Page No.:

Software Engineering

↳ Solve a problem in a systematic manner

→ As softwares are becoming more and more complex, we need a step by step and disciplined manner to write the entire software code.

→ Software life cycle ends when it becomes obsolete.

→ CMM (Capability Maturity Model)
↳ five levels.
(each company is rated according to standards followed by them)
↳ level five for company following all standards

→ Risk Management

→ Requirement Engineering (Requirement of customer should be understood)

→ After understanding requirement, we document it (SRS)

→ We are given time and cost by the customer, software has to be developed under constraint of time and cost.

→ Software project management - planning of how to develop software.

→ a quantity which can be measured.
↳ quantifiable entity

→ can be measured in terms of metrics
↳ eg. number of lines of code.
one parameter

→ Software engineering

function oriented softwares	object oriented softwares

→ UML (Unified Modelling language)

→ Software Testing
  ↳ we have to test program for right as well as wrong input.

→ All different sets of inputs that can be given should be kept in mind so that program does not stop on wrong input.

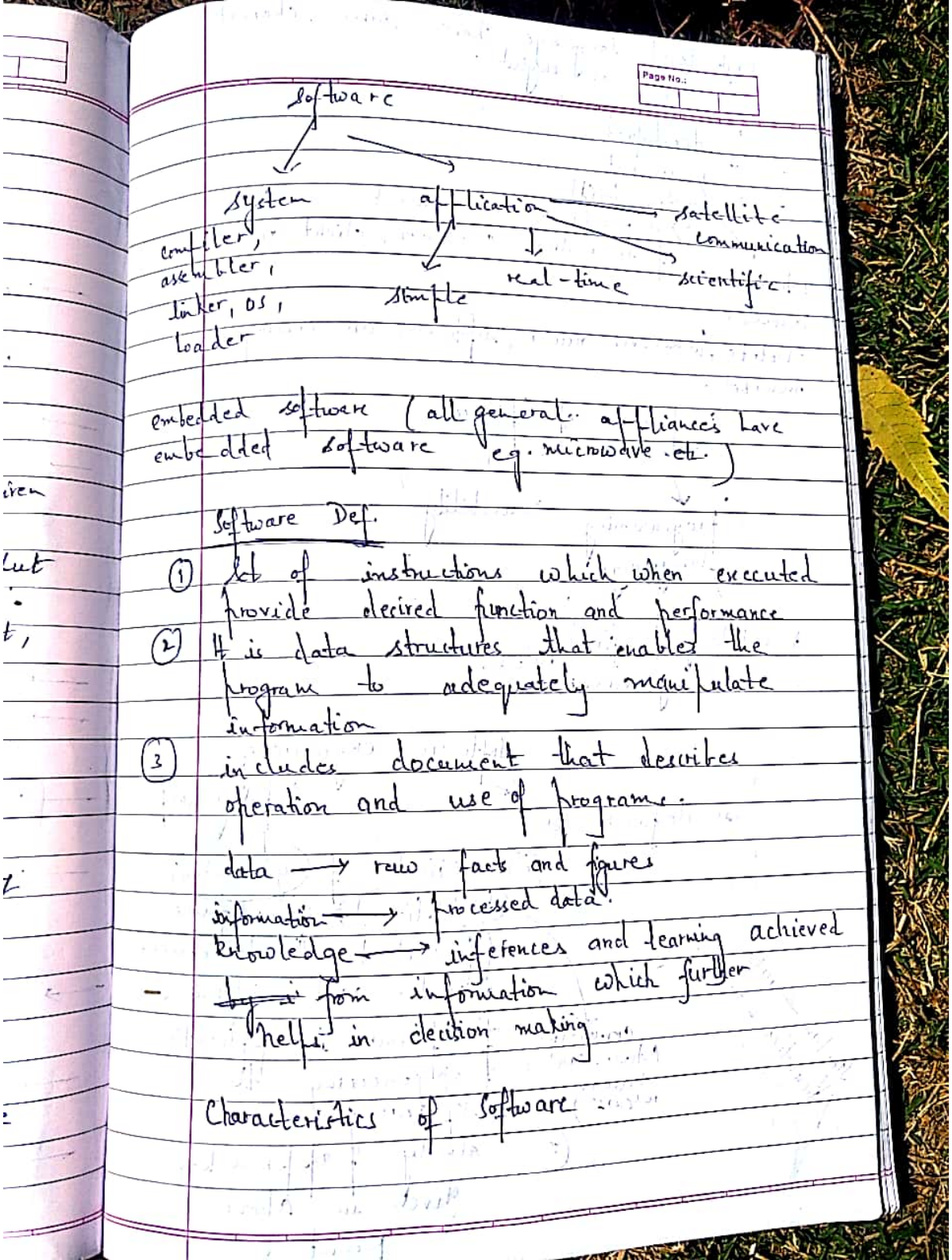
→ Software reliability → for unexpected input, software should not stop.

