# UNIT - I



## Introduction :-

**Software Engineering :-** The term SE is the product of two words, S/w, and engineering.

→ S/w is the collection of integrated programs (or) Executable programs & their associated libraries & documentation.

→ S/w consists of carefully-organized instructions & code written by developers on any of various particular computer languages.

→ Computer programs are related documentation such as requirements, design models and user manuals.

**Engineering :-** It is the application of scientific and practical knowledge to invent, design, build, maintain & improve frameworks, processes etc.

S/w product
- Requirements ①
- S/w analysis ②
- S/w design ③
- code design ④
- Testing ⑤
- deployment ⑥
- maintenance ⑦
- updates ⑧

**Why is S/w engineering required (or) purpose of S/w engineering**

1. To manage large S/w
2. For more scalability → capability to be changed in size (or) scale.
3. Cost management
4. To manage the dynamic nature of S/w.
5. For better quality management.

## Importance of s/w engineering

(2)

1. Reduces complexity → solving the problems.
2. To minimize s/w cost → To pay more to manpower.
3. To decrease time → no waste of time.
4. Handling big projects → ~~If any~~
5. Reliable s/w → If any bugs come in the s/w, the company is responsible for solving all these bugs.
6. Effectiveness. → reaching (or) doing the project fastly.

## Applications of software Engineering

→ The s/w is used extensively in several domains.

1) System software :- → Different software applications.
→ It is necessary to manage the computer resources & support the execution of application program.
Eg:- compilers, editors.

2) Networking & web applications s/w :- It provides the requi support necessary for computers to interact with each other. Eg:- HTML, PHP, XML.

3) Embedded s/w :- This type of s/w is embedd into the H/w. normally in Rom as a part of large s/y.
Ex:- washing machines, satellites.

4) Reservation s/w :- It is primarily used to store & retr information & perform transactions related to air travel other activities.

5) Business s/w :- Schools, colleges (or) shops, post It is used to support the business applications & is the most widely used category s/w.

6) Entertainment s/w.

7) AI s/w 8) Scientific s/w 9) utility s/w 10) Doc m s/

Software :- S/w is a collection of executable program, codes, and associated libraries and documentation.

(3)

Characteristics of S/w

1) S/w is developed (or) engineered, not manufactured. → modify
   ↳ new things.
2) S/w does not wear out — no expiry date.
3) S/w is custom built → Customer choices.

---

→ S/w Engineering is a technique through which we can developed (or) created S/w for computer systems.

→ In other words, S/w engineering is a process in which user needs are analyzed & S/w is designed based on the needs.

→ S/w engineers build these S/w and applications by using designing & programming languages.

* Purpose of S/w Engineering

1) To manage large S/w.
2) For greater scalability.
3) To manage the cost.
4) To manage the dynamic nature of the S/w
   ↳ changes like adding (or) updating details.
5) For better quality management.

---

IEEE defines S/w engineering as the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of S/w; i.e, the application of engineering to S/w.

## Goals of S/w Engineering

Primary goals of S/w engineering are:-

1) To improve the quality of the S/w products.
2) To increase the productivity &
3) To given job satisfaction to the S/w engineers.

**Software Engineer:-** A S/w engineer is a person who applies the principles of S/w engineering to design, develop, maintain, test and evaluate computer S/w.

**Processmodels:-** (or) ~~Sof~~ software processes

-> S/w processes are the activities for designing, implementing, testing a S/w S/y.

~~Task~~ →

-> processmodel is an abstraction of the actual process which is being described.

process models may contain activities which are part of the S/w product & the roles of people involving S/w.

-> It is a mechanism of dividing S/w development into distinct phases ~~into~~ to improve designing, product management & project management. it is also known as SDLC.

1. Evolving role of software

Evolve → develop gradually

The company has involved into a major product development.

→ S/w takes on a dual role.

1) S/w is a product – produces, manages & displays the information

2) S/w is a vehicle for delivering a product (process)

→ The process of developing a S/w product using S/w engineering principles and methods is referred to as S/w evolution.

→ S/w Evolution is a term which refers to the process of developing S/w initially, then timely updating it for various reasons, i.e., to add new features (or) to remove obsolete functionalities etc.

2) S/w is a ~~value~~ product :- It delivers the computing potential embodied by computer hardware (or) by a network of computers.

S/w is a vehicle for delivering a product (process) :-

It is information transformer – producing, managing, acquiring, modifying, displaying, (or) transmitting information that can be as simple as single bit (or) as complex as a multimedia presentation.

⇒ S/w delivers the most important product of our time – information.

- It transforms personal data
- It manages business information to enhance competitiveness.
- It provides a gateway to worldwide information networks.
- It provides the means for acquiring information.

## * Changing Nature of Software

The 7 broad categories of computer software present continuing challenges for software engineers:

1. System S/W
2. Application S/W
3. Engineering / Scientific S/W
4. Embedded S/W
5. product - line S/W
6. web applications.
7. AI S/W

(1) S/y S/W :- It is a collection of programs written to service other programs.

---

Evolution

1945 - 65 - origin
1965 - 85 - Crisis.
1990 - 2000 - Internet
2000 - 2010 - light weight
2010 - till - AI, ML, DL.

# Software Development Life Cycle (SDLC) model

→ SDLC describes the sequence of phases (or) steps to develope any software.

→ A software life cycle model also termed as process model.

→ It is a pictorial and diagrammatic representation of the of the software life cycle.

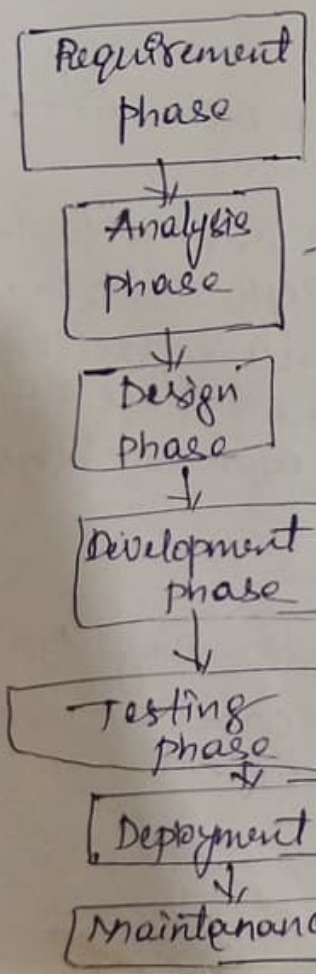
→ SDLC is a process used by the software industry to design, develope and test high quality softwares.

→ The SDLC aims to produce a high quality software that meets (or) exceeds customer expectations, reaches completion with in times and cost estimates.

→ The life cycle defines a methodology for improving the quality of software and the overall development process.

→ Without software life cycle models, it becomes tough for software project managers to monitor the progress of the project

SDLC Stages:-




① Requirement phase:-

→ Requirement analysis is the most important and fundamental stage in SDLC.

→ Business Analyst collect the requirement from the customer/client as per the clients business needs and documents the requirements in the business requirement specification (BRS) and provides the same to development team.

② Analysis phase: (or) Defining requirements

→ Once the requirement gathering is done the next step is to define and document the product requirements and get them approved from the customer. This is done through s/w requirement specification document (SRS).

→ Key people involved in this phase are project manager, Business Analyst and senior members in the team. (8)

→ The outcome of this phase is software requirement specification

(3) Design Phase :- → (Design Document Specification) DDS

High level Design (HLD) :- It gives the architecture of the software product to be developed and is done by architects and senior developers.

Low Level Design (LLD) :- It is done by senior developers. It describes how each and every feature in the product should work and how every component should work.

→ The outcome from this phase is High level Document and low level Document.

(4) Development phase :- This is the phase where we start building the software and start writing the code for the product.

→ Different high level programming languages such as C, C++, pascal, java and PHP are used for coding.

→ The outcome from this phase is source code Document of the developed product.

(5) Testing phase :- when the software is ready, it is send to the test team. once the QA makes sure that the software is error free, it goes to the next stage, which is implementation.

→ outcome of the phase is the Quality product. the testing certifies

(6) Deployment :- once the software is certified, and no bugs (or) errors are stated, then it is deployed.

(7) Maintenance :- Once when the customers start using the developed system then actual problems will come up and needs to be solved from time to time.

→ Release, upgrade.

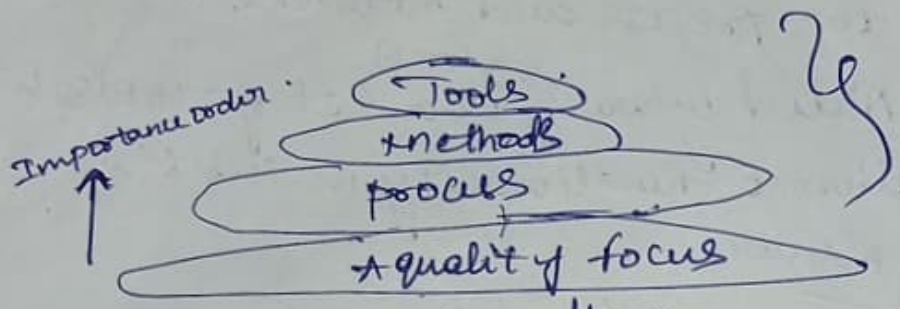
# Software Engineering - A Layered Technology



→ S/w engineering is a fully layered technology, to develop s/w we need to go from one layer to another.

→ All the layers are connected and each layer demands the fullfillment of the previous layer.

Figure :- → It allows a team to deliver even complex s/w on budget and on time.




Layers are divided into 4 parts

1) Quality focus :- → Degree of goodness, correctness, maintainability, reliability, security, provide security to s/w so that it data can be safely accessed by authorized person.

→ focus on planning, assurance, control, process improvement

Improvements → It defines the continous process improvement principle of s/w.

→ Hence it acts as a base layer (or) Bed rock for any product.

→ Functional & non-functional (structural)

2) Process :- → A framework that must be established for effective delivery of s/w.

→ "What" to do → Timely development of the s/w → management and control of s/w projects

→ Deals with Activities, actions & task.

→ It acts as a base layer.

→ Comes out with "How" to questions.

→ provide technical 'how-to's' for building a s/w

3) Methods :- → how to use.

How to do → Deals with "How to" implement.

→ Communication.

→ Has tasks for requirement and design modeling analysis.

→ Using of programming tools.

→ Testing & support.

4) Tools :-

→ Tools are integrated which means information created by one tool can be used by another

(10)

→ Environment

→ Helping hand of process.

→ Automated support / semi automated.

→ used for code, design, test or sell.

→ supports for process and methods.

→ Computer Aided software (CASE) is a set of tools working together to deliver a functionality in the s/w.

Process framework Activities

Activities
1) communication
2) planning.
3) modeling.
4) construction
5) Deployment

(5)

Tasks
1) Communication
2) requirement.
3) Analysis.
4) design
5) modeling
6) program construction.
7) Testing & support

process activities.

1) s/w specification
2) s/w Development
3) s/w validation.
4) s/w Evolution

Actions

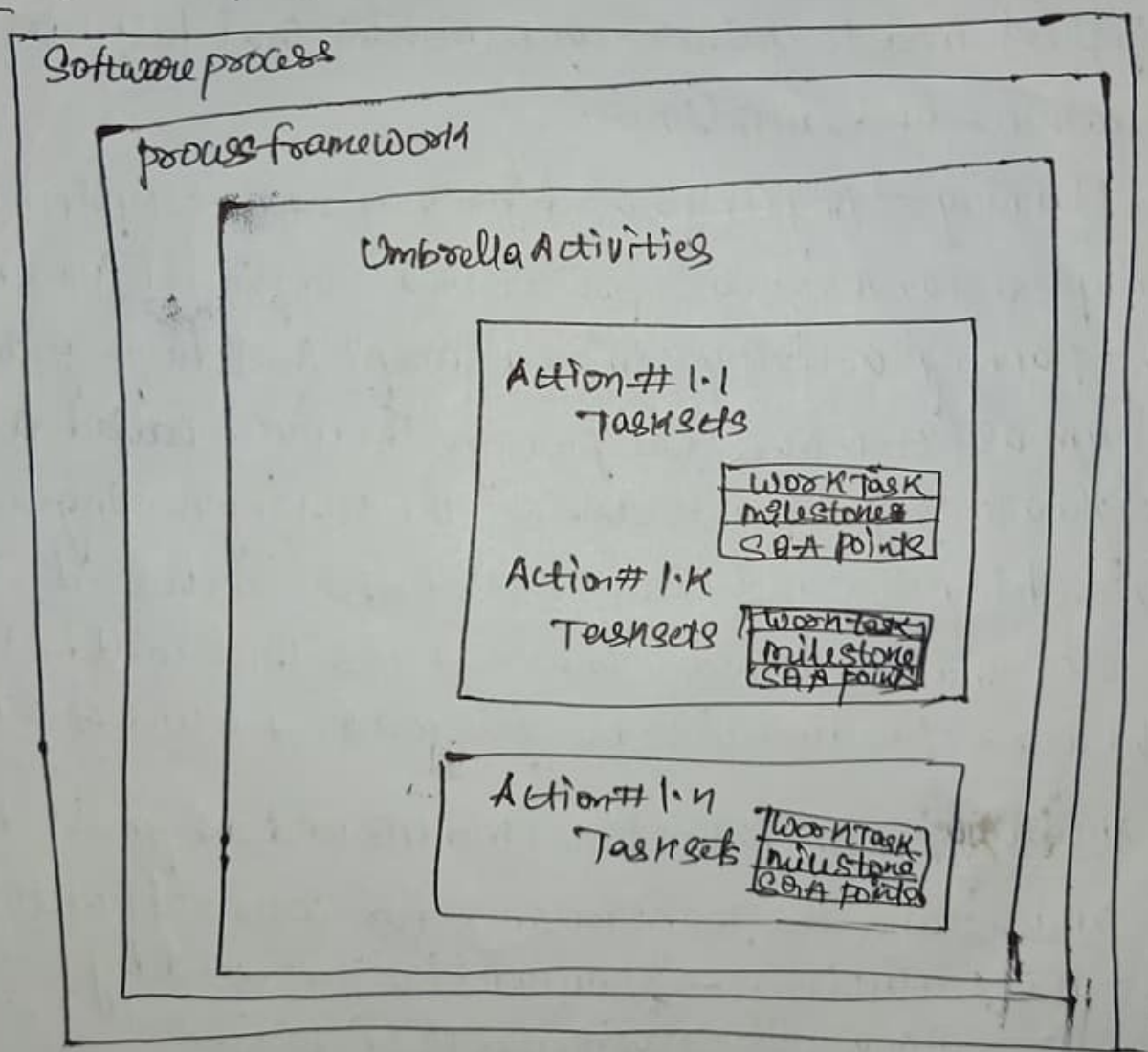
# * Software process Framework

→ Frame work is a standard way to build and deploy applications

→ Software process framework is a foundation of complete S/W Engineering process.

→ Software process frame work includes all set of umbrella activities and framework activities that are applicable to all software projects.

Figure:-




Tasks → focus on small, specific objective

Actions → set of tasks that produce a major work product

Activities → group of related tasks and actions for a major objective.

→ A generic process framework for software engineering encompasses five activities.

(12)

Process framework Activities:

① Communication:— Before any technical work can commence, it is critically important to communicate and collaborate with the customer; the intent is to understand customers (stakeholder's) objectivies for the project and to gather requirements that help define S/w features and functions.

② Planning:— Any complicated journey can be simplified if a map exists. A software project is a complicated journey, & the planning activity creates a "map" that helps guide the team as it makes the journey. The map — called a software project plan — defines the S/w engineering work by describing the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, & a work schedule.

③ Modeling:— Architectural models and design to better understand the problem and for work towards the best solution. → S/w model is prepared by
① Analysing requirements ② Design.

④ Construction:— Code generation & testing.

⑤ Deployment:— The S/w is delivered to the customer who evaluates the delivered product & provides feedback based on the evaluation.

# Umbrella activities



→ SE process framework activities are complemented by a number of Umbrella activities.

→ In general, umbrella activities are applied throughout a S/w project and help a S/w team manage and control process quality, change and risk

1. Software project tracking and control – allows the S/w team to assess progress against the project plan and take any necessary action to maintain the schedule.
2. Risk management – assesses risks that may affect the outcome of the project (or) the quality of the product.
3. Software Quality Assurance (SQA) – defines and conducts the activities required to ensure software quality.
4. Formal Technical Reviews (FTR) – asseses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.
5. Measurement – defines and collects process, project, and product measures that assist the team in delivering S/w that meets stakeholder's needs; can be used in conjuction with all other framework and umbrella activities.
6. Software Configuration management :– manages the effects of change throughout the S/w process.

(7) Reusability management – defines criteria for work product reuse and establishes mechanisms to achieve reusable components.

(8) Work product preparation & production – encompasses the activities required to create work products such as models, documents, logs, forms & lists.

# Capability Maturity Model Integration (CMMI) (5)

→ Developed by SEI in 1990's

→ CMMI helped organization to improve the quality of s/w they developed.

→ Adoption of CMMI has the significant business benifits. → It is complete SPI framework.

Software Engineering Institute Capability maturity model (SEICMM).

→ CMM is a procedure used to develop and refine an organization's s/w development process.

→ CMM is used as a benchmark to measure the maturity of an organization's s/w process.

Methods of SEICMM 2 ways

① Capability Evaluation.
② S/w process Assessment.

① process meta model intro (capa) continous represen
② Staged (maturity)

SEICMM can be used in two ways

① Capability Evaluation :-

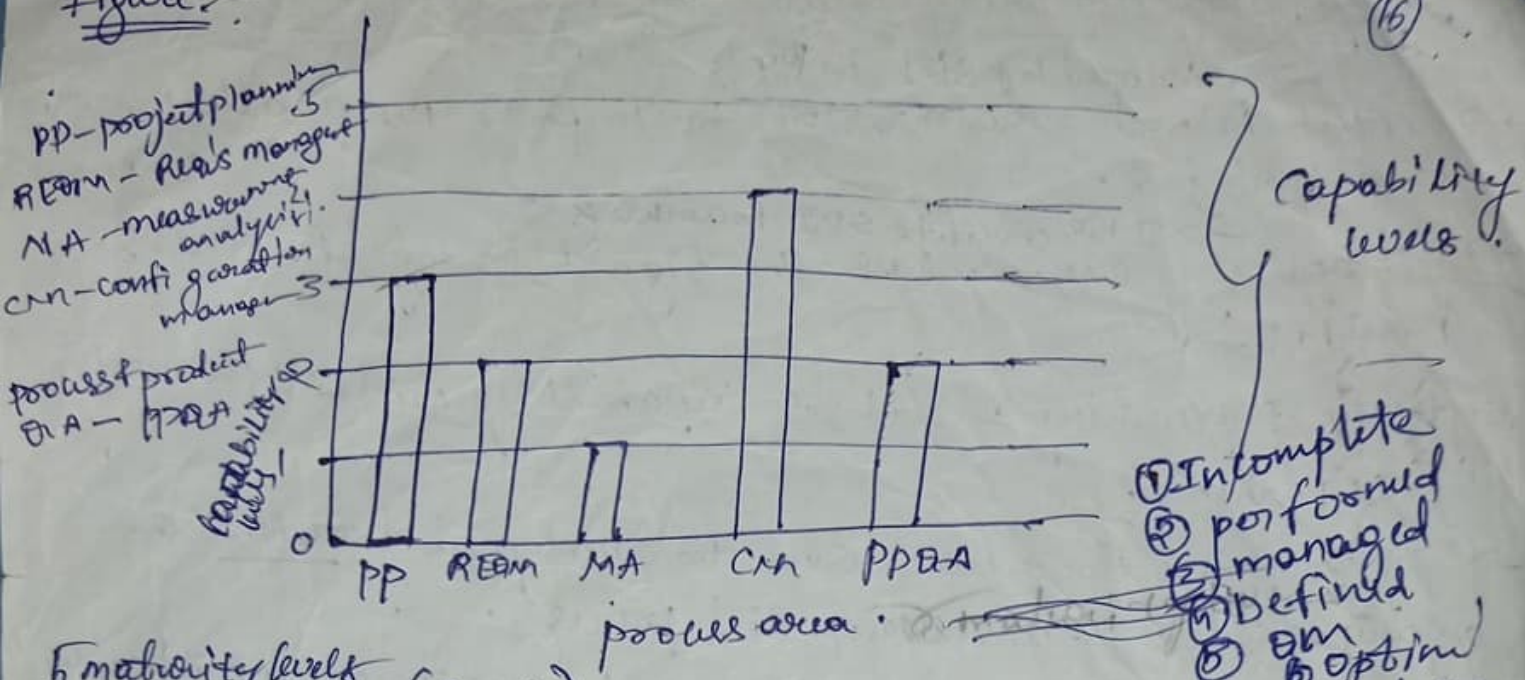
→ The Capability Evaluation provides a way to access the s/w process capability.

② S/w process Assessment :-

→ It is used by an organization with the objective to improve the process capability.

→ SEI CMM classify s/w development institutes into 5 maturity levels.

Figure:




5 maturity levels (CMM)

① Initial - level 1 :- A s/w development organization at this level is characterized by adhoc activities (or) necessary (or) needed (org - not planned in advance).

② Repeatable - level 2 :- At this level the basic process management practices such as tracking cost & schedule are established.
→ A soc use the previous successfull project approach this method is repeatable.

③ Defined - level 3 :- At this level process for both management & development activities are defined & documented.

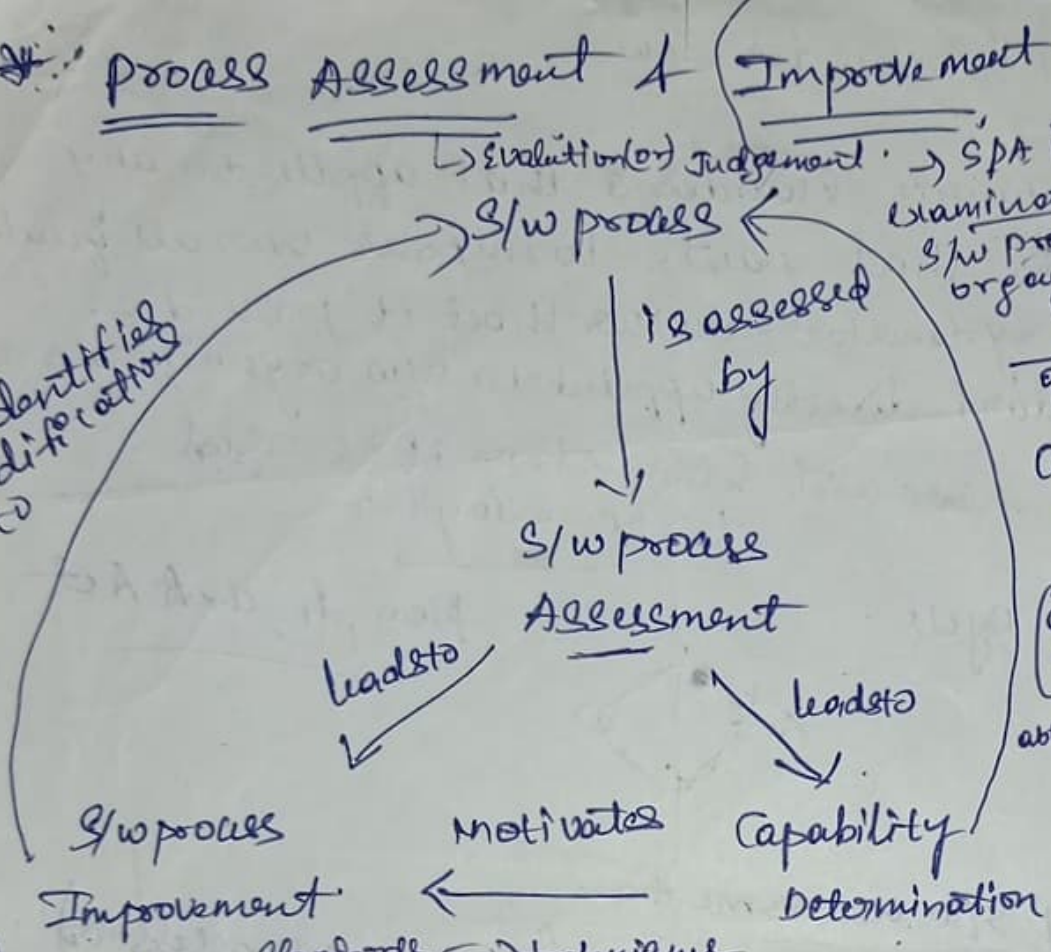
④ managed - Level 4 :- At this level, the focus is on s/w metrics (s/w metric is a measure of s/w characteristic which are measurable (or) countable).

⑤ Optimizing - level 5 :- At this stage, process & product metrics are collected.
→ Highest quality in products
→ Low risk in process performance
continous improvement

good

# Process Assessment & Improvement

↳ Evaluation (or) Judgement. → SPA is a disciplined examination (or) test of the S/w processes used by an organization based on process model.




identifies capabilities & Risks of

(assessed means evaluate (or) estimate the ability/quality.)

1) Standard CMM Assessment method for process improvement (SCAMPI)

(Standards (or) techniques)

→ It provides a five step process assessment model that incorporates five phases: initiating, diagnosing, establishing, acting & learning.

2) CMM-Based Appraisal for internal process improvement (CBA IPI)

→ It provides a diagnostic technique for assessing the relative maturity of S/w organization in SEI CMM.

3) SPICE - The SPICE (ISO/IEC 15504):-

↳ International Standard organization

→ It defines a set of requirements for S/w process assessment. (SPICE - S/w process improvement & capability determination).

The intent of the standard is to assist organization in developing objective evaluation of efficacy of and defined S/w process.

4) ISO 9001 : 2000 for S/W



→ It is a generic standard that applies to any organization that wants to improve overall quality of products, systems (or) services that it provides. This standard directly applied to S/W organizations.

→ ~~It is~~ In current situation it is used.
↳ nowadays.

→ It follows PDCA cycle. P' plan, do, check Act

3 types of S/W assessments.

① A self-assessment (First-party assessment) is performed internally by an organization's own personnel.

② A Second-party assessment is performed by an external assessment team.

③ A Third-party assessment is performed by an third party team.

## Process models :-

→ process models defines a distinct set of activities, actions, tasks, milestones and work products that are required to engineer high quality s/w.
(or)

→ A s/w process model is an abstraction of the s/w development process. → Quality of dealing with ideas

→ The models provide a road map for s/w engineering work.

→ S/w engineers and their managers adopt the process models i.e., decides the process model according to the customer needs and follows it.

→ S/w engineers have choosen a generic process framework that encompasses the following framework activities. ① communication ② planning ③ modeling ④ construction ⑤ Deployment.

→ prescriptive models became they prescribe a set of process elements framework activities, s/w engineering actions tasks, work product, quality assurance and change control mechanisms for each project.

### Process Models Types

I. Water Fall model (or) Linear-sequential life cycle model (or) classical (or) Traditional.
II. Incremental model.
} prescriptive process model. ↳ Describes "how to do" according to a certain s/w process s/y.

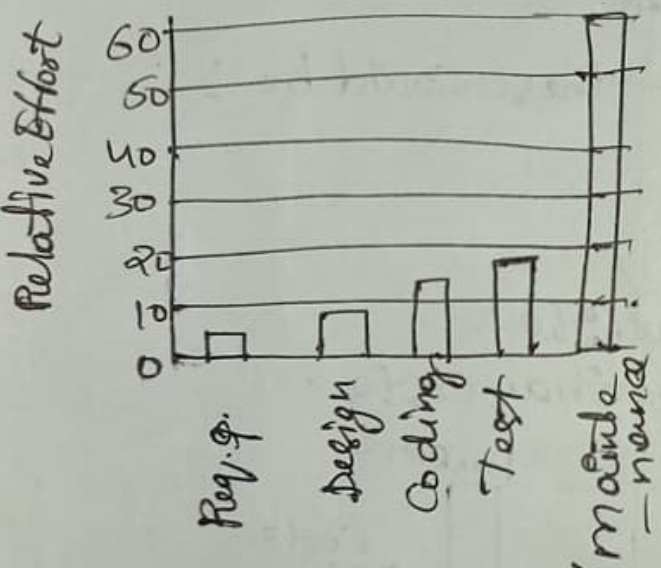
III. Evolutionary process models.
① prototype model. /II
② Spiral model
③ Concurrent development model.

IV. Unified process model.

# * Waterfall Model (Classical)

Feasibility Study → Requirements Analysis → Design → Coding → Testing → maintenance

**Relative Effort for phases:-**




→ phases b/w feasibility Study and testing - called Development phases.

→ Among all life cycle phases - maintenance phase consumes maximum effort.

→ Among development phases, Testing phase consumes the maximum effort

## ① Feasibility Study :- (Feasible means possible to do easily (or) conveniently).

→ Feasibility study has 3 dimensions:

- Technical Feasibility
- Economic Feasibility (also called cost / benefit feasibility)
- Schedule Feasibility.

→ Main aim of FS is to determine whether developing the S/w is
- Financially worthwhile,
- Technically Feasible.

→ Roughly understand what customer wants.

Activities During FS

→ Work out an overall understanding of the problem.
→ Formulate different solution strategies.
→ Examine alternate solution strategies in terms of:
- resources required,
- Cost of development &
- development time.

→ perform a cost/benefit analysis:
- Determine which solution is the best.

Cost benefit analysis (CBA).

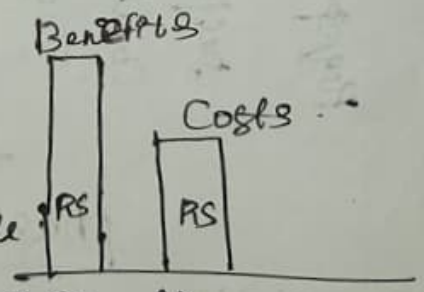
→ Need to identify all costs – these could be:
- Development costs
- setup
- operational costs.

→ Identify the value of benefits.
→ Check benefits are greater than costs.

The business case

Cost include:-
- Development
- operation

Benefits:
- Quantifiable
- Non-quantifiable




## (2) Requirements Analysis & specification

→ Aim of this phase:
- understand the exact requirements of the customer
- Document them properly.

→ Consists of 2 distinct activities:
- Requirements gathering & analysis.
- Requirements specification.

→ Gather requirements data from the customer :
- Analyze the collected data to understand what customer wants.

→ Remove requirements problems :
- Inconsistencies - not saying throughout.
- Anomalies - something deviates from what is standard. (Exp...)
- Incompleteness :- - insufficient.

→ Organize into a SRS document.

Requirements Gathering

→ Gathering relevant data :
- Usually collected from the end-users through interviews & discussions.

Eg :- for a business accounting S/W :-

Requirements Analysis

→ The data you initially collect from the users :
- usually several contradictions & ambiguities.
- Why?
- Each user typically has only a partial & incomplete view of the system.

→ Ambiguities & contradictions :
- must be identified.
- resolved by discussions with the customers.

→ Next, requirements are organized.
- into a SRS document,

## 3) Design

-> During design phase requirements specification is transformed into:
- A form suitable for implementing in some programming language.

-> Two commonly used design approaches:
a) Traditional approach,
b) Object oriented approach.

a) Traditional Design approach.

-> Consists of two activities:
=> Structured analysis (typically carried out by using DFD)
=> Structured design.

Structured design

High-level design:-
- decompose the s/w into modules,
- represent invocation relationships among the modules.

Detailed design (low-level)
- different modules designed in greater detail
-> data structures & algorithms for each module are designed.

-> First identify various objects occuring in the problem
- Identify the relationships among the objects.
- for example, the objects in a pay-roll s/w may be: employees, managers, pay-roll register, Departments etc.

b) Object oriented Design

→ object structure:
- Refined to obtain the detailed design.

→ OOD has several advantages:
- lower development effort,
- lower development time,
- Better maintainability.

(4) Coding

- Coding & Unit Testing

→ During this phase:
- Each module of the design is coded,
- Each module is unit tested.
  → That is, tested independently as a standalone unit, & debugged.
- Each module is documented.

(5) Integration & s/y testing

→ different modules are integrated in a planned manner:
- modules are usually integrated through a number of steps.

→ During each integration step,
- the partially integrated s/y is tested.

s/y testing

→ After all the modules have been successfully integrated & tested:
- s/y testing is carried out.
- s/y testing ensure that developed s/y functions according

to its requirements as specified in the SRS documented.



## 6 Maintenance

→ maintenance of any S/W:

- requires much more effort than the effort to develop the product itself.
- Development effort to maintenance effort is typically 40:60.

→ The effort spent on maintenance is 60% of the total effort spend to develop a full S/W.

Types

1. Corrective maintenance: carried out to correct errors that were not discovered during the product development phase.
2. perfective maintenance: carried out to enhance the functionalities of the S/W based on the customer's request.
3. Adaptive maintenance: porting the S/W to work in a new environment such as working on a new computer platform (or) with a new OS.

## * Incremental model (prescriptive process model)

↳ successive version model.

