se ye sare issues arise.

After completion we have to ensure that.

1. It is useful.
2. It is usable.
3. It is used.

Adhoc Approach:

↳ No planning
↳ Deliverables Not identified. (Could be a docum part of software, fe
↳ Poor understanding of user requirements.
↳ No Control & Review (Koi lead nhi hoga, to koi review nhi krega)
↳ Poor Understanding of Cost & Effort

Software Engineering Ethics:

① Technical
② Not only upholding the legal practices/laws but set of principles that are morally correct.
③ Confidentiality.
④ Competence (If you can do it, you have expertise, then take it, else ~~don't~~). Tell them that you're in a learning curve).
⑤ IP rights. (Intellectual Property rights).
↳ should be well aware of the govt. policy.
↳ jiske par rights hen to woh issue modification nikla, ya sell out krlia.
⑥ Misuse. (True bomb etc).

IEEE/ACM Code of Ethics (they properly address the ethics of SD and documented them).

① Public: (Sb public interest pe kam krna).
② Client & Employer: (Interest of client & employer also very important).
③ Product: (When you deliver a product, it should be a/c to standards and best practices).
④ Judgement: Integrity and Independence.
↳ (shows the integrity of the team)
(Judgement me independence ho but ↑
⑤ Management: (Jo standard or best approaches hon unko SD or unki management me apply kna chye).

nion: (promote the excellence in the SE profession).
SE ko image achi rakhni he to ho for its profession ko bura na kahe).
Colleagues: (Support, consider opinions, Respect Privacy etc).
Self: (shall practice life long learning)

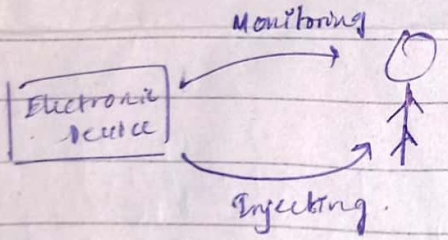
Ethical Dilemmas: (Stuck in any situation, to obey your boss or to deny to work unethically).

Case Studies:

① An Embedded System:

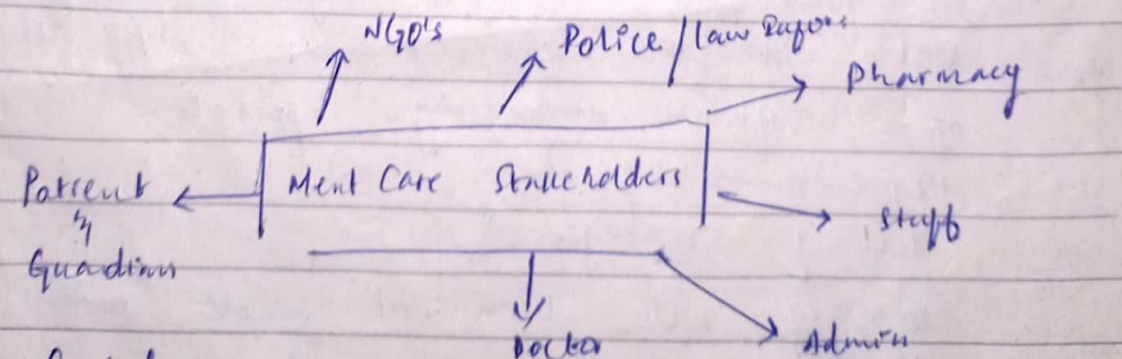
A persanal insolin pump. ⇒ (Critical Nature).
↓
life of a person

Sow How



⇒ Requirements changing is not frequent.

② Information System: (Ment Care).
Deals with mental illness and care. Promises information to different stake holders.



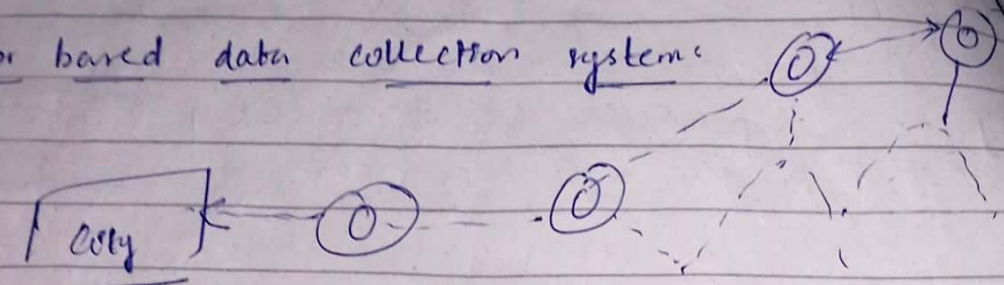
⇒ Access Control } Isko apply krege take
⇒ Privacy Protection } hr stakeholder ko sari info
na den, limited den.

(You will be in trouble)

(You will be successful)

(4)

(3) Sensor based data collection system:



Cctv me hum aap, or ap ne remote
area ki weather condition find
kiti with the help of sensors.

Multiple sensors
WSN
(Wireless Sensor Network)

(4) Digital Learning Environment: (iLearn).
like Courseria, NigiSkills etc....

Users
↳ Student
↳ Teacher

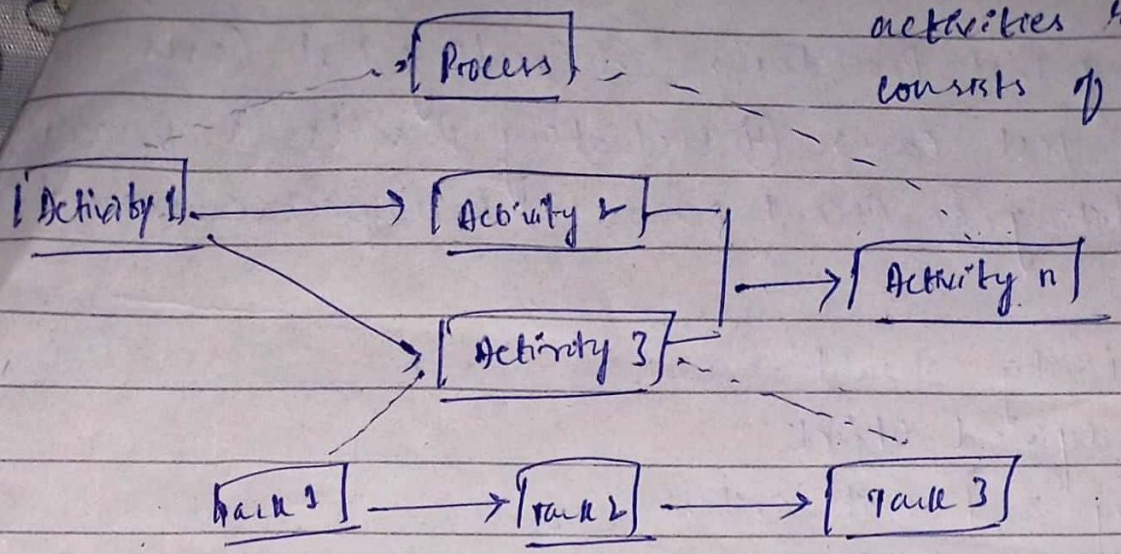
E-commerce
↳ Course Fee

Video Call -

Software Processes: (Essential in Software Projects)
↳ set of related activities. that leads to the production of the software system.

- It is important to measure the progress of a project/software by using SE approach.
- Cost estimation can be done with the help of software engineering approach.
- Requirements Engineering,
- Testing strategies.
- Dealing with technical people (coders etc) is also a challenge. ↗ and non-technical
- " " management concerned with the investment

(Process consists of activities & activities consists of tasks).



Activities:

⇒ **Project Management Phase:**

(i) Creating Processes (ii) setting the standards.
(iii) Managing Risk (iv) Performing estimations. (v) Resource Allocation
(Making measurements) (vi) Improving Processes.

⇒ Specification Phase: → Clear the req. (push the customer to clear the req.) ↗ Elicitation.

(i) Identifying Ideas. (ii) Eliciting Requirements.
(iii) Expressing Requirements. (iv) Prioritizing Requirements.
(Clearing Ambiguity)
(v) Requirements Analysis (vi) Managing Requirements
(vii) Formulating Approaches

⇒ Design & Implementation Phase:

(i) Design Architecture (ii) Design Database (iii) Design Interfaces
(iv) Creating executable codes (v) Integrating (vi) Documentation

⇒ Verification and Validation:

↓ check the consistency of all the inputs and outputs of the system.

↘ check the consistency of user's requirements.
(fulfilling the true need & expectation of the client)

(1) Developing Test Procedures (2) Creating Test Cases
(3) Executing Test cases (4) Evaluating Results. (5) Reviewing and Updating (6) Client Demo. (7) Retrospective record.

—— x ——

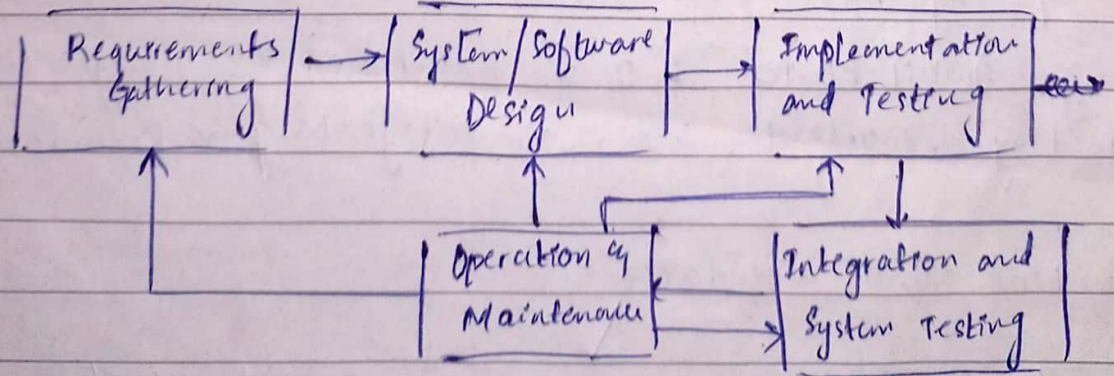
Each activity should have:
(1) Well defined Object
(2) Entry and Exit Criteria.
(3) Inputs and Outputs. → HR, → Tools & Techniques.

