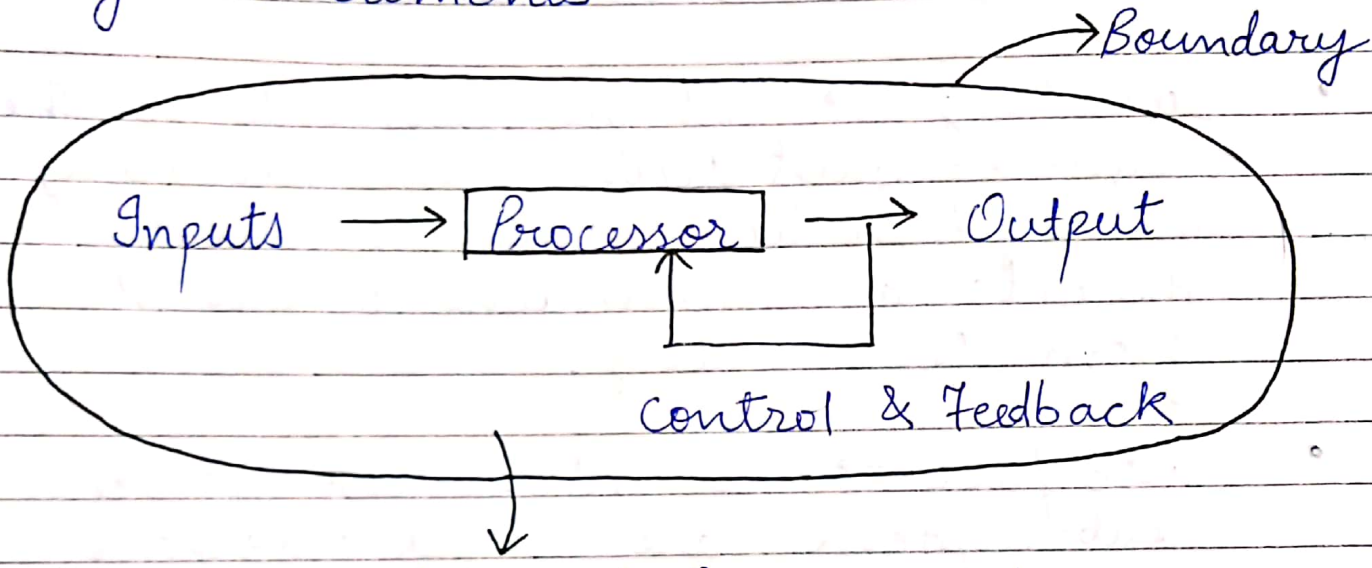
04/09/19

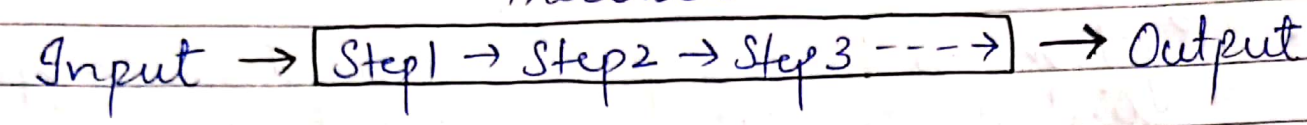
Nand
Date
Page

Ch-1 (Introduction to System)

- System is a group of independent components linked together according to a plan to achieve a specific objective.
- System Elements:

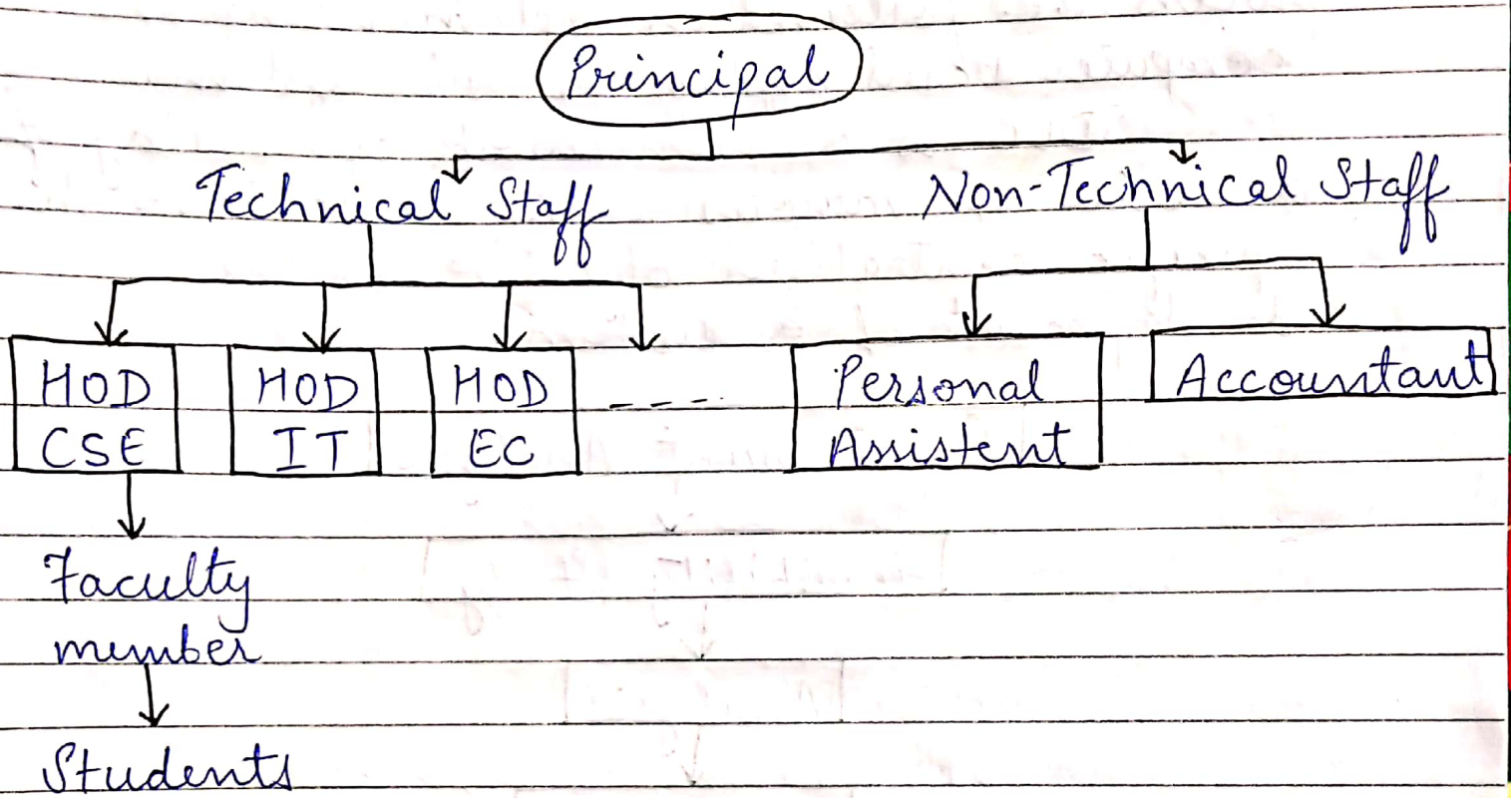


- Input → An input is the element that enters in a system to be processed by the processor.
- Output → A major objective of the system is to produce whatever is expected by the user.
- Processor → It perform the actual transformation of an input into the output through a series of steps.



- Environment → It determine the behaviour of the system, how the system behave in the environment.
- Boundary → All the elements of the system must define their boundary, the system lie within its boundary & its external environment is outside from its boundary.
- Interface → The interaction between the sub-systems are called interface.
- Features of System:
  (i) Goal / Objective
  (ii) Organization
  (iii) Interdependence
  (iv) Integration
  (v) Interaction
- Goal / Objective → The objective of the system is defined as we are going to achieve some goal through the system.