→ quality of software.
  ↳ speed
  ↳ all functionality should be there
  ↳ user friendly

→ We should be able to quantify reliability of the software.

Software

A set of programs grouped under one name used to provide a task.



embedded software (all general appliances have embedded software eg. microwave etc.)

Software Def.

(1) set of instructions which when executed provide desired function and performance
(2) It is data structures that enables the program to adequately manipulate information
(3) includes document that describes operation and use of programs.

data → raw facts and figures
information → processed data
knowledge → inferences and learning achieved from information which further helps in decision making.

Characteristics of Software

high level language have ... set as english.

→ developed
→ wears out.
→ custom built.
↳ depends on requirement of individual customer, client or user.

When it becomes obsolete, whenever newer softwares are coming in market.

Components of Software
- programming language
  - Vocabulary
  - grammar (well formed rules.)
- Reusability

Software engineering is defined as
1. application of (systematic, disciplined) quantifiable approach to the development, operation and maintenance of software ie application of engineering to software.
2. Study of approaches given in above point.

System → rule
Semantics → meaning

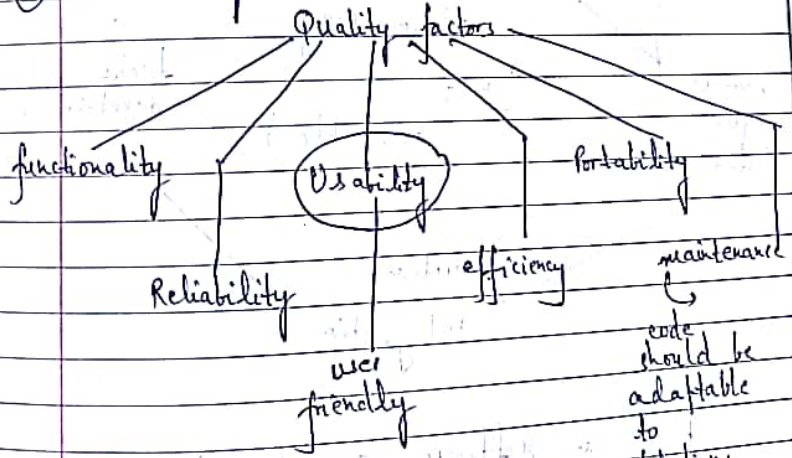
constraints (time, cost, human resource)

quantifiable ⟶ we can stick to deadline, only if we can measure it.
We try and quantity each and every activity using software project plan.

⟶ maintenance (after releasing it in market, we can change it, remove bugs and adapt it to new technology).

Qualities.

1. whether it fulfills all requirements or not.
2. performance
3. efficiency
4. reliability

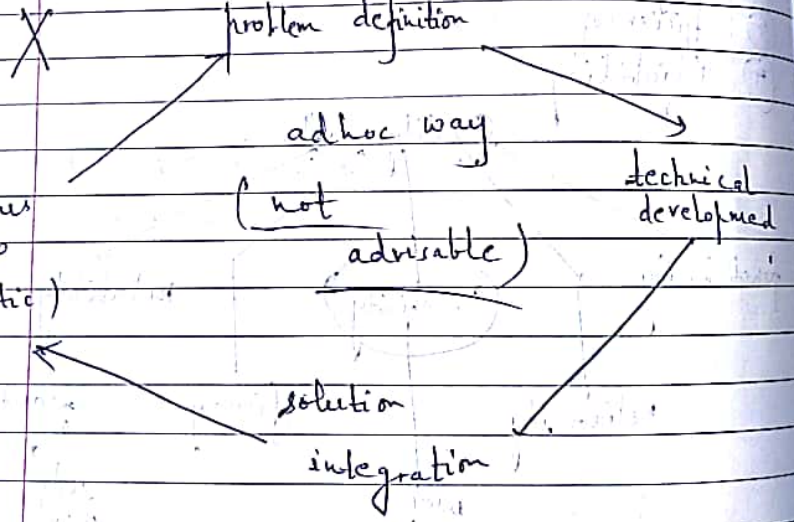


Page No.:

Software Engineering Principles

- tools → automated & semi-automated tools, help in developing software.
- methods
- P. Process
- Quality focus.

Software Process Model.



Phases in software development

Requirement Analysis → design → Coding → testing → Installation & Maintenance

(design: how)

This is an initial set of activities order.

The above model is called waterfall model or linear sequential model.

→ Requirement Analysis – activities that needs to be done. to understand problem domain, tells what all needs to be done, most important step

→ design (data structures, module, software architecture + interfaces

→ Testing should include overall testing and smallest unit testing (both right and wrong input)

→ SRS (Software Requirement specification document) / RSD (Requirement specification document)
this is a document produced after every stage.

✓ advantages of this sequence.