Verification & Validation can be applied on each activity seperately & each activity should produce data/info output for management, because each activity is related to cost.

—— x ——

Software Process Models

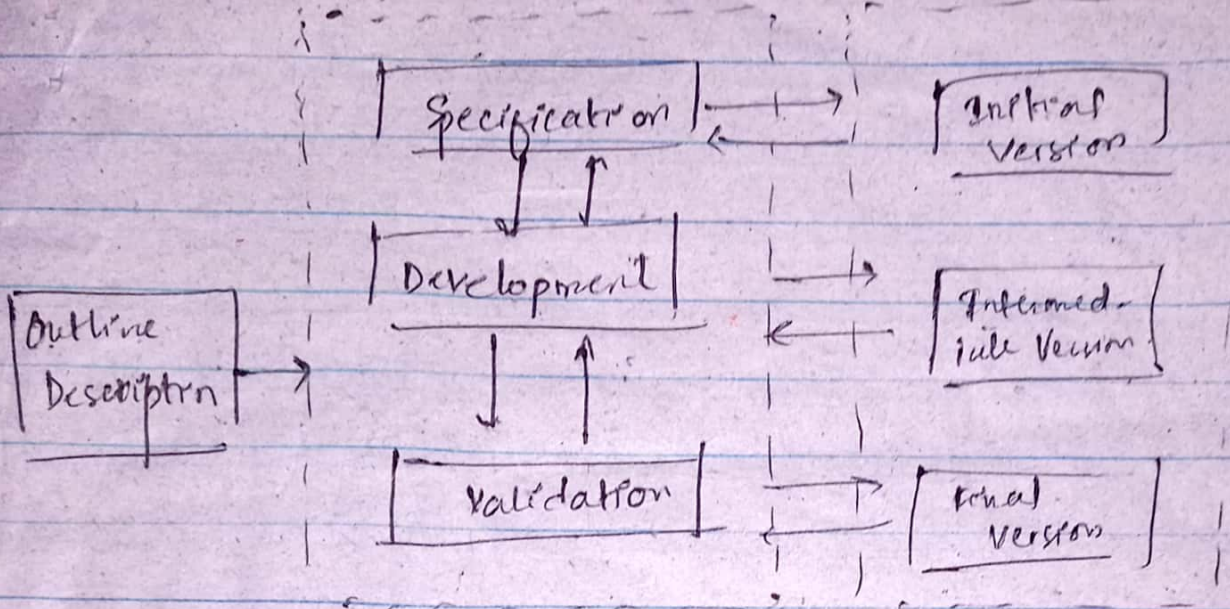
- SDLC (Software Development life cycle)
- Waterfall Model. (Requirements are finalized, embedded system).
- Incremental Development.
- Integration and Configuration.



(6) → modules by Modulis

⇒ Module by Module.
⇒ Customer interaction max.
⇒ Large projects
⇒ Early Release.
⇒ Flexible to changes.

Incremental Development :



Concurrently

The cost of changing customer's requirement is reduced.

Challenges:

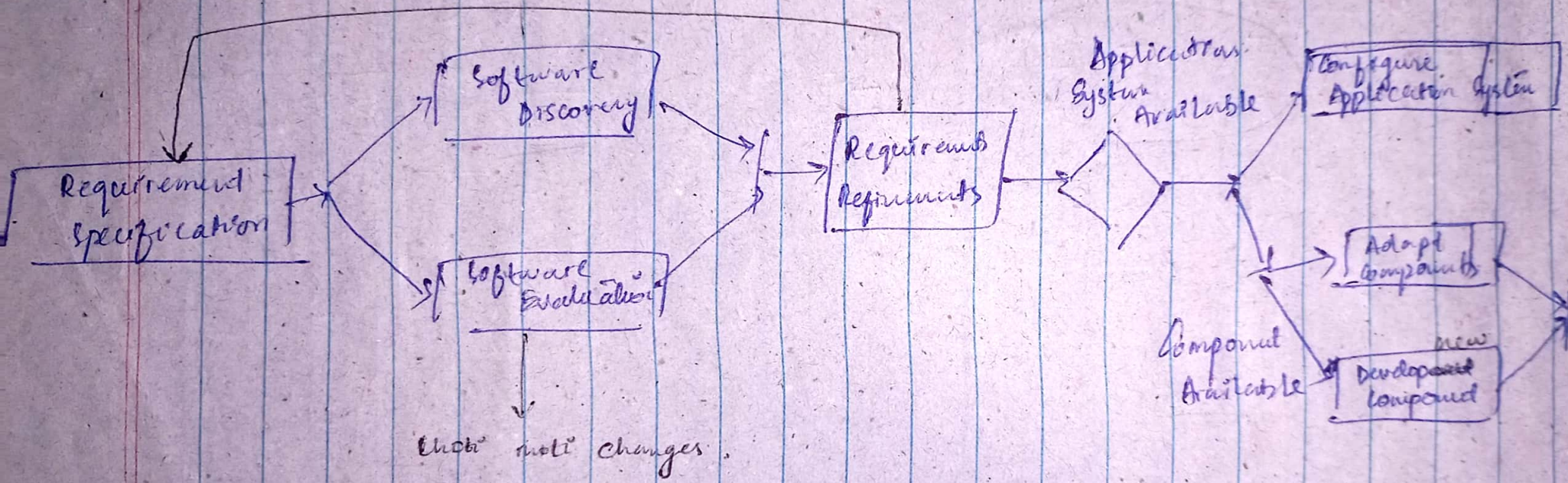
⇒ The whole process isn't clearly visible to the management.

⇒ Integration & Configuration:

COTS - Commercial off the shelf System. (already but this)

- Reused components are adapted to meet user requirement. (changes this)

Req. acc hisab se
konsa software available he



Check mate changes.