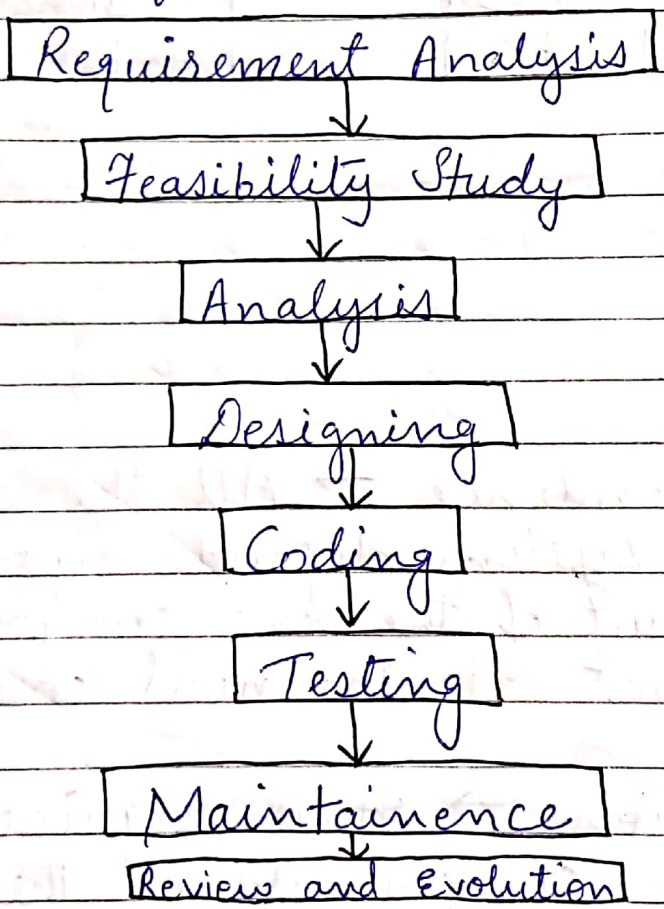
- Organisation → It concerned with the structure & arrangement of the components of the system to achieve the objective.



- Interdependence → All the components of the system depends upon each other, the output of the one component act as an input to the next component.
- Integration → It is required to assemble all the components or the sub-systems of the system.
- Interaction → It can be done by the inter-relationship of the component

of the system.

- SDLC (System Development Life-Cycle):
  A system Development life-cycle is the process i.e. followed in implementing a computer-based system or sub-system.
  SDLC is a framework consisting of a series of various tasks to achieve the specific centralized objective in the development of a system.



- Requirement Analysis → The base of this phase is to survey or initial investigation to determine whether an alternative system can solve the problem.

- Feasibility Study → It is the detailed study or investigation about the system. It is the test of a system proposal according to its work ability, impact on the organisation, ability to meet user needs, and effective use of resources.

  There are some types of feasibility:

  (i) Technical feasibility → (Hardware, software) It is related to the availibility of hardware and software to perform essential computing.

  (ii) Economic feasibility → (cost related) The new system should be economically beneficial for that, the cost & benefit analysis is performed.

  (iii) Time feasibility → The system should be implemented within the mentioned time constant.

  (iv) Legal and Ethical feasibility → The new system should exist within the legal and ethical boundaries.

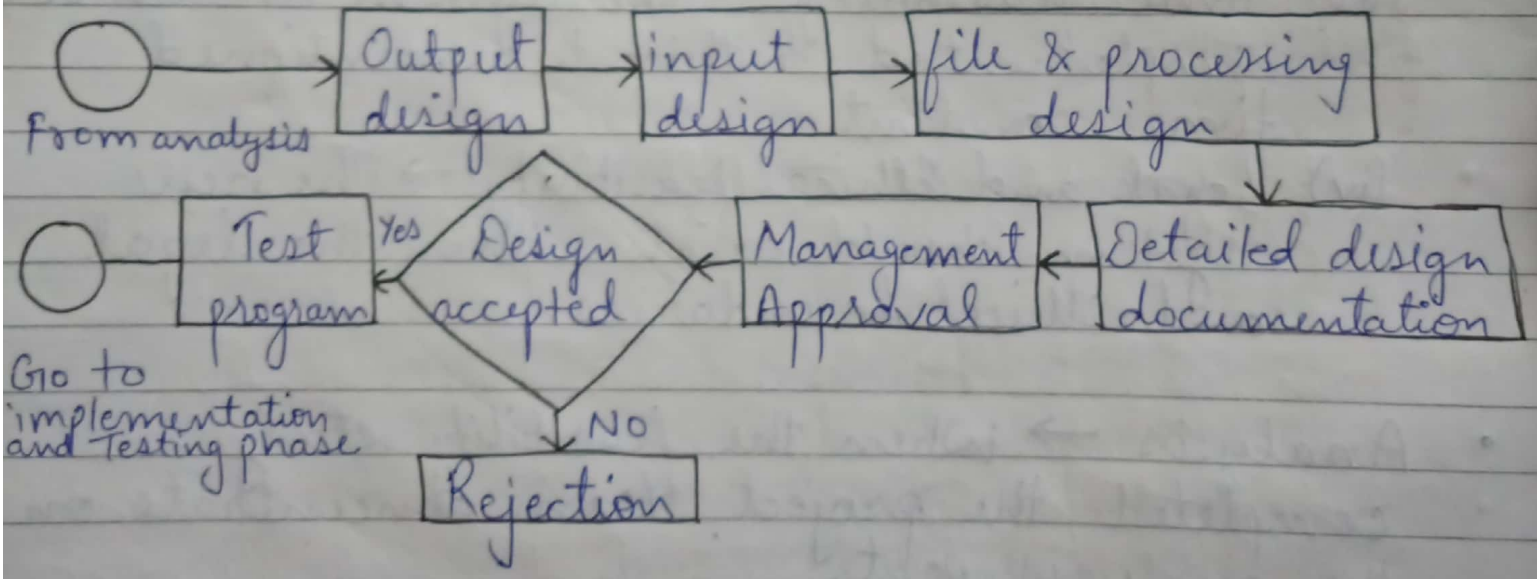
- Analysis → When the feasibility study is completed, the project team concentrate on the analysis part.

  Analysis is a detailed study of the various operation performed by a system & their relationship. There are some major objectives:

(i) Define the scope of new system
(ii) Understand the old information system
(iii) Review the feasibility & cost benefit analysis
(iv) Develop the functional, structured specification for the new system.

After analysis phase, this phase generate a document called SRS (software requirement specification).

- Designing Phase → It is done by the project manager. System design is the determination of the process and data that are required by a system. It also depends upon what are the things that user are expecting from the system. There are some steps in it:



(i) Output design → (a) Procedure to produce output
(b) Format / Representation of output
(ii) Input design → (a) Input data are designed
(b) Format of input data

(c) Sample presentations.

(iii) File & processing design → (a) files of database design to fulfill the requirement of the systems.

(b) In processing design, program construction & testing.

(iv) Detailed designed documentation → (a) Documentation of details related to the systems

(b) Documentation of estimation which affect the systems.

(v) Management Approval → (a) All the previous work evalution.

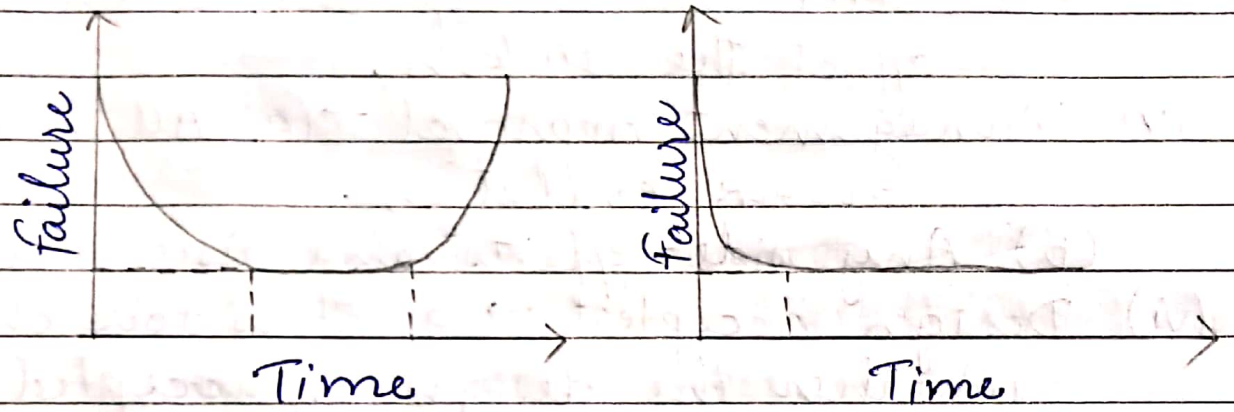
(b) Any side-effects and risk factors

(vi) Design accepted → (a) It is tobe checked whether the design is accepted or not.