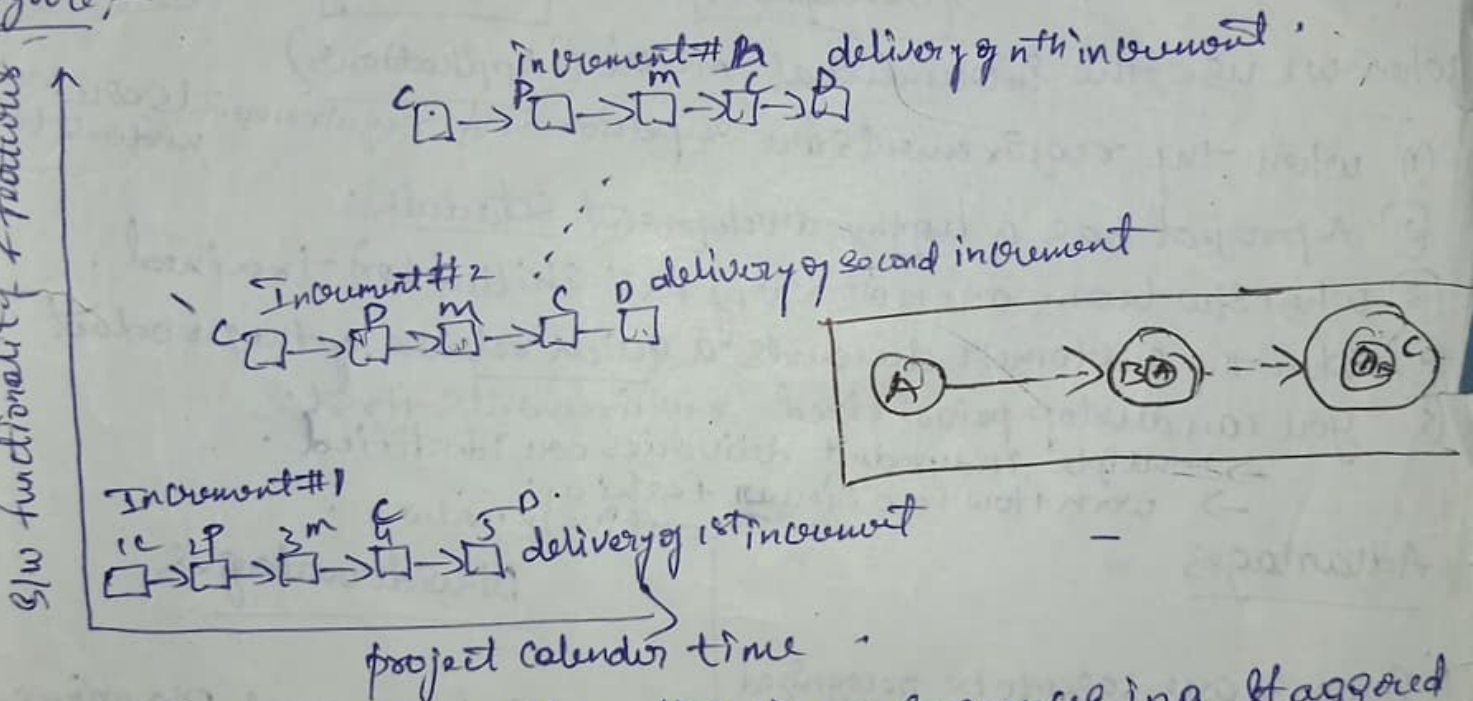
→ Incremental model is a process of s/w development where requirements divided into multiple standalone modules of the s/w development cycle.

→ Linear fashion.

→ In this model, each module goes through the requirements, design, implementation and testing phases.

→ Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

Figure:-




→ The incremental model applies linear sequences in a staggered fashion as a calendar time progresses.

→ Each linear sequence produces deliverable "increments" of the s/w.

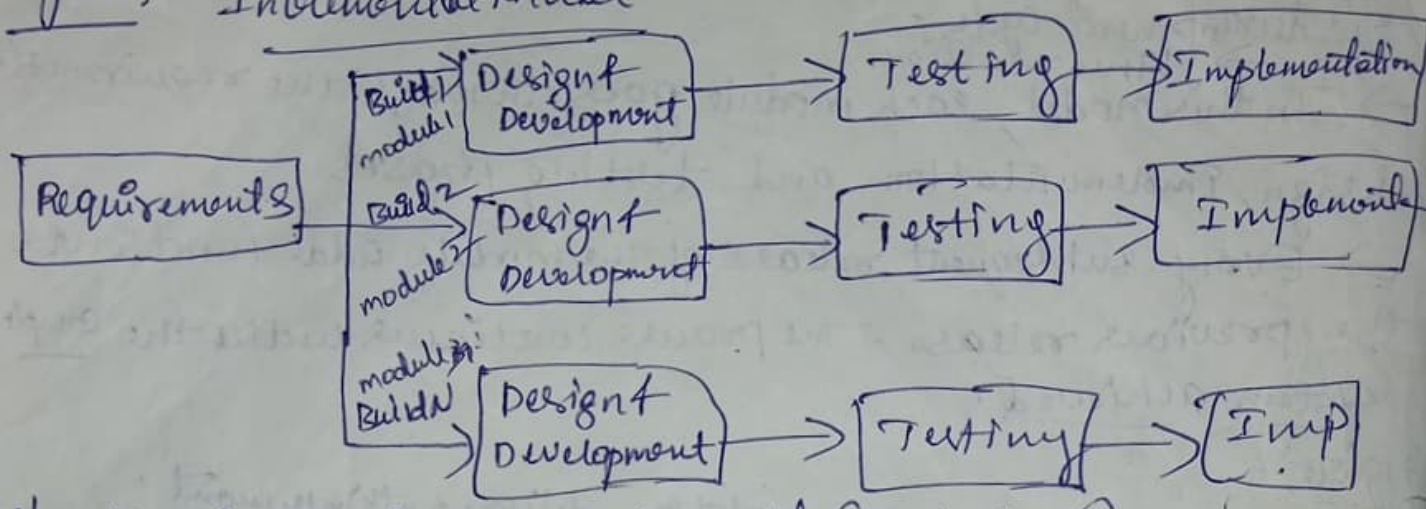
→ In the first increment s/w developed using the incremental paradigm might deliver basic file management, editing, and document production functions in this 1st increment.

→ More sophisticated editing & document production functions & capabilities in second increment.

-> Spelling & Grammar checking in the third increment.

-> Advanced page layout capability in the fourth increment.

Figure:- Incremental model




When we use the incremental model (Applications)

1. when the requirements are superior and requirements are well understood
2. A project has a lengthy development schedule.
3. when s/w team are not very well skilled (or) trained.
4. When the customer demands a quick release of the product.
5. you can develop prioritized requirements first.
   - -> multiple independent deliveries are identified.
   - -> workflow in a linear fashion.
   - -> Iterative in nature.

## Advantages

1. Errors are easy to be recognized
2. Easier to test and debug
3. More flexible.
4. Simple to manage risk because it handled during its iteration.
5. The client gets important functionality early.

## Disadvantages

1. Need for good planning
2. Total cost is high.
3. Well defined module interfaces are needed.

meeting of two points

# * Evolutionary process model

→ Evolutionary models are iterative-type models.

→ They allow to develop more complete versions of the S/w.

It has 3 types. ⇒ combination of incremental & iterative model.

1. The prototyping model
2. The Spiral model.
3. Concurrent development model.

① prototyping model (I) → dummy

→ A customer defines a set of general objectives for S/w.

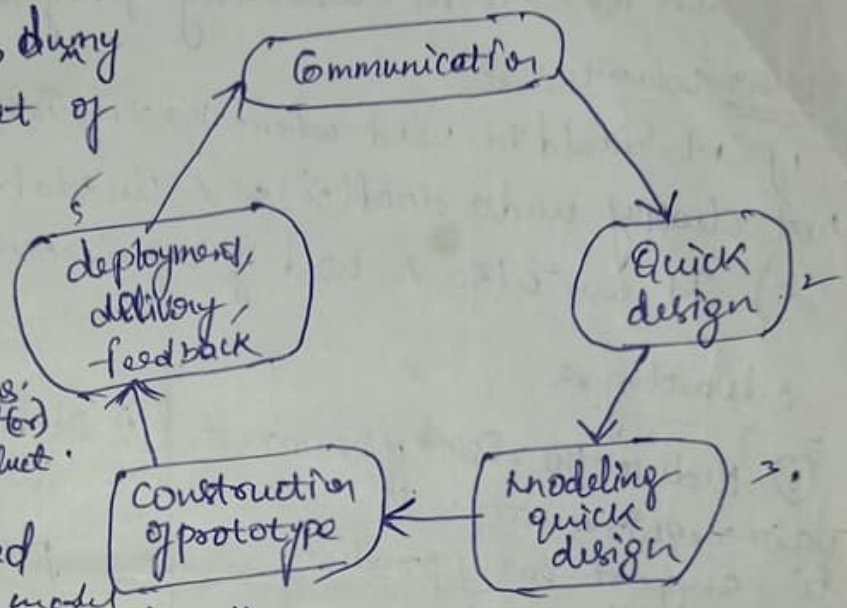
→ It does not identify detailed input, processing, (or) output requirements.

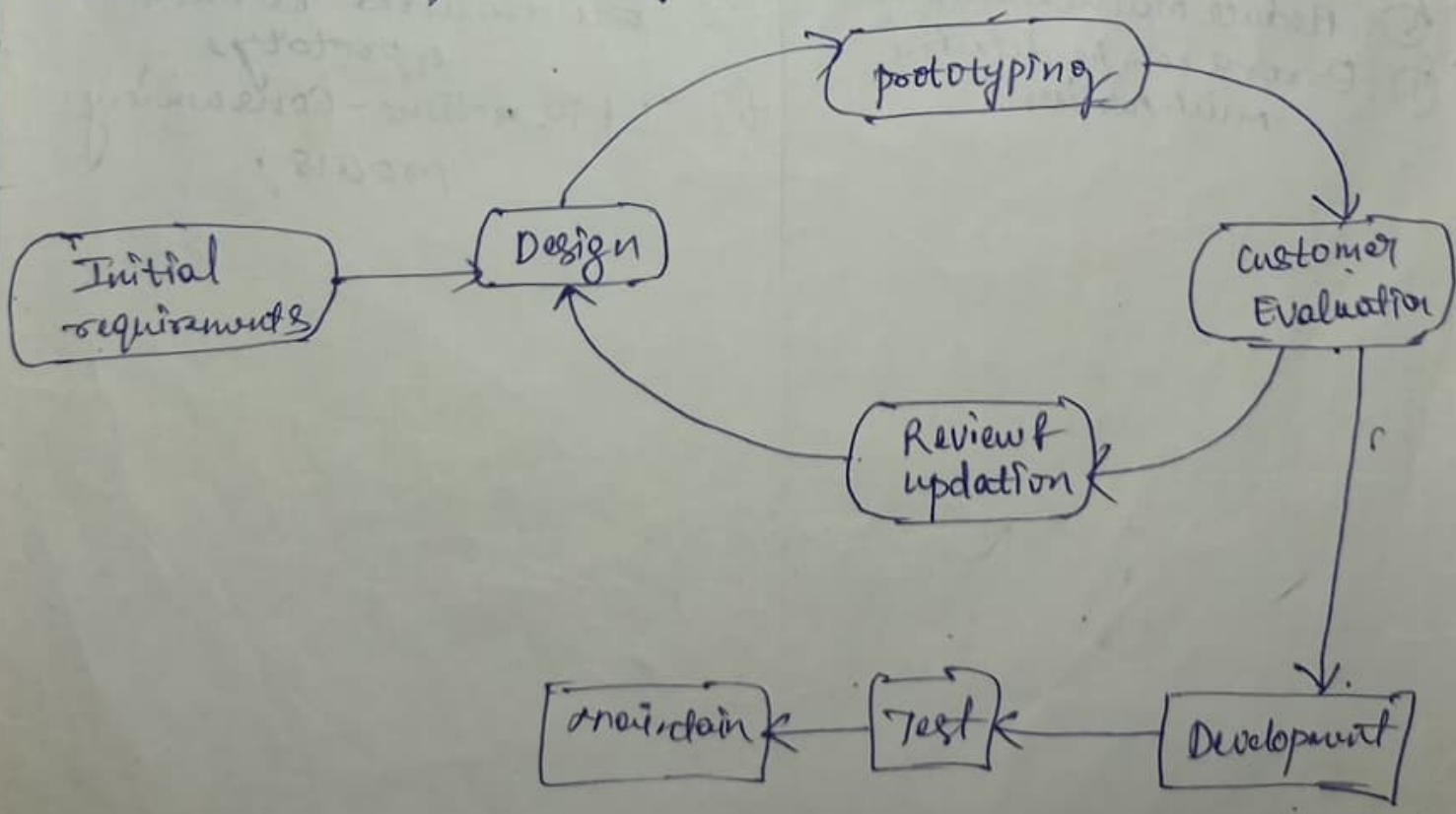
→ It uses multiple iterations.

→ prototype is nothing but first (or) preliminary version of the S/w product.

→ It is also called throughwaymodel.

Figure:-



→ In prototype model we use multiple iterations (or) requirements analysis & designs, after each iteration the design will be evaluated by the customer.

→ The prototype model is also a popular life cycle model.

→ The prototyping model can be considered to be an extension of the waterfall model.

→ It is s/w working model of limited functionality.

→ In this model, working programs are quickly produced.

Uses when to use.

1) It should be used when the requirements of the product are not clearly understood (or) are unstable.

2) It can also be used if requirements are changing quickly.

Advantages

① Reduce the risk of incorrect user requirement.

② support early product marketing.

③ Reduce maintenance cost

④ Errors can be detected much earlier.

Disadvantages

① Difficult to know how long the project will last.

② prototyping tools are expensive.

③ Special tools & techniques are required to build a prototype.

④ It is a time-consuming process.

# II. Spiral model

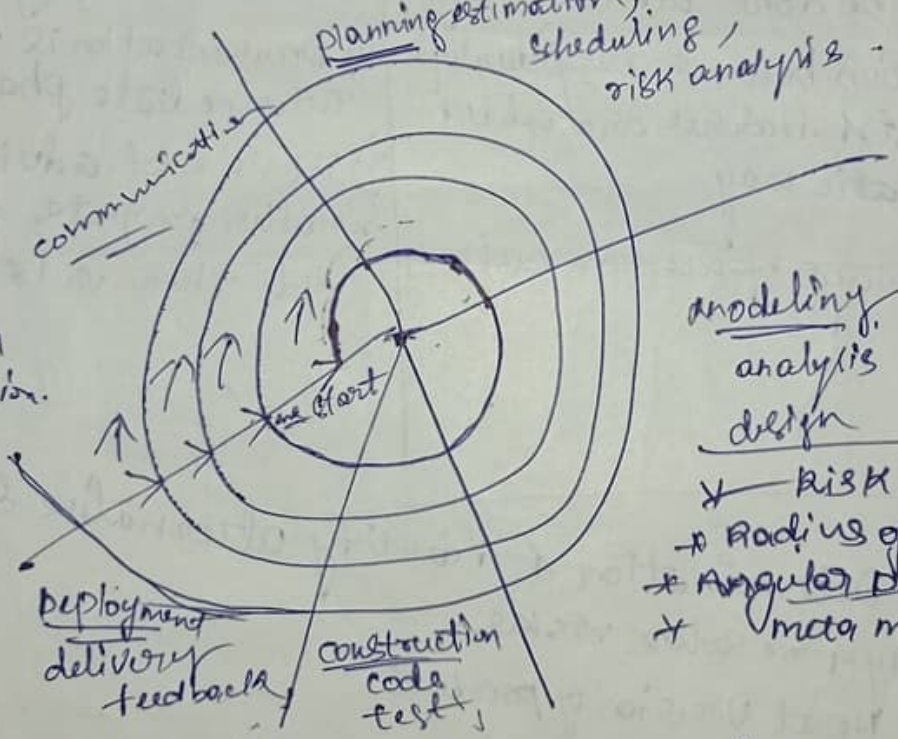
→ The spiral model is a systems development lifecycle (SDLC) method used for risk management that combines the iterative development process model with elements of the waterfall model. → metamodel.

→ It is the combination of waterfall model and iterative model.

→ The spiral model is used by s/w engineers and is favored for large, expensive and complicated projects.

→ s/w is developed in a series of incremental releases.

Figure:-




Quadrants

Phases
1) planning
2) modelling
3) Construction
4) Communication.

* Risk Handling
* Radius of spiral = cost
* Angular Dimension = progress
* meta model.

→ Each phase in spiral model begins with design goal and ends with client reviewing.

→ Inner circle is concept development.

→ Second inner circle is system development.

→ Third inner circle is system enhancement.

→ Fourth inner circle is maintenance.

## When to use Spiral model?

1) When project is large.
2) When releases are required to be frequent.
3) When risk & cost valution is important.
4) When requirements are unclear & complex.
5) When changes may require at any time.
6) For medium to high risk projects.

### Advantages

1) Additional functionality (or) changes can be done at later stage.
2) Cost estimation becomes easy
3) development is fast & features are added in a systematic way
4) There is always space for customer feedback.

### Disadvantages

1) Risk is not meeting schedule (or) budget.
2) Documentation is more as intermediate phases.
3) It is not advisable for smaller projects, it might cost them a lot.

---

1. objective Determination & identify alternative solutions
2. Identify & resolve risks.
3. Develop next version of product.
4. Review and plan for next phase.

### Advan

1) Risk Handling.
2) Large projects.
3) Flexible.
4) Customer satisfaction

### Disadvan

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

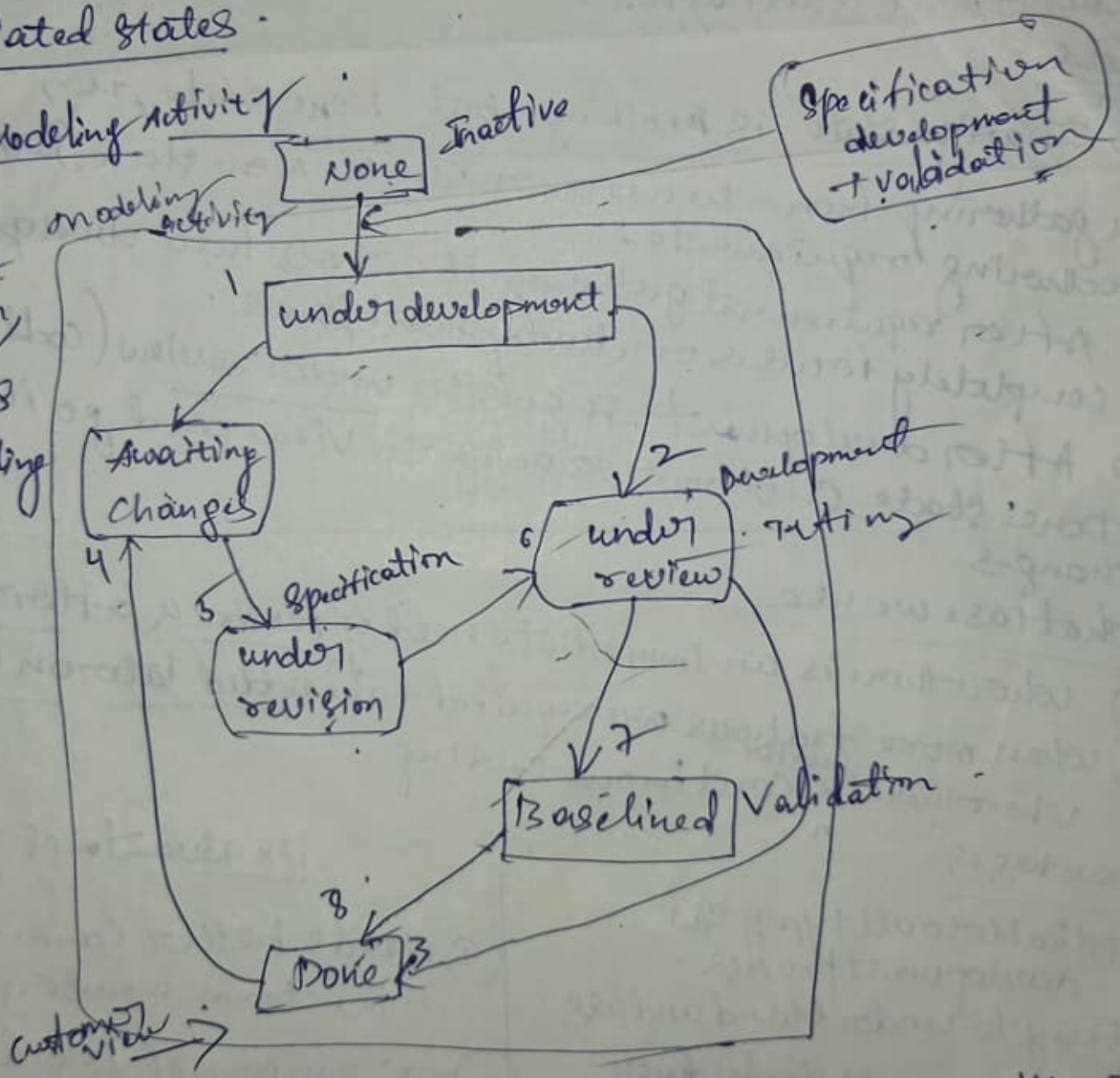
## III) Concurrent development model (CDM)

→ It is also called as concurrent ~~develop~~ process model (CPM) (or) concurrent engineering.

→ The term concurrent mean "done at the same time".

→ It can be represented ~~as~~ schematically as a series of framework activities, s/w engineering actions & tasks, and their associated states.

Figure :- Modeling activity




→ If we take waterfall model as an example, you will not know the activities going on in each phase, only after the phase is over, you get a work product or a document.

→ The concurrent process model shows the current state of activities, tasks and their associated states that remain in different phases.

-> Maintain information about each phase at the activity level.
-> The benefit of this model is that project managers know each phase is what state & why.



-> For example:- Design phase may be at an awaiting state and customer communication is in under revision state. The customer wants some changes to the design; the communication goes to awaiting changes and design goes to the under development state again.

States

-> "Inactive state" is nothing but None state, i.e., the requirements we are gathering from the customer. In the None state just we are gathering requirements.
-> After requirement gathering it enters into development state it completely focuses on development.
-> After development it goes to under review (coding & testing)
-> None state customer is going to view and going to changes

What case we use

1. where there is an immediate need of feedback after testing.
2. when more features are required to be added later on in the project.
3. when there is need for continous testing.

Advantages

1. Applicable to all types of S/W development process.
2. Easy to understand and use.
3. Gives immediate feedback from testing.
4. provides as accurate picture of current state of a project.

Disadvantages

1. Needs better comm. b/w the team members.
2. This may not be achieved all the time.

# Rapid Application Development (RAD) model

(prescriptive)

-> The RAD model is based on prototyping and iterative development with no specific planning involved. (34)

-> The process of writing the s/w itself involves the planning required for developing the product.

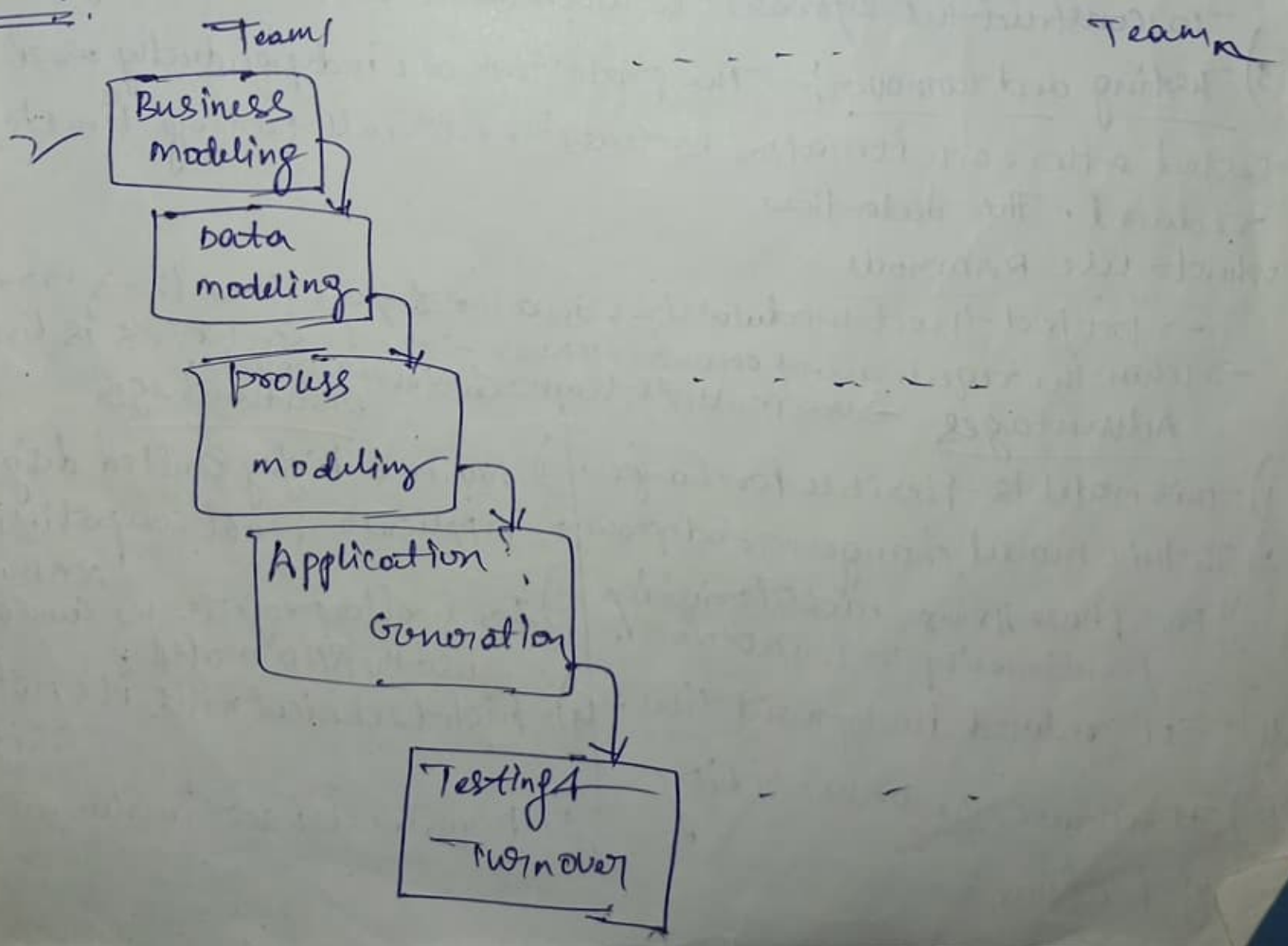
-> Using the RAD model, s/w product is developed in a short period of time.

-> RAD focuses on gathering customer requirements through workshops (or) focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes, continuous integration & rapid delivery

-> In general, RAD approach to s/w development means putting lesser emphasis on planning tasks and more emphasis on development & coming up with a prototype.

-> prototypes developed are reusable.

Figure:-



## Various Phases of RAD

1) **Business modelling**:- The information flow among business functions is defined by answering questions like
   - → what data drives the business process.
   - → what data is generated.
   - → who generates it.
   - → where does the information go,
   - → who process it and so on.

2) **Data modelling**:- The data collected from business modeling is refined into a set of data objects that are needed to support the business. The attributes are identified, and the relation b/w these objects is defined.

3) **Process modeling**:- The data objects defined in the data modelling phase are changed to fulfil the information flow to implement the business model. The process discription is created for adding, modifying, deleting (or) retrieving a data object.

4) **Application generation**:- In this the actual system is built. To construct the software the automated tools are used.

5) **Testing and turnover**:- The prototypes are independently ~~used~~ tested after each iteration so that the overall testing time is reduced. ~~The data flow~~

When to use RAD model
- → project that modularizes in a short span time (2-3 months).
- → when the requirement are well know → technical risk is limited
- → uses multiple teams on scalable project.

**Advantages**

1) This model is flexible for change.
2) In this model, changes are adoptable.
3) Each phase brings highest priority functionality to the customer.
4) It reduced development time.
5) It increases the reusability of features.

**Disadvantages**

1) Required highly skilled designers.
2) Application is not compatible with RAD.
3) For smaller projects, we cannot use the RAD model.
4) high technical risk, it's not suitable.
5) Required user involvement.

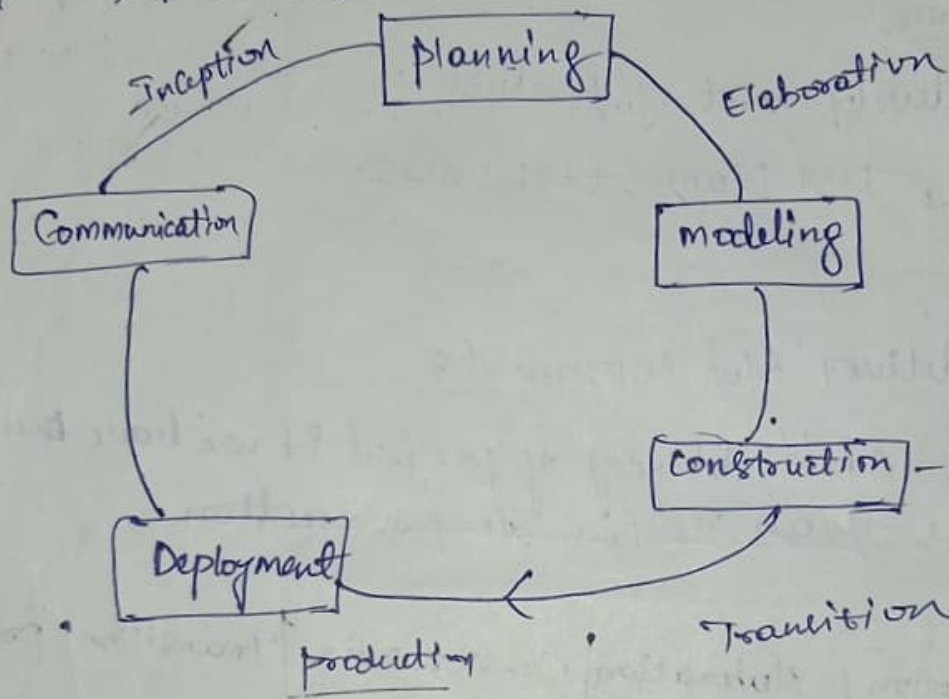
## * Unified process model (UPM) Rational Unified process (RUP)

→ The unified process model is a s/w development approach that consists of four phases of inception, elaboration, construction and transition.

(36)

→ Unified process model means combination of two ~~process~~ stages to form one phase.

Figure:- → UPM is designed and documented using UML




Inception
- communication
- planning

Elaboration
- planning
- modeling

Construction
- construction

Deployment
- production
- Transition

Phases:-

1) Inception:-

→ It is initial vision document based on use case model.

→ In this phase, you'd collect requirements from the customer and analyze the project's feasibility, its cost, risks and profits.

↳ assessment of the practicality of a proposed plan (or) method.

→ Fundamental business requirements are described through a set of preliminary use cases that describes what features and functions are derived by each major class of users.

→ Use case describes a sequence of actions that are performed by an actor, as the actor interacts with the s/w.

→ Use-cases help to identify the scope of the project & provide a basis for project planning.



2) Elaboration :-

→ To implement use-case model.
→ Analysis model.
→ S/w architecture description

3) Construction :-

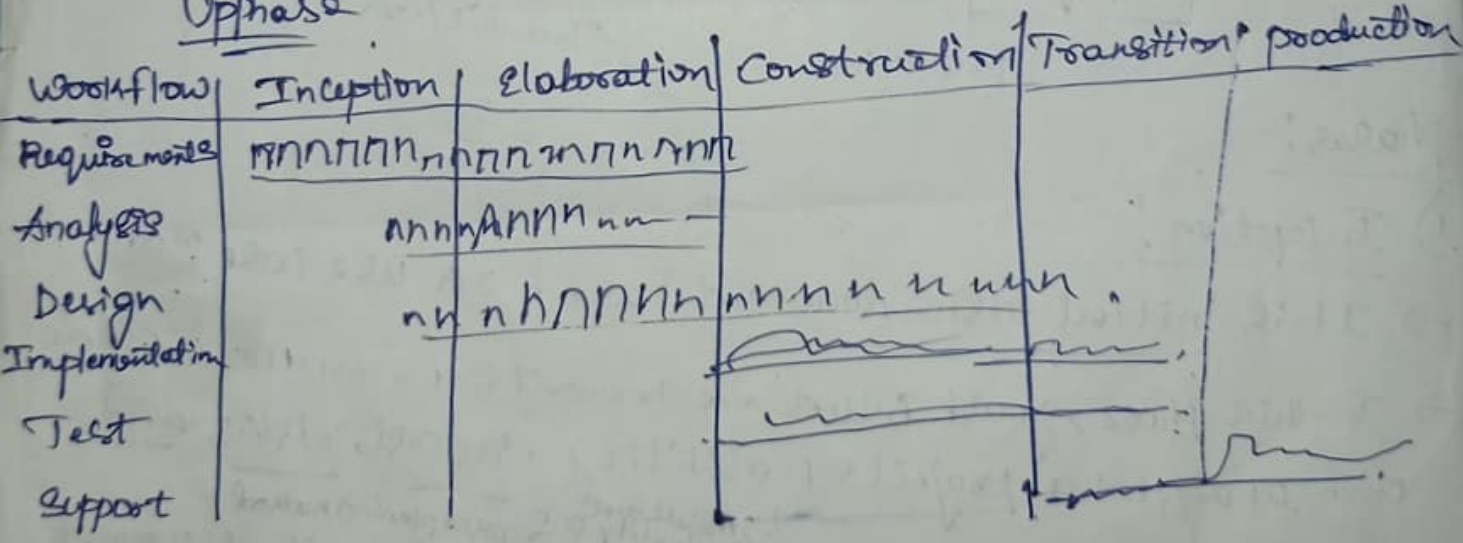
→ It means development of product.
→ Integrated, test plans, test cases.