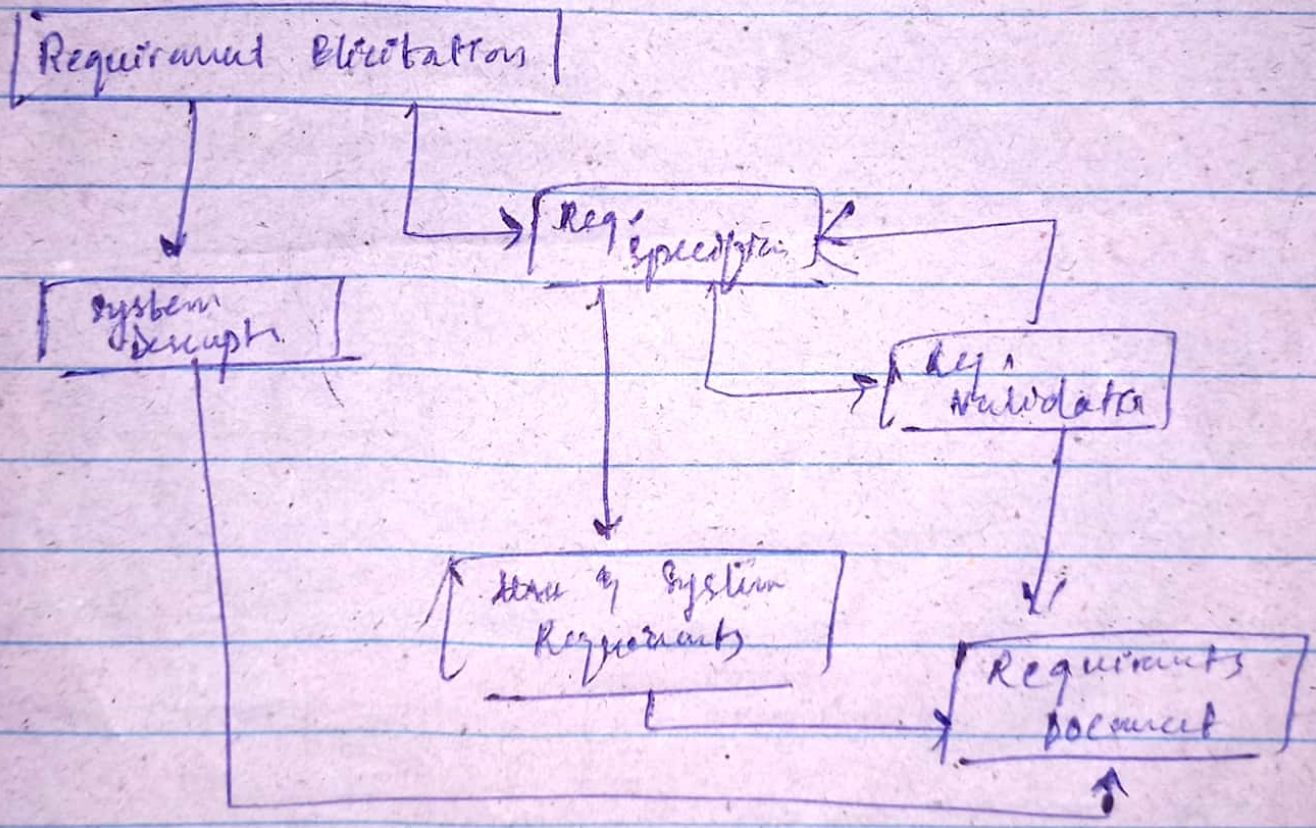
Time ↓
Cost ↓
Effort ↓

Real needs judge nhi hui ✗
Reusable comp. customize ✓

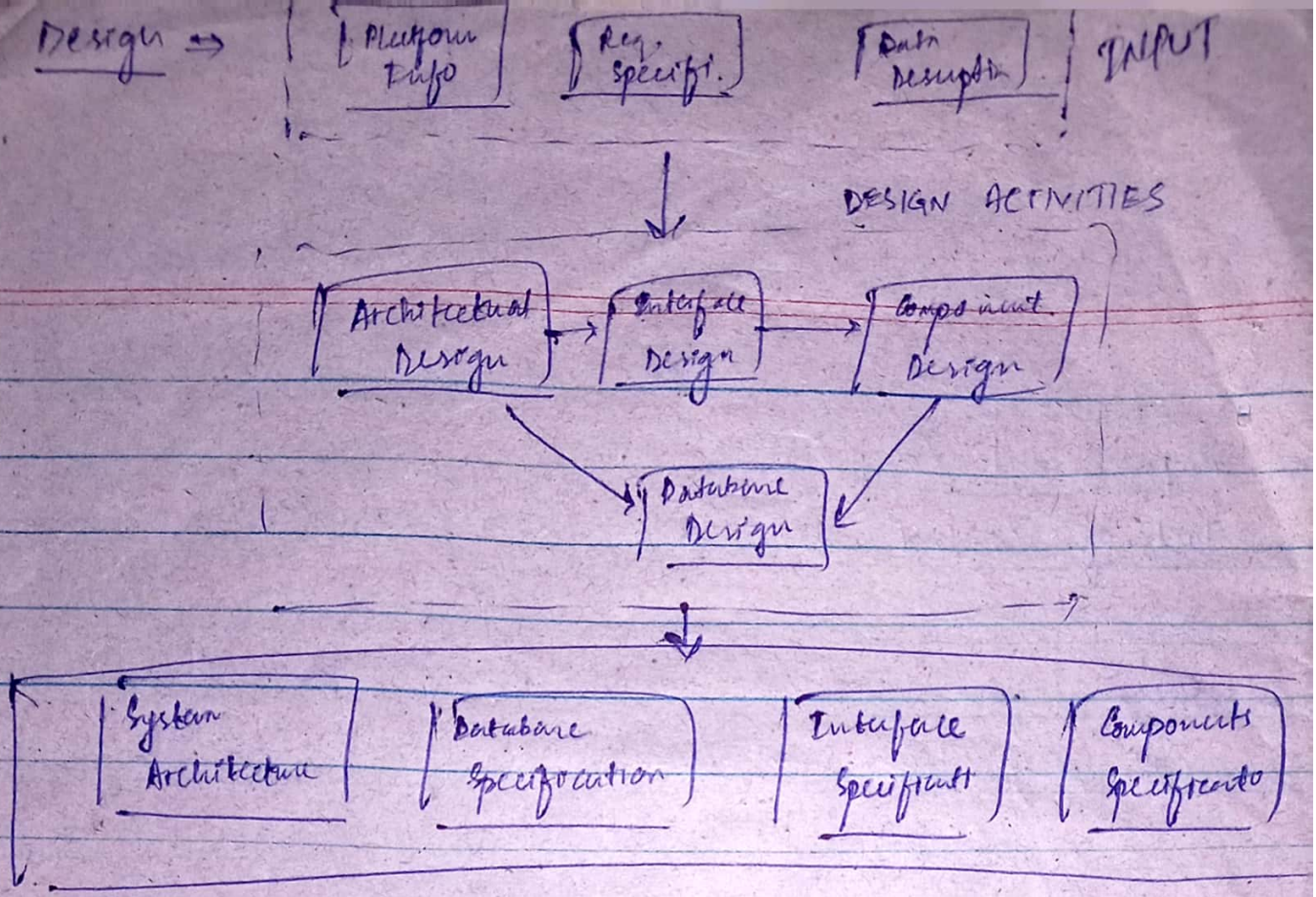
⇒ Requirement Compromises.
lost of control on software evolution.

There four process are always present,
① Specification → Development → Validation
→ Evolution.

① Specification (Requirements Engineering)

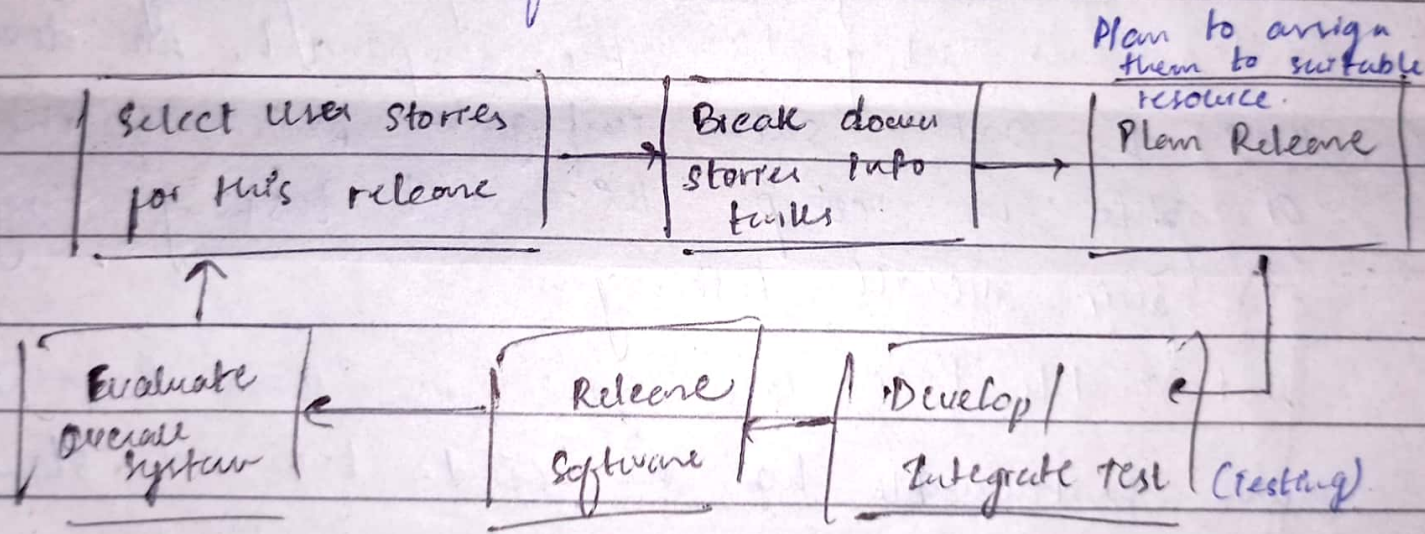


② Development → Design
→ Implementation


Design ⇒
Platform Info
Req. specifi.
Data Descripti
INPUT
DESIGN ACTIVITIES
Architectural Design
Interface Design
Component Design
Database Design
System Architecture
Database Specification
Interface Specificati
Components Specificatio

XP: Extreme Programming

⇒ Developed in 1990s
⇒ consist of multiple techniques.
⇒ related to overall SD process.
⇒ Extreme approach to iterative development.
⇒ new versions may be built (internally) (not customer) several times per day!
⇒ Increments are delivered every two weeks.
⇒ All tests must be run for every build. & the build is only accepted if tests are successful.
⇒ Select user stories for this release.



XP Release Cycle.

⇒ XP Practices:

- Incremental Planning: stories written on one diff task like, plan like, plus done story pe work karte.
- Small Releases: Minimal useful set of functionality
- Simple Design: Enough design is carried out for particular release.

→ Search.
→ Automated tools for
Test first Development

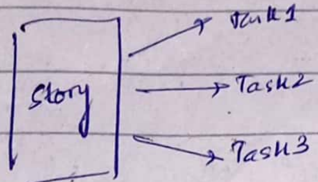
- Test first development: A set of functionality implemented while releasing any release. Kor kam krne se phle test krege.
- Refactoring: All the developers are welcome to simplify the code and refactor it continuously collectively.
- Pair Programming: Developers work in pairs, providing support to each other. (Agr ek chor ke chla gya to doosra take over krlega)
- Collective Ownership: Ek specific part me code nhi krege, poore code ka access hoga. so that agr unke code me kuch glt laga to wo point out krskta he. Anyone can change anything.
- Continuous Integration: Har component ko develop kne k sath sath integrate bhi krte rhege, or phle us comp. ki unit testing hogi or phir overall testing.
- Sustainable Pace: Ese pace me kam krna ke wo sustainable ho, na ziada tez na slow. Greedy approach (bht sare projects le lie, emp. ka time ove hgya, ye sb sustainable nhi h)
- Onsite Customer: Client ka ek banda hmari company me rhega, so that use process dikhae k sath sath feedback leti hen, He should be member of Development team. & he is responsible for bringing requirements.

XP programming methods were adapted in agile.

Influential XP Practice:

Key Practice

① User Stories for Specification.



These tasks have significant impact on cost & schedule.

② Code Refactoring. (Non-functional Requirements)
Benefits: ① Maintainability ② Extensibility

③ Test first development: Test cases banege before development. (Usko login karen to password me error aei, phir kia karege, kha more krege, ye sare test cases bnege).

④ Pair Programming: (Common Ownership) (Well informed team) (Easy to Refactor)

Agile Project Management:

SCRUM is an agile method that focuses on managing iterative development rather than specific agile practices.

↳ incremental development

→ Light weight (complicated nhi krna he, modules me krna he)

→ simple to understand (overall process bhi simple h or thora thora krege to or simple hojaega)

→ difficult to master. (Manager ya lead ko pura difficult hoga ke sb ko sath le k chlae).

2001 me manifesto meeting hui thi us me finalize hua tha.

Book (Mike Beedle, Agile Software Dev with Scrum) 2001

2002 me Scrum Aligns Alliance ki certification start ki.

2009: scrum.org ... scrumguides.org.

Scrum 3 phases:

① Initial Phase
(an outline planning)
- General Objectives
- Define Software Architecture.

② Intermediate Phase:
(a series of sprint cycle).

③ Closure Phase:
(the project wrap up).
- Finalise / Complete Documentation
- Assesment

⇒
- SCRUM
- Sprint
- Velocity
- Dev. Team
- Scrum Master

(2) Interaction Models:

(1) Use Case (2) Sequence.

→ Helps to identify user requirements.

→ Represents interaction b/w components of system.

(3) Structural Models:

(1) Class Diagram.

(4) Behavioral Models:) that ke action hote hen ek data ke upr or ek events ke upr.