- Coding → In this phase, development of source code on a specific technology is compiled.
- Testing → After developing the code of the system, it enters into the testing phase, which is called as STLC (System Testing life-cycle).
- Maintainence

- Software Engineering:
Software is designed as a collection of programs, procedure, data and documentations.

- Characteristics of Software Engineering:
(i) Software is developed or engineered not manufactured.
(ii) Software doesn't wear out.



* Failure curve of Hardware or "Bath-Tub" Curve

* Failure curve of Software.

In this, for hardware product it can be observed that the failure rate/ratio is initially high but it decrease as the components are identified and removed. After some time, the failure rate is again increased. This gives the plot of hardware reliability over the time which is called the "Bath-Tub Curve".

At the other side, for the software

failure curve, the failure rate is high at the initial level more errors are identified and after some time after removing these errors the curve will come in a stable state.

- Software Life-Cycle Method :
  A SLCM (Process Model) is a discriptive & diagramatic representation of the software life-cycle. The life-cycle represents all the activities required to make a software product.
  Different software life-cycle Models are here -
  (i) Classical Waterfall Model
  (ii) Iterative Waterfall Model
  (iii) Prototype Model
  (iv) Evolutionary Model
  (v) Spiral Model

- Project-Size estimation Techniques :
  Estimation of the size (line of code) of the software is an essential part of software project management. It helps project manager to predict the efforts and time which will be needed to build the project.
  Various majors are used in project-size estimation -
  (i) Line of Code

(ii) Number of entities in ER-Diagram
(iii) Total Number of DFD (Data flow diagram)
(iv) Function points

1.) Line of Code → Total number of lines in the source code. KLOC (1000 lines of code), NLOC (Non-comment lines of code)

2.) No. of Entities in ER-Diagram →

- Software Quality Assurance:
  It is a set of activities for ensuring quality in software engineering process (life-cycle). It ensure that developed software meets and compile with the defined or standardised quality specifications.

- SQA Activities: (Explain all activities)
  (i) Setting the checkpoints (deadline)
  (ii) Creating an SQA management plan
  (iii) Apply software engineering techniques
  (iv) Executing formal technical review (FTR)
  (v) Having a multi-testing techniques/ startegy.
  (vi) Controlling change

(vii) Performing SQA audit (inspection)
(viii) Manage good relations
(xi) Maintaining records & reports

20/9/19

Ch-2 (Software Engineering)

★ Software Engineering introduction :-

- SE is the application of a systematic, discipline, quantifiable approach to the development, operation and maintenance of Software.
- It is Based on Layered Technology :

Tools
↓
Methods
↓
Process
↓
A Quality Focus

- Quality Focus → Any Engineering approach must rest on an organisational commitment to quality.
- Process → It is the glue that holds the technology layers together and enables rational and timely development of computer software. Process defines a framework for a set of Key Process Areas (KPAs) that must be established for

defective delivery of S.E. Technology.

- Method → S.E. methods provides the technical how-to-'s for building software methods incompass a broad array of task that includes requirement analysis, design, program, construction. testing & support.
- Tools → S.E. Tools provide auto-mated / semi-automated support for the process and the methods

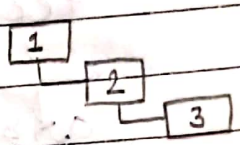
★ A Generic-View of S.E. :-

(i) Analysis
(ii) Design
(iii) Construction
(iv) Verification
(v) Management of Technical-Entities

★ Process Models :-

(i) Linear Sequential or Waterfall Model
(ii) Prototyping Model
(iii) Rapid Application and Development Model (RAD)
(iv) Component Based development Model (CBD)
(v) Evolutionary Software Process Model :-
  a) Incremental development Model
  b) Spiral Model

1.) Linear Sequential or Waterfall Model :-
The most basic process model. All the steps are sequentially attached & flows in one direction only.
Advantage : Step by Step working

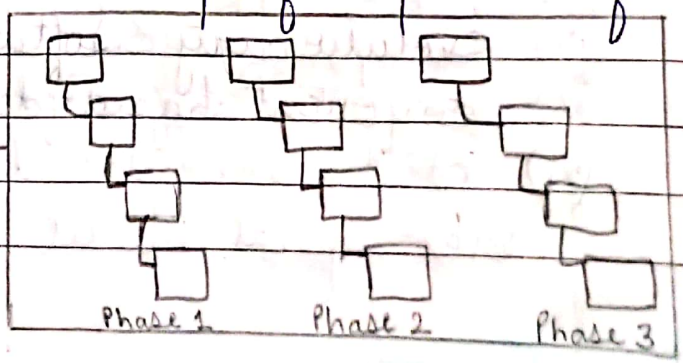


Drawback : If the developer comes on 3rd step and the user tells to modify and change anything, this model fails. It is Because it is not possible to again perform first two steps.

2.) Prototyping Model-
It depends on a formula. User tells requirements and a prototype is made on the base of requirements and modifications are done to satisfy the requirements and a physical original model is made afterwards.
Drawback : User can deny about the non-familiarity of the model, this model fails.

3.) RAD Model-
Best when a user provides a specific period of time for completion of model. The Best model under a specific period of time / time limits.

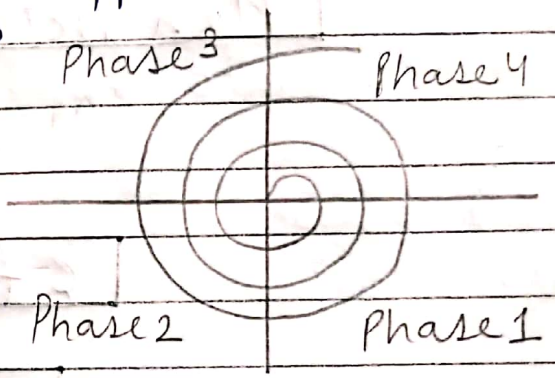
Combination of Waterfall and Prototyping ←



4.) Evolutionary Software Process Model:-

a) Incremental development Model-
Consideration of Production is available here. It mainly focuses on first incremental analysis. Every increment/upgradation is studied and analyzed.

b) Hybrid /Spiral Model (Prototyping + Sequential) -
Based on Risk Driven Approach:
(i) Management Risks
(ii) Technical Risks
(iii) Economical Risks
(iv) Financial Risks



5.) Component Based Development Model (CBD)-
It is developed because of getting rid of making models continously. It is fully Object Oriented Technology based.