1. fixed way of working
2. step by step (no overlapping or disordering)
3. easy to understand procedure

✓ disadvantages

→ lack of flexibility, every phase is separated, so if there is a change in later phases, we need to go all the way back.

→ time consuming
→ if partial requirements are available, we cannot start.

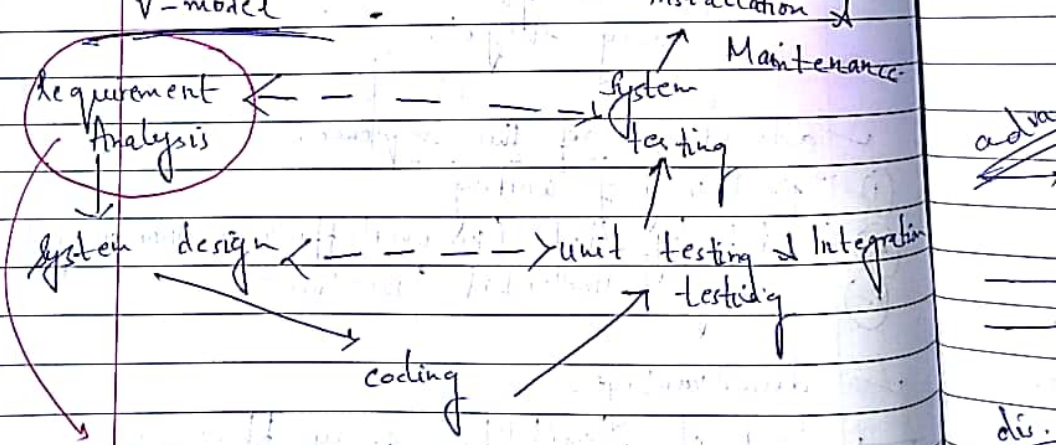
→ we cannot show the advancement to client until it is completely done.

→ teams cannot work in overlapping function as the process is not iterative, some teams have to wait for other teams, so at times we have blocking states.

→ if error in first stage is unknown, fixing it in later stage is not possible.

→ The customer cannot express his/her requirements fully in the beginning thus not enabling us to start the process.

V-model



Verification testing (helps removing various errors done after this at first level itself)

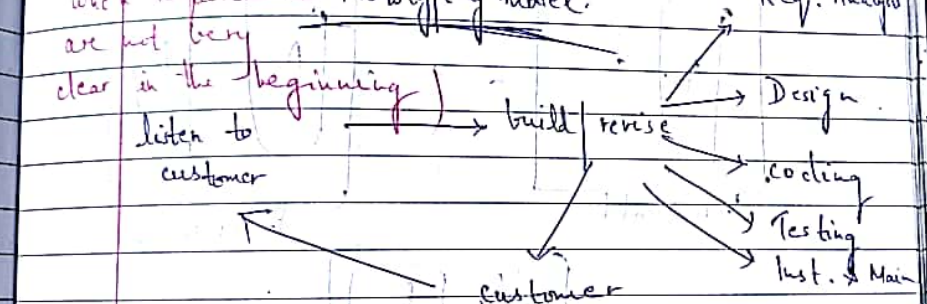
→ verified with the help of customer, expert or

third party
→ review again.
→ dry run (after designing)
→ This verification testing helps removing a lot of errors in initial stages.
→ Requirements should be consistent, shouldn't involve fluctuations.

advantage over waterfall model → testing phase starts early in development cycle.

(Valid in scenario where requirements are not very clear in the beginning)

Prototyping model.

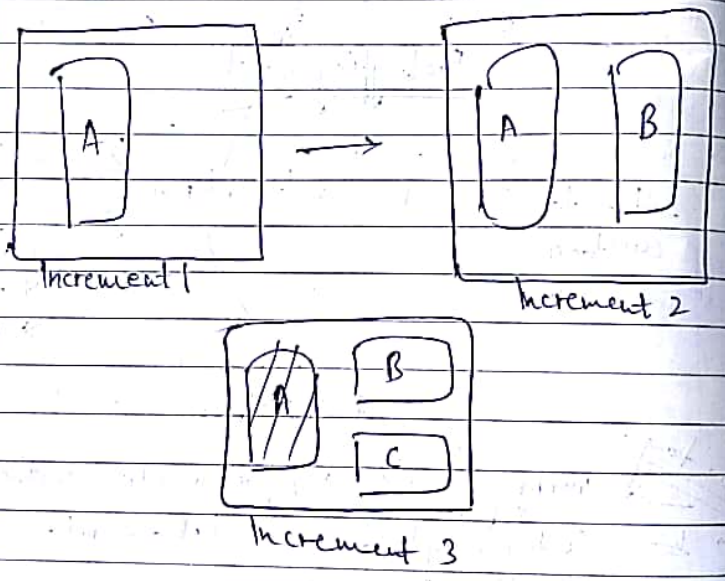


advantage
→ working model can be made without knowing all requirements.
→ Customer feedback is taken into consideration during the process.
→ Customer is satisfied.
→ Interaction helps in better understanding.

dis.
→ time and cost will be high
→ documentation has to be done again & again.
→ wide variety of users have different requirements.