4) Transition :-

Trying to deliver S/w increments

5) Production :- After delivery of project if we have any modifications than we can do production

Upphase



# Software Requirement Specification (SRS)

→ A SRS is a document that capture complete description about how the system is expected to perform.

→ Behaviour of a system to be developed.

→ 3 types of requirements.

1) Functional requirements
2) Non-functional requirements
3) Domain requirements.

1) Functional requirements:- → requirements which are related to function/work aspects of s/w fall into this category.

→ It describes functionality of system services.

Ex's Calculations, Data processing.

2) Non-Functional requirements:-

→ This define s/y properties & constraints.

Eg:- Storage requirements, reliability, response time.

→ Requirements are expected characteristics of a given s/w.

3) Domain requirements:- These are the requirements which are clarified of a particular category (or) domain projects.

Ex:- school (or) college s/w maintain.

---

Functional requirements:-

FR characterized into 4 statements.

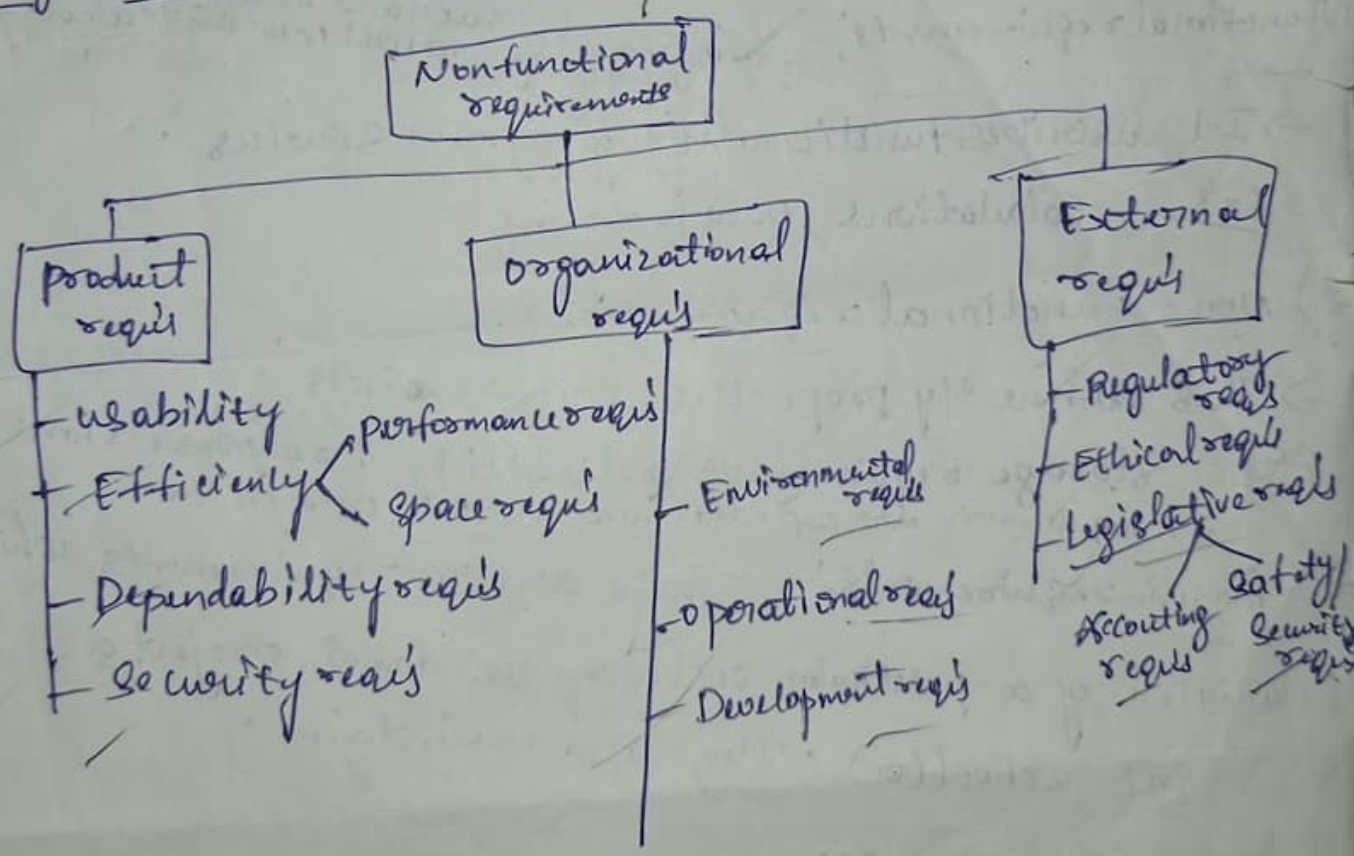
① It describes that How s/y should react to external environment.
② How s/y should respond when certain i/p's are given.
③ What o/p to be generated when certain i/p's are given
④ How every function of s/y should behave — (Behaviour of function).

Funct -

M.S → OS
function — oper — nonfunct

USO easy & simple to operate, Quick response, efficiency handling, operational costs, Customer supports.

Types of Non-functional requirements:



## * User Requirements

User requirements are defined using natural language, tables and diagrams as these can be understood by all users.

**Problems with natural language**

⇒ Lack of Clarity

→ precision is difficult without making the document difficult to read.

⇒ Requirements Confusion

→ functional & non-functional requ's tend to be mixed-up.

⇒ Requirements amalgamation (unite)

→ Several different requirements may be expressed together.

1) Easy & simple to operate
2) Quick Response
3) Effectively handling operational errors
4) Customer support

⇒ Guidelines for writing requirements

→ Invent a standard format & use it for all requirements

→ Use language in a consistent way.

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

→ Avoid the use of computer jargon (special words)

# System Requirements

→ More detailed specifications of system functions, service & constraints than user requirements.

→ They are intended to be a basis for designing the system.

→ They may be incorporated into the system contract.

→ System requirements may be defined (or) illustrated using s/y models.

## Problem with NL (Natural Language) specification

① **Ambiguity**

→ The readers & writers of the requ's must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult.

② **Over-flexibility**

→ The same thing may be said in a number of different ways in the specification.

③ **Lack of Modularization**

→ NL structure are inadequate to structure s/ys requirements.

④ Tabular specification

⑤ Graphical models

⑥ Sequence of diagrams.

# Software Requirements

→ The software requirements are description of features & functionalities of the target system.

→ Requirements convey the expectations of users from the S/w product.

→ The requirements can be obvious or hidden, known (or) unknown, expected (or) unexpected from Client's point of view.

## S/w requirements are necessary

→ To introduce the concepts of user and system requirements.

→ To describe functional & non functional requirements.

→ To explain how S/w requirements may be organized.

## What is requirement?

→ A requirement is a capability (or) condition required from the system.

(or)

→ The description of the services provided by the system & its operational constraints.

## Requirements Engineering

→ The process of finding out, analysing documenting & checking these services & constraint is called requirement engineering.

→ The process of establishing the services that the customer require from a system and the constraints under which it operates and is developed.

→ The requirements themselves are the description of the S/w services and constraints that are generated during the requirements engineering process.



Types of requirement:

① User-requirements :- Definition

The S/w must provide the means of representing & accessing external files created by other tools.

② System-requirements :- Definition

→ A structured document setting out detailed description of the system's functions, services & operational constraints

→ Defines what should be implemented so may be part of a contract b/w client & contractor.

* Functional requirements

Statements of services the system should provide how the system should react to particular inputs & how the system should behave in particular situations.

# Requirements

→ A requirement is a capability (or) condition required from the system.

→ What is involved in requirements analysis & specification?

- Determine what is expected by the client from the system (Gather & Analyze).

→ Document those in a form that is clear to the client as well as to the development team members (Document).

## Activities in Requirements Analysis & Specification

Requirements Gathering — Fully understand the user requis

↕

Requirements Analysis — Remove inconsistancies, anomalies, etc from requis.

↕

Requirements Specification — Document requis properly in an SRS Doc.

↕

SRS Document

→ Good SRS reduces development cost.

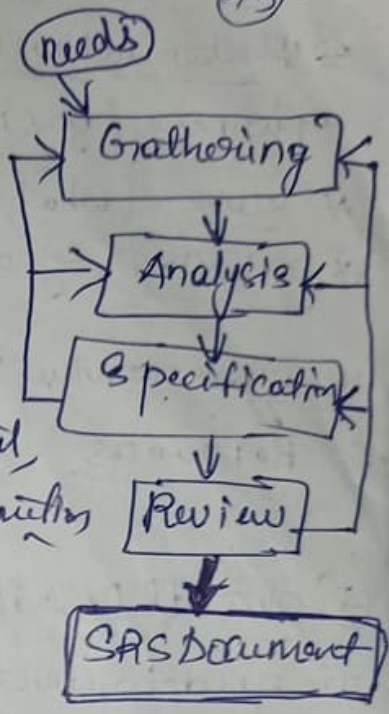
- Req. errors are expensive to fix later
- Req. changes cost a lot
- Good SRS can minimize changes & errors.

An Example

Cost of fixing errors in req, design, coding, acceptance testing & operation increases exponentially.

# Software Requirements Specification (SRS)






⇒ Main aim - systematically organize the requirements arrived during requ analysis.

→ Document requirements properly.

## SRS Document

Useful in various contexts:

- Statement of user needs, - Contract document,
- Reference document, - Definition for implementation

→ SRS Document is known as black-box specification.

- The sy is considered as a black box whose internal details are not known

* Software requirements Document (or) SRS Document.



(1) Introduction (we need to write intro)

* purpose (you need to cover purpose of the project)
* scope - (what is scope of the project)
* Definition (synonyms, acronyms (or) new words (or) new names in the project.
* System overview } sys overview will be cover in the introduction
* References. } part & References will be covered

(2) overall Description

we need to cover

* product perspective
* product functions
* User characteristics
* Constraints, Assumptions
* Dependencies

(3) Specific Requirements

* External Interfaces
* functions
* performance requirements
* logical database requirements
* Design constraints.
* Key features.

(4) Validation checks

* Validity check
* Consistency check
* Realism check.
* Verifiability check.
* completeness check.

(5) Validation Techniques.

* Requirement reviews.
* prototyping techniques.
* Test case generation.

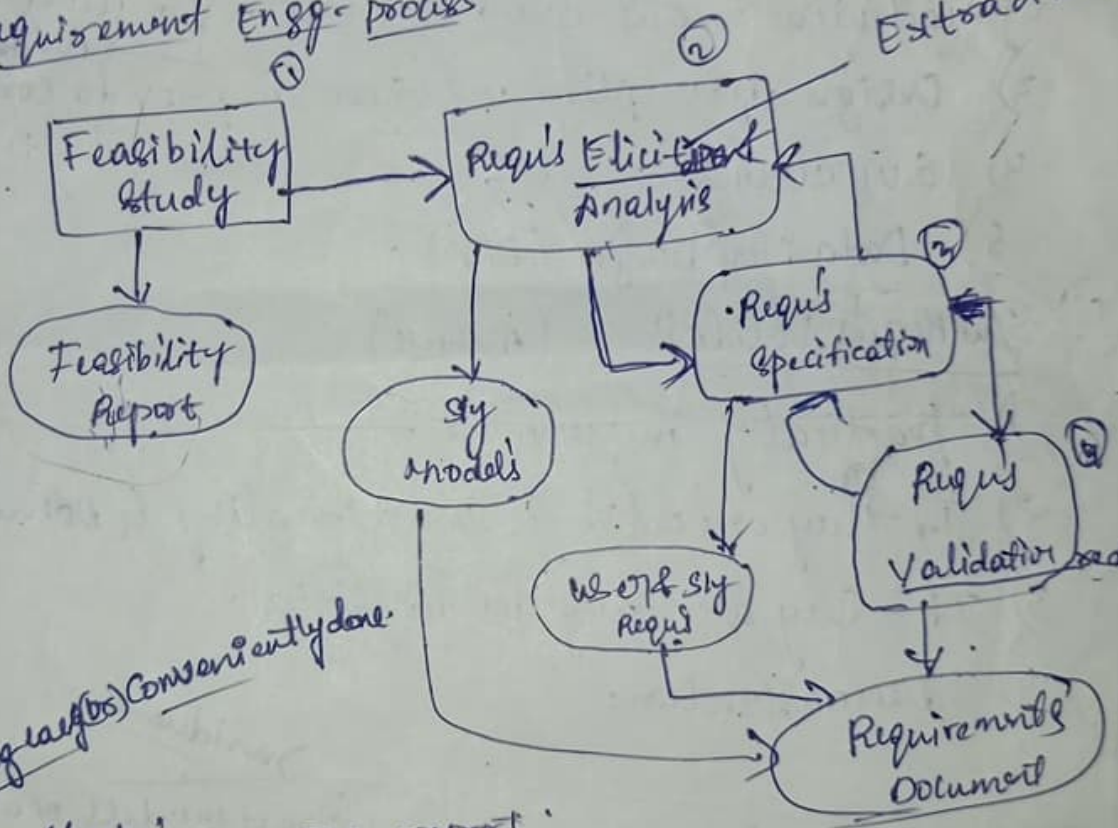
# Requirements Engineering process

Def → The process of gathering, analyzing, documenting, (SRS) Validating, managing requirements.

4 Steps
1) Feasibility Study
2) Requirement gathering
3) S/w requirement Specification (SRS)
4) S/w requirement Validation.

participants
1) S/w Engineers
2) managers
3) customers
4) Users

Figure:- Requirement Engg. process




the manner

Being lazy(bs) Conveniently done.

① Feasibility Study:- — F.S report.

→ Client approaches the organization.

→ A rough idea about what all functions that S/w must perform & which all features are expected from the S/w.

→ After S/w analyst analyze ~~the~~ we get feasibility study report with minimum requis.

Types:- ① Technical feasibility:- Evaluate current technologies.
② Operational Feasibility:- To solve business problem & customer requir—
③ Economic:- decides the generate financial Profits for organiz

→ The output of this phase should be feasibility study report.

(2) Requirements Elicitation & Analysis → Extraction.

(48)

→ Analyst & engineer gathering requis from the customer.

Eliciation → Extraction (what were requis we are gathering).

That requis extracting, extending & developing & analyze) with sly model.

(3) SRS:- System analyst created a SRS document.

→ SRS should

1) User requis are in natural language (Specifications) → tables & diagrams.
2) Technical requis are in structured lang → Reusable code & reusable components
3) Design description written in pseudo code → text based algorithmic
4) GUI screens.
5) Data Flow Design (DFD).

(4) S/w Requis Validation:- (Re-check)

1) practically implemented.
2) If they are valid & functionality & Domain of S/w.
3) If they are any ambiguities.
4) Demonstration.

→ analysis

Sly model is the process of developing abstract models of a s/y, with each model presenting a different view (or) perspective of that s/y.

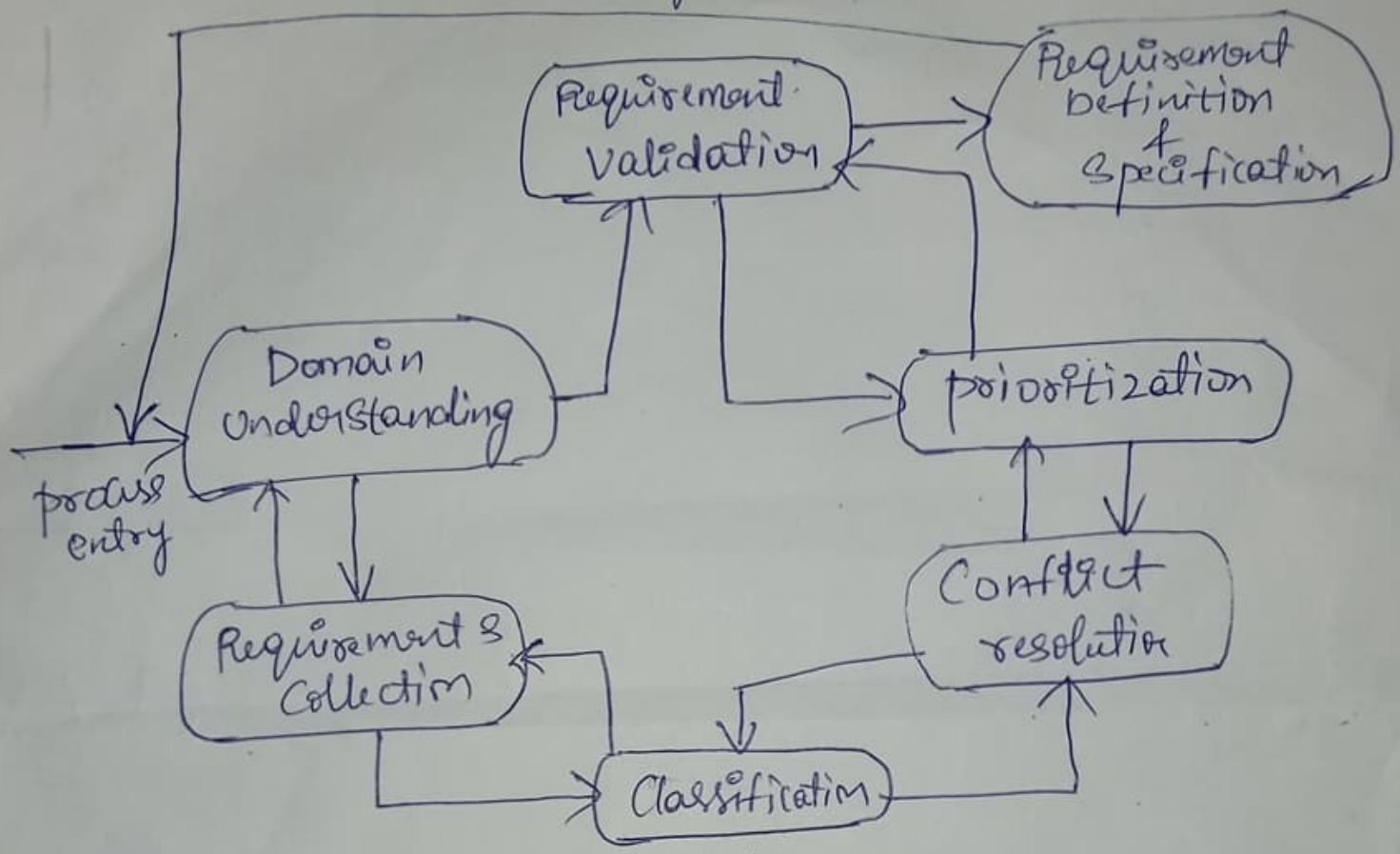
Types:- context, Behavioral, Data model, Object model, Structural model

Requis Validation Techniques.

1. Requis Reviews/Inspections
2. prototyping
3. Test-case generation
4. Automated consistency analysis.

# * Elicitation and Analysis process

↳ Also known as the gathering of requirements.



# U-II Design Engineering

Part-1

→ Design Engineering encompasses (covered / have or hold within) the set of principles, concepts and practices that leads to the development of a high quality s/w (or) product.

→ Design principles establish an overriding (more important than other consideration) ~~philosoph~~ philosophy that guides the designer in the work that is performed.

→ Design concepts must be understood before the mechanics of design practice are applied.

→ It leads to the creations of various representations of s/w.

## Design :-

→ Design is what virtually (almost) every engineer wants to do.

→ It is the place where creativity rules - customer's requirements, business needs, and technical considerations all come together in the formulation of a product (or) a s/y.

## Important :-

① Design allows a s/w engineer to model the s/y (or) product that is to be built.

② Design is the place where s/w quality is established.

## * Design process :-

→ Software design is an iterative process through which requirements are translated into a "blue print" for constructing the s/w.
↳ Design plan (or) technical drawing

→ The design is represented at a high level of abstraction (quality of dealing with ideas)

→ The design process is an step by step and repetitive procedure where s/w is simulated according the requirements.

→ The simulation (imition of some material) will be in detailed (high level of abstraction).

→ Design process is required for better quality software.

→ This quality is achieved by a series of Formal Technical reviews (FTR).

Characteristics (or) Goals

① Design must consider and fullfill all the customer's requirement those are implicit and must implement explicit requirements contained in the analysis model. (It must implement all the explicit requirements contained in the analysis model).

② Design must be readable, understandable and as a guide for those who generate code and for the those whose test & subsequent support of the s/w.

③ Design should provide a complete picture of the s/w, addressing data, functional & behavioral domains.

Design Quality (s/w quality is defined as how well the s/w works)

→ These are the few guidelines which are the characteristics of a good design. → characteristics of items used by designers

Quality guidelines

① A design should exhibit an architecture that,
a) has been created using recognizable architectural styles (or) patterns.
b) is composed of components that exhibit good design characteristics.
c) can be implemented in an evolutionary fashion, thereby facilitating implementation and testing.

② A design should be modular (partitioning elements).

③ A design should contain distinct representation of data, architecture, interfaces and components. (To understand all)

④ design should explore the data structures useful for the classes to be implemented. (In design all you will decide which DS to be used).

⑤ design should contist of components that have independent functional characteristics.

⑥ A design should lead to interface that reduce the complexity. (manage to make it easier).

⑦ design should be achive from iterative method (Checking continuously)

The above are the mandatory guide lines to be followed while designing.

## Quality Attributes (FURPS)

→ The FURPS quality attributes represent a target for all s/w design:

**Functionality:** functionality is assessed by evaluating the feature set and capability of the program, generality of functions and security overall system.

**Usability:** is assessed by considering human factors, overall aesthetics, consistency and documentation.

**Reliability:** is evaluated by measuring the frequency and severity of failure, the accuracy of output results, and their mean-time-to-failure (MTTF), the ability to recover from failure, and the predictability of the program.

**Performance:** is measured by processing speed, response time, resource consumption, throughput, and efficiency.

**Supportability:** combines the ability to extend the program (extensibility), adaptability, serviceability - these three attributes represent a more common term maintainability.

**Design Concepts:** — Following is a set of fundamental s/w design concepts

① Abstraction
- highest level of abstraction.
- lowest level of abstraction.
- procedural abstraction
- data abstraction.

② Architecture
- Structured models.
- Framework models
- Dynamic models
- Process models
- Functional models.

③ Patterns

④ Modularity

⑤ Information Hiding.

(6) Functional Independence
- Cohesion
- Coupling.

(7) Refinement

(8) Refactoring

(9) Design classes
- User interface classes
- Business domain "
- process classes
- persistent classes.
- s/y classes.

1) Abstraction:- (hiding unwanted data/code) By hiding the model will be easily understood.

→ Abstraction simply means to hide the details to reduce complexity and increase efficiency (or) quality.

→ Different levels of abstraction are necessary and must be applied at each stage of design process so errors that are present can removed efficiently.

* At the highest level of abstraction:- a solution is stated in broad terms using the language of the problem environment.
(broad terms means we are just giving a high level of details but not in details about that).

* At lower level of abstraction: A more detailed description of the solution is provided.

* procedural abstraction: Refers to sequence of instructions that have a specific and limited functions. (Step-by-step) Ex:- open for a door.

* Data Abstraction: It is a named collection of data that describes a data object
Ex:- door type, swing direction, opening mechanism, wieght, dimensions.
encompasses Attributes

2) Architecture :- (design ~~architecture~~ is a structure.) (most popular architecture is data driven architecture (or) data centred architecture / data interacting all client and s

It is the structure (or) organization of program & program components (modules), their interaction and structure of data that are used by components → used to represent major sy elements & their interactions.

Architectural modules

* Structured modules - organized collection of program components
* Framework modules - increase the level of design abstraction
* Dynamic modules - Represents the behavioral aspects indicating changes as a function of external events.
* Process modules - Focus on the design of the business (or) technical process.
* Functional modules - can be used to represent the functional hierarchy of a s/y.

3) pattern :- (a repeated form)

→ The pattern simply means a repeated form of design in which the shape is repeated several times to form a pattern

→ The pattern in the design process means the repeation of a solution to a common recurring problem with a certain context

The input of each pattern is to provide a description that enables a designer to determine.

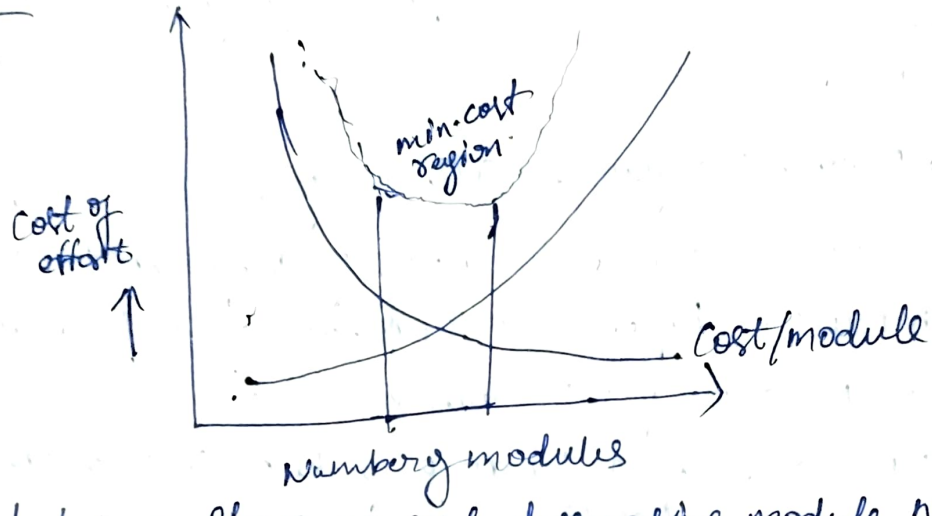
* whether the pattern is capable to the current work.
* whether the pattern can be reused
* whether the pattern can serve as a guide for developing a similar but functionally (or) structurally different pattern.

4) Modularity :- (Sub divided the system)

Modularity simply means to divide the s/w (or) project into smaller parts to reduce the ~~components~~ complexity of the s/w (or) product.

→ In the same way, modularity in design means to subdivided a s/w into smaller parts so that test parts can be created independently & their designing & construction will easily.

Figure:-




→ The left curve shows as cost decreasing module numbers are increasing.

→ The right curve shows as modules increasing the cost is also increasing.

Hence always select medium module based on project cost and complexity.

5) Information Hiding :- (hide the information)

Information hiding simply means to hide the information, the information so that it cannot be accessed by unwanted party (users (or) Hackers).

6) Functional Independent :- (independent methods)

As a project has multiple functions/methods they all should be independent to each other.

Cohesion :- It is an indication of the relative functional strength of a module. → It is the concept of intra module.
→ Represents the relationship within the module.

Coupling :- It is an indication of the relative interdependence among modules.
→ It is a concept of intermodule.
→ It represents the relationship b/w the modules.

7) Refinement :- (remove impurities)

-> Refinement simply means to find the impurities of present & increase the quality.

-> Refinement is very necessary to find out any bugs/errors of present optimisation is also done here.

-> Refinement is actually a process of elaboration.

8) Refactoring :- (reconstruct something)

-> Refactoring simply means to reconstruct something in such a way that it doesnot affect the behavior of any other features.

-> Refactoring in s/w design means to reconstruct the design to reduce complexity and to make it simple without affecting the behavior (or) its functions.

9) Design classes : The model of s/w is defined as a set of design classes.

Different types of design classes are :

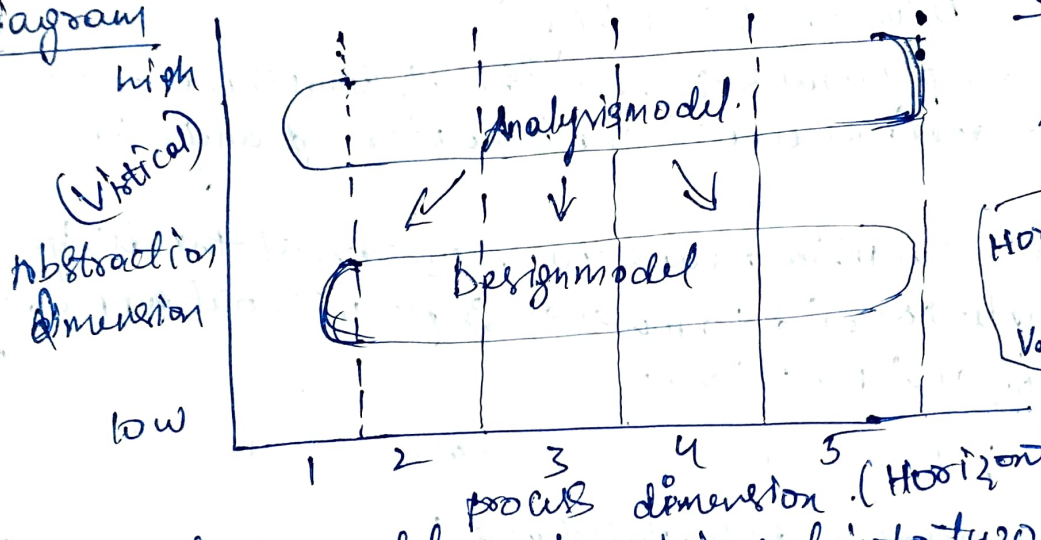
* User interface classes : define all abstractions that are necessary for human computer interaction.
* Business Domain :
* process classes :
* persistent classes :
* s/y classes :

**Design model:-** Designing a model is an important phase and is a multi process that represents the data structure, program structure, interface characteristic, and procedural details.

→ It is mainly classified into 5 categories (or) elements.

① Data / class elements ② Architectural elements ③ Interface elements ④ Component level elements ⑤ Deployment-level elements.

Diagram




→ Horizontal (process dimension)

↑ Vertical (abstraction dimension)

Horizontal - layer by layer.

Vertical - step by step

→ The design model can be viewed into two different dimensions.

1) Horizontally - The process dimension indicates the evolution of the parts of the design model each design task executed.

2) Vertically - The abstraction dimension represent the level of details as each element of the analysis model is transferred into design model & then iteratively refined.

→ The elements of the design model use many of the same UML diagram used in the analysis model.

→ They are only refined & elaborated.

→ More implementation specific is provided.

We will elaborating them of the diagram is in 2 steps, we will making into 5/6 steps.

## 1) Data design elements :-

→ It is also referred as "data architecturing".

→ Data design creates a model of data and or information that is represented at a high level of abstraction (customer/user view of data)

→ As in any project data plays a key role, hence designing it properly helps a lot in the further (coding, testing)

→ The structure of data is the most important part of the s/w design.

## 2) Architectural Design elements :-

→ Architectural design is nothing but a photocopy of the end software.

→ The ADE provides us overall view of the system.

To achive the following

a) we need to have knowledge on the application domain.

b) Relationship b/w UML diagram 4

c) we need to have proper architectural styles & patterns

## 3) Interface Design Elements :-

→ These tells us how information flow into and out of the system

→ ~~It includes~~ They communicate b/w the components defined as part of architecture.

→ Following are the important elements of the interface Design

Include a) The user interface.

b) External interface.

c) Internal interface.

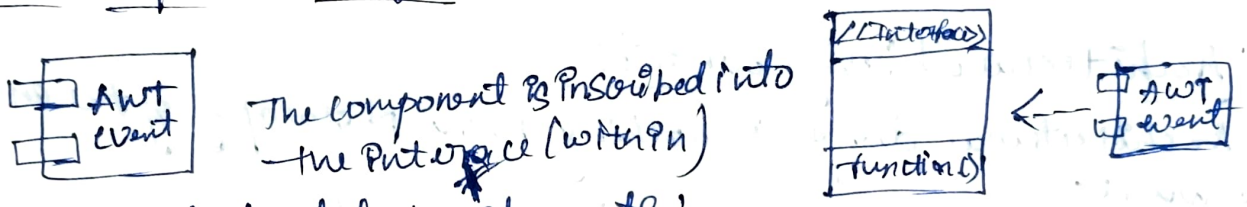
## (4) Component level design elements :-

→ The component level design for the software completely describes the internal details of each software component.

→ It provides all details of a given ~~statement~~ S/w component.

→ In order to achive this the component design describes the representation of data structure.

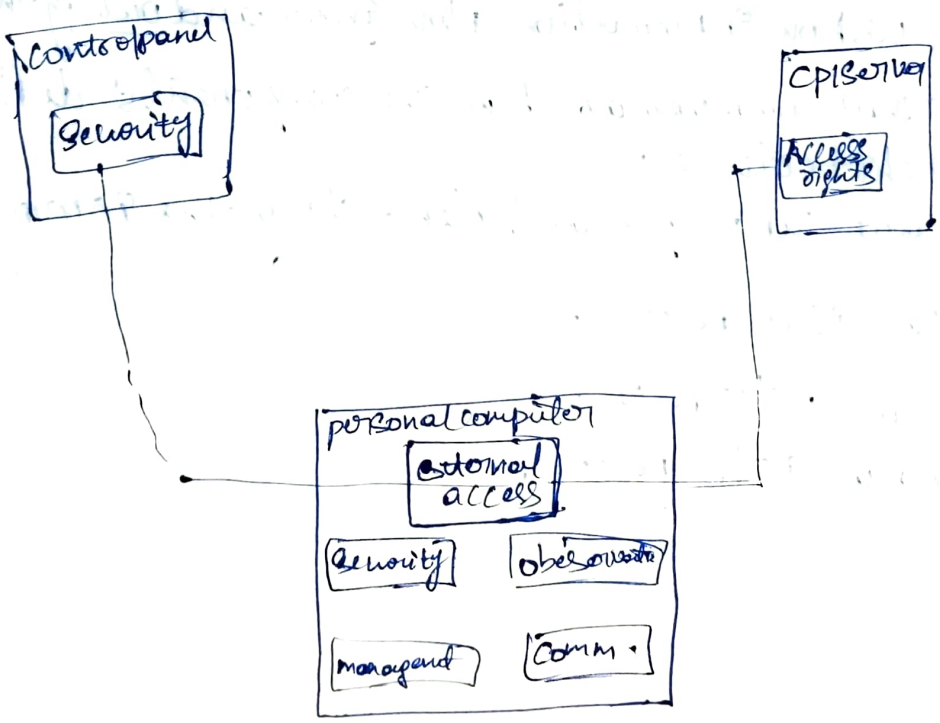
UML component diagram :-




## (5) Deployment level design elements :-

→ These indicate how software functionality & subsystem will be allocated within the physical computing environment that will support the S/w.

fig: Deployment level diagram.