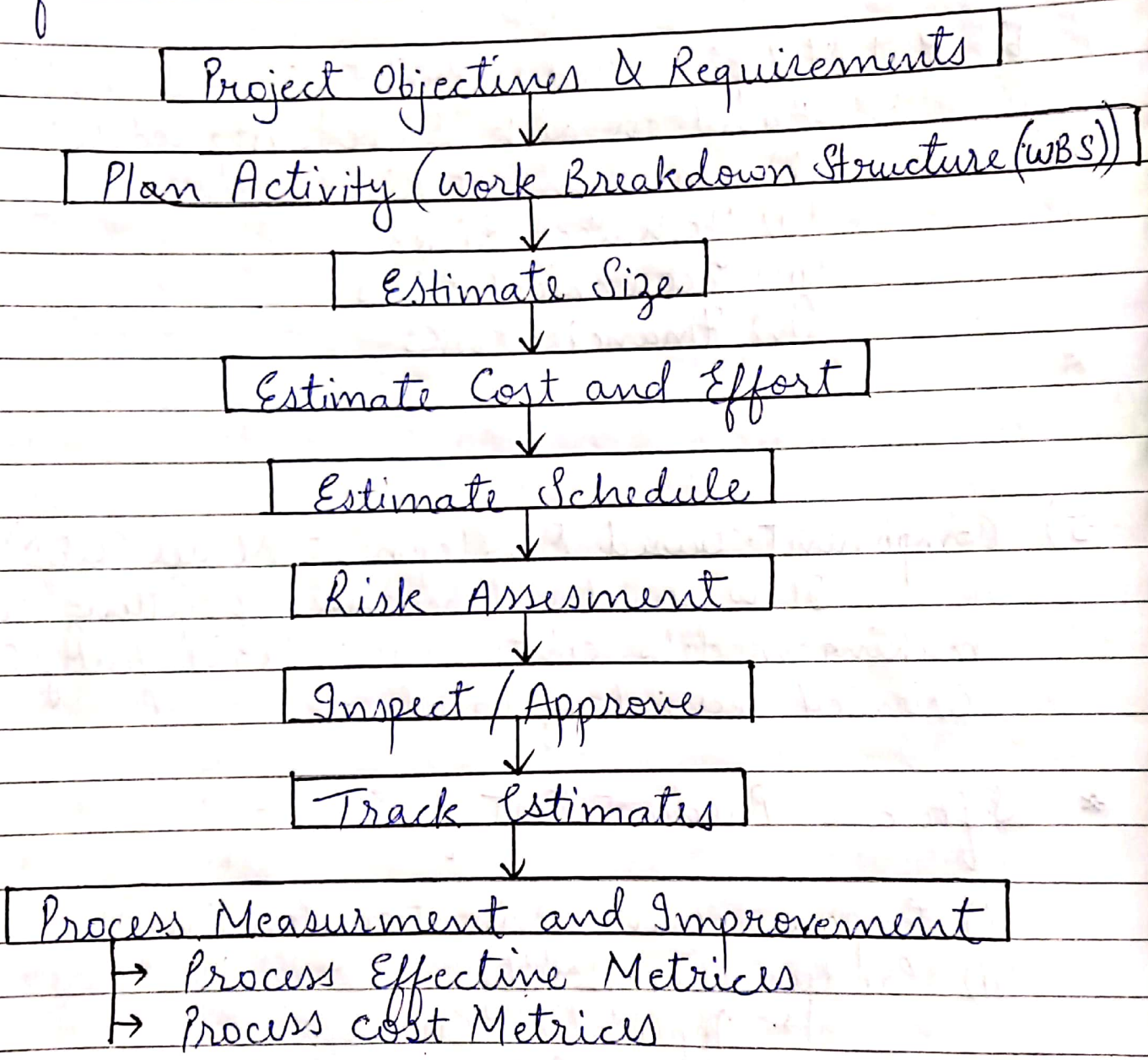
* Software Product Cost factors:-

(i) Experience in application domain
(ii) Product Complexity
- → Application Programs
- → Utility Programs
- → System Programs

(iii) Project Size
(iv) Available Time

(v) Programmer Ability
(vi) Level of Technology
(vii) Required level of Technology

imp. ★ Software Cost Estimation Process -



Note: i) Here, Metrics means group of parameters
(ii) Cocomo Model is used for cost estimation.
↓
imp

* Decomposition Techniques -

- Software cost estimation is a form of problem Solving and in most cases the problems to be solved are too complexed to be considered in a single form. Therefore, the problem is decomposed into complements in order to achieve an accurate cost estimate.

- Two Approaches are there:-
  (i) Problem based estimation
    a) FP → Function Point
    b) LOC → Line of Code ($S = (So + 4Sm + Sp)/6$)
  (ii) Process based estimation

- FP → This matrix is used to measure the functionality delevired by the system.
  There are four parameters in function point -
  (i) EI = No. of External Inputs
  (ii) IO = No. of Internal Outputs
  (iii) ILF = No. of Internal Logical Files
  (iv) EIF = No. of External Interface Files
  Formula : FP = Count × WF (weighting factor)
  ↓ Average

* Cost Estimation Model -

(i) Algorithmic (Eg: COCOMO Model) used 80%.
(ii) Non-Algorithmic (Eg: Price to Win) failure Model

→ 10 Marks

- Constructive Cost Model (COCOMO Model) - First invented in early 80's developed by Barry Boehm. It is basically used in cost estimation.

Category to judge project : (Size → KDLOC)
a) Organic (Size ≤ 50 KDLOC)
b) Embedded (Size ≥ 300 KDLOC)
c) Semi-Detached (Size ≤ 300 KDLOC)

1.) Organic Project → Eg: Small Business system, Library Management
2.) Embedded Project → Complex Projects (Eg: Software used in Military Hardware)
3.) Semi-Detached → Eg: OS, Compiler design

- Three Sub-Models in COCOMO Model : } Hierarchy
(i) Basic Model
(ii) Intermidiate Model
(iii) Advance Model

1.) Basic Model → Formula : $E = A \times (size)^B$ (size → KDLOC)

E → Effort, Person Months (PM)
A → constants
B → constants

Eg :

Project Time	A	B
Organic	2.4	1.05
Semi-Detached	3.0	1.12
Embedded	3.6	1.20

Project Type : Organic
Size : KDLOC

$E = 2.4 \times (30)^{1.05}$

$E = 85$ PM approx.

2) Intermidiate Model → It consider software reliability and complexity as parameters along with size and effort.
There are total four parameters.

Step 1: Calculate → E an initiate estimate of effort

Step 2: Identify a set of 15 parameters, all parameters are rated against a numerical value called multiplying factors. → EAF

Step 3: Calculate Total Effort = EAF X Ei

Eg: P+ = organic

Size = 45 KDLOC

P+	A	B
organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

→ $3.2 \times (45)^{1.05} = 174$ PM

EAF = 0.8895 (1.15 X 0.85 X 0.91 X 1.00)

Reliability, Complexity, Application Example, program Capability

3.) Advanced Model → Here, effort is calculated as a program size and a set of cost drivers for each phase of software engineering. This model incorporates all characteristics of the intermediate model and provides procedures for

adjusting the phase wise distribution of the development schedule. It has four phases.

a) Requirement planning & project design (RPPD)
b) Detail Design (DD)
c) Code & Unit Test (CUT)
d) Integration and Test (IT)

For all these rating, cost drivers are assigned multiplying factors and these are called multiplying factors for analyst capability (ACAP). ACAP is calculated using 25 parameters.

Eg: Total Effort = Ei X ACAP
= 174 X ACAP
= 174(ACAP) PM

* System Analysis -

- What is software requirement?

→ Requirement is a condition or a capability possed by software or system component in order to solve a real world problem. Requirements describes how a system should act, appear or perform.

For this, when users request for software they posses an approximation of what the new system should be capable of doing.