→ When to stop the prototyping process is an issue
→ System Analyst who has domain specific knowledge decides how many cycles of prototyping should be done.
→ developer will make quick changes and will give sub standard product to the customer.
→ iteration is difficult to manage

Incremental Model



In this, ~~we~~ in first installment we meet requirement A and give it to customer who then gives the feedback. In increment 2, we deliver B set of requirements. The earlier module can be modified or not. In increment 3, in addition to delivering C, we modify A.

Here we have

linear seq. process + prototyping

linear seq. process → normal sequential model.

prototyping → we are repeatedly giving it, to be checked by the customer

In 1 set, we target only basic set of requirements such as writing into file, editing etc. In 2nd requirement, we add spell check, font etc.

advantage → customer feedback of each module.

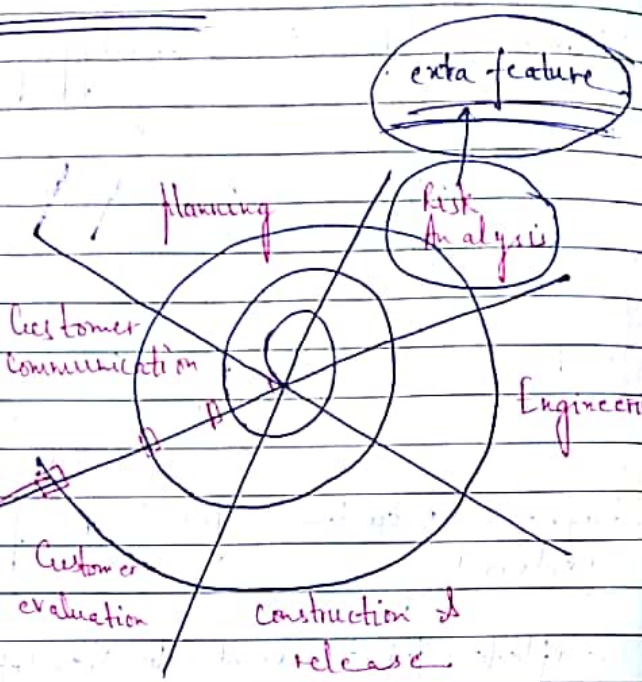
disadvantage

1. complete project cannot be developed & delivered together.
2. problems have to be initially divided and hence we need to know all requirements in beginning.
3. We need to provide interfacing b/w various modules which involves extra time and cost. Individual testing of module, interfaces and their simultaneous testing needs to be carried out.
4. We have to decide which section has to be developed first. for eg. to create a module of library, we need to create module of student & staff.

One more advantage → resources can be divided in parts.

find two domains where Spiral model will be appropriately used.

Spiral Model.



- evolutionary
- iterative
- controlled
- systemic
- linear seq. model

} feature

→ viable for large scale softwares.

→ We have to identify the probability of failure and also the factors which lead to risk.

→ after finding factors, we need to control them.

→ with complicated softwares like scientific and satellite comm., we do risk analysis and hence we apply spiral model.

→ not used for softwares like library management etc.

advantage

→ customer feedback, risk analysis, iterative etc.

disadvantage.

→ very complex.

→ who will decide how many times the iteration is to be done.

→ documentation is very tough.

→ if some risk factor is left in beginning, it is difficult to be dealt by in the end.

Rapid Application Development (RAD) model.

→ Software is to be made in a very small time period, every process is done quickly, normally 60 days are given to complete the task.

AGILE software Development - (We do not follow any sequence, constant comm. with customer)

→ activities can be executed in adhoc way as per the requirements of customer and availability of customer.

XP (extreme processing)

SCRUM

Page No.:

→ This model is different from all other models, customer meets on daily basis, not suitable for very large scale projects, requirements are met as and when they occur, documentation is very less.
→ not possible for large software.
→ for small softwares, the large time and cost requirement of documentation can be done away with.
→ constant customer interaction and small cycles of interaction.

Requirement Analysis / Engineering

→ demands, desires, need, functionality of the customer.
→ understanding the requirements is called requirement analysis.
→ If requirement analysis is done properly, there is high probability to get good final end product.

Requirement Analysis

→ results is specification of software, operation characteristics such as function, data and behaviour, indicate software integrate ~~software~~ interface with other software elements, establish constraints that software must need
→ provides software designer with a representation of information, function and behaviour that

can be transmitted to data, architecture, interface and component level designs.