* Creating an architectural design
  - Software Architecture I
  - Data design II
  - Architectural styles and patterns III
  - Architectural design IV

(components are nothing but functions (or) module used in the program)

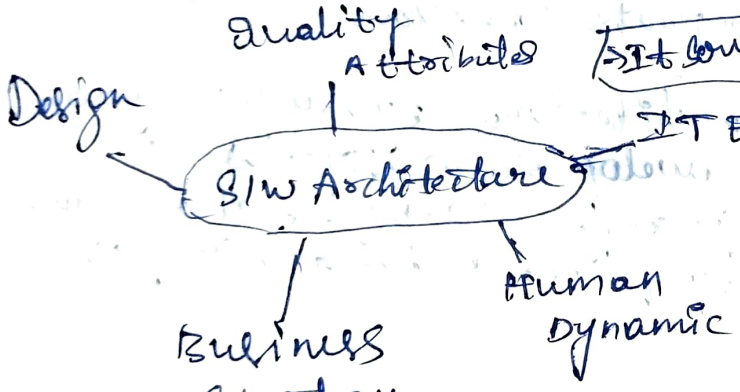
## I. Software Architecture → (Designing structure of something).

→ S/w architecture of a program (or) computing system is the structures of the system. If we want build (or) develop any architecture we need some

→ which comprise → S/w Components
→ details (about data structures & algorithms which are hidden in the S/w)
→ relationship among components
→ data flow, control flow, dependencies.

5000 LoC (code diagram)
500 to 5 just write

→ It serves as a blueprint for a S/y.
→ It provides an abstraction to manage the S/y complexity & establish a comm. and coordination mechanism among components.



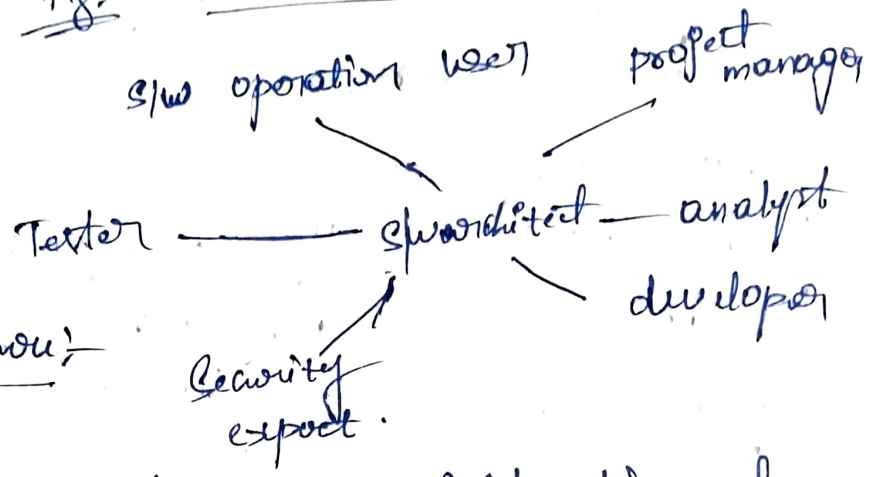

Use →
→ It reduce the risk associated with the construction of S/w

Imp:-
→ It provides a representation facilitates communication among all stack holders.
→ It highlights early design decisions.
→ It is relatively small, how the system is structured and how its components work together.

Adv

1) Stakeholder Comm.
2) System analyst
3) Large-scale reuse.

Fig: S/w architecture.




Elements of S/w Architecture:-

① Components:- It is a abstract unit of S/w instruction and internal state which provides transformation of data.

② Connectors:- The connectors are abstract mechanism that communicate, cooperate and coordinate among components.

③ Configuration topologies:- It is nothing but structure of architectural relationships among components, connectors.

④ System models:- Different architectural styles are going to be used to develop S/w architecture.

Architecture:- Architectural design represents the structure of data and program components that required to build a computer based s/y.

II. Data Design:- Data Design actions translate data objects into data structure, further into a database architecture.

(It is a region of storage that contains a value (or) group of values)

→ It contains two levels.

① Data Design at the Architectural level:-

→ In todays S/w business environment, where a lot of data and database are used to the challenge is to extract useful information.

→ To solve this challenge, business, IT community has developed data mining techniques, also called knowledge Discovery in Database (KDD), to extract (or) import ~~info~~ from the database.

→ An alternate solution called as Data warehouse is a large, independent db that has access to data that are stored in db.

② Data Design at the Component level:- Data design at the component level, focuses on representation of data structures that are directly accessed by one (or) more s/w components.

Principles:-

→ A s/w design and programming language should support the specification and realization of abstract data types.

→ A library of useful data structure and operations that may be applied to them should be developed.

→ The representation of a data structure should be known only to those modules that must make direct use of data contained with in the structure.

→ All data structure & the operations to be performed on each should be identified.

→ The systematic analysis principles applied to function & behaviours should be applied to data.

III: Architectural Styles

→ The builder has used an architectural style as a descriptive mechanism to differentiate the house from other styles.

→ Each style describes a system category than encompasses

1) A set of components (eg: a database, computational modules) that perform a function required by a system.

2) A set of connectors that enables "communication, coordination & cooperation" among components.

3) Constraints that define how components can be integrated to form the system and

4) Semantic models that enable a designer to understand the overall properties of a system by analying the known properties of its constituent parts.

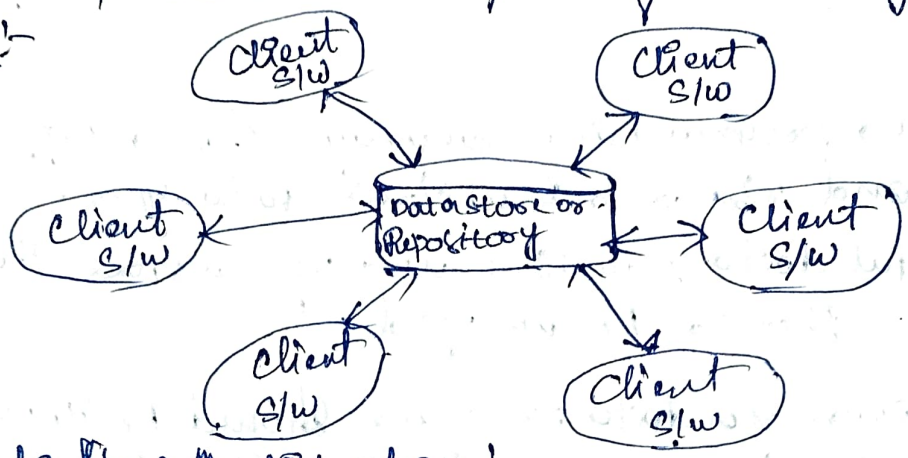
1) Data-Centered Architecture:- A data store (files (or) db) resides at the center of its architecture and is accessed frequently by other components that update, add, delete (or) modify data within the store.

→ Data-centered architecture helps integrity.

→ Pass data b/w clients using the blackboard mechanism.

→ The processes are independently executed by the client components.

Figure:-

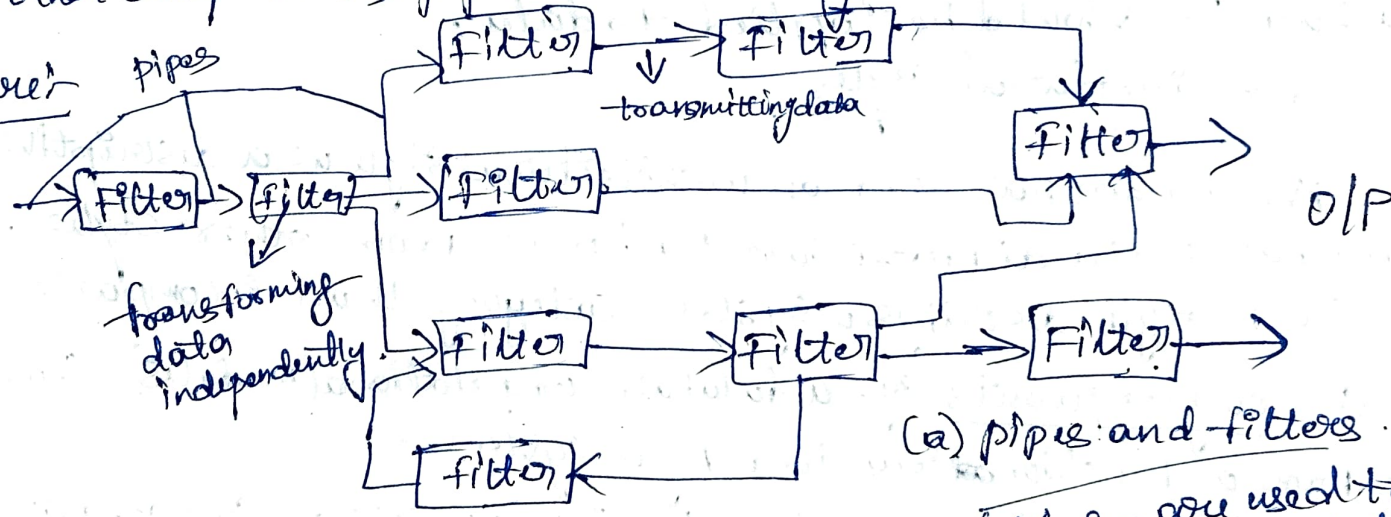



2) Data Flow Architecture:-

→ This architecture is applied when i/p data are to be transformed through a series of computational (or) manipulative components into o/p data.

Eg:- A pipe and filter pattern, filters transforming data independently of other components, pipes transmitting data.

Figure:-




(a) pipes and filters.

(b) batch sequential.

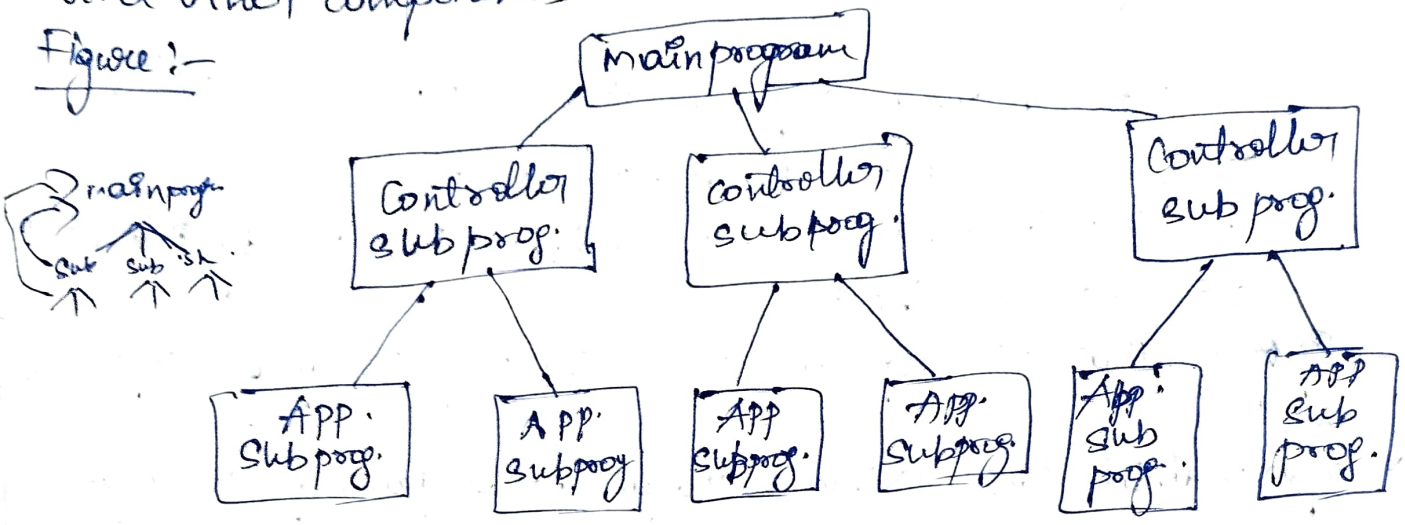
Pipes - are used to transmitting data.

Filters - are used to transforming data independently

## 3) Call and Return Architecture

Main program / subprogram Architecture :- Decompose function into a control hierarchy where a "main" program invokes a number of program components, which in turn may invoke still other components.

Figure :-




## 4) Object-oriented Architecture :-

The components of a s/y encapsulate data and the operations that must be applied to manipulate the data. Communication and coordination b/w components are accomplished via message passing.

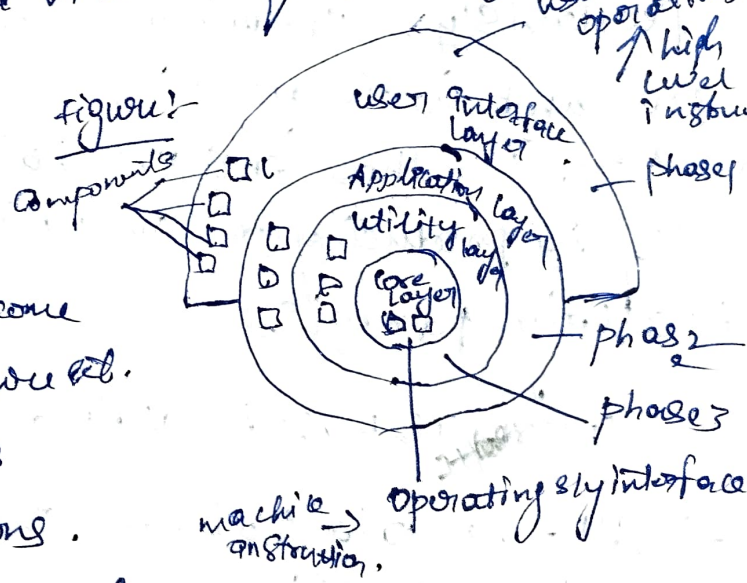
## 5) Layered Architecture :-

→ A number of different layers are defined, each accomplishing operations that progressively become closer to the machine instruction set.

→ At the outer layer, components service user interface operations.

→ At the inner layer, components performs operating s/y interfacing.

figure :-



# IV Architectural Design

→ S/w Architectural Design represents the structure of data and program components that are required to build a Computer based system.

→ The architectural design is not a operational (ready for use) s/w, but it is representation.

→ Architectural design, representing the s/w in Context:

- Sys engineer must model the context.
- Context diagram - modules in which the s/y interacts with external entities

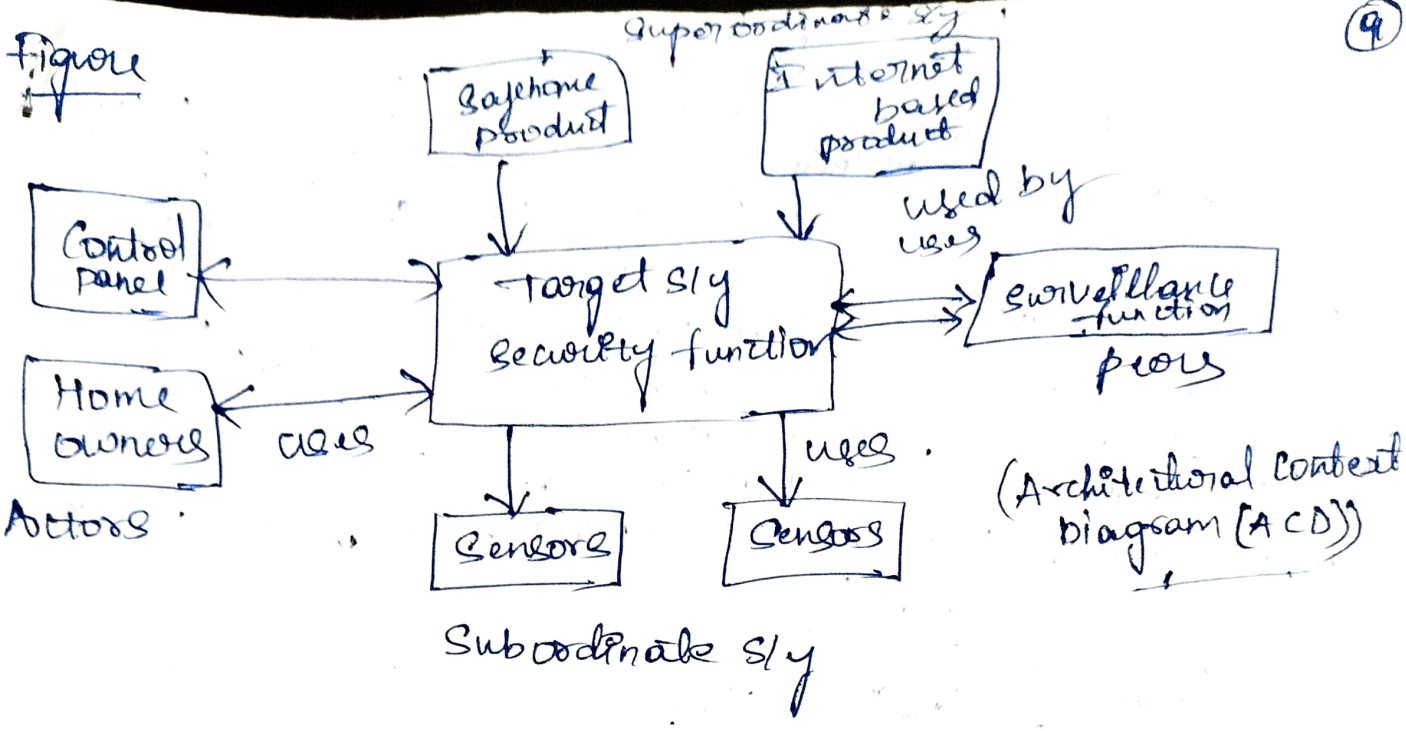
→ System that interoperate with the target s/y are represented as:-

* Superordinate s/y:- using the target s/y as part of some higher level processing scheme.
* Subordinate s/y:- used by the target s/y to provide data (or) processing needed to compute the target s/y.
* Peer level s/y:- producing or consuming information needed by peer and the target s/y.
* Actors:- those entities that interact with the target s/y by producing (or) consuming information that is necessary for requisite.

→ Interfaces must be defined (relationship)

→ All the data that flow into (or) out of the target s/y must be identified.

Figure

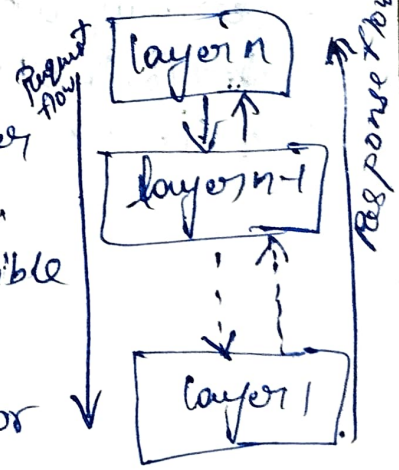



* Architectural patterns:-

→ An architectural pattern is a general, reusable solution to a commonly occuring problem in s/w architecture within a given context.

① Layered pattern:- This pattern can be used to structure programs that can be decomposed into groups of subtasks, each of which is at a particular level of abstraction. Each layer provides services to the next higher layer.

⇒ 4 layers




1) presentation layer:- (User interface layer) the UI layer where user see and enter data into an application.

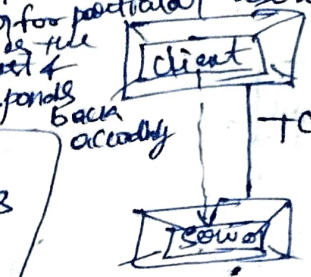
2) Business layer:- (Domain layer) This layer is responsible for executing business logic as per the request.

3) Application layer:- (Service layer) This layer acts as a medium for comm. b/w the 'presentation layer' and 'data layer'.

4) Data layer:- (persistance layer) This layer has a db for managing data.

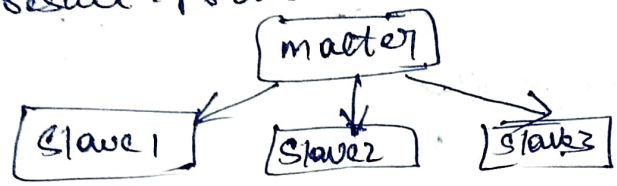
⇒ Open System Interconnection (OSI) model layers.

② Client-Server pattern :- It has two major entities. They are a server and multiple clients. → How the server has resources (data, files) and a client requests the server for particular resource. that the server provides the request & responds back accordingly

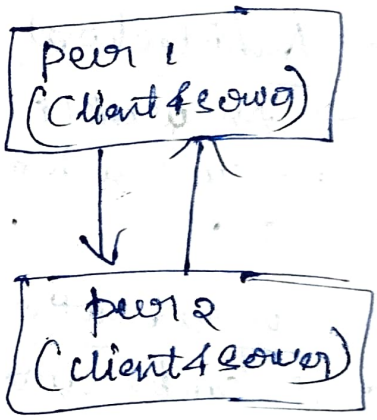



Eg:- Email, www, file sharing apps, Banking etc.

③ Master-Slave pattern :- This pattern consists of two parts is master & slaves. The master component distributes the work among identical slave components, and ~~computes~~ computes a final result from the results which the slaves return.




④ ~~Pipe-fi~~ Peer-to-peer pattern :-




→ In this pattern, individual components are known as peers.

→ Peers may funtion both as a client, requesting services from other peers, and as a server, providing services to other peers.

→ A peer may act as a client or as a server (or) as both, and it can change its role dynamically with time.

# * Modeling Component-level design  Unit-2  part-B

## Component level design :-

→ Component level design occurs after the first iteration of the architectural design.

→ A component level design can be represented using some intermediate representations (eg:- graphical, tabular (or) text based) that can be translated into source code.

→ In design basically, we follow 3 steps (levels) analysis, architecture design, & component design. So, component level design is the last level. (Stage)

→ Once the component level design stage done then we will go automatically coding stage.

→ The design of data structure interfaces, and algorithm should conform to well established guidelines to helps us avoid the introduction of errors.

→ If the component level design done 100% correctly the basic errors like structural & algorithm errors will be resolved.

## Component :-

→ A component is a modular building block for computer s/w (or) → It is a small module of a huge program (or) project.

→ "A component is a modular, deployable and replaceable part of a system that encapsulate implementation and exposes a set of interfaces".

→ It is an interface b/w two or more components & it is a component it is connected to multiple components also it does a particular functionality.

→ Based on different point of view the meaning of component changes

* Object oriented view
* Conventional view
* Process related view.

→ Component level design is the definition and design of components and modules after the architectural design phase.

→ Component level design defines the data structure, algorithms, interface characteristics, and communication mechanisms allocated to each component for the s/w development.

Views

1) Object-oriented View: Under the object oriented view, an individual component contains a set a collaborating classes, or a group of objects with common properties, operations and relationships to other components. Classes can be divided into two categories.

* Analysis Classes (problem domain) representative of people, locations, or other physical objects.
* Design Classes (infrastructure) classes with in the context of s/w that allow an application to be executed.

2) Conventional View :- A component is viewed as a functional element of the s/w, meaning that it is part of a program that incorporates the processing logic and required internal data structures to perform its designated task.

3) Process Related View :- It especially advocates for the advantage of component reusability, requiring that each process-related component contains:

* A complete description of the interface.
* A definition of the specific function that each component is meant to perform.
* The communication & collaboration with other components that each component requires.

* Characteristics of Components :-

1. Reusability
2. Replaceable
3. Not Context Specific
4. Extensible
5. Encapsulated.
6. Independent.

# * Designing Class based Components

→ When an object-oriented s/w engineering approach is chosen, Component level design focuses on the elaboration of problem domain specific classes and the definition and refinement of infrastructure classes contained in the requirement model.

## I) Basic design principles

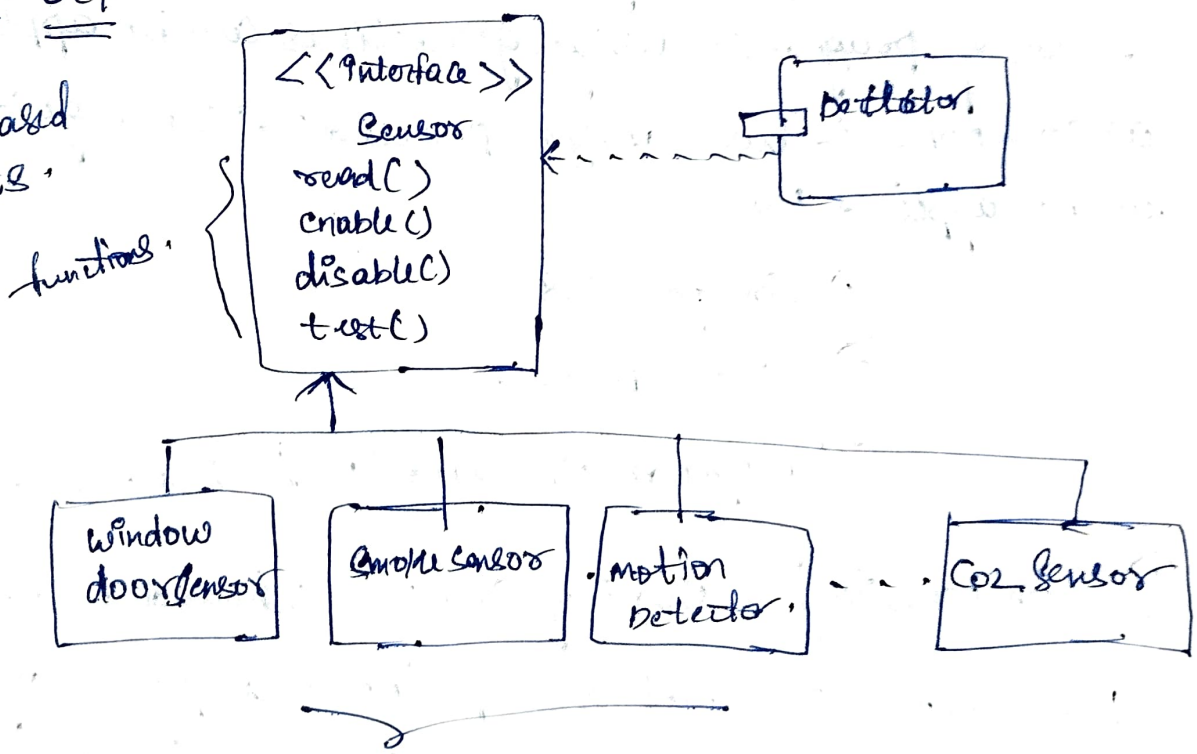
Four basic design principles are applicable to component-level design and have been widely adopted when object-oriented s/w Engineering applied.

### 1) The open closed principle (OCP):-

→ A module (or) Component should be open for extension but closed for modification.

→ The designer should specify component in a way that allows it to be extended without the need to make internal code or design modifications to the existing parts of the component.

Figure:- OCP



2) The LISKOV Substitution principle (LSP)

→ Replace of main class.

→ Subclasses should be substitutable for their base class.

→ ~~For suppose subclass is there~~ If there is subclass ~~from~~ main class the subclass should represent all the characteristic of some main class.

3) Dependency inversion principle (DIP)

→ Depend on abstractions. Donot depend on concretions.

→ The more a component depend on other concrete components (rather than on the interface) the more difficult it will be to extend.

4) The interface segregation (separation) principle (ISP)

→ many client specific interface are better than one general purpose interface.

→ for server class, specialized interfaces should be created to serve major categories of client.

II.) Component-Level Design Guidelines

A set of pragmatic design guidelines can be applied as component-level design proceeds. These guidelines can be applied to

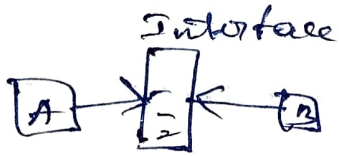
1. Components
2. Interfaces.
3. Dependencies and Inheritance.

1) Components :- Naming conventions should be established for components that are specified as part of the architectural model and then refine & elaborated as part of the component level ~~design~~ model.

→ Architectural Component names should be drawn from the problem domain and should have meaning to all stakeholders who view the architectural model.

(2) Interfaces :- Interface provides important information about communication and collaborations b/w two components

Figure :-




→ It acts as a broker b/w two components. It just has some functions & operations

(3) Dependencies & Inheritance :-

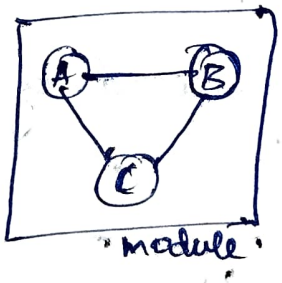
It is a good idea to model dependencies from left ~~idea~~ to right and inheritance from bottom to top.

III) Cohesion :- (maximum Cohesion) is important

→ Cohesion is a measure of the degree to which the elements (or) components of the module are functionally related.

→ A good s/w design will have high cohesion.