Data → Data driven → Activity Sequence

Events → Event driven (minimal data) → State Diagram

(1) Development Testing:

i, Unit Testing ii, Component Testing iii, System Testing

↓ Automated

↓ Interface Testing

↓ usecase

(2) Release Testing: (Black box)

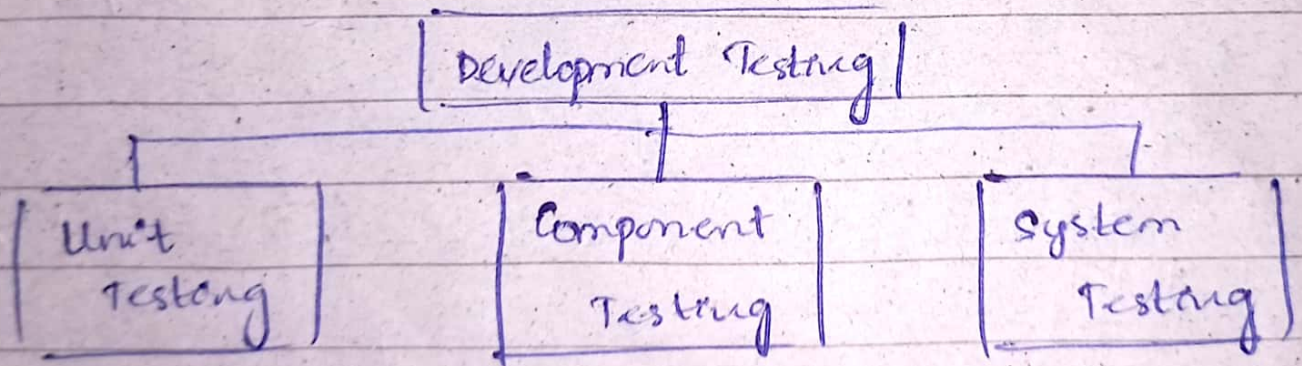
↳ Performance, Req based

(3) User Testing:

i Alpha ii, Beta iii, Acceptance

(1) Development Testing: where the system is tested during development to discover bugs & defects.

There are three stages of development testing:



It is a defect testing process, where the aim of testing is to discover bugs in the software.

dependability nhi hoti, data direct client k
par se ashai, ADI wgera nhi h leoi

f. Unit Testing:

stem Testing ⇒ It is the process of testing program components, such as methods or object classes.

usecase

⇒ It is a defect testing process, where the aim of testing is to discover bugs in the software

⇒ Agar hm ek class object ko test krege to us se related sare operations ko test krna hoga.

ested

fects. ⇒ Hm diff. test cases bnate hen or apne sys. ko test krte hen.

ting:

⇒ login / signup form he to us me validations hugo multiple, or agr hm one by one test krne beth gae manually to bht lambi hogi.

⇒ Is ke liye hm automated testing use krte hen. Kuch ese tools or frameworks hte hen jo hmen provide krte write or run your test case.

the

An automated testing has three parts:

1. A setup part, where you initialize the system with the test case.

Automated f1. integrate krdete hen code se.

iii. A call part, where you call the object or method to be tested.

iii. An assertion part, compare result of the call with expected result. If assertion evaluates to true, the test has been successful; if false, then failed.

~~three~~ Two testing strategies: → guidelines ko follow krke unit testing. guidelines prev. exp se bnai hogi phle jo error face kie unge.

① Partition testing ② Guideline based Testing.

ii) Component Testing:

Login (Page) → Authentication (Login/Signup)
↓
Component.

→ Also known as Integration Testing.

ii. System Testing:

→ Ab poora system test krege.
→ Comp. jo integrate nhi hue the phle unhe put kia pher system test kra poora.

(2) Release Testing:

→ Release Testing ki team project me include nhi hoti he.

→ Hm unke project dekr hen or testing krwate hen.

Happy flow testing: Part of unit testing, sahi inputs daal k check kre.

Abnormal Testing: Glt inputs daal k chk kra. (Checking error handling).

Interface Testing: Part of component testing, ek interface input le rha,

(Adv of TDD).

Regression Testing: Ap ne jo new code likha he usse aaja se purana wala break to nhi hogaya.

If yes, to apko pata chl jaega.

Manually exp hogi, so automated krte hen.

ii) Requirement Based Testing:

→ Client ki req fulfill hoti h ya nhi.
→ I/O to bata check krein client ne jo kaha tha woo hota ya nhi.

Use-Case Testing: (Part of System Testing)

Use-cases ki help se hi testing karte hai ki jese use case banae wese kam ho bhi rha ya nahi.

Strategies:

① Partition Testing: → input of same group.

→ Black box testing approach.
→ partition h 6 digit no ka, exactly woo likhege to chlega warna error page pe le jaega

6 digit wale pe hi show krega.

③ User Testing:

User ko deke hai or woo testing karta he.

→ Alpha Testing: User hmare pas aya

or usne testing kri, dev. team k sath beth k.

→ Beta Testing: Limited users ko dedete hen, testing k lie.

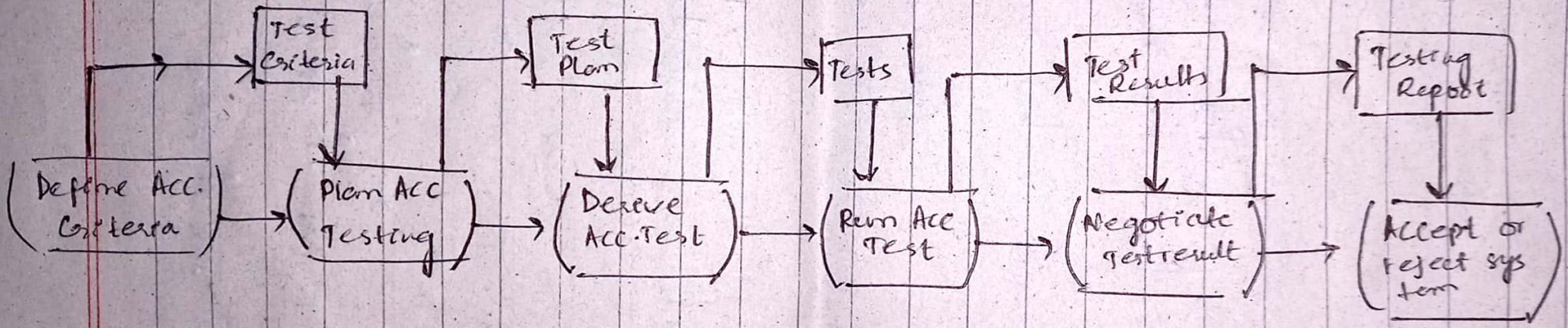
Jb os me mistk identify hjate hen jbhi isko Sahe kste.

→ Acceptance Testing: Uses dekhta jo req. he uske wo accept hrti ya nhi.

Jis diz k lie bna he wo kam hiha ya nahi.

Sari req fulfill huhi ya nahi.

Agile me alag se acceptance testing nhi hoti kyu be user shru se ho rhrshte hta he hmare sath.



Acceptance Testing.

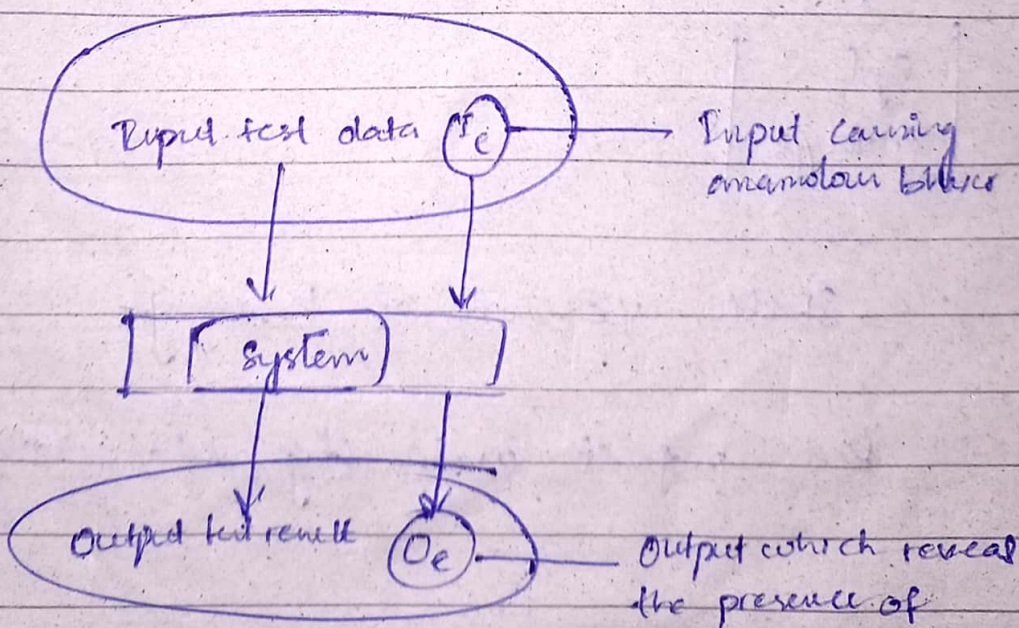
Program Testing :

→ Project run kia, test kia, jo defect he wo identify kia.

→ Ek or chiz achieve kiye hen successful hoen, k jo user ne manga he wahi chiz mil rhi ya nahi.