→ problem recognition.
→ evaluation & synthesis (whether problem is feasible or not)
→ Modeling - data & control flow
→ specification - data input & output. How control will be transferred from one section to another.
→ reviewing
→ format

informal set of req → Requirement Analysis → formal req → SRS → RE

fizzy req

initial contract b/w customer & developer ← Software requirement specification document.

Software Requirement is defined as

(1) condition or capability needed by a user to solve a problem or to achieve an objective.
(2) condition or capability that must be met or possessed by system or system component to satisfy a contract, standard specification or other formally imposed document.

(3) a documented representation of a condition or capability as in (1) and (2).

Requirements

functional	non-functional	domain
Whatever functionality is performed by the system and gives desired output is functional requirement, it is something that works on the data.	Constraints which are imposed on the system, one particular functionality should not take a lot of time then we should have a fast, safe, reliable functionality non-functional req. does not pertain to data	Requirements of a particular domain and should exist in all the systems pertaining to the particular domain

Requirement Analysis

process → Req elicitation

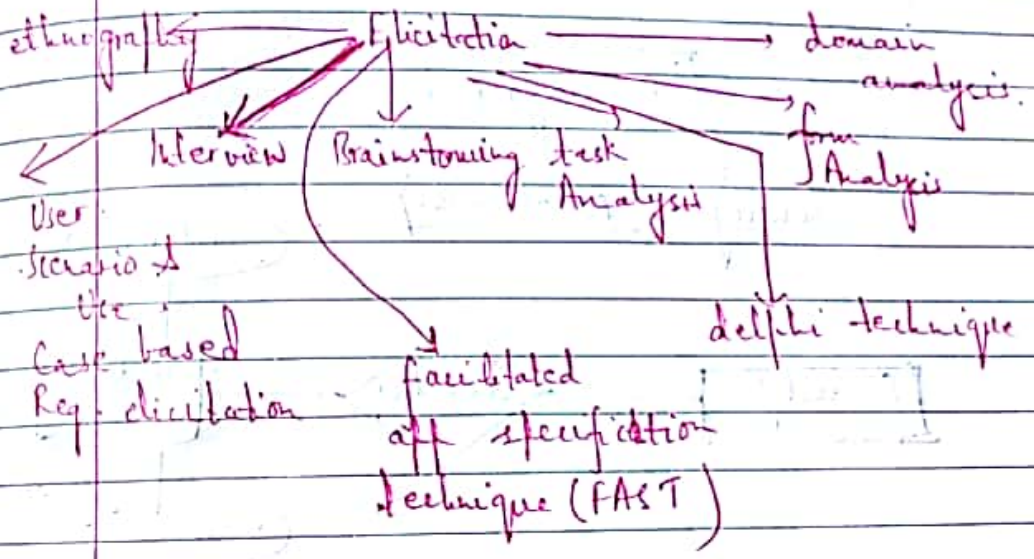
→ req. specification

→ req. validations

— elicitation - pen down the requirements, gathering of information.

→ req. specification - representing with help of diagrams, charts, graphs etc.

→ req. validation - for eg. if we have er-model and data-flow model, they should be consistent with each other. This is also important for change management process.



(1) ethnography - go to required place, observe and gather information. as much as possible.

(2) Interview
- structured (prepare a questionaire)
- open ended.

(3) brainstorming - group discussion of bw customers & developers.

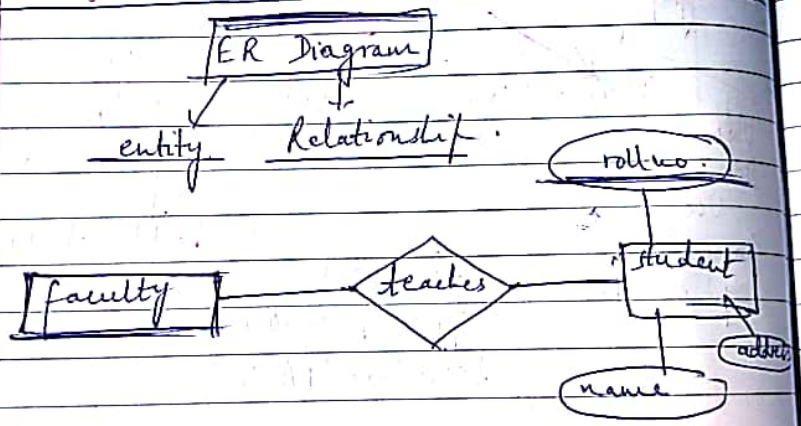
(4) task analysis - hierarchial structure is made, bigger task is divided into smaller functions.

Page No.:

(5) form analysis - we fill a form for getting admission, opening account.

(6) domain analysis - analyse the legacy systems to create a new product and incorporate new features as well.

(7) User scenario - narrate the sequence of events in the form of a story
story is required not only for correct scenario but also for alternate scenario.