Figure

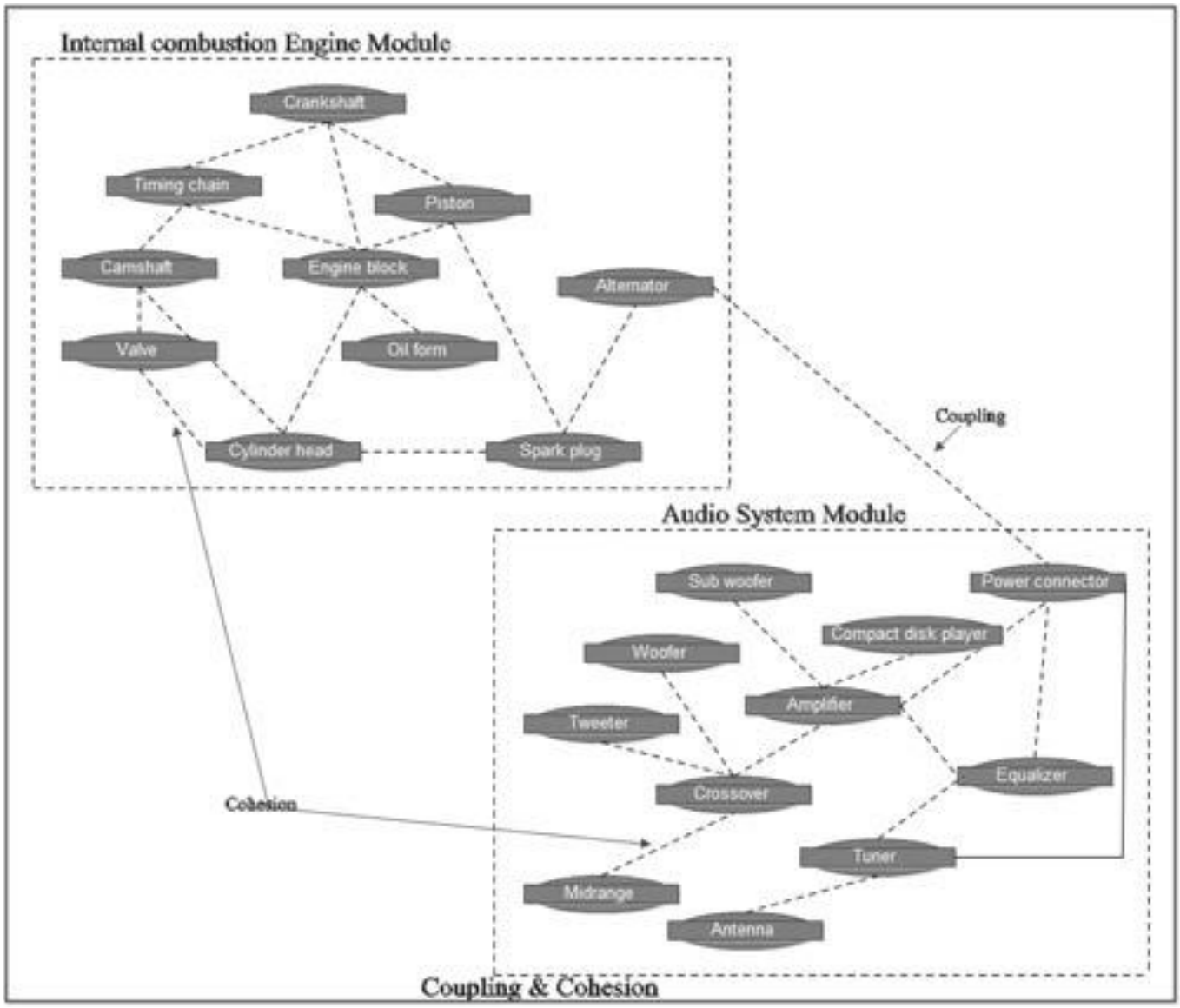


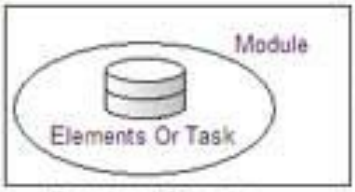

Types

1) Functional Cohesion
2) Sequential "
3) Communication "
4) procedural "
5) Temporal "
6) Logical "
7) Coincidental "

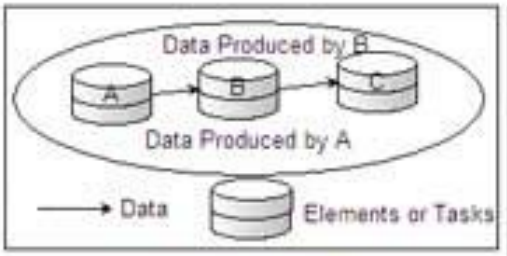
Best — high

low — worst

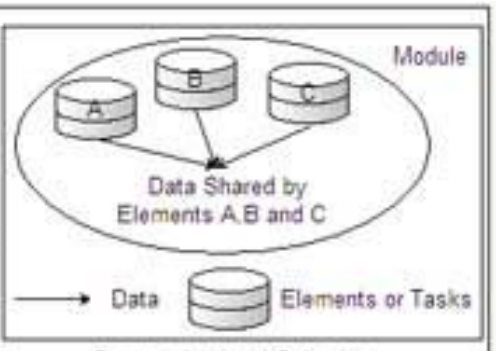

Internal combustion Engine Module
Crankshaft
Timing chain
Piston
Camshaft
Engine block
Alternator
Valve
Oil form
Cylinder head
Spark plug
Coupling
Audio System Module
Sub woofer
Power connector
Compact disk player
Woofer
Amplifier
Tweeter
Equalizer
Crossover
Cohesion
Tuner
Midrange
Antenna
Coupling & Cohesion



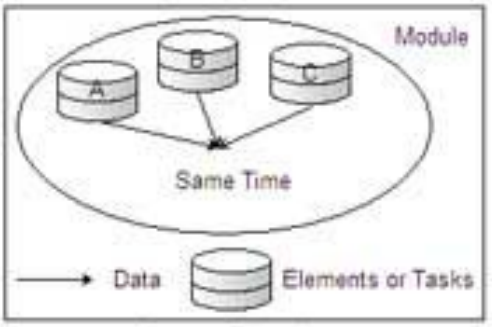

Functional Cohesion



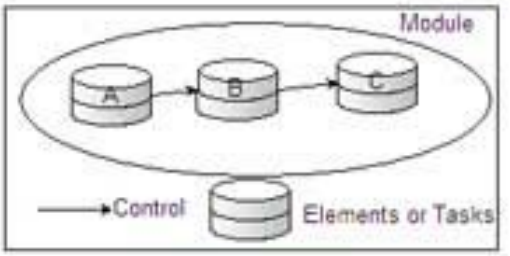

Sequential Cohesion



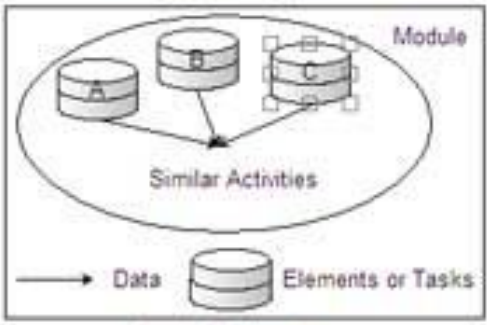

Communicational Cohesion



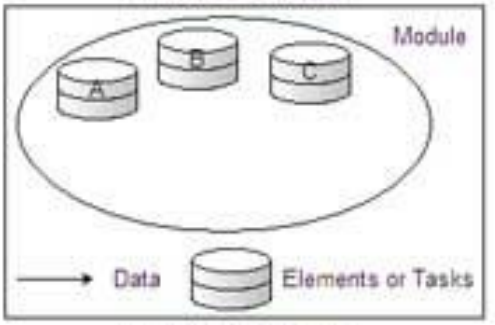

Temporal Cohesion




Procedural Cohesion




Logical Cohesion

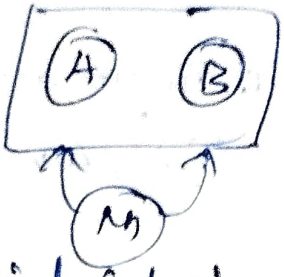



Coincidental Cohesion

Types of Cohesion

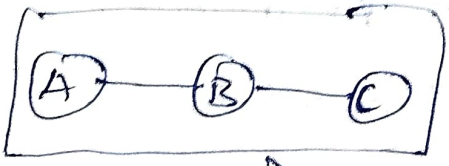
1) Functional Cohesion :- Two (or) more functions performing similar task should be placed in a single module. (If they are related)

Fig:



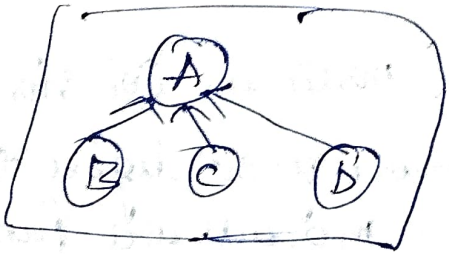

2) Sequential Cohesion :- An elements outputs some data that become the input for other elements data flow between the part.

Fig:-



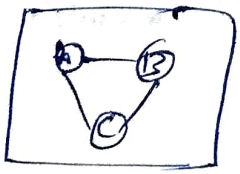

3) Communication :- Two (or) more elements operate on the same input data

Fig:-



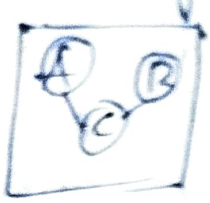

4) procedural :- Elements of procedural cohesion ensure the order of execution one after other

5) Temporal :- Elements are related by time involved (parallely (or) at a time).



6) Coin Cidental :- Elements are not related to any module hence they are can kept any where

7) Logical :- The elements are logically related but not functionally




## IV) Coupling (min coupling)

Coupling is the measure of the degree of independence b/w the modules. A good S/W will have low Coupling.

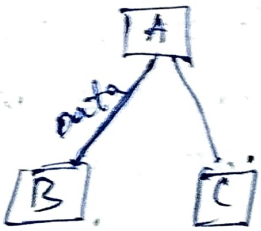
Types

1) Data Coupling
2) Stamp "
3) Control "
4) External "
5) Common "
6) Content "

Best low ~~high~~

1) Data Coupling :- When data of one module is passed to another module, this is called as Data Coupling.

Figure

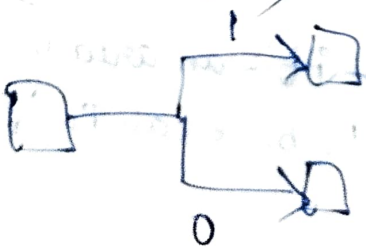



~~low~~ high worst.

2) Stamp Coupling :- In this the complete data structure is passed from one module to another module.

Figure

3) Control Coupling :- If the modules communicate by passing control information then they are said to be control coupling. (like flags)

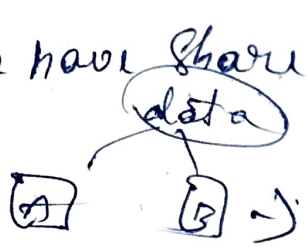



"1" is executed
"0" is not executed

single flag.

4) External Coupling:- In external coupling the module depends on other modules external to the s/w (like DB / server).

5) Common Coupling:- The module have share global data structure / global data.




→ dependent on one part of data

6) Content Coupling:- Here one module modifys data of another module.

* Conducting Component level design

→ Component level design is elaborative in nature.

→ The designer must transform information from the analysis and architectural model into a design representation that provides sufficient details to guide the construction (Coding & testing) activity.

The following steps represent a typical task set for component-level design, when it is applied for an object-oriented sy.

Step ① Identify all design classes that correspond to the problem domain.

→ Using the requirements and architectural model, each analysis class and architectural component is elaborated.

→ All design (or) classes that correspond to the problem domain are identified.

Analysis classes

Problem domain - it is an area of expertise (or) application that needs to be examined to solve a problem.

Step ② Identify all design classes that correspond to the infrastructure domain. (Identify infrastructure design class)

→ All design class that correspond to infrastructure domain are identified.

→ These classes are described in the requirements model and are often missing from the architectural model, but they must be described at this point.

→ Classes and components in this category include..

→ ① GUI components (often available as reusable components),

→ ② operating s/y components, → ③ Object & data management components.

Step ③ Elaborate all design classes that are not acquired as reusable components. (Elaborate design classes)

→ Specify message detail, when class or components collaborate

→ Identify appropriate interface for each component.

→ Elaborate attributes & define data structures required to implement them.

→ Describe processing flow within each operational in details.

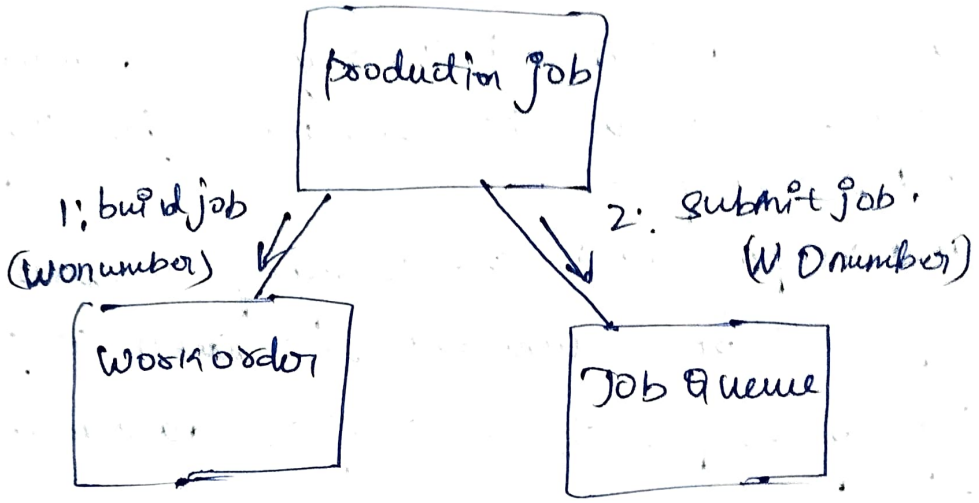
Step ③a Specify message details when classes (or) components collaborate. (collaboration details)

→ The requirements model makes use of a collaboration diagram to show how analysis classes collaborate with one another.

→ message can be elaborated by expanding this syntax in the following manner.

(guard condition) sequence expression (return value) := message name (argument list).

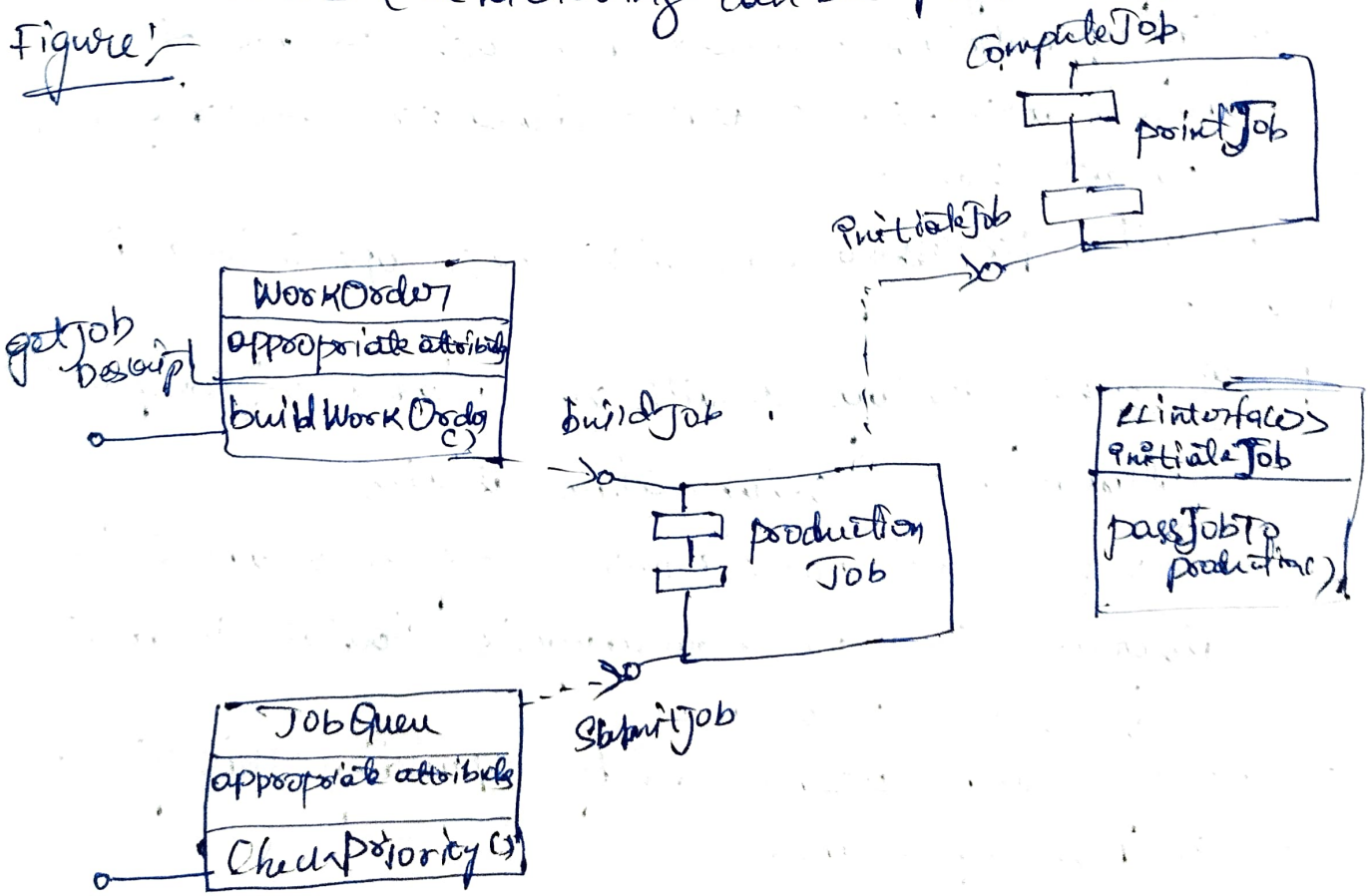
Figure:- Collaboration diagram with messaging.




Step 3b Identify appropriate interfaces for each component

Print job interface "Initiate job" (appropriate interface), which doesn't exhibit sufficient cohesion because it performs three different sub functions (refactoring can be performed).

Figure:-



Step (3c): Elaborate attributes and define data types and data structures required to implement them (Elaborate attributes) (15)

→ In general, data structures and types used to define attributes are defined within the context of the programming language that is to be used for implementation.

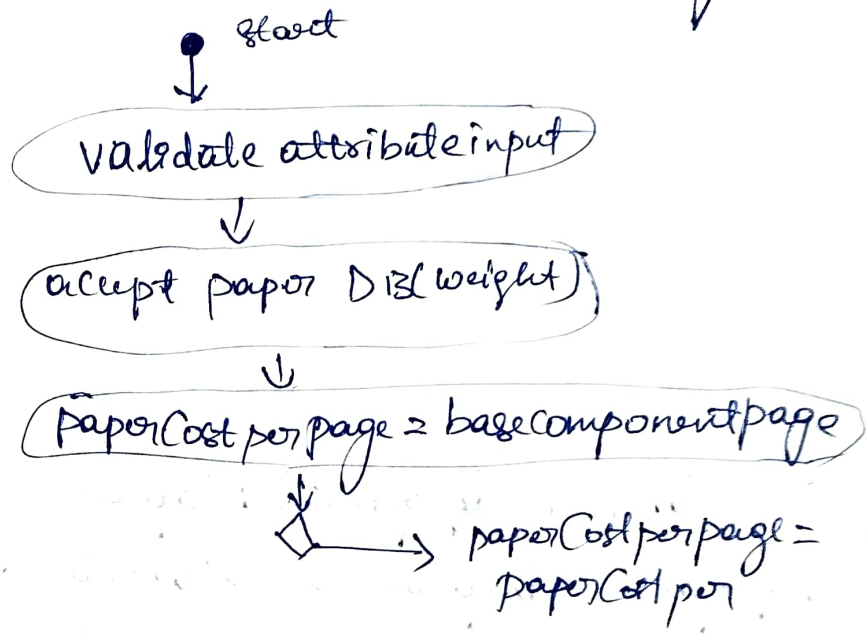
→ UML defines an attribute's data type using the following syntax:

name: type-expression initial-value {property string}

Step (3d): Describe processing flow within each operation in detail. (Describe processing flow)

→ The process flow is described in detail using activity diagram.

Figure:-

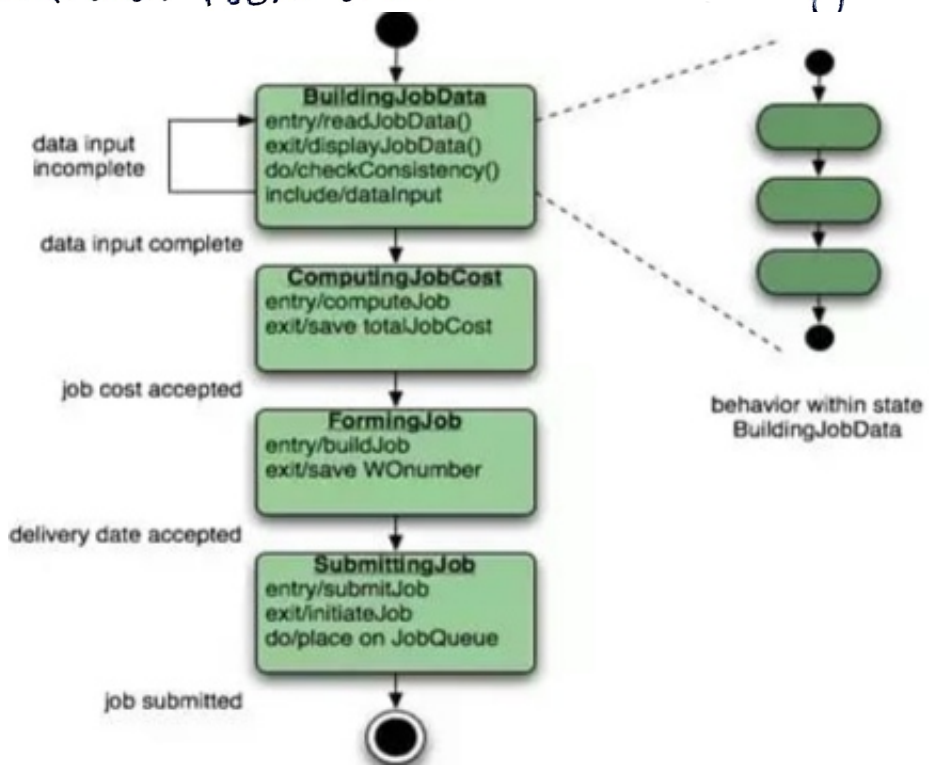


Step(4) persistance data sources

The persistance data sources are described (db + files) & the classes required to manage them are identified.

Step(5) Elaborate behavior

→ It is sometimes necessary to model the behavior of a design class. Transition from state to state having the form




Step(6): Elaborate deployment Diagrams

→ Deployment diagram are elaborated to represent the location of key package or component.

Step(7):- (Redesign/re consider)

The first component-level model you creation will not be as complete, accurate (or) consistent as the nth iteration you apply to the model.

## * Golden rules - User Interface

Theo mandel lains 3 golden rules.

1) Place the user in control. I
2) Reduce the user's memory load II
3) Make the interface consistence III

→ These golden rules actually form the basis for a set of user interface design principles that guide this important aspect of s/w design.

User Interface design :- User interface design begins with the identification of user, task and environment requirements. Once user tasks have identified, user scenarios are created and analyzed to define of a set of interface objects and actions.

### I) Place the user in control

→ Define interaction in such a way that the user is not forced into performing unnecessary actions
→ provide for flexible interaction.
→ User interaction to be interruptible and reversible.
→ Hide the technical internal from the casual user

### II) Reduce user memory load :-

→ Reduce demands on users short team memory
→ establish meaningful defaults.
→ Define short cuts.
→ Disclose information in a progressive fashions.

## III. Make interface consistent

→ Allow user to put the current task into a meaningful context.

→ Maintain consistency across all applications.

→ use the past interaction models.

## * User interface design

→ UI design creates an effective communication b/w a human and a computer.

→ It is the front end application view to which user interacts inorder to use the S/W.

→ User can manipulate and control S/W and H/W by means of UI.

Eg:- Computers, mobile phones, cars, Airplanes, music players etc.

→ It provides a platform for human-computer interaction.

→ The S/W becomes most popular if its user interface is :-

* Attractive
* Simple to use
* responsive in short time
* Clear to understand
* Consistent on all interfacing screens.

→ UI is broadly divided into two categories

1) Command line interface (CLI)
2) Graphical User Interface (GUI)

### 1) Command Line Interface (CLI)

→ CLI has been a great tool to interact.

→ The user needs to remember the syntax of command and its use.

→ less amount of computer resources as compared to GUI

→ CLI provides a command prompt, where the user types the command and feeds to the system.

CLI elements

a) Command prompt
b) Cursor
c) Command.

2) Graphical User Interface (GUI)

→ GUI provides the simple interactive interface to interact with the system.

→ GUI can be a combination of both h/w and s/w.

GUI elements

1) Window — Application window
2) Tabs — Dialogue Box
3) Menus — Text-Box
4) Icons — Buttons.
5) Cursor

* User Interface Design models

→ UI analysis and design process is an iterative process & it can be represented as a spiral model.

It consists of 4 frame work activities

1) User, task, & environment analysis, & modeling :-

→ From each category requirements are gathered.

→ Based on the requirements developer understand how to develop the interface.

→ Once all the requirements are gathered a detailed analysis is conducted.

→ The analysis of user environment focuses on the physical work environment.

2) Interface design :-

→ The goal of this phase is to define the set of interface objects and actions i.e. control mechanisms that enable the user to perform desired tasks.

→ This phase serves as the foundation for the implementation phase.

3) Interface construction:- UI development tools may be used.

-> As iterative design process continues a UI toolkit that allows the creation of windows, menus, device interaction, error messages, commands etc.

4) Interface Validation:- Correctness of s/y is validated.

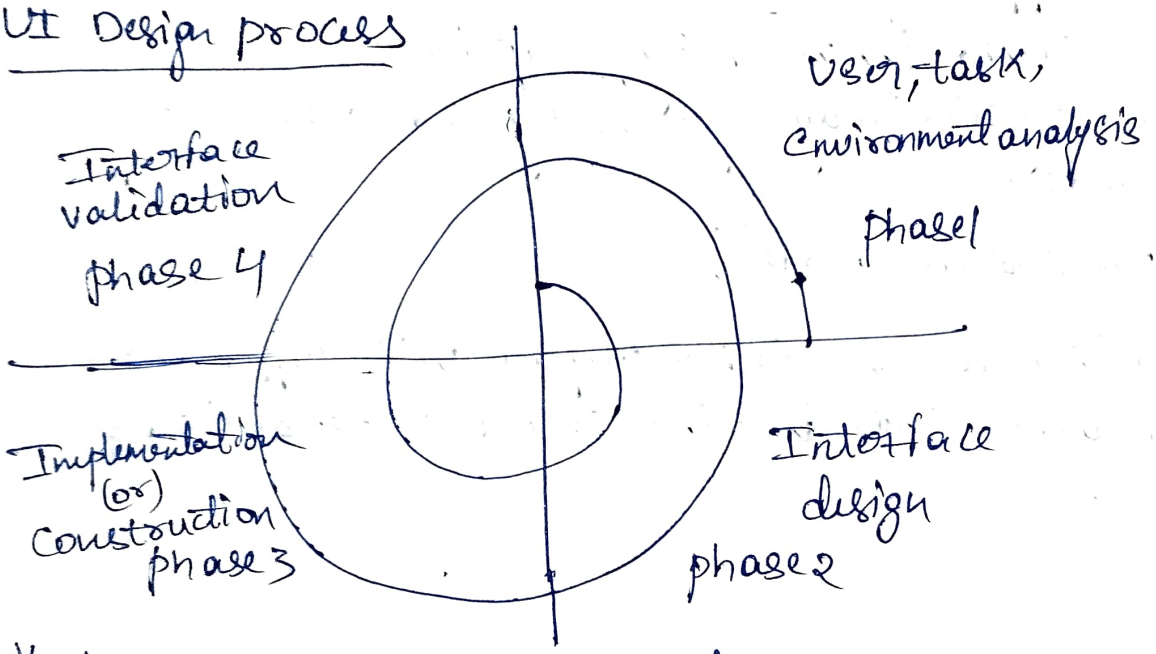
-> It should be able to perform tasks correctly.

-> It should be able to handle a variety of tasks.

-> It should achieve all the user's requirements.

-> It should be easy to use and easy to learn.

UI Design process




* User Interface Analysis and Design

-> The overall process for analyzing and designing a UI begins with the creation of different model of system function.

4 models

1) User model (established by human engineer (or) s/w engineer)

-> Establish the profile of the end-user of the s/y (Based on age, gender, education background & various other considerations)

-> This is an important criteria since, it is the end user who is going to deal with the product for ever after developed by a given s/w development team.

3 categories

a) No users (new & inexperienced)
b) Knowledgeable, intermediate users.
c) Knowledgeable, frequent users.

2) Design model :- (Created by a s/w engineer)
-> It is derived from analysis model of requirements & also based on users of the system.

3) Users Mental model :- (Developed by users when interacting with the application).
-> often called the user's s/y perception.
-> Consists of the image of the s/y that users carry in their hands.

4) Implementation model :-
-> The interface look & feel coupled with supporting information that describes interface semantic & syntax.

## Cohesion

→ Cohesion is a concept of Intra module

→ It represents relationship within a module.

→ Max. Cohesion is important.

→ It is a measure of the degree to which the elements (or) components of the module are functionally related.

→ A good S/w design will have high Cohesion.

→ It functional strengths of modules.

Types

1) Functional cohesion — High, Best
2) Sequential "
3) Communication "
4) Procedural "
5) Temporal "
6) Coincidental "
7) Logical — low, worst.

## Coupling

→ It is a concept of Inter-module

→ represents relationships b/w the module.

→ Min. Cohesion is negligible.

→ It is a measure of the degree of Independence b/w the modules

→ A good S/w design will have low Coupling.

→ It represents Independence among modules.

Types

1) Data Coupling — Low, ~~worst~~
2) Stamp "
3) Controll "
4) External "
5) Common "
6) Content ". — ~~best~~, High

# UNIT - 3

## Introduction

## Unified Modeling Language (UML)

→ UML is a general purpose modelling language.

→ The main aim of UML is to define a standard way to visualize the way a s/y has been designed.

→ It is used for specifying, visualizing, constructing and documenting the primary artifacts (by product of s/w development that helps describe the architecture, design & function of s/w) of the s/w s/y.

→ It is quite similar to blueprints used in other fields of engineering.

→ It is not a programming language, it is rather a visual language.

→ We use UML diagrams to portray the behavior and structure of a system.

→ UML helps s/w engineers, businessmen and s/y architects with modelling, design and analysis.

→ It helps in designing and characterizing, especially those s/w systems that incorporate the concept of object orientation.

→ UML makes the use of elements and forms associations b/w them to form diagrams.

→ UML diagrams is classified as

1) Structural Diagrams

→ Static.

→ capture static aspects (or) structure of a system.

→ It includes, class diagrams, Component diagrams, object diagrams etc.

2) Behavior Diagrams.

→ Dynamic.

→ Capture dynamic aspects (or) behavior of the system.

→ It includes, use case diagrams, Interaction diagrams, Activity, Sequence etc.

* Principles of Modeling.

-> Four basic principles

1) Choose your model well

-> The choice of model profoundly impacts the analysis of the problem and the design of the solution.

(or)

-> The right models will highlight the most nasty development problems. wrong models will mislead you, causing you to focus on irrelevant issues.

2) Every model may be expressed at different levels of precision

-> The same model can be scaled up (or down) to different granularities.

3) The best models are connected to reality

-> Simplify the model, but dont hide important details.

-> In s/w, the gap b/w the analysis model and the sys design model must be less.

4) No single model is sufficient

-> a set of models is needed to solve any non trivial s/y.

# Conceptual model of UML

- Building Blocks
  - Things
    - Structural things
      - Classes
      - Interfaces
      - Collaborations
      - Use case
      - Components
      - Node
    - Behavioral Things
      - Interaction
      - State machines
    - Grouping Things
      - package
    - Annotational things
      - ~~Node~~
      - Notes
  - Relationships
    - Association
    - Dependency
    - Generalization
    - Realization
  - Diagrams
    - Class
    - Objects
    - Use case
    - Sequence
    - Collaboration
    - State chart
    - Activity
    - Deployment
    - Component
- Rules
  - Names
  - scope
  - visibility
  - Integrity
  - Execution
- Common mechanisms
  - Specifications
  - Adornments
  - Common Divisions
  - Extensibility mechanism
    - Stereotypes
    - tagged values
    - Constraints

* Conceptual model of the UML

→ To understand UML, you need to form a conceptual model of language, 4 this requires 3 major elements.

1. UML's Basic Building blocks
2. The rules
3. mechanisms.

# I Building Blocks of UML

3 types

1) Things 2) Relationships 3) Diagrams

- 1) Things in the UML :-

→ 4 kinds

a) Structural things
b) Behavioral "
c) Grouping "
d) Annotational "

a) Structural things :- Static part of a model, representing elements that are either conceptual or physical

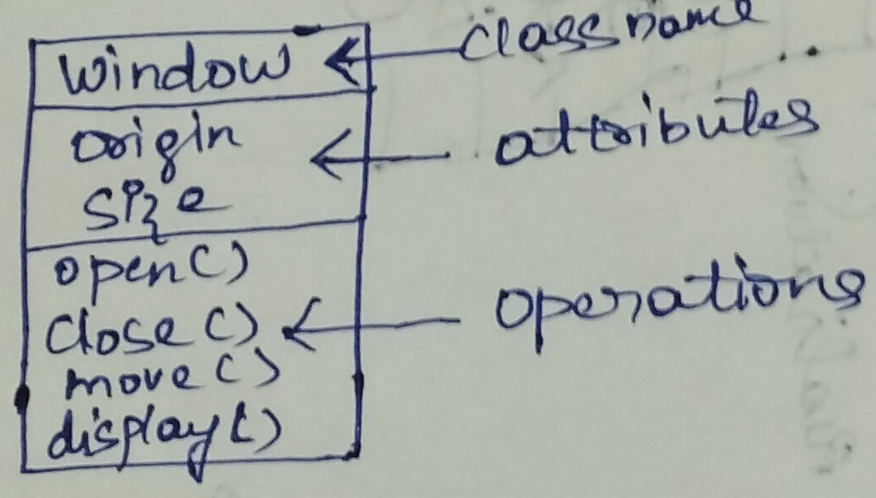
- Classes, - Interfaces, - Collaborations, - Use cases, - Active Classes, - Components, - Nodes.

- Class

→ Class is a description of a set of objects that share the same attributes, operations, relationships and semantics.

→ Class implements one (or) more interfaces.

Diagram

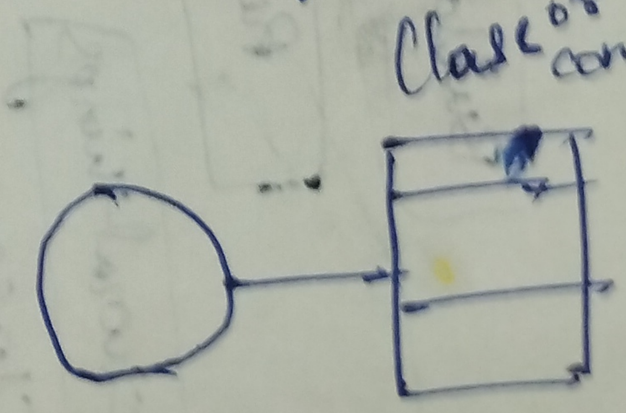



- Interface

→ a Collection of operation that specify a service of a class (or) component.

→ Describes externally visible behavior of that elements.

→ Attached to class (or) component that realizes the Interface.



- Collaboration

→ Defines an interaction, which have structural as well as behavioral dimensions.

→ A given class may participate in several collaborations.

(chain of responsibility)

- Usecase :

→ A description of set of sequence of action that a s/y performs that yields to observable result of value to a particular actor.

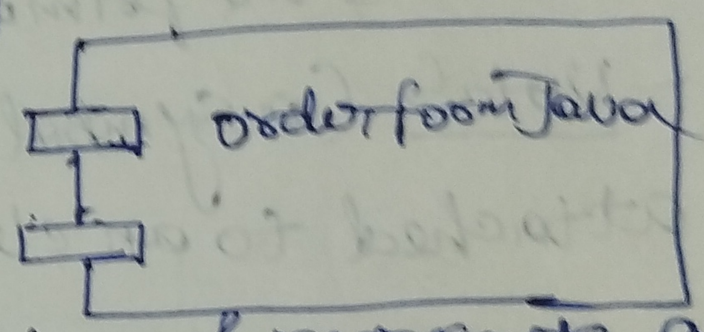
(place order)

→ It is realized by collaboration.

- Component

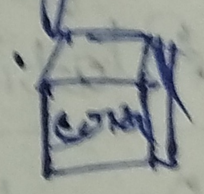
→ a physical and replaceable part of a s/y that conforms to and provides the realization of a set of interfaces.

→ represents physical packaging of classes, interfaces and collaboration.




- Node

→ a physical element that exists at runtime & represents a computational resource having at least some memory and often processing capability.