→ Iske 2 goals hte hen, phla ye k jo chye user ko

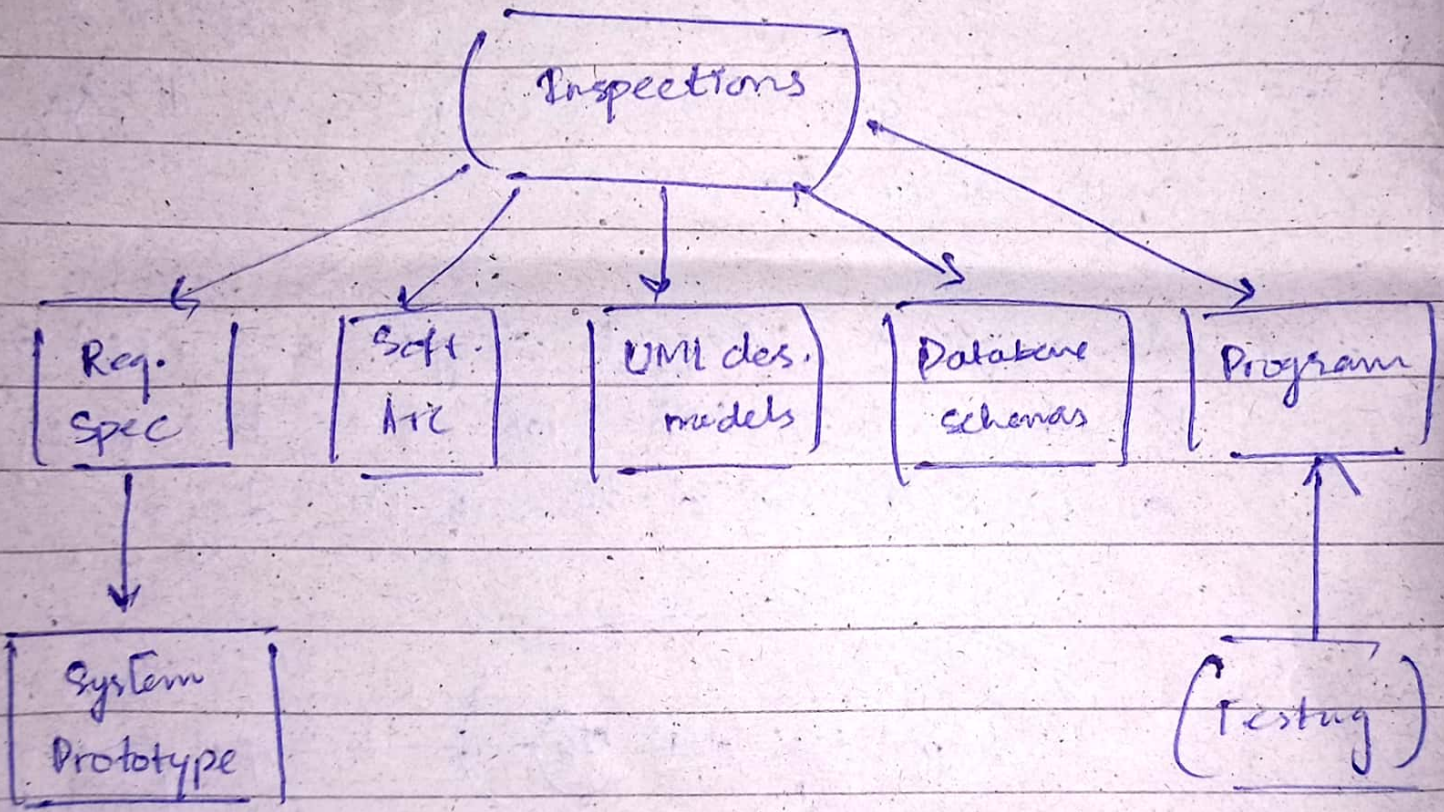
→ dosra ye k defect dega, esa nhi hoga k kam nhi krha or error bhi nhi arha, phir validation ye k dega.



→ Inspection Testing (Program Testing):

Kahi se kuch bhe catha k check ke skte.

Srf prog. run ks k. nhi, coster design, doc, specs kuch bhi.



Static version of testing.

Run project wali dynamic testing.

Test Driven Development

Enhanced form of Test first development.

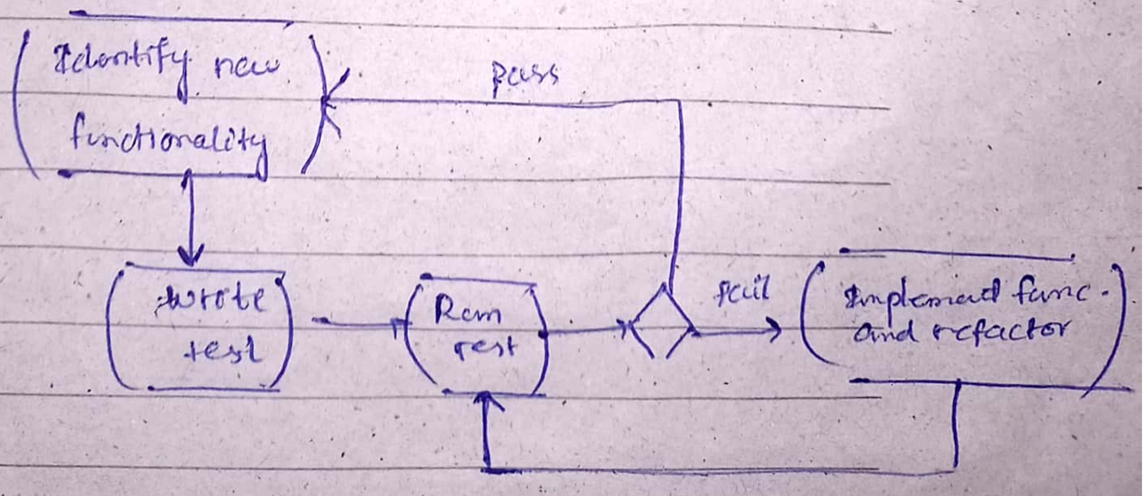
Phle test likhte hen, phir code likhte.

Testing sath sath hote he.

Incremental approach use krte hen.

Decide krlete hen next few lines me hmen kia krna he, test likhte hen phir run krte baki poreme k sath ta k ye na ho k end sahi chl rha lekin ek sath mile les rha.

Ab code likhege, pass hogaya (test pass hogaya) to next one pe move krege.



Verification VS Validation

↓ Are we building the product right?

↓ The software should conform to its spec

↓ Are we building the right product?

↓ Software should do what user really requires.

1. Code Coverage.
2. Regression Testing.
3. Simplified Debugging
4. System Documentation.

Requirements Engineering

Date

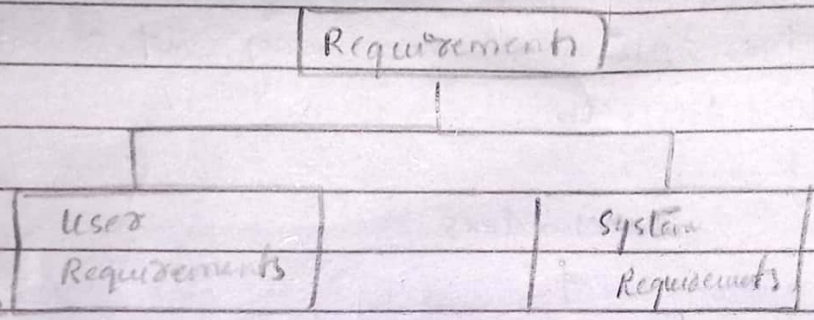
The process used for Requirements Engineering vary widely depending on the application domain, the people involved and the organisation developing the requirements.

Requirements:

"The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organisation's needs"

- Prof. Alan M. Davis.

Types of Requirements:



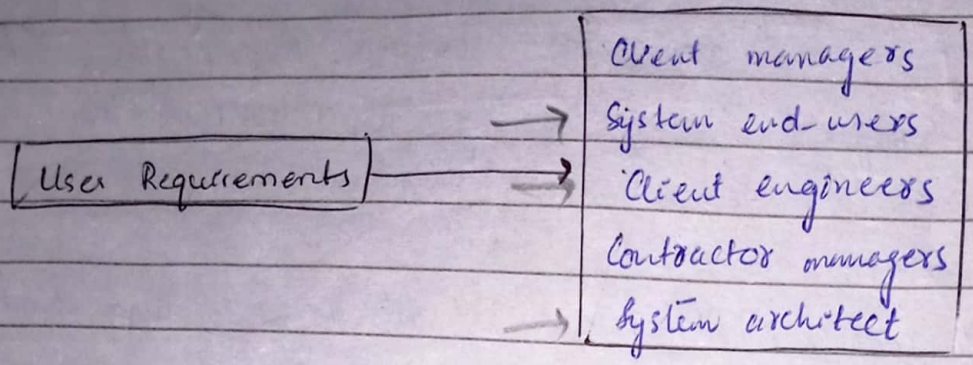
User Requirement ~~Specification~~ Definition.

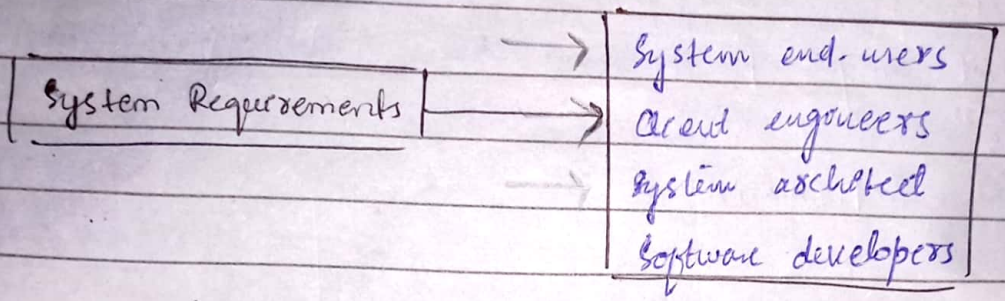
The Medcare system shall generate monthly management reports, showing the cost of drugs prescribed by each clinic during that month.

System Requirements Specification

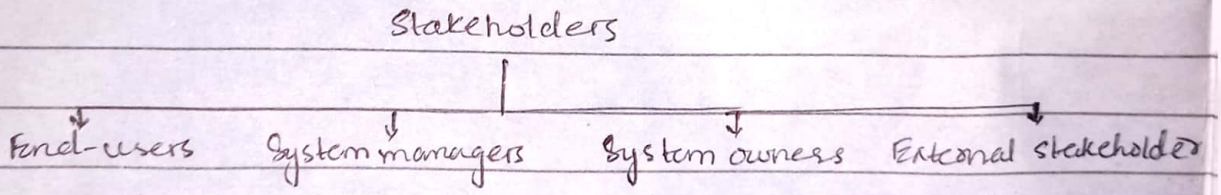
On last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.

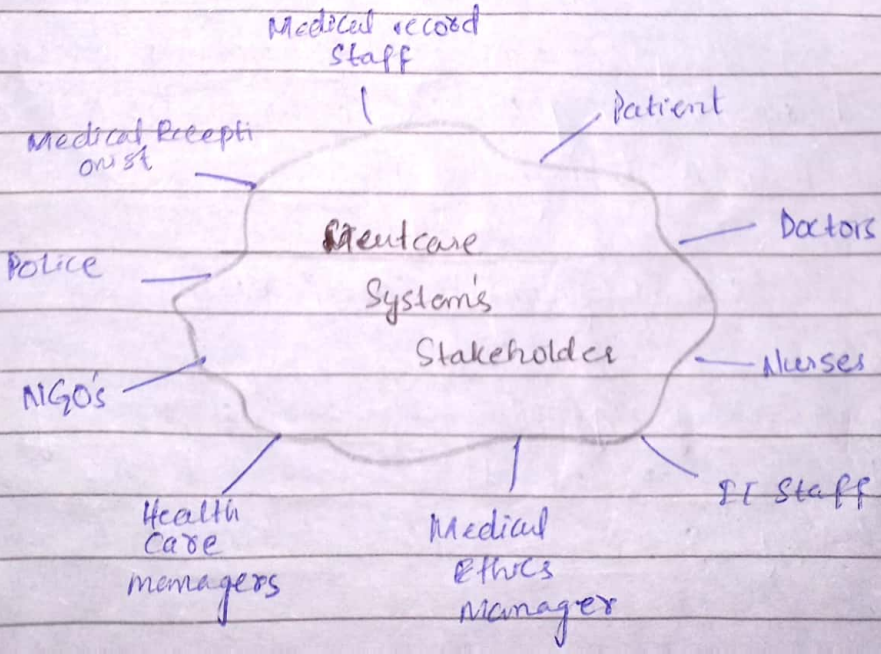
Readers of Different Types:





System Stakeholders: Any person or organization who is affected by the system in some way and so who has a legitimate interest.