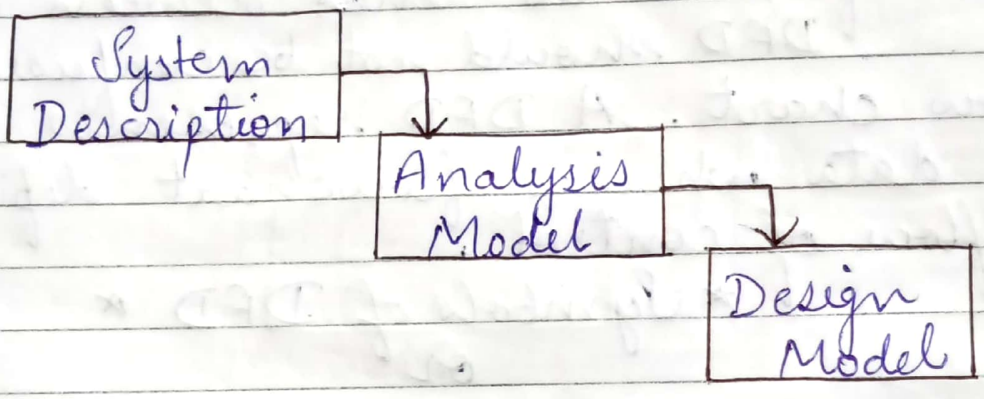
- Feasibility Analysis
- Requirement Analysis according to Ethical rules :-

IEEE defines:

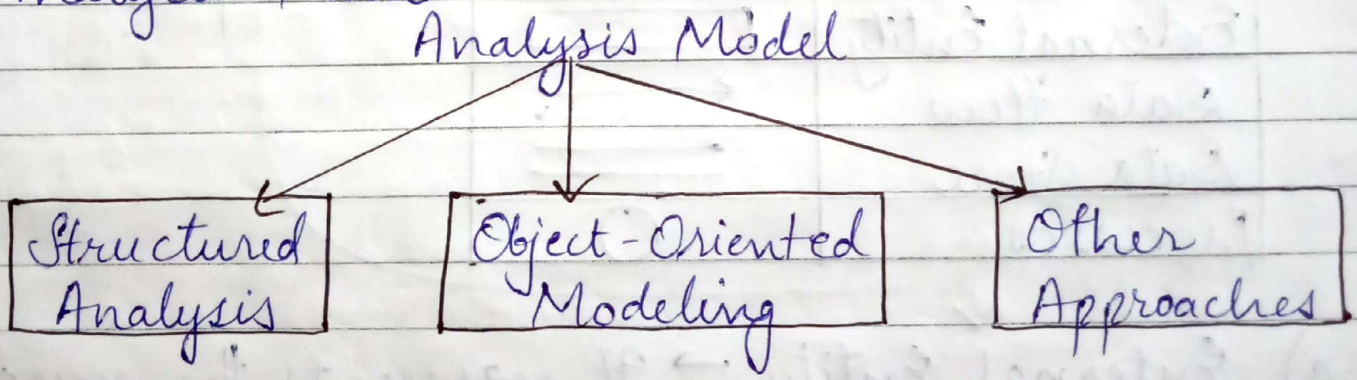
a) The process of studying, user needs to arrive at a definition of a system, hardware or software requirements.

b) The process of studying & refining system hardware or software requirements.

* Analysis Model as Connector *



- Analysis Model :-



a) Structured Analysis

- Data Flow Diagram (DFD)
  - Level 0 (Context Diagram)
  - Level 1
  - Level 2
  - Level 3 (Detailed Hierarchy)
- Data Dictionary

6) Object-Oriented Modeling (OOM)

* Data - Flow Diagram (DFD):

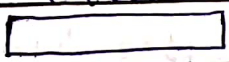
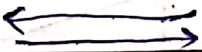
IEEE defines DFD (also known as Bubble Chart or Workflow diagram (WFD)) as a diagram that depicts data sources, data sinks, data storage and processes perform on data as nodes and Logical Flow of Data as links between the nodes.

DFD should not be confused with a flow chart. A DFD represents the flow of data whereas flowchart depicts the flow of control.

* Symbols of DFD *

or

* Notations *

Name	Notation	Description
External Entity		
Data Flow		
Data Store		
Processes		

a) External Entity → It represents the source or destination of data within the system. Each external entity is identified with a meaningful & unique name.

b) Data Flow → Represents the moment of data from its source to destination within the system.

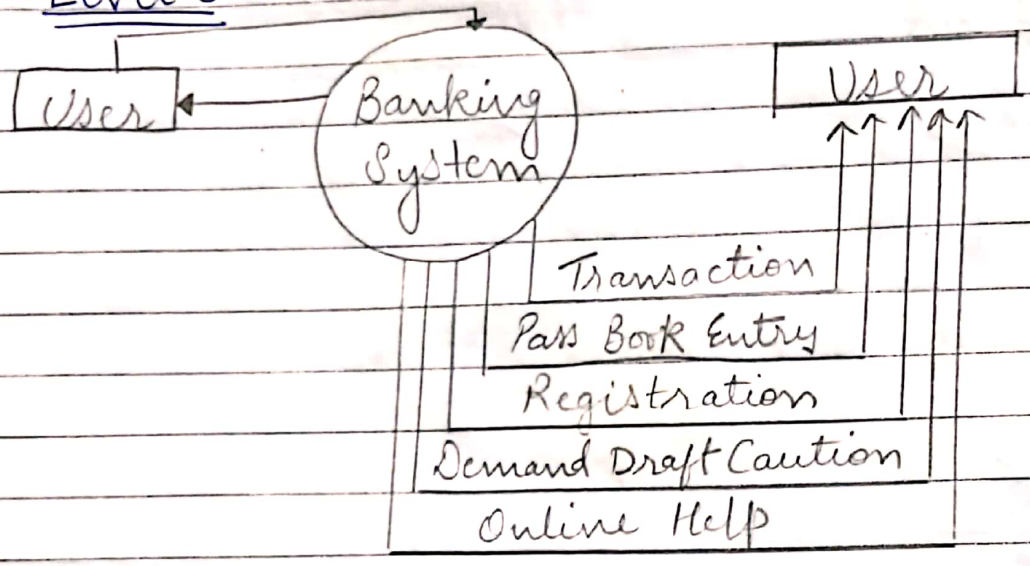
c) Data Store → Indicates the place for storing

information within the system.

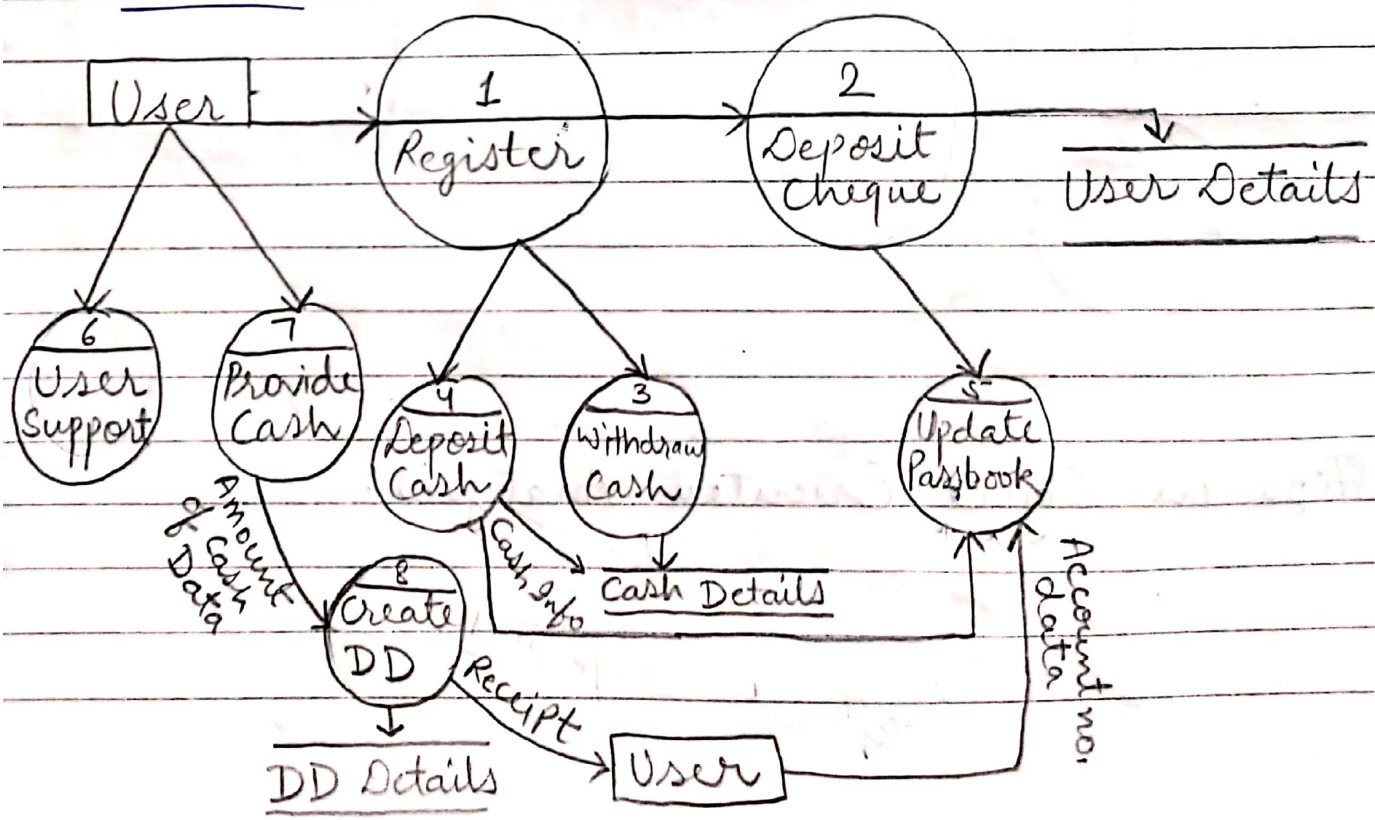
d) Processes → Shows a transformation or manipulation of data within the system. A process is also known as 'Bubble'.