→ A set of components may reside on a node and may also migrate from node to node.

b) Behavioral things

→ Behavioral things are dynamic part of UML models.

→ These are the verbs of a model representing behaviour over time and space.

→ Two primary kinds of behavioral things

- Interaction (display messages)
- State machine (waiting states)

c) Grouping Things

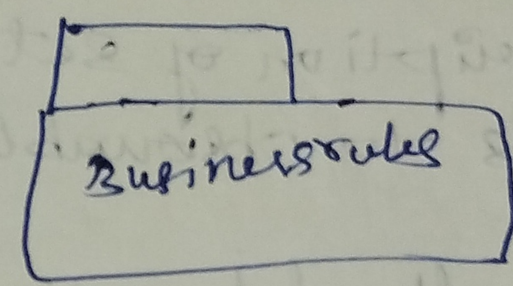
→ Grouping things are the organizational parts of UML models.

→ These are the boxes into which a model can be decomposed.

→ There is one primary kind of grouping.

- Packages (mechanism for organizing elements into groups)

- a package is purely conceptual meaning that it exists only at development time.

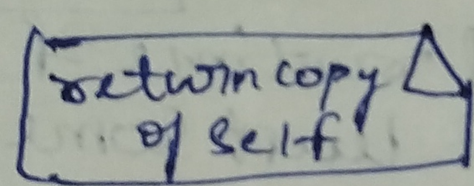



d) Annotional things.

→ Annotional things are the explanatory parts of UML models. These are the comments. you may apply to describe, & remark about any element in a model.

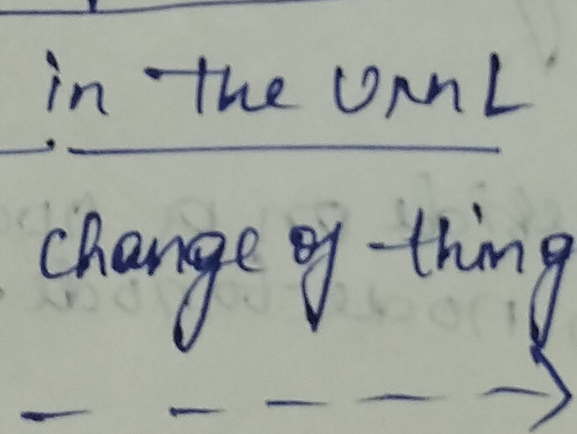
→ There is one primary kind of annotional things.

- Notes (a symbol for rending constraints and comments attached to an element or a collection of elements)

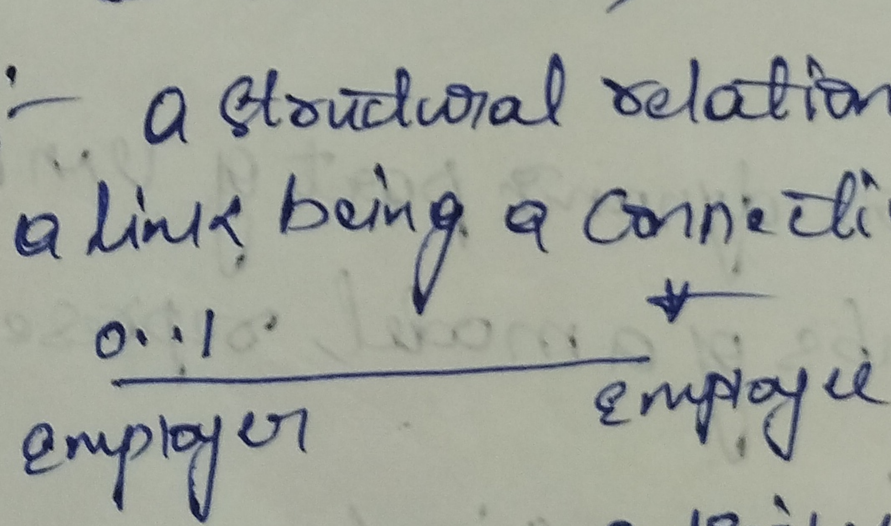



2) Relationships in the UML

a) Dependency :- change of thing may affect the other



b) Association :- a structural relationships that describes a set of links, a link being a connection among objects.




→ It contains a multiplicity and role names.

⇒ (The description of a set of links b/w objects)

c) Generalization :-

→ It is also called a parent-child relationship.

→ In generalization, one element is a specialization of another general component. → It may be substituted for it.

→ It is mostly used to represent inheritance.

⟶▷

d) Realization :-

→ One entity denotes some responsibility which is not implemented by itself and the other entity that implements them. → This relationship is mostly found in the case of interfaces.

- - - - ▷

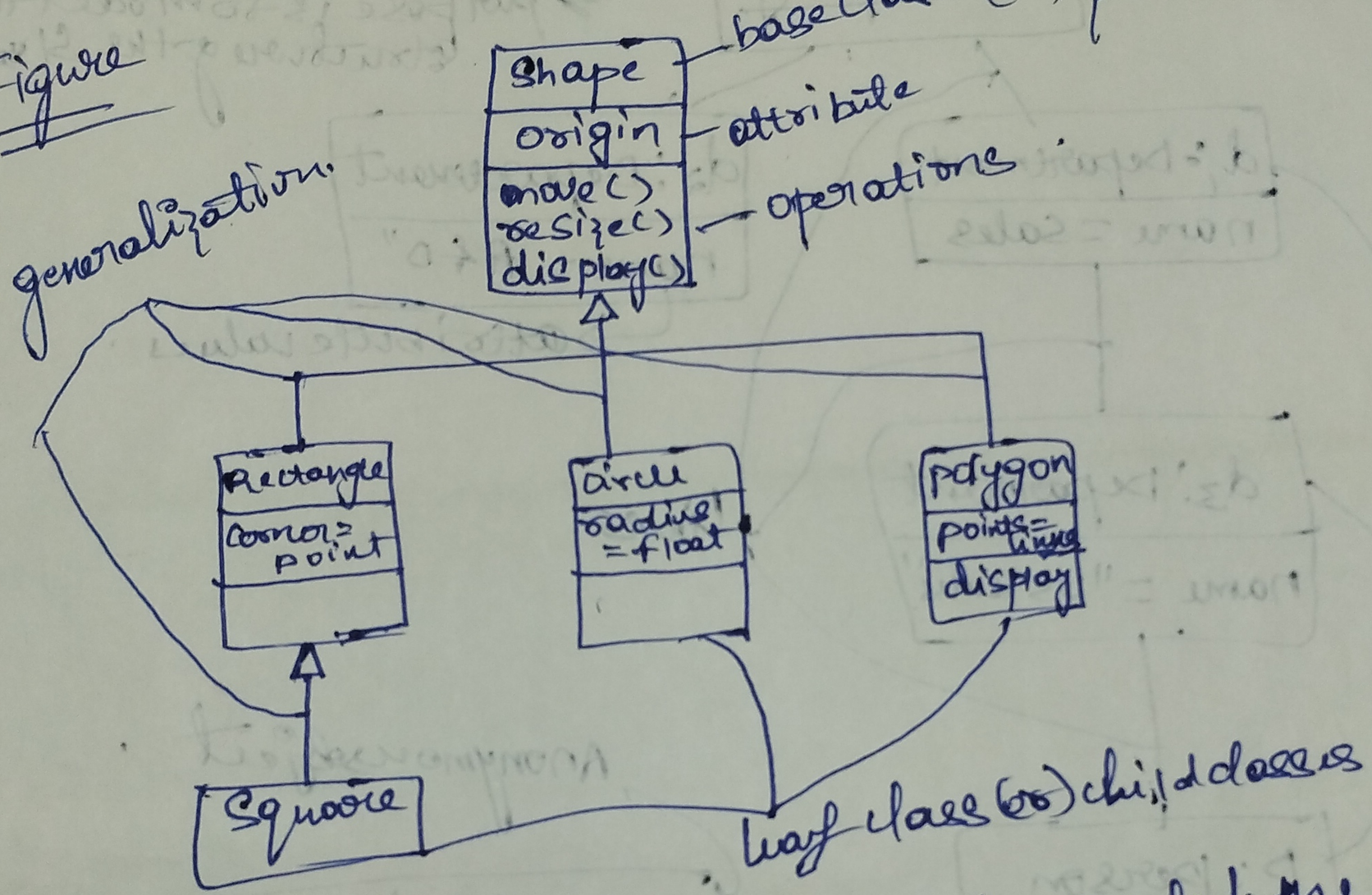
# * Class Diagram

→ Class diagram shows a set of classes, interfaces, and collaborations and their relationships.

→ Most common diagram found in modeling object orient systems.

→ Address the static design view of a s/y.

Figure




→ The purpose of class diagram is to model the structure/skeleton of the s/w s/y

→ Class diagram is a static diagram.

→ It represents the static view of application.

→ It describes the attributes & operations of a class & also the constraints imposed on the s/y.

## Purpose of Class Diagram

→ Analysis & design of the static view of an application.
→ Describe responsibilities of a system.
→ Base for component & deployment diagrams.
→ Forward & reverse engineering
→ A collection of class diagram represent the whole system.

Following points should be remembered while drawing a class diagram :-

① The name of class diagram should be meaningful to describe the aspect of the system.
② Each element & their relationships should be identified in advance.
③ Responsibility (attributes & methods) of each class should be clearly identified.
④ For each class minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
⑤ Finally, before making the final version, the diagrams should be drawn on plain paper & reworked as many times as possible to make it correct.

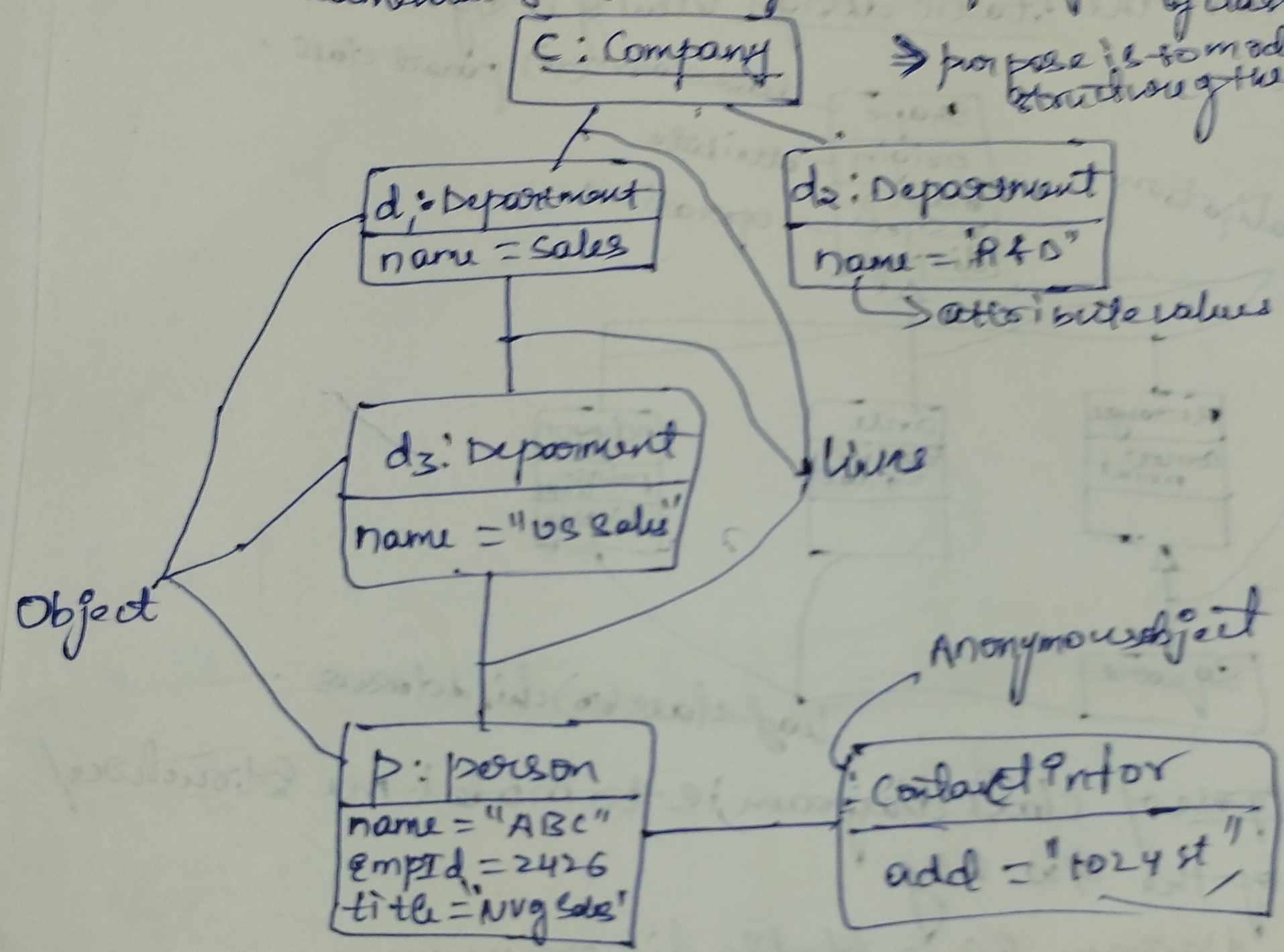
## * Object Diagram

→ Object diagram shows a set of objects and their relationships.

→ Object diagram represent static snapshots of instance of the things found in class diagram.

→ The fundamental of object diagram completely stands on basis of class diagrams

⇒ purpose is to model the structure of the sys sys.




→ It may include data values of entities(or) attributes in the sly.

⇒ object diagram shows how these objects behave at run time.

→ object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams.

→ It represent an instance of a class diagram.

## Purpose of Object Diagrams

⇒ The purpose is to capture the static view of a system at a particular moment.

1. Forward & reverse engineering
2. Object relationships of a system.
3. Static view of an interaction.
4. Understand object behaviour & their relationship from practical perspective

Following things are to be decide before starting the construction of the diagram:

1. Object diagram should have a meaningful name to indicate its purpose.
2. The most important elements are to be identified
3. The association among objects should be clarified.
4. Values of different elements need to be captured to include in the object diagram.
5. Add proper notes at points where more clarity is required.

# Common Modeling Techniques

of
Class Diagram

# Modeling Simple Collaborations

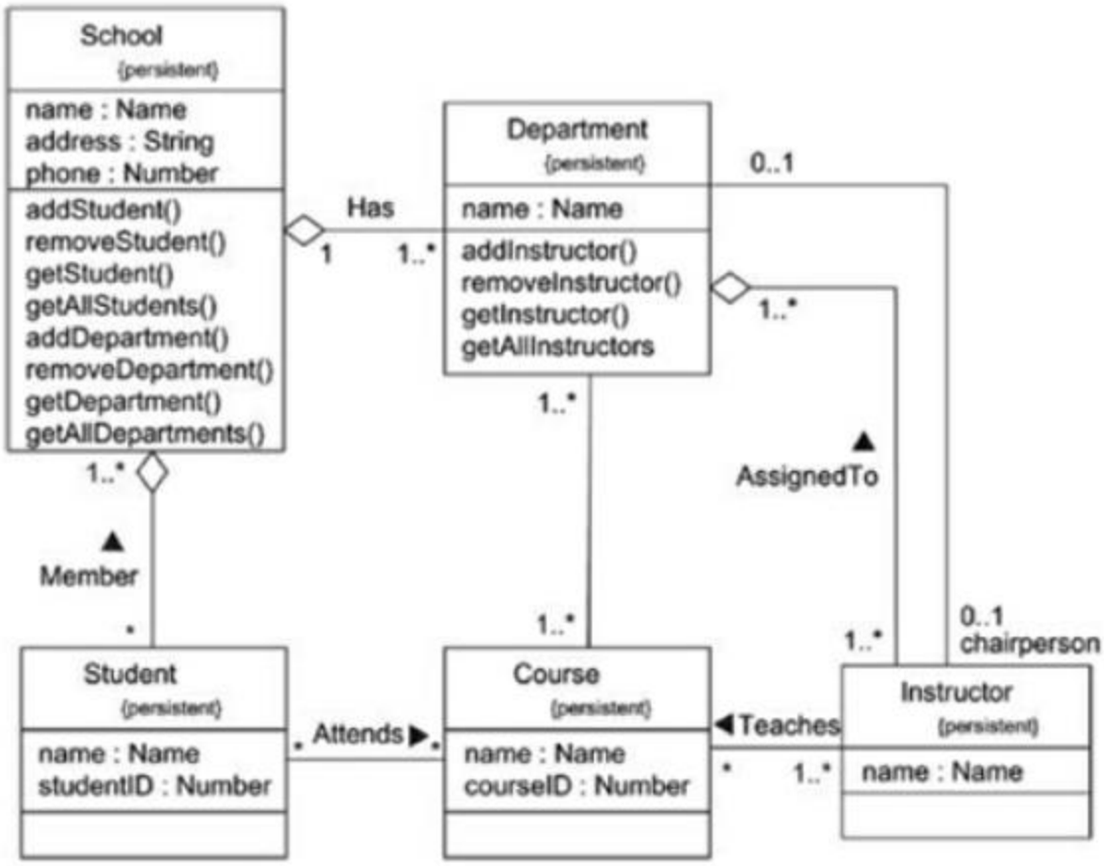
To model simple collaborations,

1. Identify the function or behavior of the part of a system you would like to model.
2. For each function or mechanism identify the classes, interfaces, collaborations and relationships between them.
3. Use scenarios (sequence of activities) to walk through these things. You may find new things or find that existing things are semantically wrong.
4. Populate the things found in the above steps. For example, take a class and fill its responsibilities. Now, convert these responsibilities into attributes and operations.

# Modeling a Logical Database Schema

To model a schema,

1. Identify the classes whose state must be saved over the lifetime of the application.
2. Create a class diagram and mark these classes as persistent by using tagged values.
3. Provide the structural details for these classes like the attributes, associations with other classes and the multiplicity.
4. Minimize the common patterns which complicate the physical database design like cyclic associations, one-to-one associations and n-ary associations.
5. Provide the behavior for these classes by listing out the operations that are important for data access and integrity.
6. Wherever possible, use tools to convert the logical design to physical design.




# Forward and Reverse Engineering

To forward engineer a class diagram,

1. Identify the rules for mapping the models to the implementation language of your choice.
2. Depending on the semantics of the language, you may want to restrict the infor-

3. Use tagged values to specify the target language.
4. Use tools to convert your models to code.

# Common Modeling Techniques

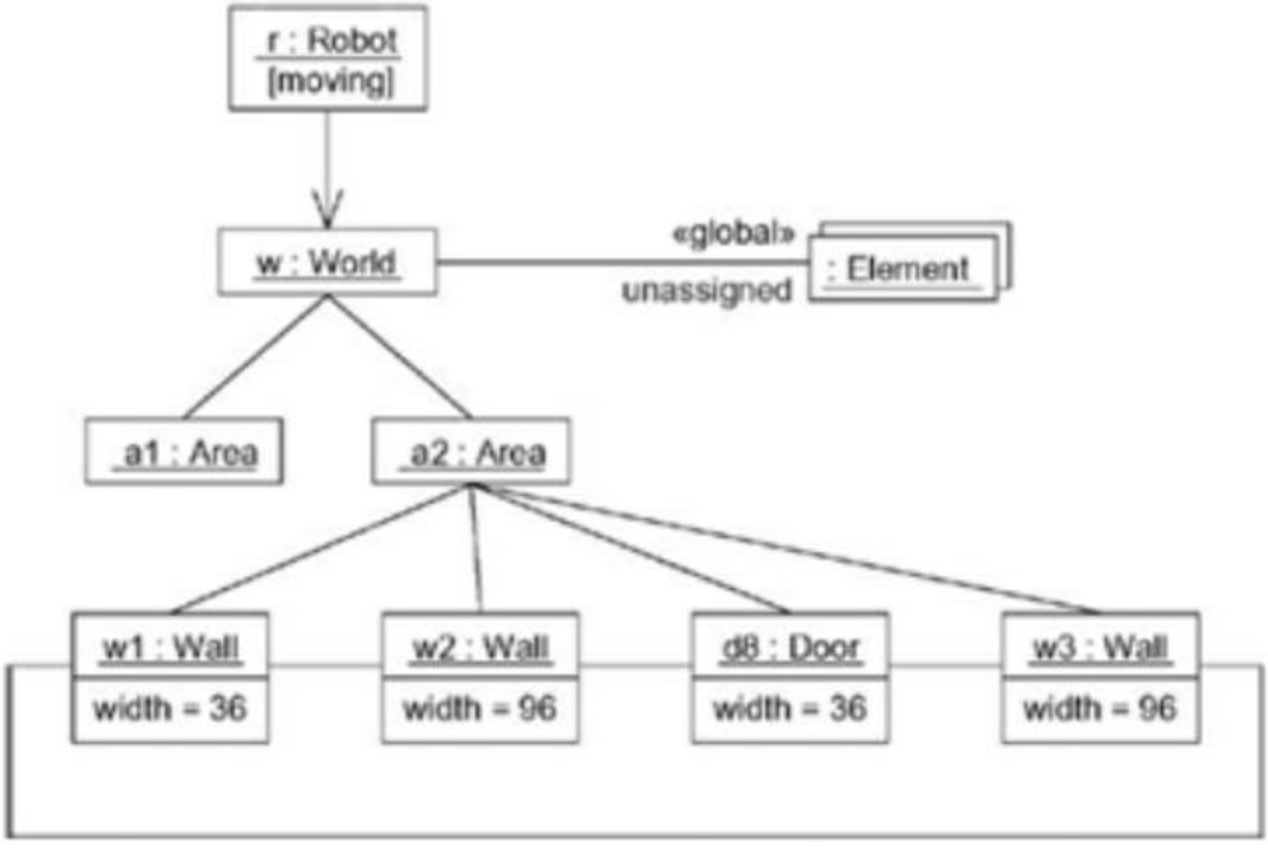
of
Object Diagram

## Modeling Object Structures

To model an object structure,

- First, identify the function or behavior or part of a system you want to model as collection of classes, interfaces and other things.
- For each function or mechanism identify the classes and interfaces that collaborate and also identify the relationships between them.

- Consider a scenario (context) that walks through this mechanism and freeze at a moment in time and identify the participating objects.
- Represent the state of objects by listing out the attributes and their values.
- Represent the links between objects which are instances of associations.



# Forward and Reverse Engineering

Forward engineering a object diagram is theoretically possible but practically of limited value as the objects are created and destroyed dynamically at runtime, we cannot represent them statically.

To reverse engineer a object diagram,

1. Choose the target (context) you want to reverse engineer.
2. Use a tool to stop execution at a certain moment in time.
3. Identify the objects that collaborate with each other and represent them in an object diagram.
4. To understand their semantics, expose these object's states.

Part B. Behavioral modeling UNIT-III

→ It contains.

1) Interaction diagrams ← Sequence diagram. / Collaboration diagram
2) Use Case diagrams.
3) Activity diagrams
4) State chart diagrams
5) Component diagrams.
6) Deployment diagrams.

Scanned by Scanner Go

## 1) Interaction Diagrams

→ Interaction is a part of dynamic behavior of the sly.

→ This interactive behavior is represented in UML by two diagrams known as Sequence diagram and Collaboration diagram. The basic purpose of both the diagrams are similar.

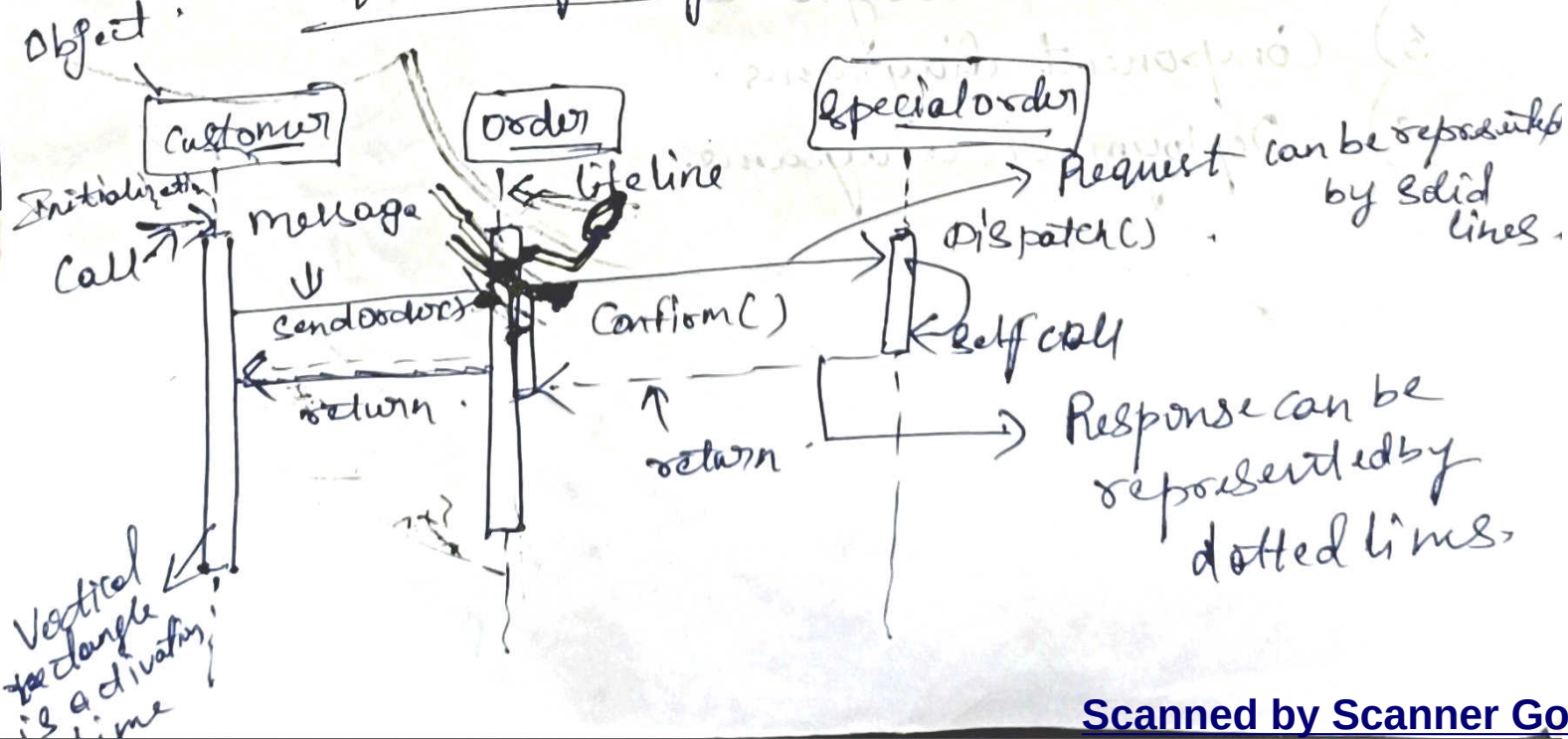
→ The purpose of interaction diagram is to visualize the interactive behavior of the sly.

- To capture the dynamic behavior of a sly.
- To describe the message flow in the sly.
- To describe the structural organization of the objects.
- To describe the interaction among objects.

### a) Sequence diagram

→ It is an interaction diagram that emphasizes the time-ordering the messages.

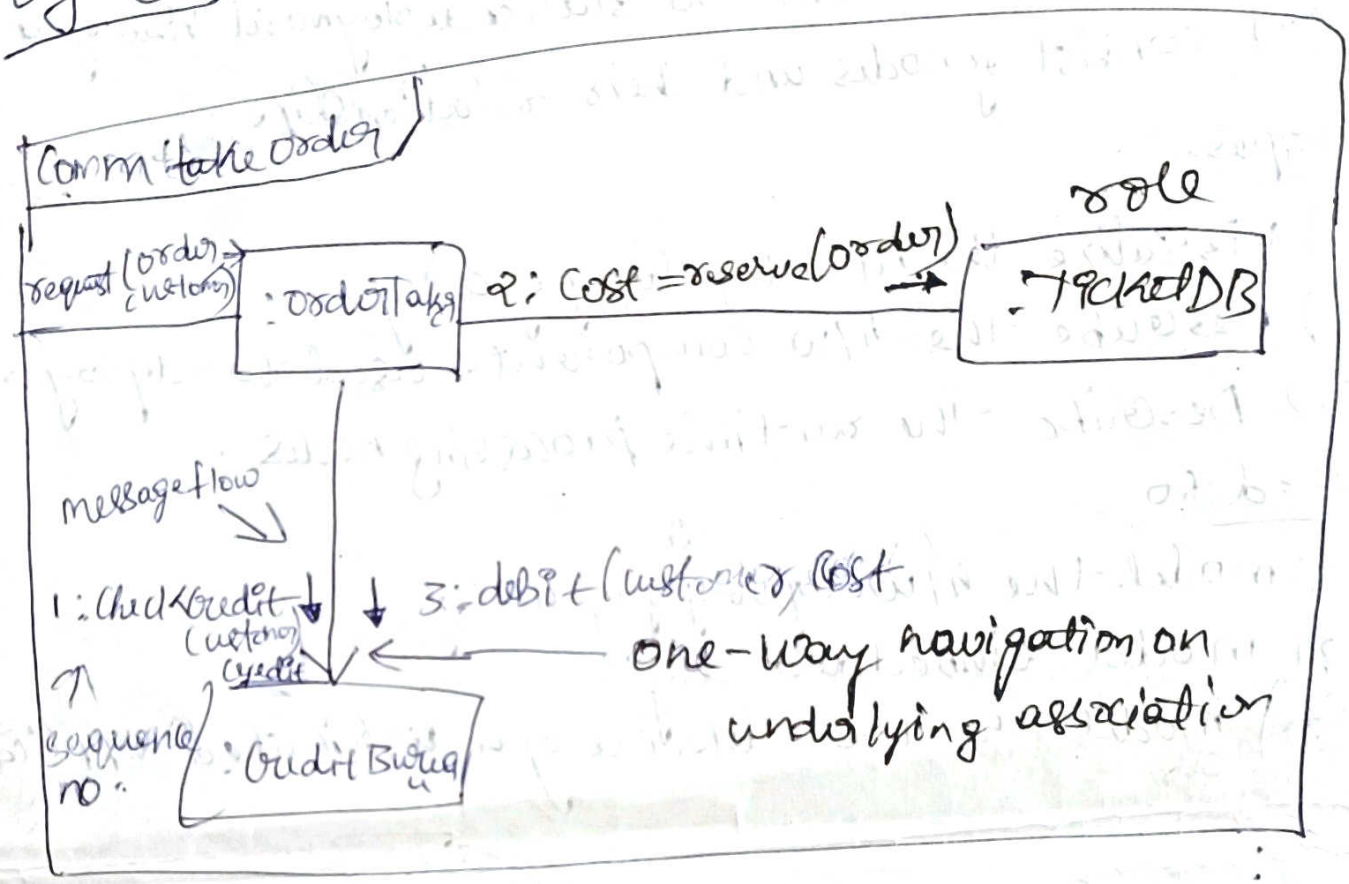
Sequence diagram of an order management sly




Scanned by Scanner Go

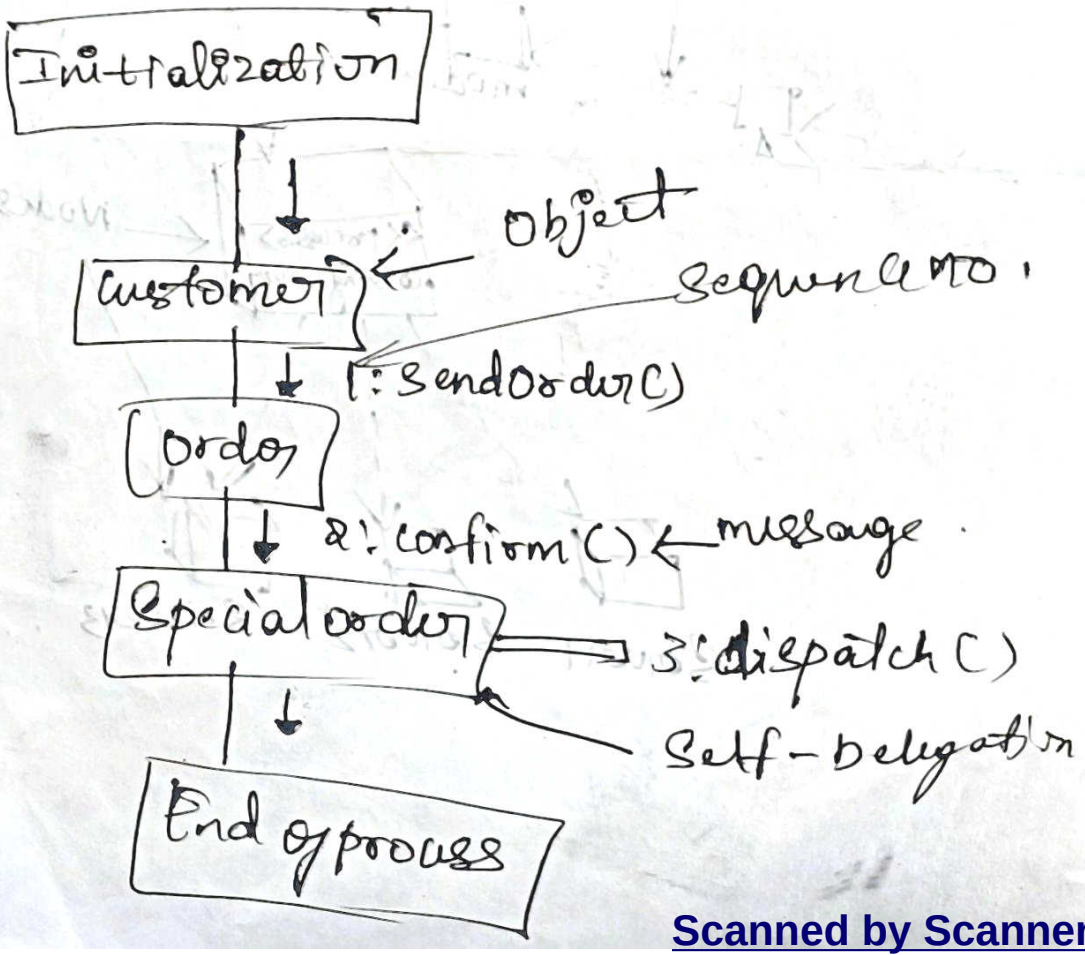
# Collaboration / Comm - Diagrams

fig:- Eg - ①




(or)

Eg - ②




Scanned by Scanner Go

## * Use Case diagrams.

→ It is used to represent the dynamic behavior of a s/y.