Cardinality
1:1, 1:m, m:1, m:m

generalization
~~specification~~
specialization

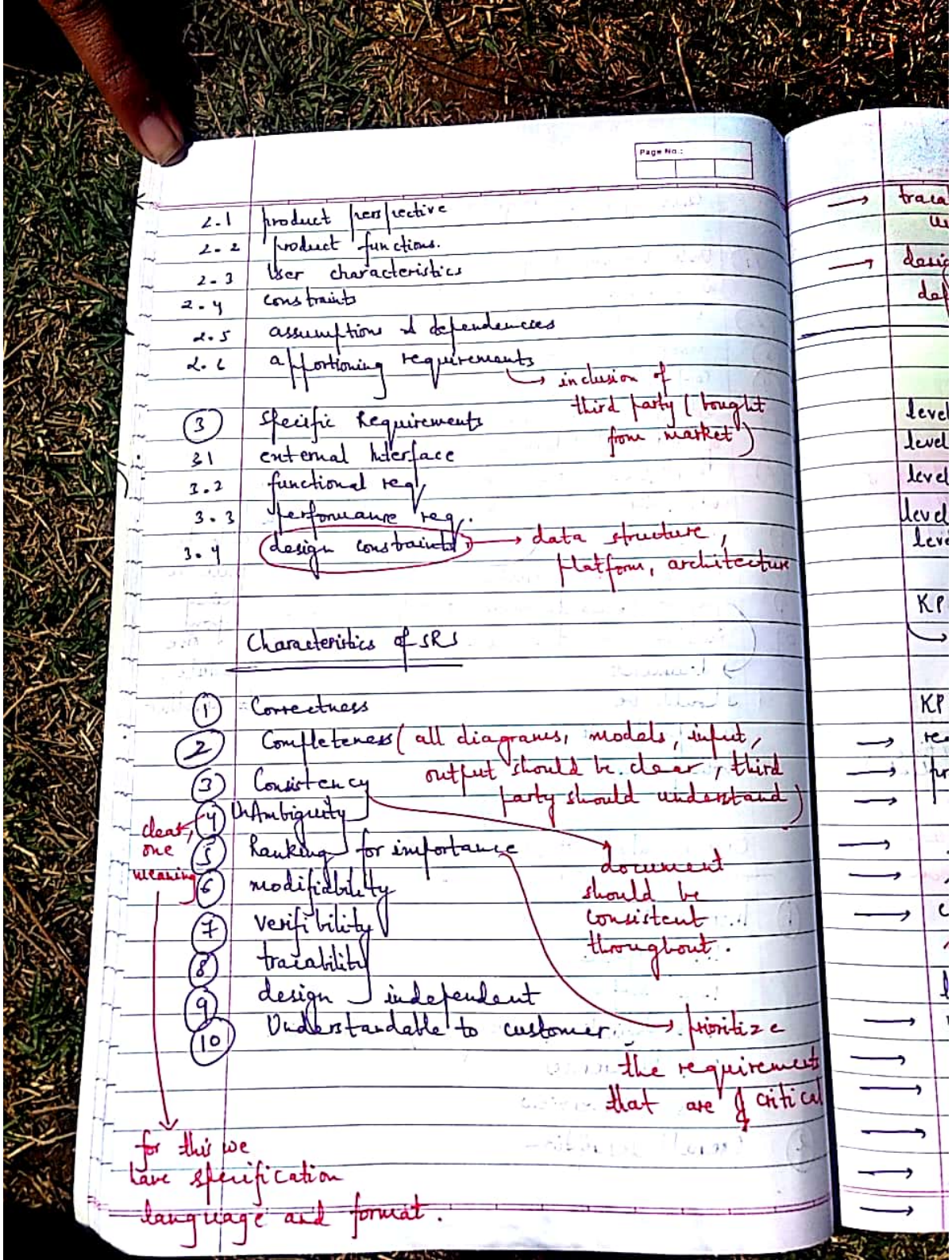
static view of the system (ER diagram)

Challenges in requirement elicitation

1. Understanding large or complex system requirements is a challenge
2. undefined system boundary
3. user or customer not clear about their needs.
4. Conflicting requirements
5. lot of requirements are put under category of to be determined.
6. partitioning the system suitably to reduce complexity
7. validating and tracing requirements → we should be able to trace a thing from one module to another
8. identification of critical requirements.
9. proper documentation of requirements
10. meeting all the constraints of customer → document should be acceptable to both parties

Organization of SRS (IEEE 830) → find one more format.

1. Introduction
   - 1.1 purpose
   - 1.2 Scope
   - 1.3 definitions, acronyms, abbreviation
   - 1.4 References
   - 1.5 Overview
2. Overall description



2.1 product perspective
2.2 product functions.
2.3 User characteristics
2.4 constraints
2.5 assumptions & dependencies
2.6 apportioning requirements → inclusion of third party (bought from market)

(3) Specific Requirements
3.1 external interface
3.2 functional req
3.3 performance req.
3.4 design constraints → data structure, platform, architecture

Characteristics of SRS

(1) Correctness
(2) Completeness (all diagrams, models, input, output should be clear, third party should understand)
(3) Consistency → document should be consistent throughout.
(4) Unambiguity (clear, one meaning)
(5) Ranking for importance → prioritize the requirements that are critical
(6) modifiability
(7) Verifiability
(8) traceability
(9) design independent
(10) Understandable to customer

for this we have specification language and format.