- DFD of Banking System: (Case Study)

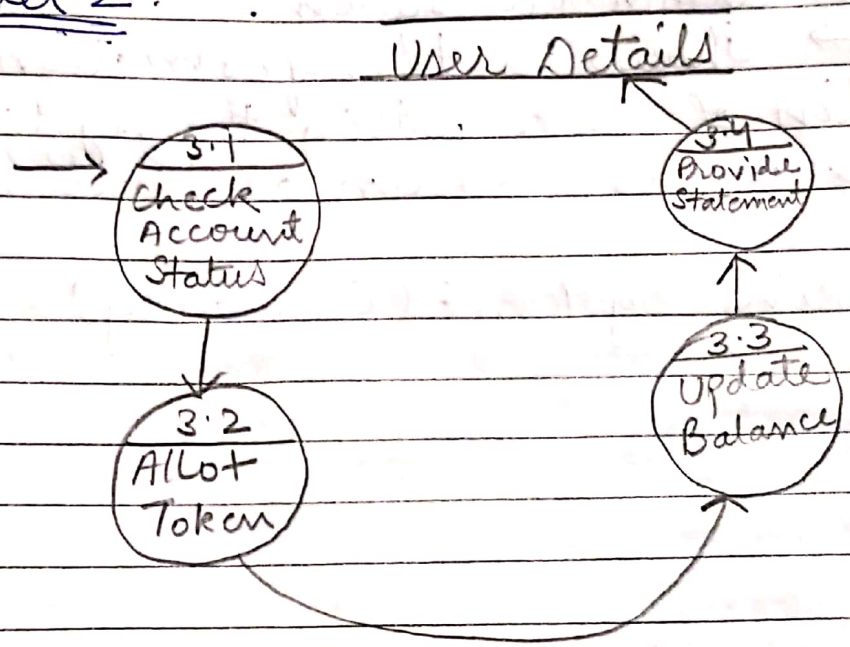
Level 0:



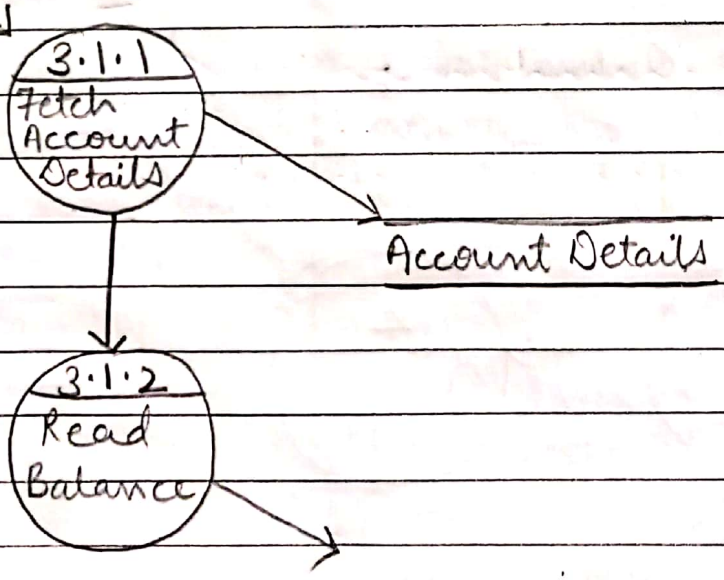
Level 1:



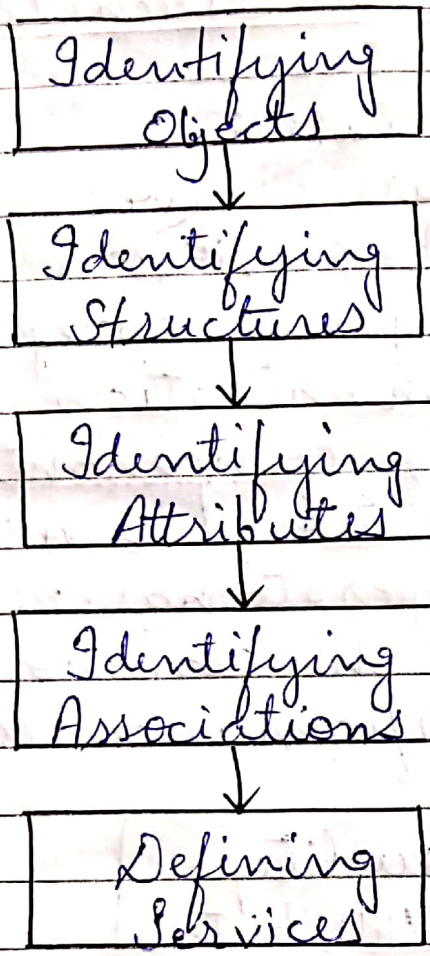
Level 2 :



Level 3 :



* Steps in Object-Oriented Analysis :

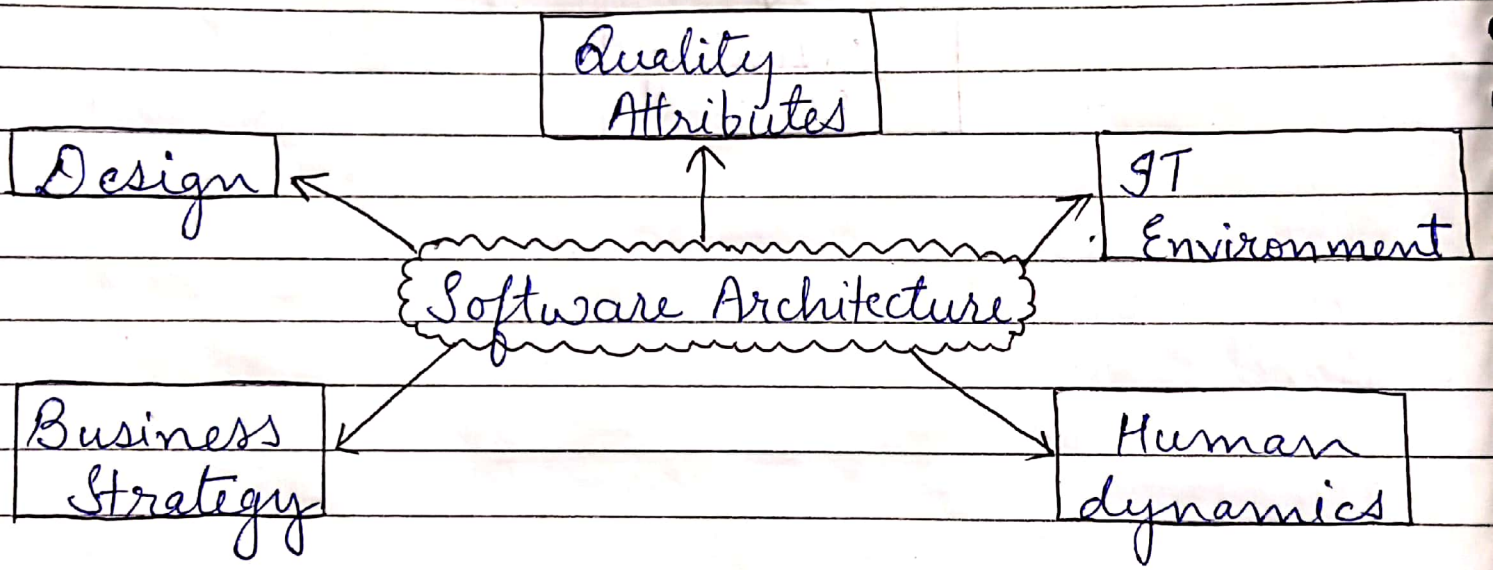


08/11/19

Unit-3 (Software Architecture & design)

* **Software Architecture:**

Architecture serves as a blueprint for a system. It provides an abstraction to manage the system complexity and establish a communication and coordinator machnism among components. It defines a structured solution to meet all the technical and operational requirements while optimizing the common quality attributes like performance & security.