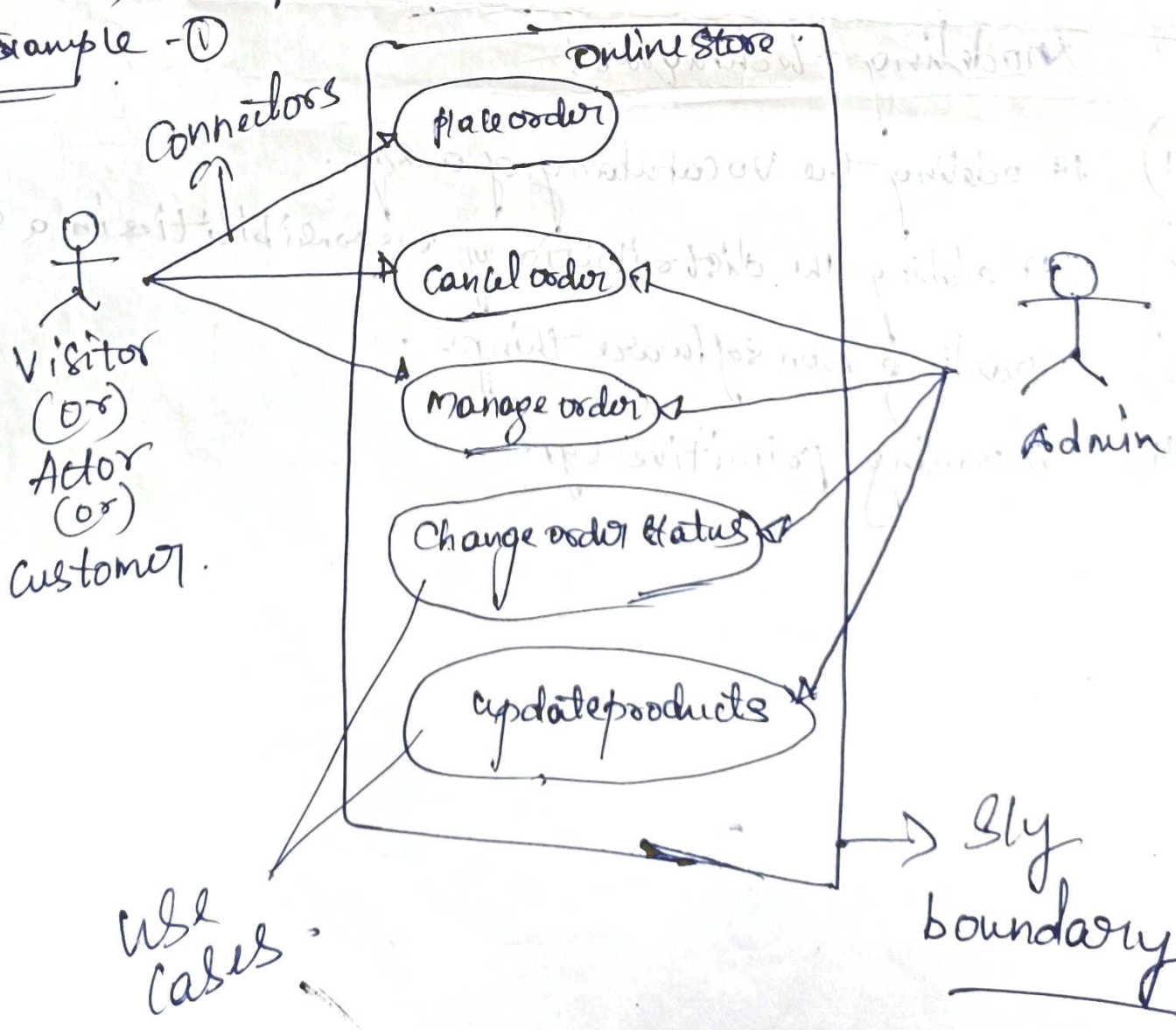
→ It encapsulates the systems functionality by incorporating use cases, actors, and their relationships.

→ The main purpose of use case diagram is to portray the dynamic aspect of a system.

Following are the purposes of a use case diagram :-

1) It gathers the system's needs
2) It depicts the external view of the s/y.
3) It recognizes the internal as well as external factors that influence the s/y.
4) It represents the interaction b/w the actors.

Example - ①




Scanned by Scanner Go

# Use Case Diagram

## Representation of Use Cases

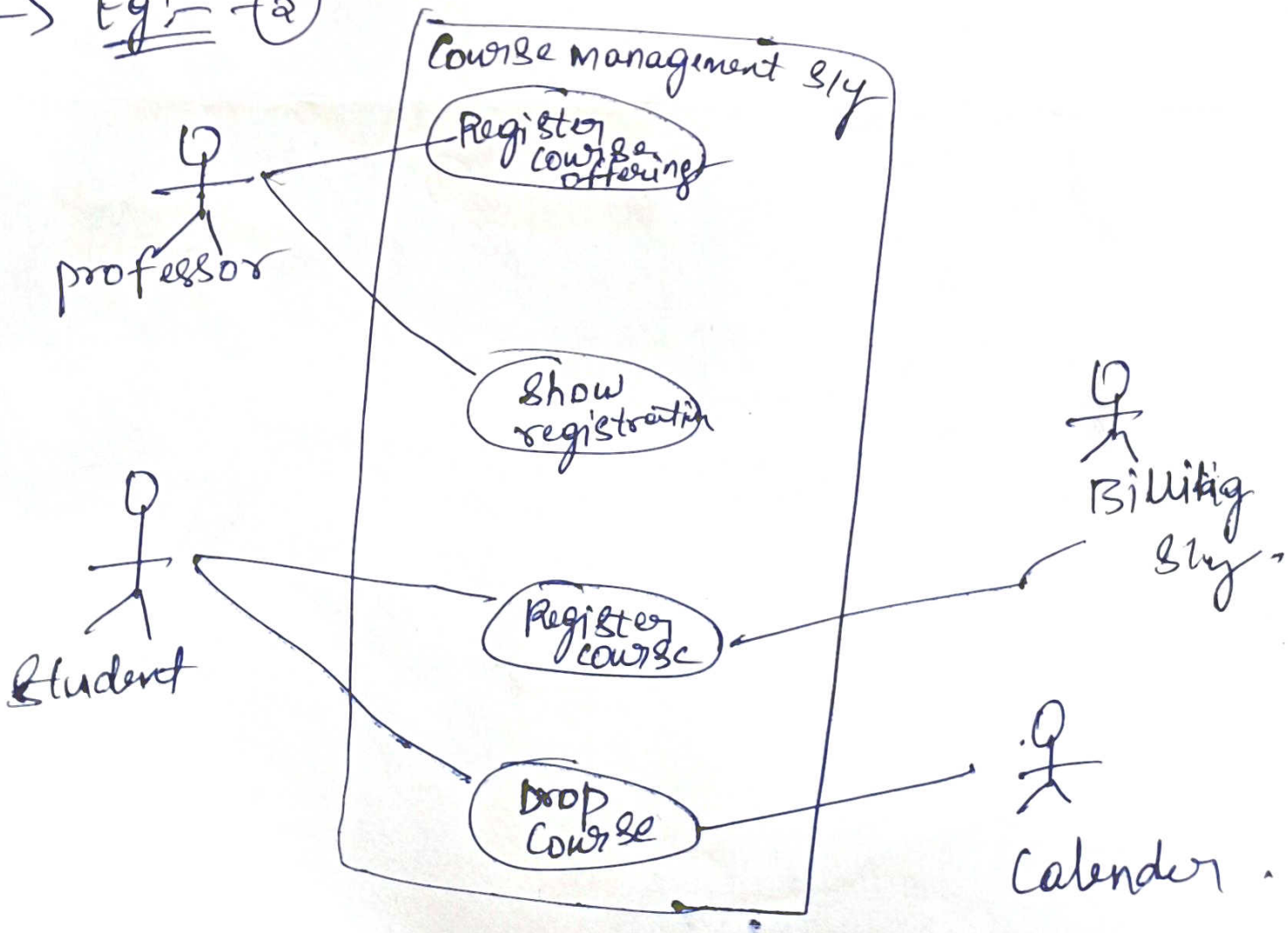
→ A Usecase is represented by an ellipse

→ System boundary is represented by a rectangle

→ Users are represented by stick person icons (actor)

→ Communication relationship b/w actor & use case by a line _____ which association relationship (or) connector.

→ Eg:- (2)




Scanned by Scanner Go

# * Activity Diagrams

- It is used to describe the dynamic aspects of the s/y.
- It is basically a flowchart to represent the flow from one activity to another activity.
- The activity can be described as an operation of the s/y.
- purpose of activity diagram is similar to other four diagrams. → we model sequencial & concurrent activities using an activity diag
- It captures the dynamic behavior of the s/y.
- Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another. → It very similar to flowchart.
- It shows different flows such as parallel, branched, concurrent, and single.

purpose of an activity diagram can be described as –

- Draw the activity flow of a s/y.
- Describe the sequence from one activity to another
- Describe the parallel, branched and concurrent flow of the s/y

following diagram is drawn with the four main activities

- Send order by the customer
- Receipt of the order
- Confirm of the order
- Dispatch the order.

uses

- modeling work flow by using activities
- modeling business seqs.

Scanned by Scanner Go

Following are the components of an Activity Diagram.

~~(1) Activities~~

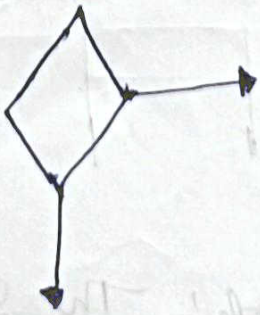
Activity Diagram Notations

(1) Initial State :- The starting state before an activity takes place is depicted using the initial state.

(2) Action / Activity State :- An activity represents execution of an action on objects (or) by objects.
→ we represent an activity using a rectangle with rounded corners.

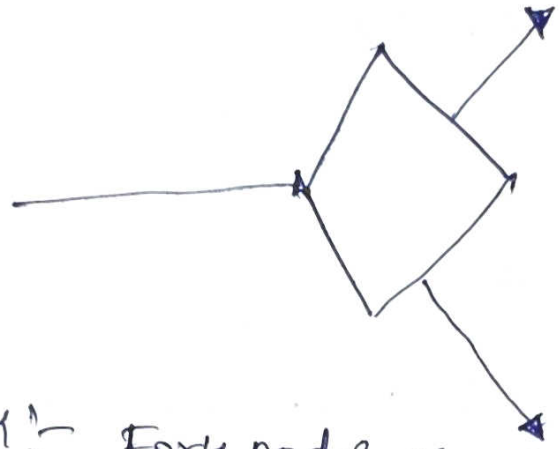
(3) Action flow / control flows :- are also referred to as paths & edges. They are used to show the transition from one activity state to another.

(4) Decision node and Branching :- when we need to make a decision before deciding the flow of control, we use the decision node.

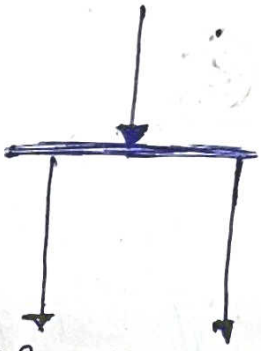


(5) Guards :-
A Guard refers to a statement written next to a decision node on an arrow sometimes with in square brackets

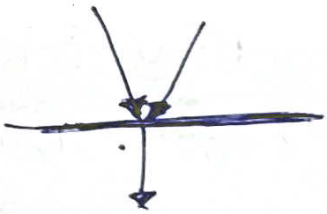
Scanned by Scanner Go



(6) Fork :- Fork nodes are used to support concurrent activities.



(7) Join :- Join nodes are used to support concurrent activites converging into one. For join notations we have two (or) more incoming edges ⟹



(8) Merge/Merge Event :-

we can merge two/more activities into one if the control proceeds onto the next activity irrespective of the path chose.

⟹

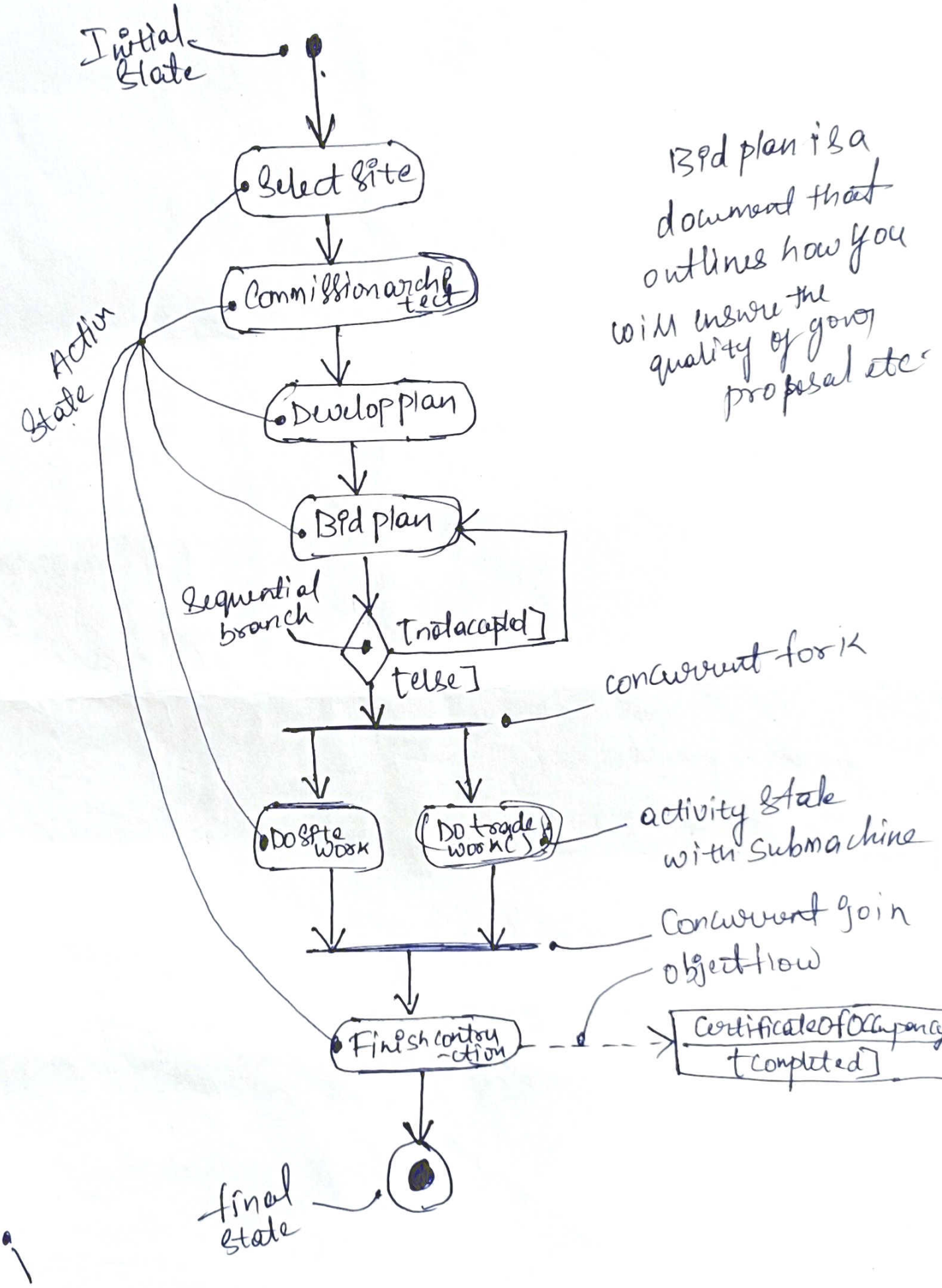
(9) Swim lanes :- used for grouping related activities in one column.

(10) Time Event :-

(11) Final State/End State :- The state which the s/y reaches when a particular process / activity end is known as a Final State / End state.

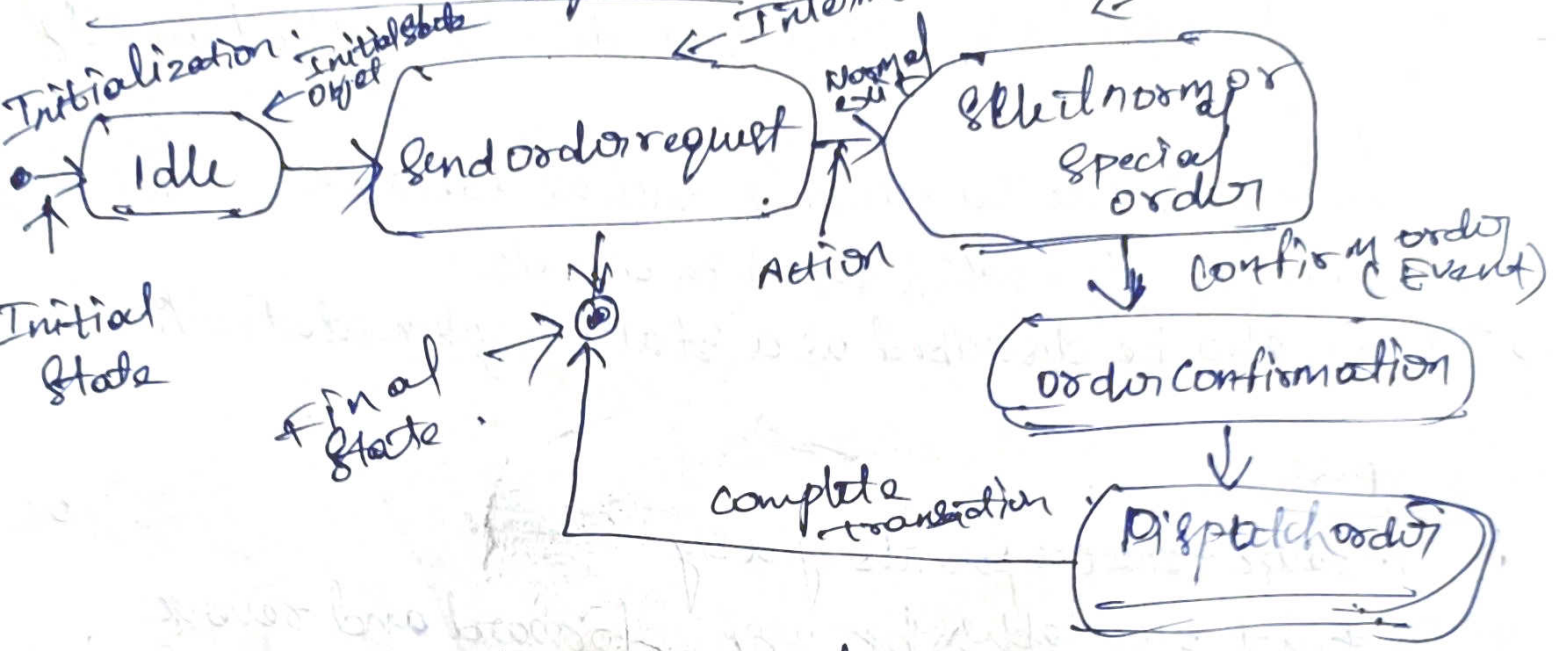
Scanned by Scanner Go

Q:- Activity diagram.




Scanned by Scanner Go

# Start chart diagrams




→ To model the dynamic aspect of the sys.

→ To model the life time of a reactive sys.

→ To describe different states of an object during its lifetime

Before drawing

→ Identify the important objects to be analyzed.

→ Identify states.

→ " events

⇒ It describes the statemachine. SM can be defined as a machine which defines different states of an object & these states are controlled by external or internal events.

⇒ The name of the diagram itself clarifies the purpose of the diagram and other details.

⇒ describes the flow of control from one state to another state.

Def:- State chart diagrams, shows a state m/c, consisting of states, transitions, events & activities.

Scanned by Scanner Go

## Component Diagram

⇒ Component diagrams are used to model the physical aspects of a system.

→ physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.

→ It can also be described as a static implementation view of a s/y.

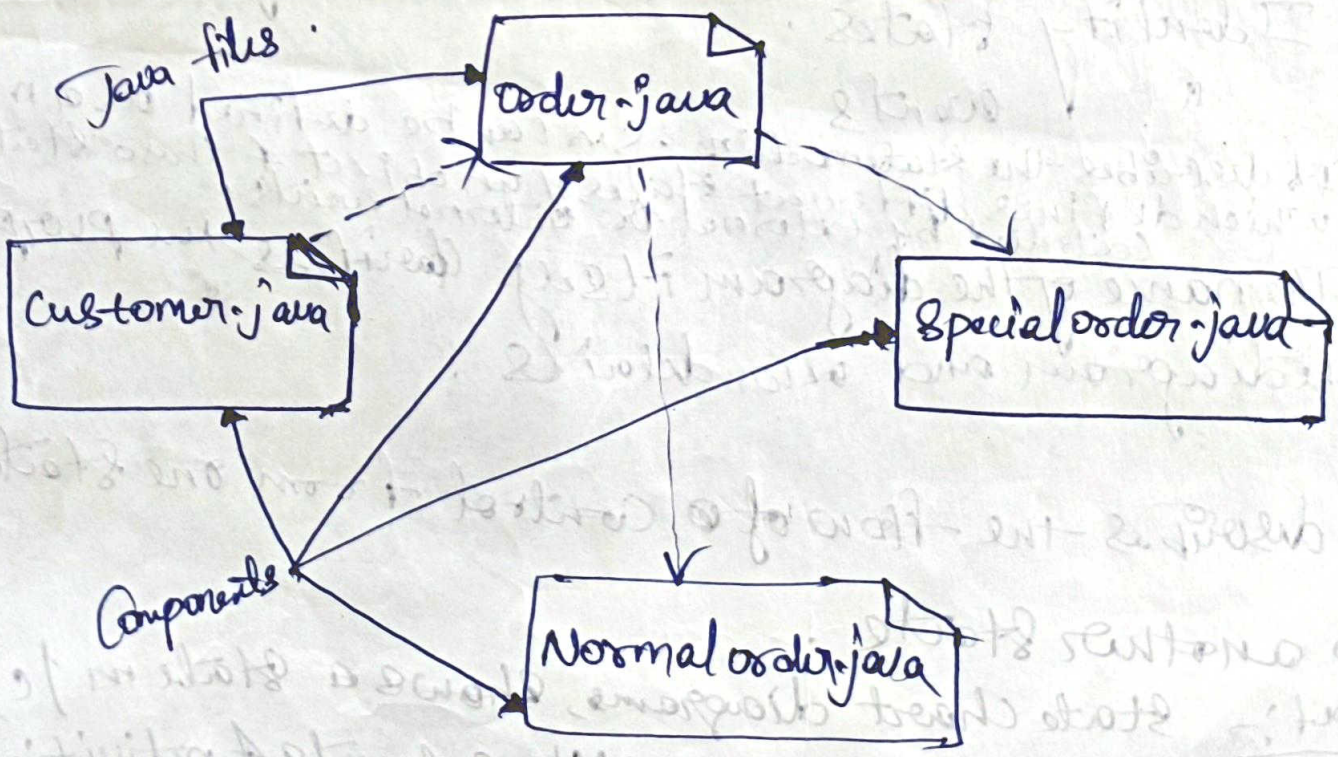
→ **purpose**

1) Visualize the components of a s/y.
2) Construct executables by using forward and reverse engineering.
3) Describe the organization & relationships of the components.

**used to**

1) model the components of a s/y
2) model the database schema.
3) model the executables of an application.
4) model the system's source code.

**Eg:**




Scanned by Scanner Go

## Deployment Diagrams

→ Deployment diagrams are used to visualize the topology of the physical components of a s/y, where the s/w components are deployed.

→ It is used to describe the static deployment view of a s/y.

→ It consist of nodes and their relationships.

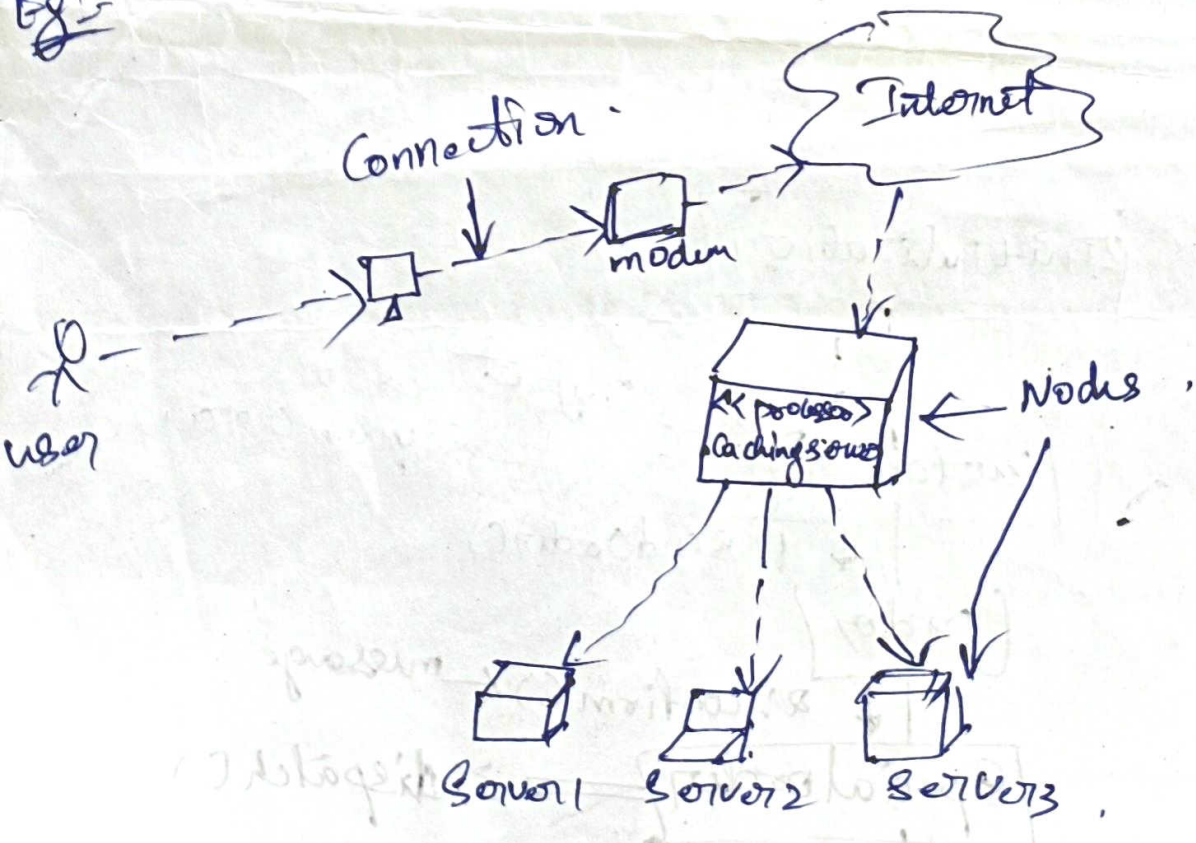
**purpose**

1) Visualize the h/w topolopy of a s/y.
2) Describe the h/w components used to deploy s/w Compo
3) Describe the runtime processing nodes.

**used to**

1) model the h/w topology of a s/y
2) model embedded s/y.
3) model the h/w details of a distributed application.

Eg:-




Scanned by Scanner Go

UNIT-4

part-A

* Software Testing

-> It is a process used to identify the correctness, complete quality of developed computer s/w

-> It also helps to identify errors, gaps (or) missing requirements.

-> It can be done either manually (or) using s/w tools.
(or)

=> s/w testing is a type of investigation to find out if there is any default (or) error present in the s/w so that the errors can be reduced (or) removed to increase the quality of the and to check whether it fulfills the specifies requir (or) not.

following objects (or) Aim (or) Goal

① When the number of errors found during the testing is high, it indicates that the testing was good and is a sign of good test case.

② Finding an unknown error that's wasn't discovered yet is a sign of a successful and a good test case.

=> The main objective of s/w testing is to design the tests in such a way that it systematically finds different types of errors without taking much time and effort so that less time is required for the development of the s/w.

Scanned by Scanner Go

**Error:-** An error is a mistake made in the code, that's why we cannot execute (or) compile code.

**Raised/Elevated** by developers & automation test engineers raise the error.

**Types:-** Different type of error are. ① Syntatic error. ② User-Interface error ③ flow control, ④ Error handling ⑤ calculation error etc.

**Reasons** for having an error are as follows!

* Errors in the code. * mistake of some values.
* If a developer is unable to compile or run a program successfully. * Confusions & issues in programming.
* Invalid login, loop, & syntax. * Inconsistency between actual & expected outcomes.

**Preventing ways:-** * Enhance the s/w quality with s/w review & programming * Detect the issues & prepare a suitable mitigation (risk) plan.

* **Bug:-** A bug is the informal name of defects, which means that s/w (or) application is not working as per the requirement.

**Raised/Elevated** by the test engineers, they submit the bug.

**Types:-** * logic bugs * Algorithmic * Resource.

**Reasons** for having bugs are * missing coding * wrong coding * Extra coding.

**Preventing ways:-** * Test driven development, offer programming lang. support * Evaluating the code systematically.

Scanned by Scanner Go

* **Defect** :- It is the difference b/w the actual outcomes & expected outputs.

**Raised by / Elevated** :- by testers identify the defect and it was also solved by the developers in the development phase / stage.

**Types** * Based on priority

1. High
2. Medium
3. Low

Based on Severity

1. Critical
2. Major
3. Minor
4. Trivial.

**Reason** :- Incorrect & wrong i/ps

**preventing ways** :- * Implementing several innovative programming methods * use of primary & correct s/w development techniques.

* **Fault** :- It is a state that causes the s/w to fail to accomplish its essential function.

**Raised / Elevated by** :- Human mistakes cause fault.

**Types** :- * Business logic faults, * functional & logical faults * faulty GUI * performance faults * Security faults * S/w / h/w fault.

**Reason behind** :- * It may occur by an improper step in the initial stage, process / data deficiency

**preventing ways** :- * Peer review * Assess the functional necessities of the s/w * Execute detailed analysis

* **Failure** :-

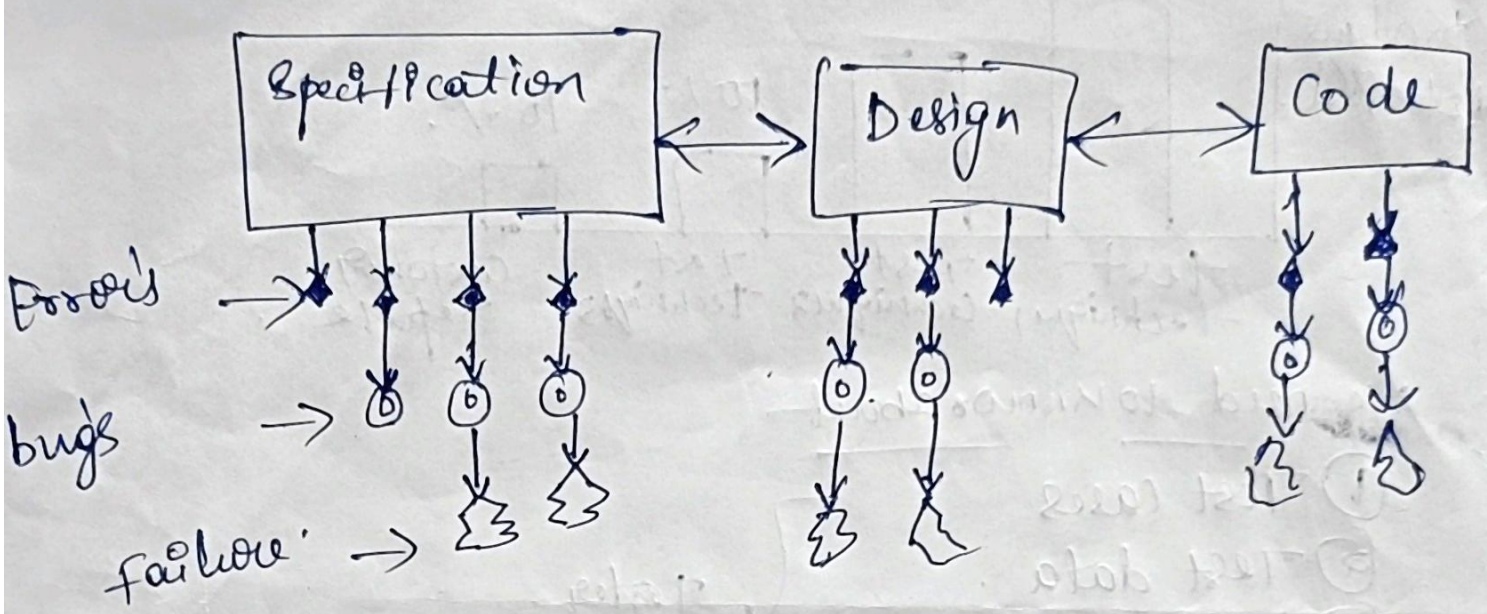
If the s/w has lotsy defects it leads to failure (or) causes failure.