→ traceability (there should be proper flow and unique names for everything)
→ design independent (should not be abstr dependent on platform, language etc.)

CMM (Capability Maturity Model)

level 1 - adhoc } no KPA defined
level 2 - repeatable
level 3 - defined
level 4 - managed
level 5 - optimized

KPA - Key process Areas
↳ set of activities to be done at various levels.

KPA required in level 2
→ requirement management
→ project planning
→ software project tracking (scheduling) & oversight.
→ sub contract management
→ software quality assurance
→ configuration management (versioning of software with new requirements into picture)

level 3 (organization oriented)
→ organization process focus
→ training program (train a human resource)
→ organization process definition
→ integrated software management
→ software product engineering (which process model is followed)
→ peer reviews (feedback)

measuring a process
→ cost
→ labour
→ time

level 4
→ quantitative process management
→ software quality management

level 5
→ defect prevention
→ technology change management
→ process change management

→ find any Indian company with CMM level 5.
→ find if software industry needs both CMM & ISO. find a company which satisfies this.

Requirements specification

- static view (ER diagram)
- functional view → function decomposition diagram
  - data flow diagram → flow of data from one process to another
  - data dictionary → data about data
  - process specification → describe function in English or pseudocode
- dynamic view (time dependent) → state transition diagram

Page No :

data flow diagram

process

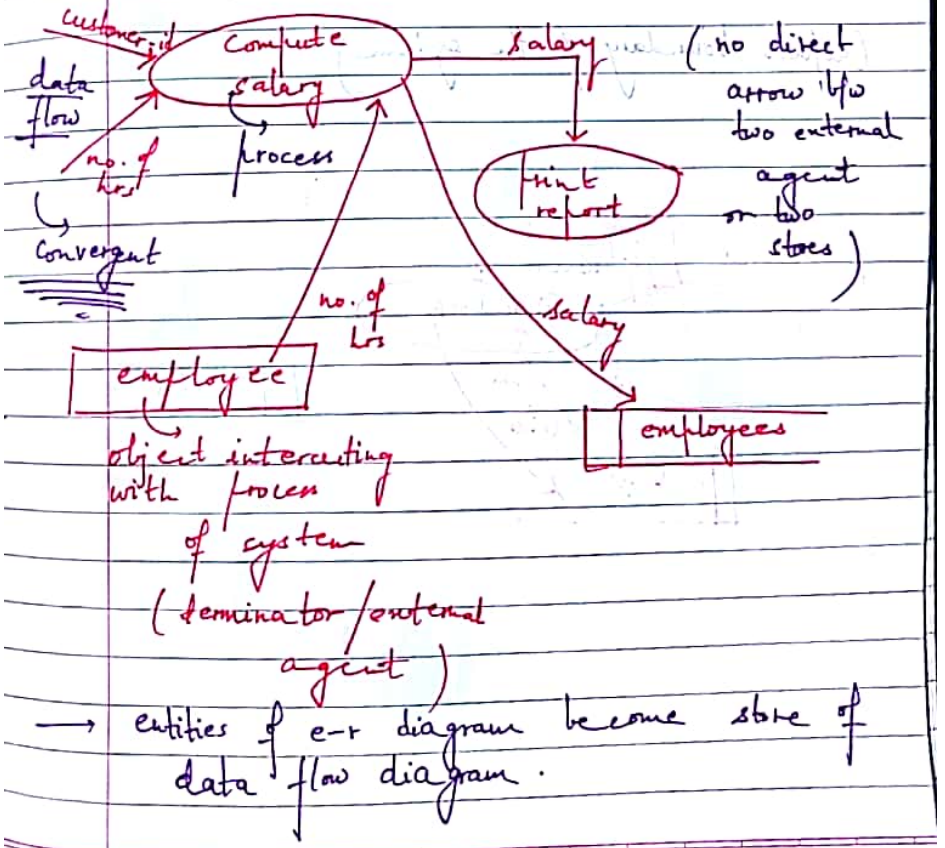
data flow

terminator / external

store (database)