System Requirements

Functional Requirements

- Describe functionality or system service
- High level statements of what system should do.
- FR of MC System:
  → A user shall be able to search the appointments lists for all clinics.

Non-functional Requirements

- They define system properties and constraints e.g. reliability response time & storage requirements.
- Particular IDE, programming language or development method, security requirements
- Not met, system is useless.
  → Product Requirement
  → Organizational "
  → External Requirement.

Metrics for specifying non-functional requirements:

(1) Speed (2) Size (3) Ease of use (4) Reliability
(5) Robustness (6) Portability.

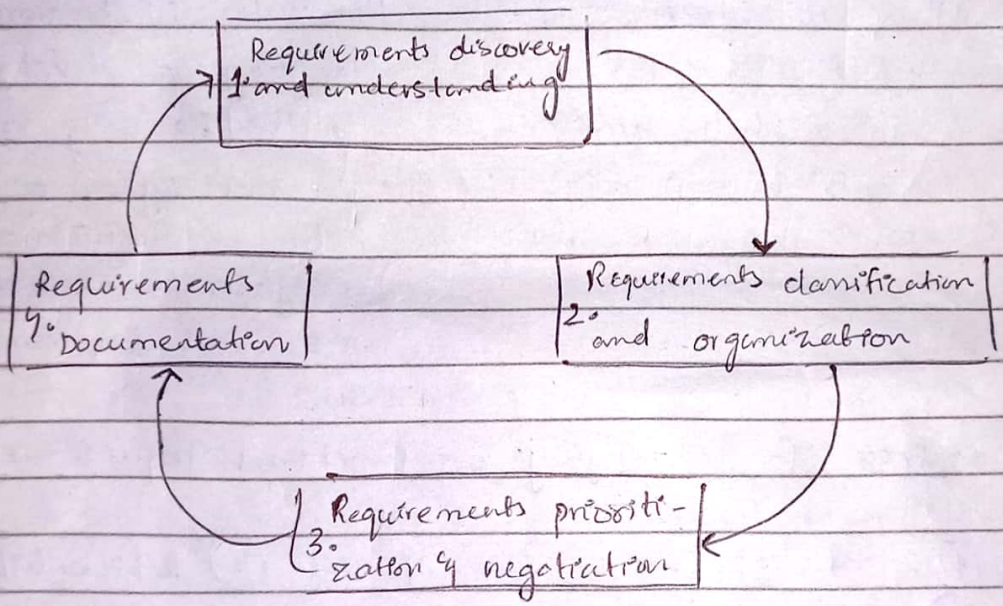
Requirements Engineering:

The broad spectrum of tasks and techniques that lead to an understanding of requirements is called Requirements engineering.

It involves three key activities:
- Elicitation and analysis
- Requirements specification
- Requirements validation

Requirements Elicitation:

During requirements elicitation, software engineers work with stakeholders to find out about the (1) application domain, (2) work activities, the (3) services and system features that stakeholders want, the (4) required performance of the system (5) hardware constraints and so on. Eliciting and understanding requirements from system stakeholder is a difficult process for several reasons.



The requirements elicitation & analysis.

1. Requirements discovery and understanding: This is the process of interacting with stakeholders of the system to discover their requirements.
- Domain requirements from stakeholders and documentation are also discovered during this activity.

2. Requirements classification and organization:

- This activity takes the unstructured collection of

requirements, groups and related requirements and organizes them into coherent clusters.

3. Requirement prioritization & negotiation:

- When multiple stakeholders are involved, requirements will conflict.
- This activity concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation.
- Usually, stakeholders have to meet to resolve differences and agree on compromise requirements.

4. Requirements Documentation:

The requirements are documented and input into the next round of the spiral.

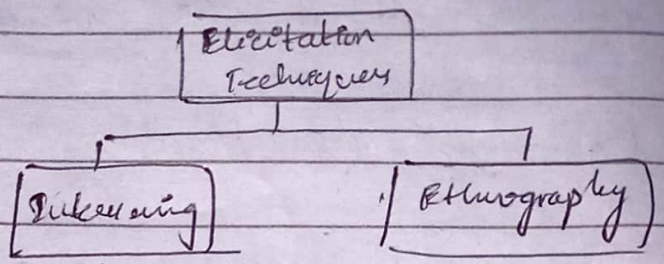
An early draft of the software requirements document may be produced at this stage,

~~Or~~ the requirements may ~~be produced at this stage~~ simply be maintained informally on whiteboards, wiki's or other shared spaces.

At this stage, it is important to use simple language and diagrams to describe the documents.

Techniques of Requirements Elicitation:

1. Interviewing
2. Ethnography.

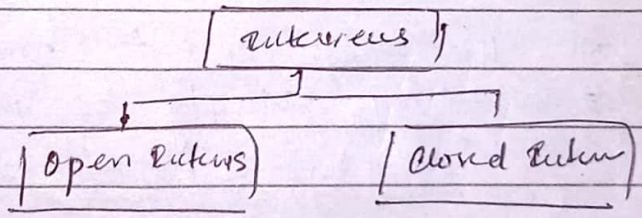


1. Interviewing:

- Formal or informal interviews with system stakeholders are part of most requirements engineering processes.
- In these interviews, the requirements engineering team puts question to stakeholders about the system that they currently use and the system to be developed.
- Requirements are derived from the answers to these questions.

Interviews may be of two types:

Closed Interviews
Open "



(2)

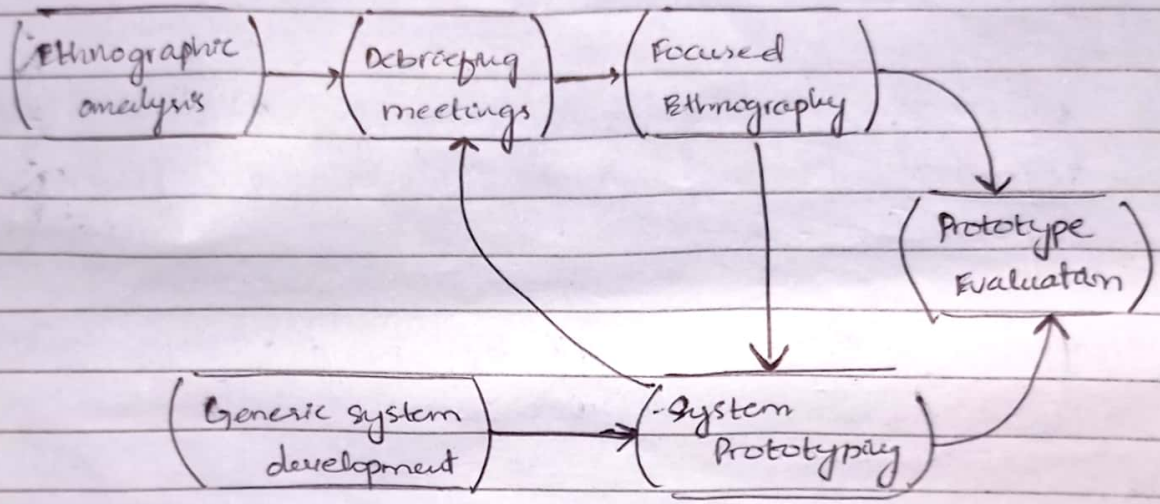
2. Ethnography:

It is an observational technique that can be used to understand operational processes,

and help derive requirements for software to support these processes.

(2)

- An analyst immerses himself or herself in the working environment where the system will be used.
- the day to day work is observed and notes are made of the actual tasks in which participants are involved.



Ethnography and prototyping for requirement analysis.

Problems with interviews:

(1) All application specialists (users) use jargons specific to their area of work that requirement engineers may misunderstand.

(2) Some domain knowledge is so familiar to stakeholders that they find it difficult to explain or they think it is so fundamental that it is not worth mentioning.

(2) Requirements Specification :

- The process of writing down the user & system requirements in a requirement document.

- Users requirements have to be understandable by end-users and customers who do not have a technical background.

Where as, system requirements are more detailed requirements and may include more technical information.

The requirements may be a part of contract for the system development.

28

Ways of writing system requirements specifications.

- (Natural language) - used numbered sentences.
- (Structured Natural language - natural language on a standard form or template.
- (Design description languages - used a language like programming language.
- (Graphical Notations) UML, use cases & sequence diagrams.
- (Mathematical specifications) - finite machines or set.

Guidelines for writing requirements : (Natural language)

⇒ Invent a standard format and use it for all requirements.

⇒ Use "shall" for mandatory requirements, "should" for desirable requirements.

⇒ Use text highlighting to identify key parts of the requirement.

⇒ Avoid the use of computer jargons.

⇒ Include an explanation (rationale) of why a requirement is necessary.

✗ Problems with Natural language:

1	Lack of clarity:	Precision is difficult without making the document difficult to read.
2	Requirements Confusion:	Functional & non-functional requirements tends to be mixed up.
3.	Requirements Amalgamation:	Several different requirements may be expressed together.

Structured specifications:

⇒ An approach to writing requirements where the freedom of the requirements writer is limited and the requirements are written in a standard way.

⇒ This works well for some types of requirements e.g. requirements for embedded control system.

Form based specifications:

- A desc. of function or entity
- Desc. of inputs & where they come from.

- Desc. of outputs & where they go to.
- " " actions to be taken.
- Pre & Post conditions.
- The side effects of the functions.