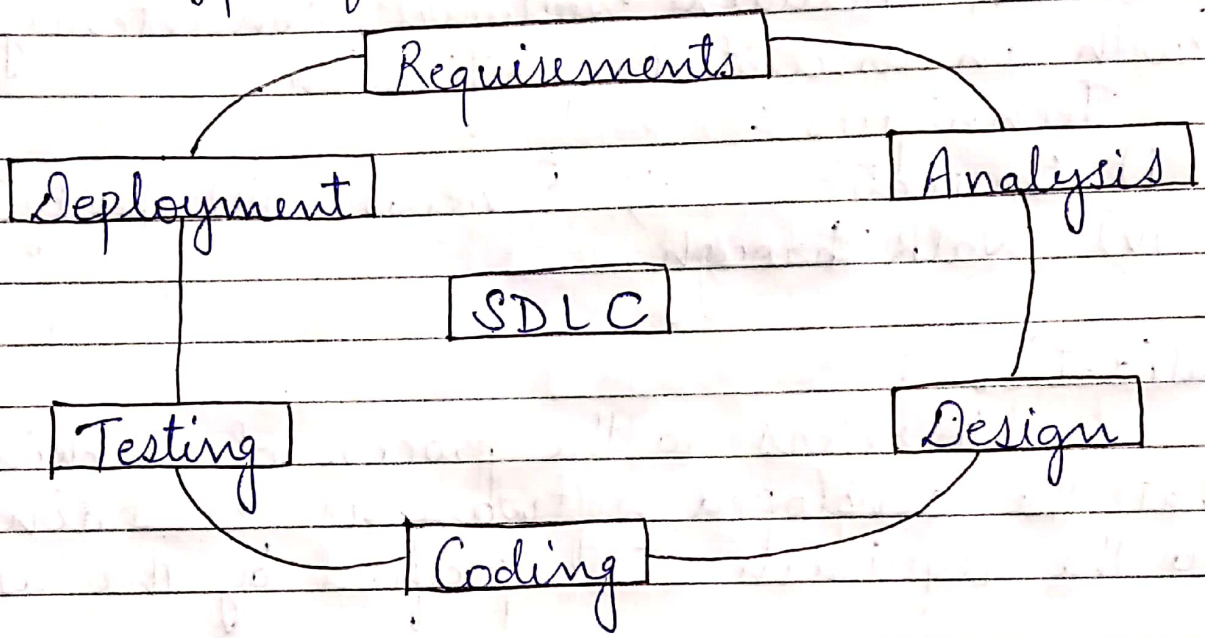


15/11/19

Unit-5 (Testing and Maintenance)

* Overview :
  - → Testing Introduction
  - → Testing Concepts
  - → Test plan
  - → Testing Principles
  - → Testing Strategies
  - → Types of Testing

*



* Testing :

Testing is a important phase of SDLC. Its main objective is to detect errors in the software.

IEEE formal definition, Software Testing is a process which is used to indentify the correctness, completeness and quality of software.

Two Important phases of Testing are -

a) Validation b) Verification

(A) Verification → "Are we developing the software right ??"

(B) Validation → "Are we developing the right software ??"

* Verifications:
Verification refers to checking or Testing of items, including software, consistency with an associated specification.
Techniques:
i) Inspection ii) Reviews iii) Analysis
iv) Walk Through

* Validation:
It refers to the process of checking that the developed software is according to the requirements specified by the user.

* Principles of Software Testing:
(i) Define the expected output.
(ii) Inspect output of each test completely.
(iii) Include Test cases for invalid and unexpected conditions.
(iv) Test the modified program to check its expected performance.

* Test Plan:
A Test plan describes how testing

would be accomplished.

It is defined as a document that describes the objectives, scope, method and purpose of software Testing. This plan identifies test items, features to be tested, testing task and the persons involved in programming these tasks.

Test environment, Test design & measurement techniques that are to be used.

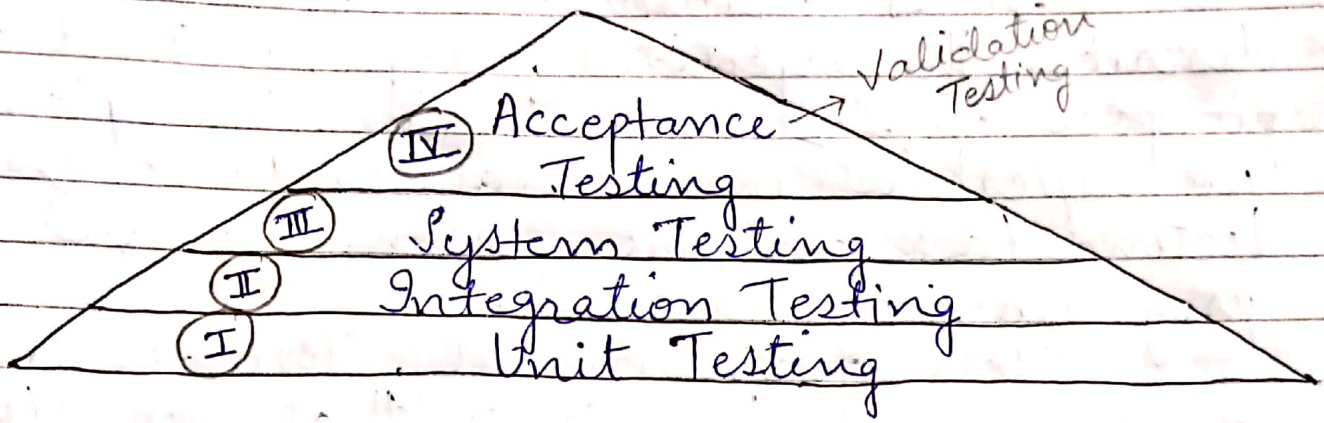
* Components of Test Plan:
  i) Responsibilities (Duty assignment)
  ii) Assumptions (avoid misunderstandings)
  iii) Test (Testing scenerio)
  iv) Communication (get familiar & share problems)
  v) Risk Analysis (risk possiblities)
  vi) Defect Reporting (errors during reporting)
  vii) Environment (supportable platform)

* Test Case Design:

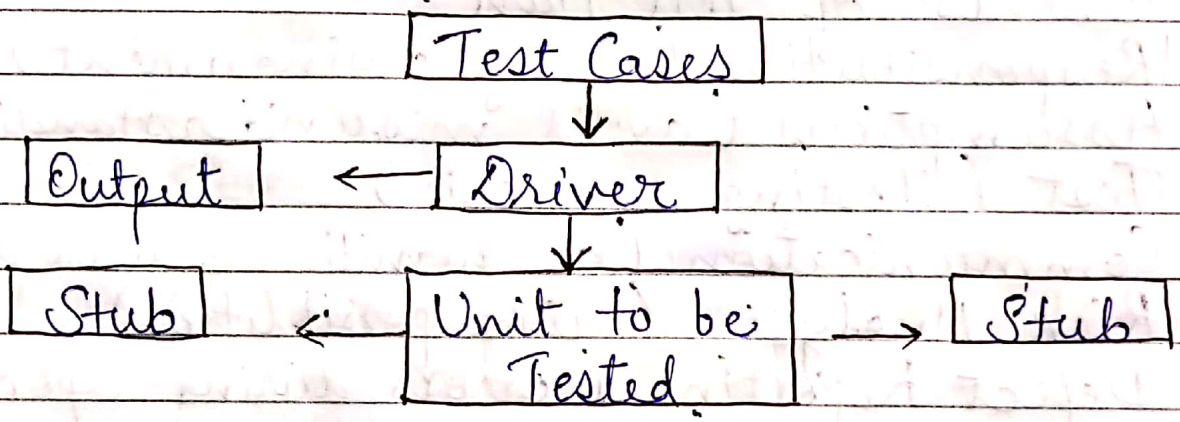
A Test Case is a document that describes an input, action or event and its expected result, In order to determine whether the software or part of the software is working correctly or not.