employees → additional

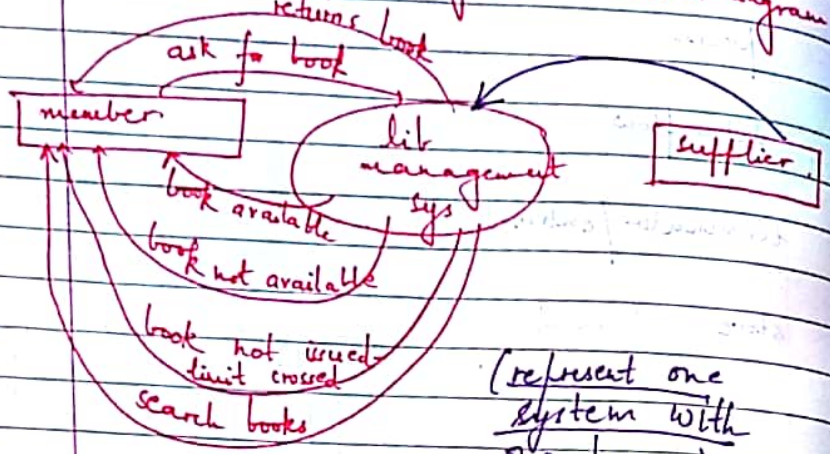
employees



→ entities of e-r diagram become store of data flow diagram.

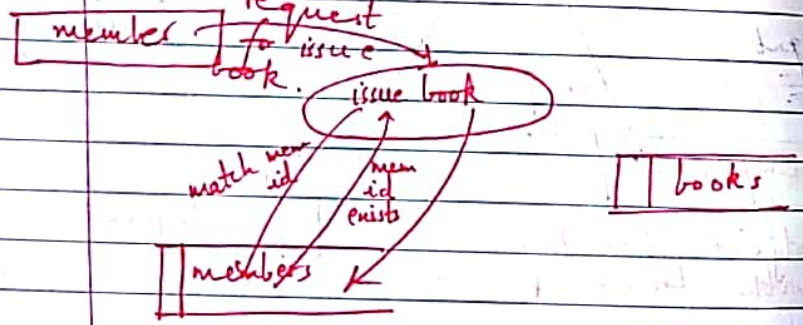
Page No.:

level 0 data flow diagram - content diagram



(represent one system with one process)

(defines boundary of the system)



Data Dictionary

Page No.:

Conventions

** , || ⟶ comment
[ ] ⟶ one of several choices
@ ⟶ key field of store
/ ⟶ choice
= ⟶ is composed of
+ ⟶ and
( ) ⟶ optional
n Σ 3 m ⟶ at least n and at most m.

telephone no. = (country code) + (area code) + number
country code = Σ digit 3 3
area code = Σ digit 3 5
number = 7 Σ digit 7 8
digit = [0|1|2|3|...|9]
customer = Σ customer 3 n
customer = @ customer_id + customer_name
+ address + telephone_no
address = state + city + country + [sector_no | plot_no]
+ house_no

function decomposition diagram

hierarchially decomposes all functions from top to bottom.

Process specification

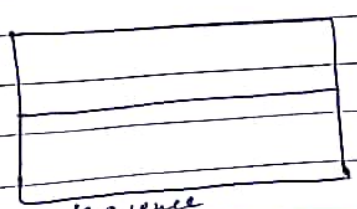
- Structured English
- Decision Table
- Decision Trees
- Flowcharts
- Nassi-Shneiderman charts/diagram

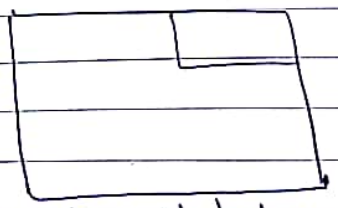
Structured English

Input two number
Set a = 5 and b = 10
Compare a and b
If a is smaller than b
then
display a is smaller than b.

add 7 to item_total.
Do while
Repeat Until
Do Case

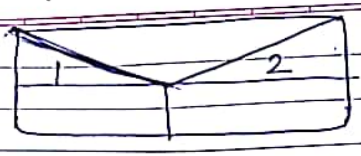
Nassi Schidermann charts



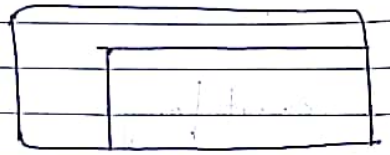


Page No.:

1

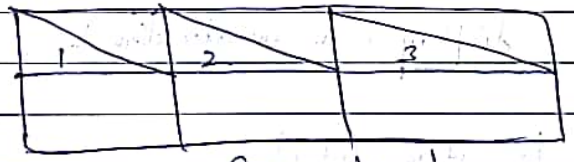


Conditional struct



Do while loop

Do until loop



Case struct

Page No.:

set cash_sale to 0
set sale_status to cash
set total_sale = 0
set credit_sale to n
Read Daily_sale, Sale_Status, EOF
Total_Sale = TotalSale + Daily_Sale.
Sale_status = 'cash'

YES	N.O
cash_sale = cash_sale + daily_sale.	credit_sale = credit_sale + daily_sale

Return total_sale, cash_sale, credit_sale.
until EOF = Yes.