Tabular Specifications:

Example of Insulin Pump:

Condition	Action.
• Sugar level falling ($r_2 < r_1$)	CompDose = 0
• " " stable ($r_2 = r_1$)	"
• " " increasing and rate of increase decreasing	"
• Sugar level increasing and rate of increase stable or increasing	CompDose = round ((r2 − r1)/y)) If round = 0 then : CompDose = MinimumDose.

(3) Requirements Validation:

It is the process of checking that requirements define the system that the customer really wants.

It is very important because the errors in the req. doc. can lead to extensive rework costs when these problems discovered during development.

There are different types of checks in the req. validation process. Some of them are:

Date (a)

Checks of Req. Validation

- Validity Checks
- Consistency Checks
- Completness checks
- Realism Check
- Verifiability

i) Validity checks: Checks that the requirements reflect the real needs of system user.

ii) Consistency checks: Req. in the doc. should not conflict.

iii) Completness Check: Req. doc. should include all functions & constraints intended by system user.

iv) Realism check: By using knowledge of existing technologies the req. should be checked to ensure that they can be implemented within the proposed budget for the system.

v) Verifiability: Write a set of tests that can demonstrate that the delivered system meets each specified requirement.

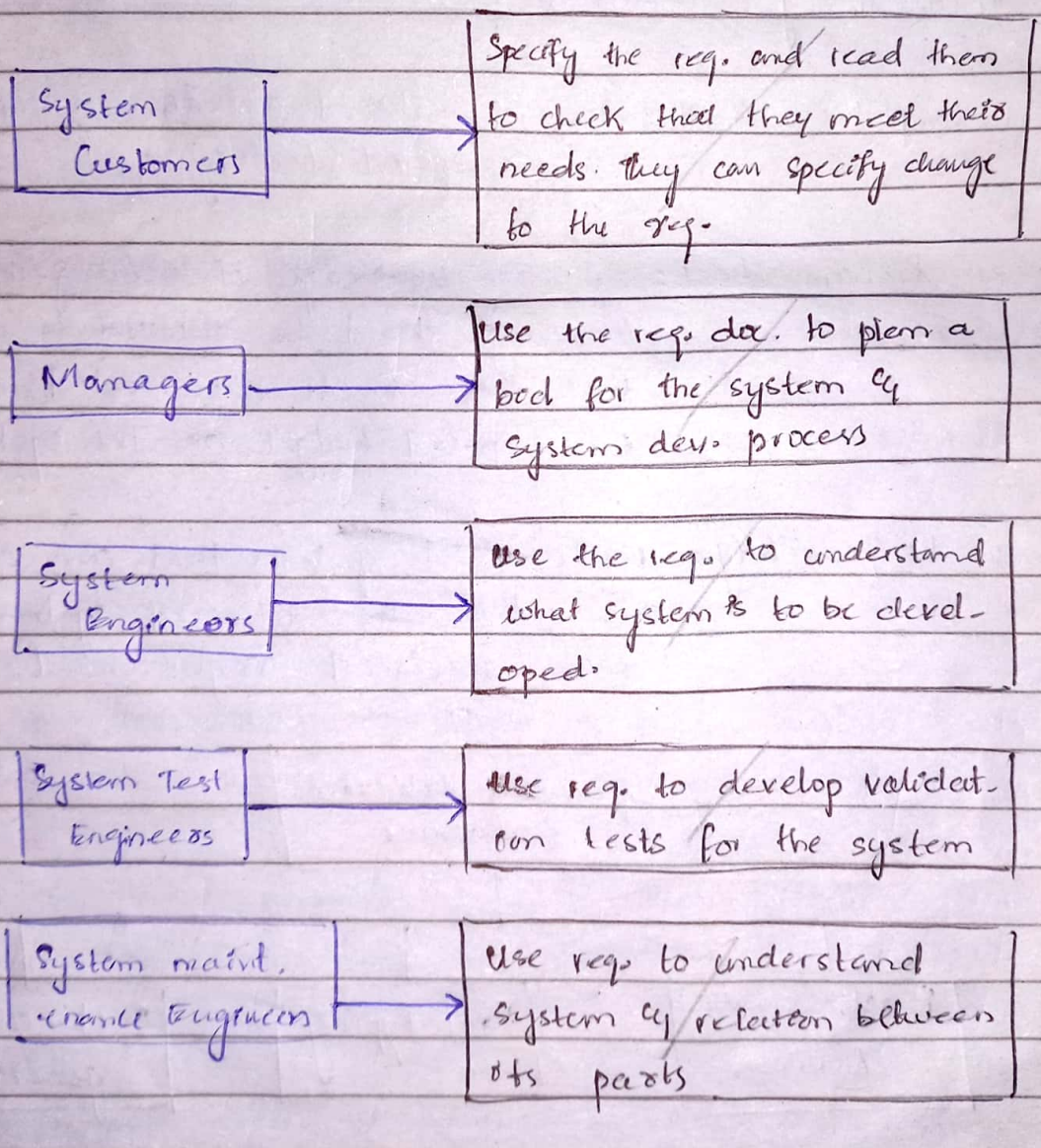
Requirement Validation Technique

- Requirement Reviews
- Prototyping
- Test-case generation

Software Requirements Document

Date (13)

- The software req. doc. (sometimes called the software req. specification or SRS) is an official statement of what the system developers should implement.
- It includes both user & system requirements.
- The below figure shows possible users of the document and how they use it.



IEEE Standards for SRS. Date 13

Chapters	Description.
1. Preface	This defines the expected readership of the document & describe its version history, including rationale for the creation of new version & summary of changes made in each version.
2- Introduction	Describes the need for the system & briefly describe the system's functions & explain how it will work with other systems.
3. Glossary	This defines the technical term used in the document.
4. Users requirements definition.	Describe the services provided for the user. It uses natural Language, diagrams or other notations that are understandable to customers.
5. System architecture	Presents an high-level overview of the anticipated system architecture.
6- System requirements specifications	Describes the functional & non-functional requirements in detail.
7. System models.	This chapter includes graphical system models ~~showing the rela~~ such as object models, data-flow models, or semantic data model.
8. System Evolution.	Describes the fundamental assumptions on which the system is based, changing user needs and so on.

9. Appendices	Provide detailed, specific information related to application being developed. For example hardware & database descriptions.
10. Index	Several indexes to the document may be included like alphabetic index, index of diagrams, index of functions and so on.

Security Engineering

- sub-field of computer security. • Data to backstop the malicious attacks so as to damage risk the han.

Credit card details from an system

Security Dimensions: (Three security dimensions you have to take to secured system).

1) Confidentiality: Info. disclosed to people or program that are unauthorized.

2) Integrity: Info. may be damaged or corrupted, making it unusual or unreliable.

a worm that deletes data in a system

3) Availability: Access to a system or its data that is normally available may not be possible. (Server down bcoz to system is overload compromise bcoz).

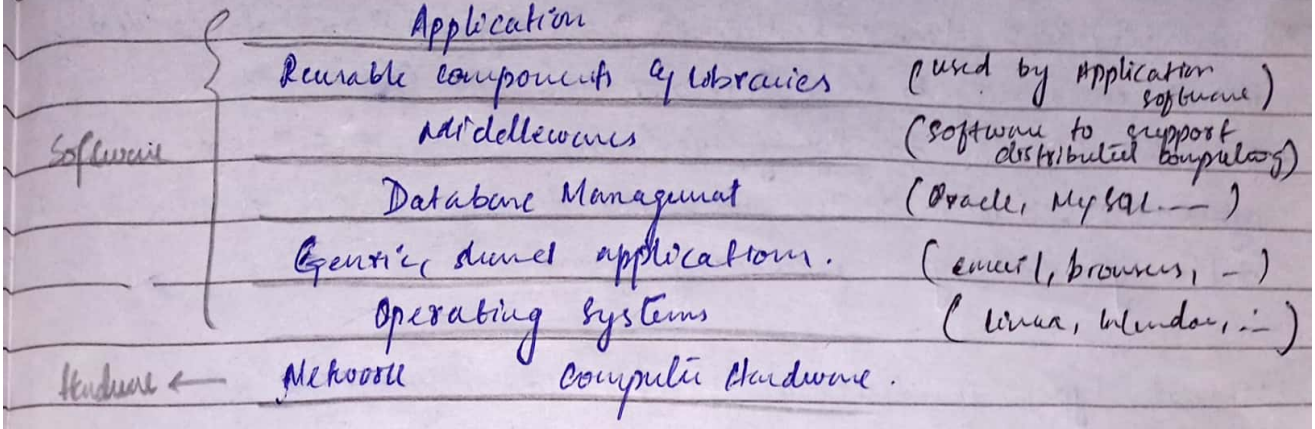
Security levels:

i) Infrastructure security: Concerned with maintaining the security of all systems & networks that provide an infrastructure & set of shared services to organization.

ii) Application security: concerned with the security of individual applications systems or related group of systems.

iii) Operational security: concerned with the secure operation and use of the organization's systems.

System layers where security may be compromised.



Here, we can see how an application system relies on infrastructure of other systems in its operations.

Our focus.

Application Security VS Infrastructure security

↓ ↓

SE problem where system is designed to resist attacks.

System management problem, inf. configured to resist attac

⇒ Operational Security: • human & social issues • logged on • Acc pro.
• there is trade-off b/w sys. security & sys. effectiveness.

⇒ Dependability: Dep. of a computer system is a property of the system that reflects its trustworthiness.

i, Availability: The ability of system to deliver services when requested.

ii, Reliability: The ability ---- deliver services as specified.

iii, Safety: Ability ... to operate without catastrophic failure.

iv, Security: Ability to protect itself against deliberate or accidental intrusion

v, Resilience: To resist & recover from damaging events.

i, Security & Reliability: If system is attacked, & data is corrupted due to this attack then this may induce system failures that compromise the reliability of the system.

ii, Security & availability: A common attack on a web-based system is denial of service attack where a web server is flooded with

iii, Security & Safety:
→ Attack hua system pe or data corrupt hogaya, ab hm us system ka code analyse kr ke dekhenge k execution or source code same he ya nhi, agr nhi hua to mtlb safety-related failures may be induced & safety for software is invalid.

iv, Security & Resilience: Resilience is a system characteristics that reflects its ability to resist & recover from damaging events.
• The most probable damaging event on networked soft. sys. is a cyberattack, so most of the work now done on resilience is aimed at deterring, detecting & recovering from such attacks.