It should be developed in such a manner that it checks software with all possible inputs.

★ Software Testing Strategies :



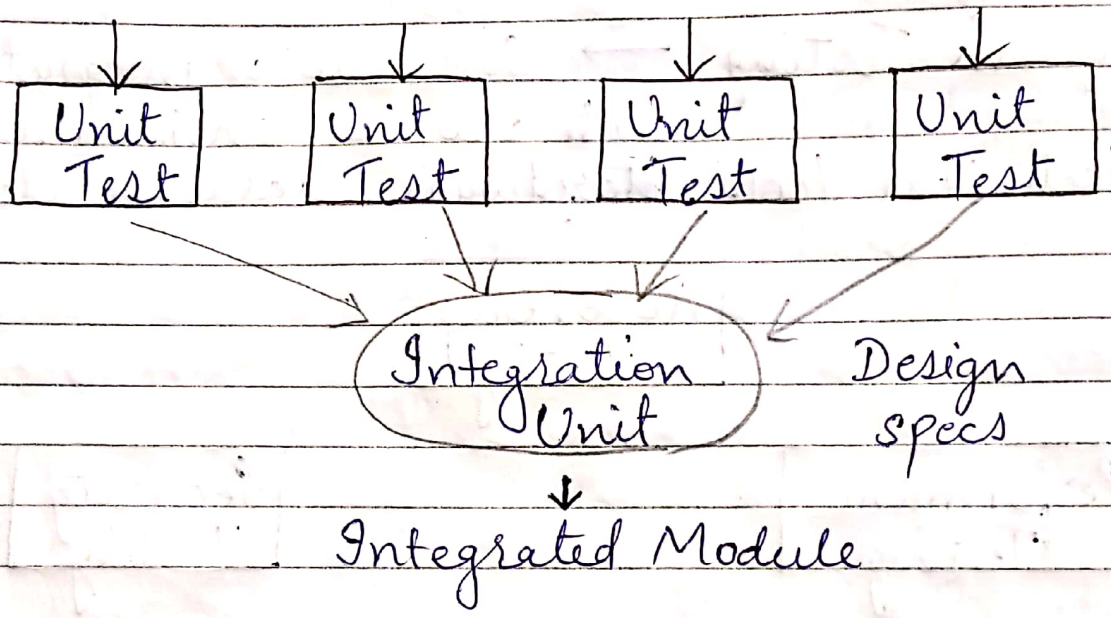
(i) Unit Testing → Software parts are tested individually.



- Driver → It is a module that passes input to the unit to be tested. It accepts test case data and then passes the data to the unit being tested.
- Stub → It is a module that works as a unit referenced by the unit being tested. It uses the interface of the subordinate unit does minimum data manipulation & returns control back to the unit being tested.

(ii) Integration Testing → All the modules have been unit tested.

- Integration Test approaches -
  a) Big-Bang
  b) Incremental Integration



- Big-Bang → Here, all modules are integrated then the entire program is tested.
- Incremental Integration → Test the program in small increments.

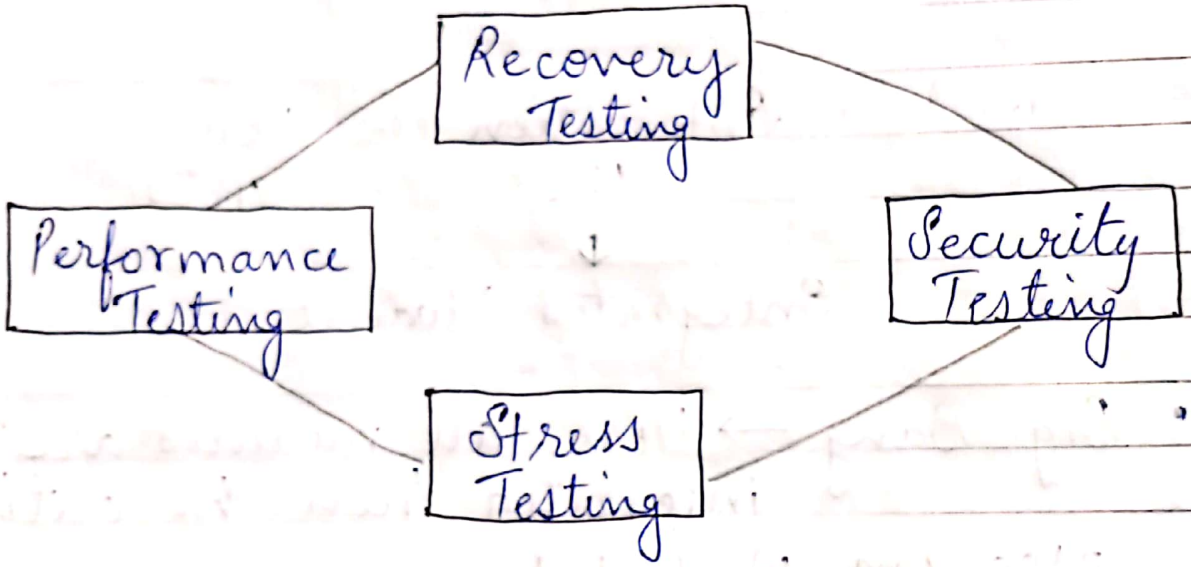
  Four patters -
  a) Top-down integration (parent-child mode)
  b) Bottom-up integration (child-parent mode)
  c) Regration integration (Re-Testing mode) (Expensive & Time-consuming)
  d) Smoke Testing integration (It is performed when s/w is underveloped)

- Smoke Testing → All modules are developed

and integrated in a form of cluster.

- It is performed when a new cluster developed & it checks the particular cluster is suitable to sent for further testing or not.

(iii) System Testing → Software is integrated with people, environment, technical tools, Hardware, database, etc.



- Stress Testing → Test the software with abnormal situations.
- Performance Testing → Speed, accuracy, stability should be checked.
- Security Testing → Application & System Security should be checked.
- Recovery Testing → Recovery of the damaged software or backup.

(iv) Acceptance - Validation → It is perform to determine software meets all the functional, behavioural, and performance requirements or not. Here, software is evaluated by a group of users, either at the developer site or user's site.

Types -

a) Alpha → It is conducted by the users at the developer's site.
(Internal Acceptance)
Eg: Online games relaunched.

b) Beta → It is conducted at the user's site.
(External)

- Alpha Testing → It is also said Internal-Acceptance Testing.
- Beta Testing → It is also said External-Acceptance Testing. Live Testing is conducted.

* Types of Testing :

(i) White-Box Testing → It is also called structural testing. It checks the internal structure of a program for this testing, tester should have a proper knowledge of the program code & the purpose for which it is developed.

Objective :

... To ensure that the test cases exercise each path through a program.

ii) Black-Box Testing → It is also called Functional Testing. It checks the functional requirement & examines the input & output data of these requirements. To recover interface errors.

iii) Gray-Box Testing → It is a mixture of White-Box and Black-Box testing. It is especially used in Web-applications because these applications are built around loosely-integrated components, operate and interact. It can be applied on the real-time systems.