———×———.

Threat types: i, Interception Threats: Allow attacker to gain access to an asset. (access to the records of individual patients).
ii, Interruption Threats: Allow an attacker to make part of the system unavailable (denial-of-service attack on DB server so DB is unavailable)
iii, Modification threats: Allow attacker to tamper with a system asset. (interfere & cause damage)
(Ex MCS, attacker alters or destroy a patients record).
iv, Fabrication threats: Allow attacker to insert false information into a system. (false transactions add & send money to attacker)

Security Assurance: ① Vulnerability Avoidance (No external system) ② Attack detection and elimination (Antivirus) ③ Exposure limitation & recovery. (backup policy).

Organizational Security Policies: Hr organization ke security policies hoti hen or agr nhi hoti hen to bnani prti hen.
i, The assets that must be protected: 80 k lie nhi, srf confidential k lie (vrna cost issue)
ii, The level of protection that is required for diff. types of assets:
→ Not all assets needs the same level of protection. Sensitive personal info - high level of security required, for other, the consequence of loss may be minor, so lower level of security required.
iii, The responsibilities of individual users, managers & the organization: use of strong pw, log out of comp, lock offices, etc.
iv, Existing security procedures & technologies that should be maintained
Jo existing system hen security k jese login, Authentication, token system etc. Amen budget ko dekhte hue wahi use krlene chye, khud bnana zaruri nahi, although hmen unki limitations ka pta he but phir bhi wahi sahi hen budget k lehaz se.

Date

Risk Management: Manager ka kam he k risk ko assest kre ya manage kre. Hm sirf ye nhi dekhege k hackers se kia risk hen software ko bl k hm sochte esa na ho team chor k chli gae kon atak gae, etc. Risk management involves:
① Preliminary RA ② Life cycle (Design) risk Asset. ③ Operational RA.

① Operational Risk Assessment: • Because of human error. • Social or human issue • log out nhi kia • Share pw • save krdia pw. ab ye chize koi misuse krskta he, entertainment k lie bhi misuse bhe. Hmen team ko aware/guide krna he tak wo esi chizen na kren.

② Design Risk Assessment: Zaruri nhi ke phle risk assestment krlen or phir dev. ho. Software Dev. life cycle me kisi bhi stage pe aake risk assesment/management kr gaskte he, developers dev. krte hue bhi krskte hen agr unhe koi vulnerability dikhe to, Security req. change hugi & addition of new req.

③ Preliminary Risk Assessment: Ye primary approach hoti he risk assestment ki beginning of project me krte hen. Analysis of high level risk of system. Outcome: Identify security requirements

Types of Security Requirements: (Picture) & Classification.

The preliminary risk assestment process for security requirements.

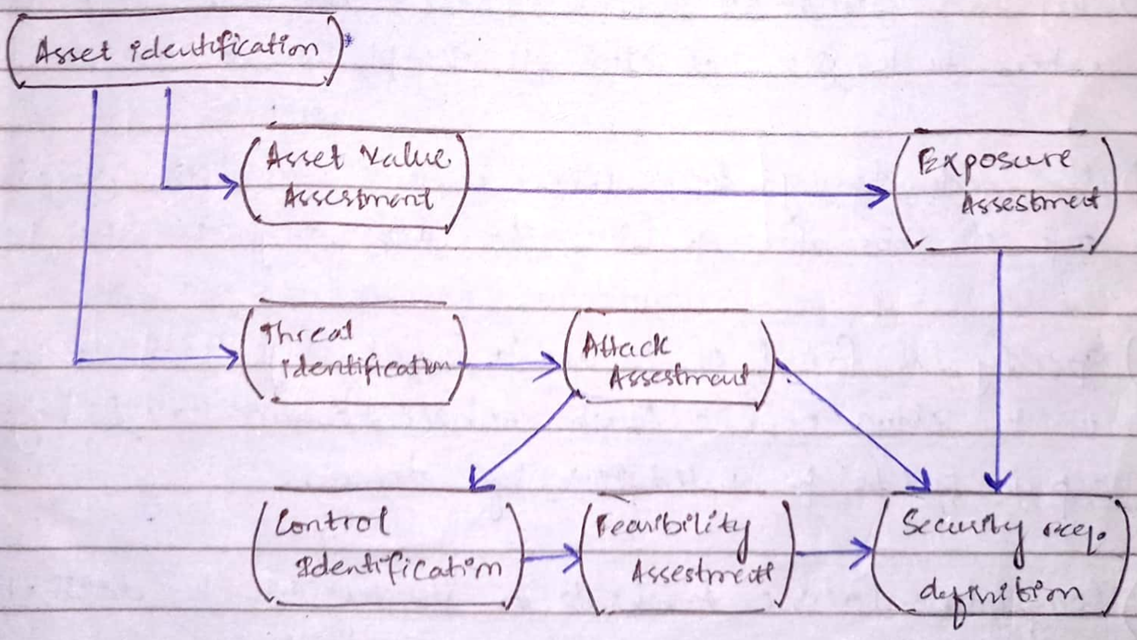


Table (Picture), Misuse Case,

Design Compromises: Agar bht security lga di to us se kia kia msle hote hen, password validation (frustrated user), session expire, check ur one example etc.

Design Guidelines for secure systems engineering:

1) Base Decisions: Kuch policies bna denge, jese koi bhi apna pw share nhi krega kisi se, or agr ap ne kia to action lia jaega, case hoga, ye project ya ye report confidential h etc.

2) Avoid Single point of failure: Agar pw hack bhi hgya to two factor auth usko bcha le, failure to ho ho jb sare layers fail hojae, ek k hone se fail na ho.

3) Fail securely: Agar system fail bhi hogaya he to uska data phr bhi secure rhe esa na ho k bilkl hi open source hjae.

4) Balancing Security: Pw ko bhi sare validations etc.

5) log user actions: koi system pe kia kr rha he, uski history rkhen ta k jesi koi kuch glt attempt kre to hm pkr len use.

6) Use redundancy: copies rkhen data ki, multiple jagah k agr ek bhi damage hua to dosri jaga se fetch krlen.

7) Specify the format of inputs: jo input chye wahi enter ho us k elawa na ho kuch wrna problem cause hogi, so input fields pe validation lga denge.

8) Compartmentalize Assets: Jo Dr. jis patient ko dekh rha he use srf uski info mile ye na ho k baki sare patients ki mil jae. Dr ek patient ki detail uska hi Dr dekh skega sare Dr nhi to risk reduce hjaega.

Date

9) Design for deployment: Design the system to avoid deployment problems (take security measures so Google play store won't reject).

10) Design for recoverability: Design the system to simplify the process of recovering of data after attack.

—— x ——

System Modeling

- Process of developing abstract models of a system
- System ka view ya perspective show krte he.
- It is now almost based on notations in the UML.
- It helps the analyst to understand the functionality of system & models are used to communicate with customer

Existing & Planned system Models: Agar humare pas existing models hen to hm un me thori modifications kr k use krskte hen. Or new models create krte hen during req. specifications.

Types of Models:

① Context Models (external perspective):
→ Model the context or environment of the system.

② Interaction Models: (Interaction Perspective)

Context → Context Activity → Business Process Model.

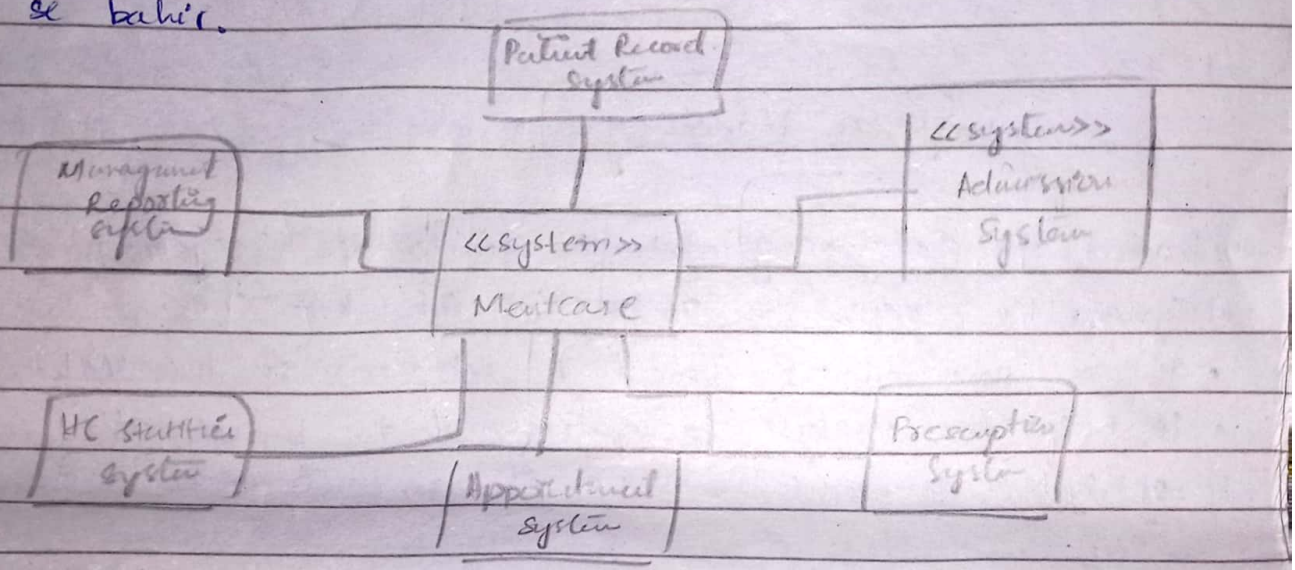
Context Models

Date

→ Ek system boundary bna denge, jese koi 3rd party system hai to wo hm ne boundary ke bahir rkha.

→ 2 tarike ke boundaries bnaege.

→ Boundary ke endr apna system eg sub system rkhege or baki ke 3rd party systems or stakeholders ko boundary se bahir.



Context Model – show other systems in environment, not how the system is used.

Process Model – reveals how the system is used.

Activity diagrams used to define business process model.

↓

Diagram of MC