The failure finds by the manual test engineer through development cycle.

Scanned by Scanner Go

Eg: ⓪ fault, defect(or) bug

♦ Error/mistake  ⚡ failure




* Few error facts

→ An experienced programmers make many errors.

→ Avg. 50 bugs per 1000 LOC

⇒ After testing the S/W

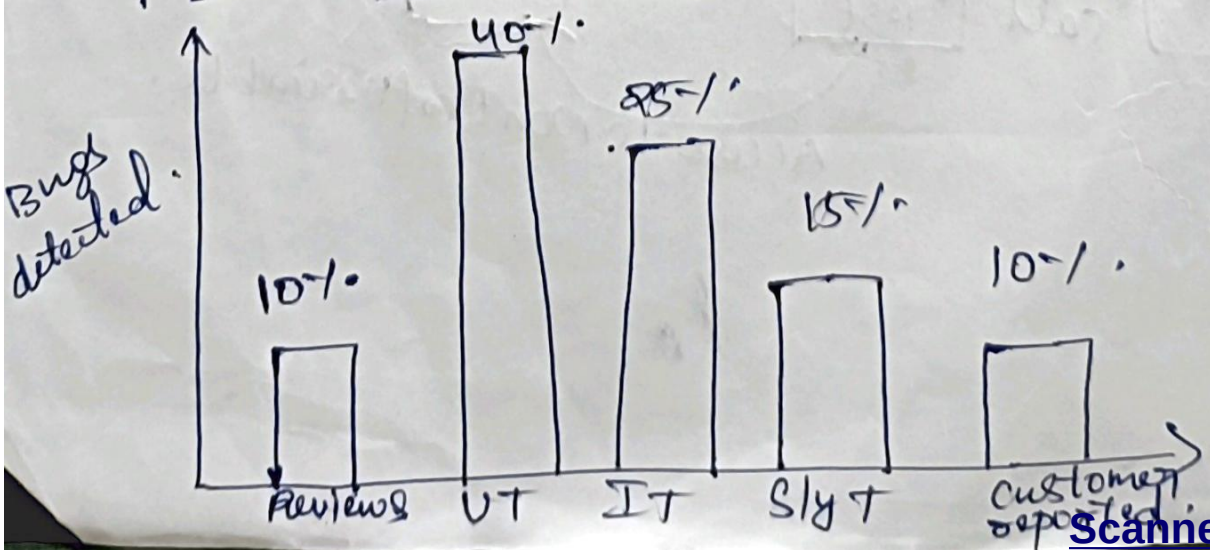
→ About 1 bug per 1000 LOC

* Bug distribution

→ For specification/ design 60%

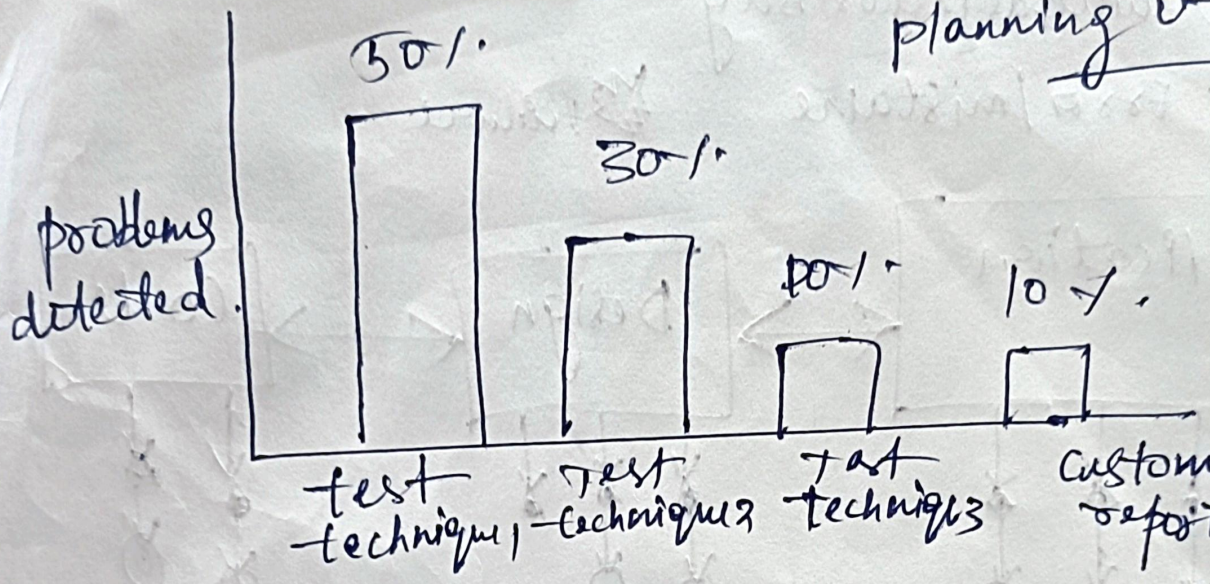
→ For implementation 40%

planning Test effort




Scanned by Scanner Go

planning VT effort




we need to known about

1. Test cases
2. Test data.
3. Test suite Design
4. Test plan

⇒ prepare failure list

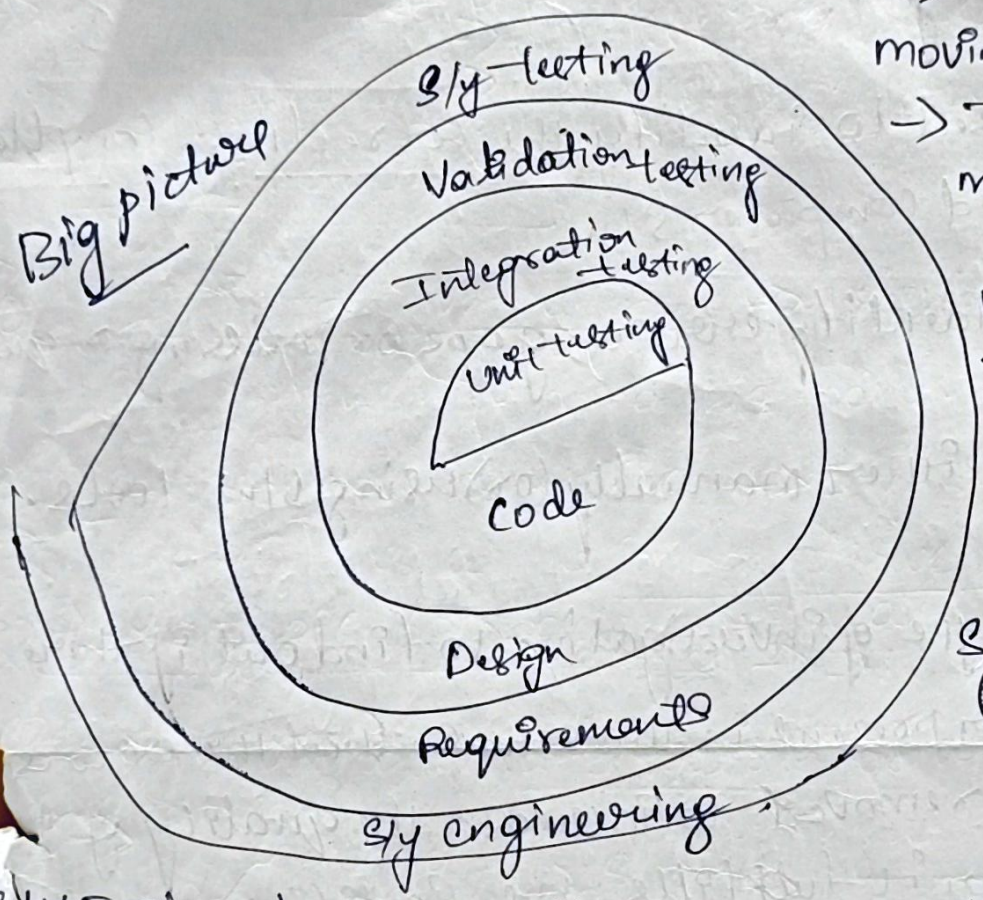
} Tester.

1. Debug to locate errors
2. correct errors

} Developers.

Scanned by Scanner Go

## * A Strategic Approach to Software Testing




→ The S/w is developed by moving in the spiral (Inwards)
→ Testing is conducted by moving in the spiral (outwards)

Unit testing ⇒ code
Integration ⇒ Design ↓ Architecture
Validation ⇒ requirements ↓ (SRS)
S/y testing ⇒ S/y engineering

**S/y Engineering** :- defines the role of the S/w & leads to requirement analysis.

**Unit Testing** :- focuses on testing each module / component independantly based on implementation.

**Integration Testing** :- Components are integrated together step by step to form a complete S/w.

**Validation testing** :- requi established are validated against the developed S/w.

**System testing** :- The S/w is tested with other S/y elements as a whole.

Scanned by Scanner Go

## * Strategies for conventional software

I. Unit Testing → Unit-test considerations
→ Unit-test procedures.

II. Integration Testing Strategies.

1) Top-down Testing.
2) Bottom-up testing.
3) Regression testing → Verify (or) modify (or) update in a s/w without affecting overall working functionality
4) Smoke testing → determines suitable (or) not
↳ continue testing.

## * A Strategic Approach to Software Testing.

→ Verification and Validation.
→ Organizing for S/W Testing.
→ S/W testing Strategy - The Bigpicture (Spiral representation)
→ Criteria for completion of Testing.

## * S/W testing strategies

1. Identify and specify the requirements
2. Specifying the objectives of testing in a clear & detailed manner.
3. Identifying & developing users profile.
4. Developing a test plan.
5. Build Robust S/W.
6. Conducting Formal Technical Reviews (FTR).
7. Using effective Formal reviews
8. Developing continous development approach.

Scanned by Scanner Go

# * A Strategic Approach To Software Testing

→ Testing is a set of activities that can be planned in advance and conducted systematically.

## Generic characteristics

→ ① To perform effective testing, you should conduct effective technical reviews. By doing this, many errors will be eliminated before testing commences.

→ ② Testing begins at the component level and works "outward" toward the integration of the entire computer-based s/y.

→ ③ Different testing techniques are appropriate for different s/w engg. approaches and at different points in time.

→ ④ Testing is conducted by the developer of the s/w and (for large projects) an independent test group.

→ ⑤ Testing and debugging are different activities, but debugging must be accomodated in any testing strategy.

→ A strategy for s/w testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major s/y functions against customer requirements.

low-level test ⇒ verify.

high-level test ⇒ validate.

Scanned by Scanner Go

# Verification and Validation

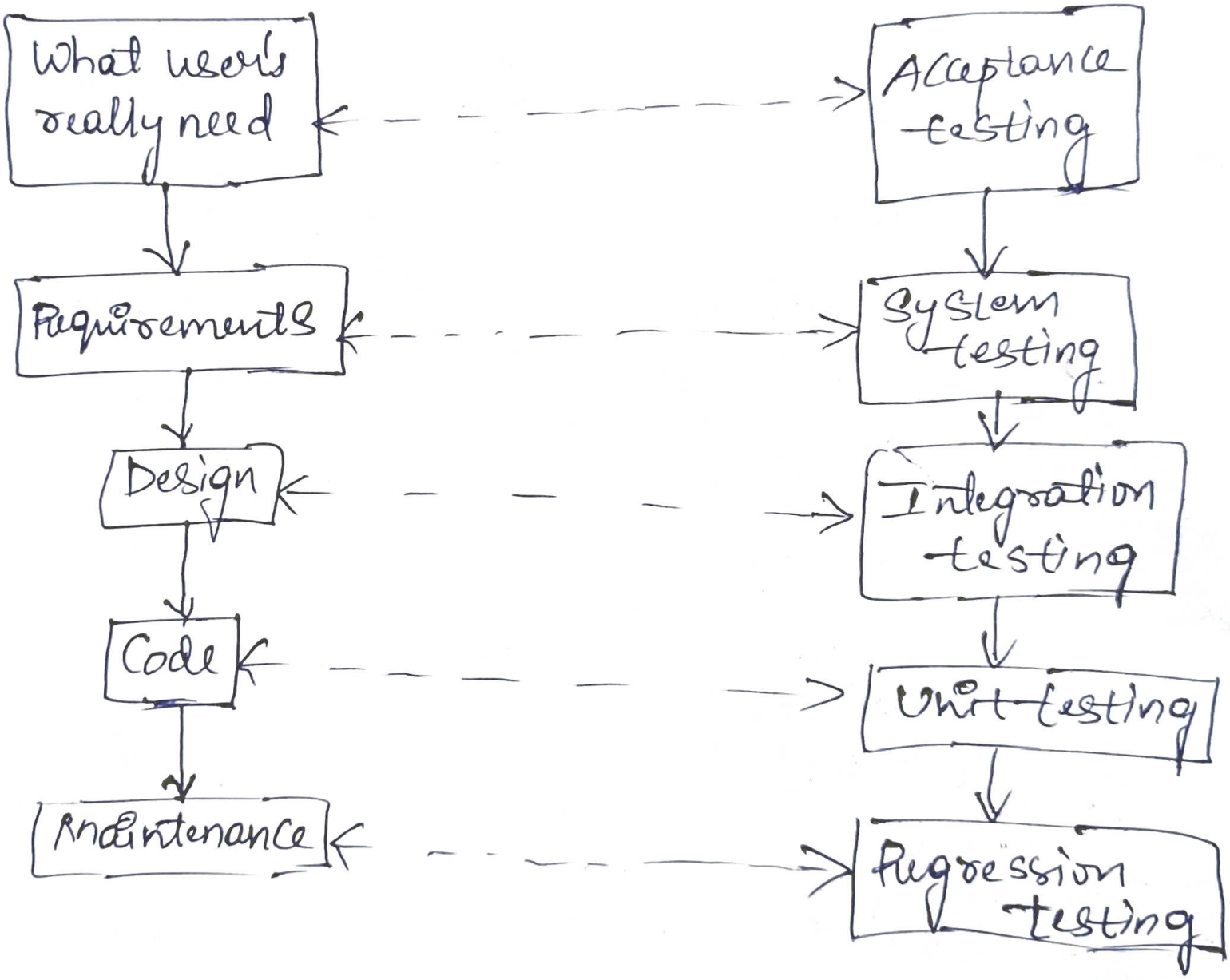
**Verification** :- "Are we building the product right?"

**Validation** :- "Are we building the right product?"

| Verification | Validation |
|---|---|
| → We check whether we are developing the right product (or) not. | → We check whether the developed product is right. |
| → It is also known as static testing. → Done by developers | → It is also known as dynamic testing → done by testers. |
| → It includes different methods like inspections, reviews, and ~~walk throughs~~ UT & IT | → It includes testing like functional testing, integration, & user acceptance testing / ST |
| → It is a process of checking the work-products of a development cycle to decide whether the product meets the specified requirements. | → It is a process of checking S/W during or at the end of the development cycle to decide whether the S/W follow the specified business requirements |
| → Quality assurance comes under Verification testing. | → Quality control comes under Validation testing. |
| → Finding the bugs early in the development phase of the product | → Finding bugs which are not caught in the verification process |
| → Verification is done before the validation testing. | → Validation testing is executed by the team to test the application. |
| | → After verification testing, validation testing takes place |

Scanned by Scanner Go

# Testing levels




Scanned by Scanner Go

## Strategies for Conventional Software

-> There are many strategies that can be used to test software.

### Unit Testing.

-> Unit testing is defined as a type of s/w testing where individual components of a s/w are tested.

-> Unit testing of the s/w product is carried out during the development of an application.

-> UT is very cost effective way of reducing's bugs.

-> An individual component (or) module may be either an individual function or a procedure.

-> It can detect algorithm error (or) a programming error.

-> Unit testing is first level of testing done before integration testing.

-> UT reduces debugging effort substantially.

-> It is a type of testing technique that is usually performed by developers.

-> without UT errors become difficult to track down

-> without UT debugging cost increases substantially

-> The unit test focuses on the internal processing logic and data structures within the boundaries of a component.

-> This type of testing can be conducted in parallel for multiple components.

Figure:

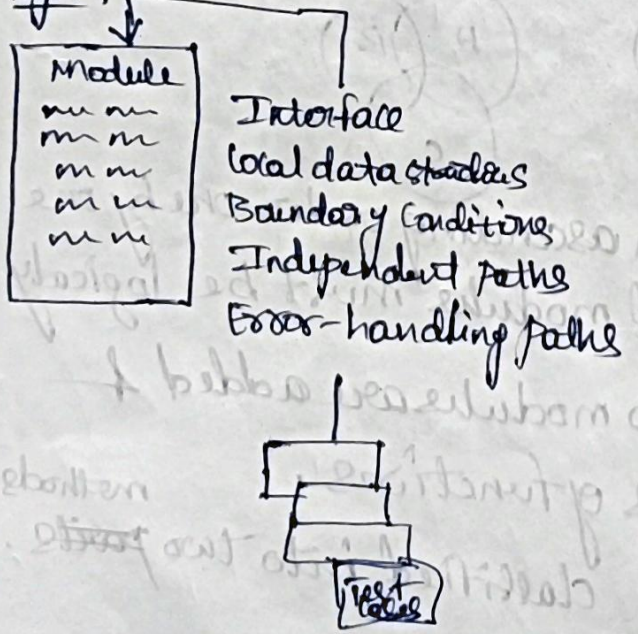
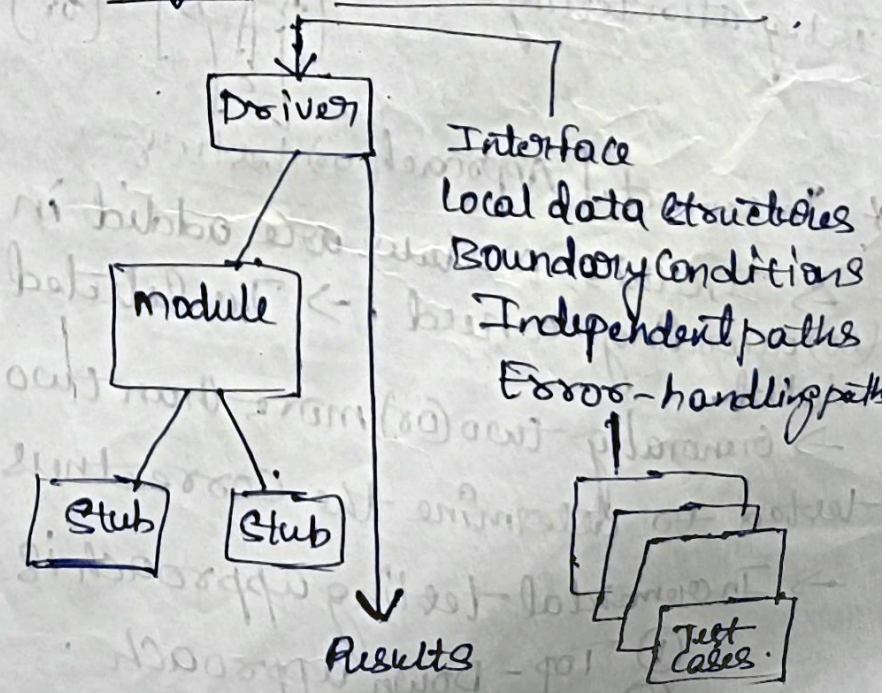
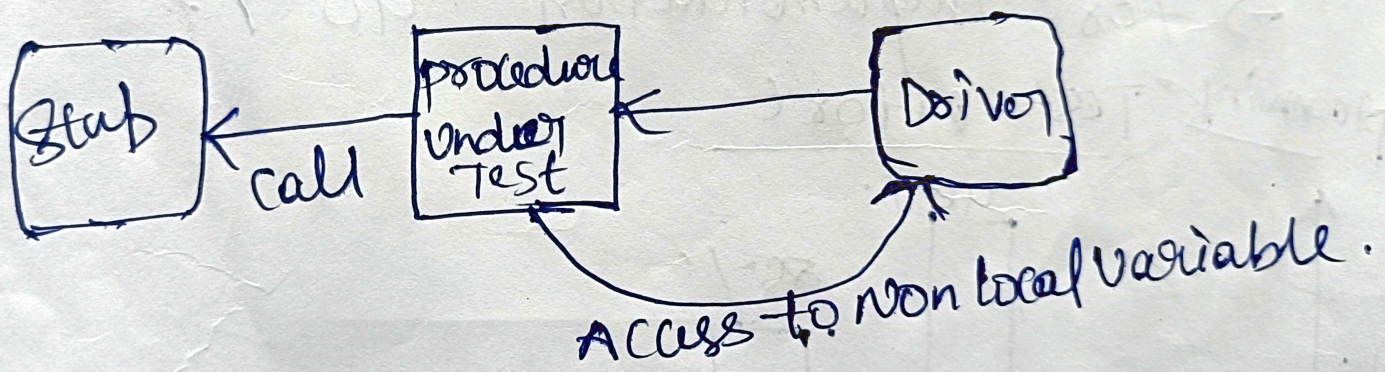



Figure:- Unit-test environment




Scanned by Scanner Go

Unit testing figure




Scanned by Scanner Go

* Stub :- A stub is a small piece of code that takes the place of another component during testing.

→ The benefit of using a stub is that it returns consistent results, making the test easier to write.

→ And you can run tests even if the other components are not working yet. Driver :- Simulates the behaviour of a function that calls & supplies necessary data to the function being tested.

II) Integration Testing

Scanned by Scanner Go

# White Box Testing

→ White box techniques analyze the internal structures the used data structures, internal design, code structure,

→ White box test cases designed on code structure of program.

→ It is a test-case design philosophy that uses the control structure described as part of component-level design to derive test cases.

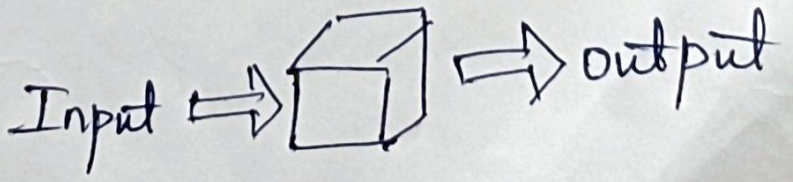
→ Using white-box testing methods, you can derive test cases that

1. guarantee that all independent paths within a module have been exercised at least once,
2. exercise all logical decisions on their T & F sides.
3. exercise all loops at their boundaries & within their operational bounds, &
4. exercise internal data structures to ensure their validity.

→ White box testing also known as.

- Glass-box testing
- Structural testing.
- Clear-box "
- Code-based "
- Transparent ".

Figure :-




Scanned by Scanner Go

## White-Box Testing Strategies

* **Coverage-based**: Design test cases to cover certain program elements.

→ Several white-box testing strategies have become very popular:

1. Statement Coverage
2. Branch Coverage
3. path "
4. Condition "
5. MC/DC "
6. Dataflow-based.

* **Fault-based**: Design test cases to expose some category of faults.

→ Type

1. Mutation testing.

Scanned by Scanner Go

## ① Statement Coverage :-

→ Design test cases so that every statement in the program is executed at least once.

→ Unless a statement is executed, we have no way of knowing if an error exists in that statement

$$\text{Coverage measurement} = \frac{\text{No. of executed statements}}{\text{Total no. of statements}} \times 100$$

**Example ① :-**

```
Read A
Read B
if A > B
  print "A is greater than B"
else
  print "B is greater than A".
endif.
```

**case ①**

If A = 7, B = 3

No. of statements Executed = 5

Total statements = 7

Statement coverage = 5/7 × 100 = 71.00%.

**case ②** If A = 4, B = 8

No. of statements executed = 6

Total statements = 7

Statement coverage = 6/7 × 100 = 85.20%.

Scanned by Scanner Go

## * Control Structure Testing (or) Condition testing

→ It is a testcase design method that exercises the logical conditions contained in a program module.

→ Simple condition
→ Compound "

(or) (and) (not)
( | , & ¬ ) ⇒ Boolean operators

### Types of errors in condition

→ Boolean operator error.
→ Boolean variable "
→ Boolean parenthesis "
→ Relational operator errors.

⇒ Data flow Testing

⇒ Loop Testing

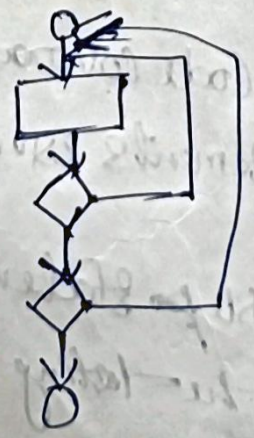
- Simple loops
- Nested loops
- Concatenated loops.
- Unstructured loops.

→ This method selects test paths of a program according to the locations of definitions & uses of variables.

**Simple loop**

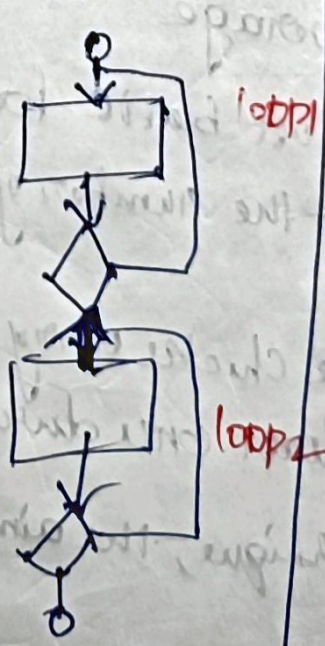
only one condition

**Nested loop.**

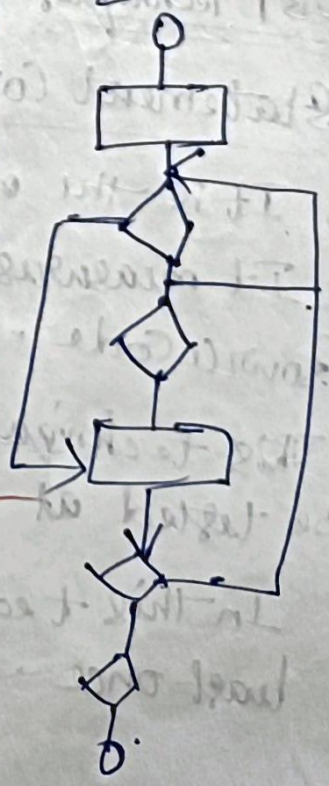


more than one loop in one condition in another.

**Concatenated loops**




**Unstructured loops**



Scanned by Scanner Go

## * Path Coverage (or) path Testing *

→ Design test cases such that:

- All linearly independent paths in the program are executed at least once.

→ Defined in terms of

- Control flow Graph (CFG) of a program.

### path Coverage - based testing

→ To understand path coverage based testing - we need to learn how to draw control flowgraph (CFG) of a program.

→ A CFG describes - the sequence in which different instructions of a program get executed

- The way control flow through the program.

### How to draw CFG?

→ Number of all statements of a program

→ Numbered statements:

- represent nodes of CFG.

Scanned by Scanner Go

→ Every program is composed of:

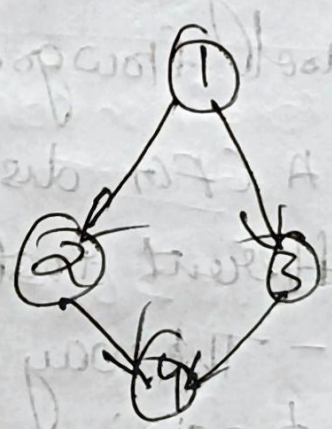
- sequence
- Selection
- Iteration

**Sequence:**

1. a = 5
2. b = a*b-1;

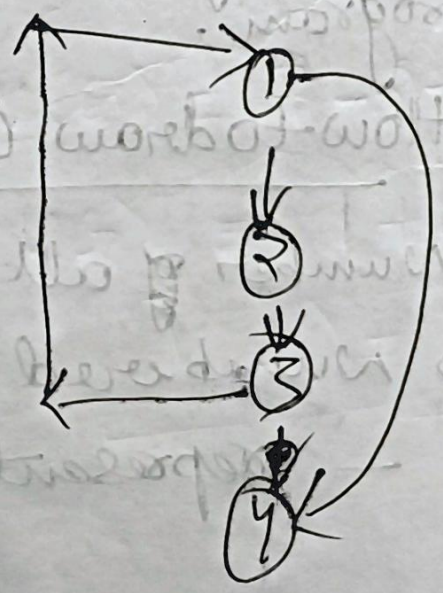
**Selection:**

1. if (a > b) then
2. c = 3;
3. else c = 5;
4. c = c*c;




**Iteration**

1. while (a>b){
2. b = b*a;
3. b = b-1; }
4. c = b+d;




Scanned by Scanner Go

**Path** :- A node and edge sequence from the starting node to a terminal node of the control flow Graph (CFG).

- There may be several terminal nodes for program.

**Linearly Independent path**

-> Any path through the program that:

- Introduces at least one new edge;
- not included in any other independent paths.

**Independent path**

-> It is straigh forward:

- To identify linearly independent paths of simple programs.

Scanned by Scanner Go

## * McCabe's Cyclomatic metric

→ It provides a practical way of determining the maximum number of test cases required for basis path testing.

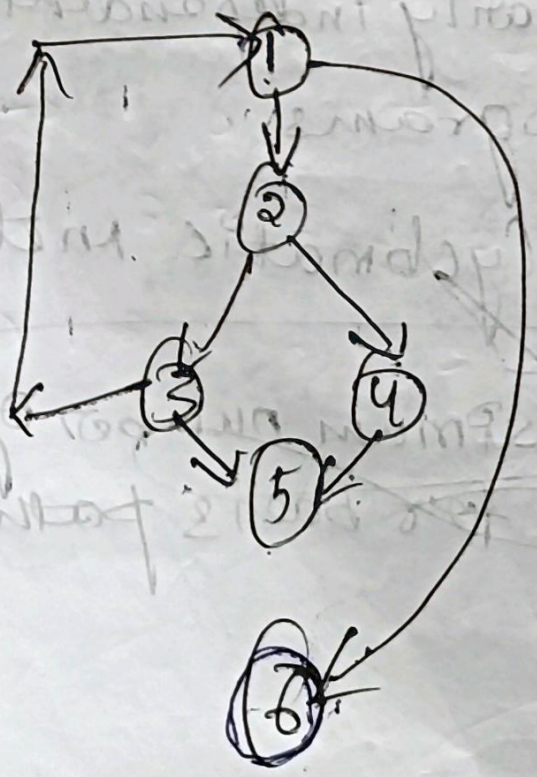
Given a CFG G, Cyclomatic complexity V(G):

(1) $\boxed{V(G) = E - N + 2}$ *

N - No. of nodes in G

E - No. of edges in G

Example CFG Fig (1)




$V(G) = 7 - 6 + 2$

$V(G) = 1 + 2$

$\boxed{V(G) = 3}$

Scanned by Scanner Go

→ another way of computing

(2)

Cyclomatic complexity:

- inspect CFG
- determine number of bounded areas in the graph

$$\boxed{V(G) = \text{Total number of bounded areas} + 1}$$

→ Any region enclosed by a nodes & edge sequence.

from fig (1):

No. of bounded areas is 2

→ $\boxed{V(G) = 2 + 1 = 3}$ *

→ McCabe's metric provides:

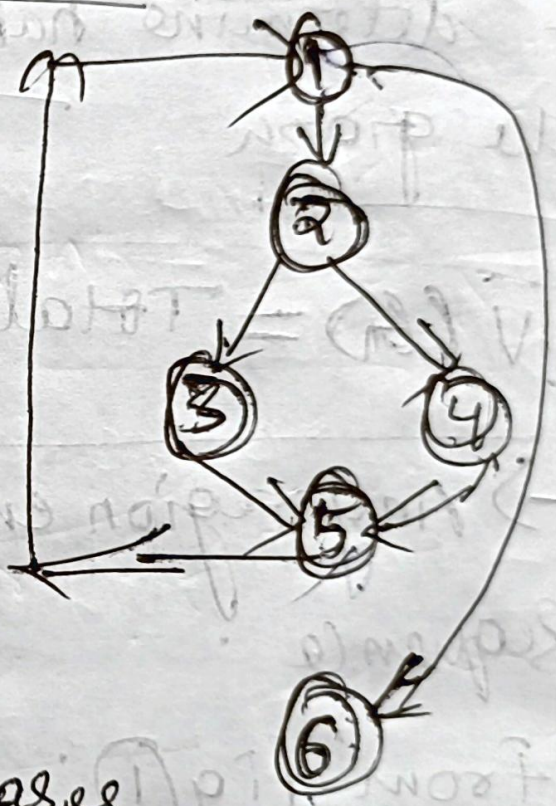
- A quantitative measure of testing difficulty & the reliability.

Scanned by Scanner Go

# Test case Derivation

Eg CFG

```
int f1(int x, int y) {
while (x != y) {
if (x>y) then
    x = x-y;
    else y = y-x;
}
return x; }.
```




## Derivation of test cases

No. of independent path : 3

→ 1, 6 test case (x=1, y=1)

→ 1, 2, 3, 5, 1, 6 test case (x=1, y=2)

→ 1, 2, 4, 5, 1, 6 test case (x=2, y=1)

Scanned by Scanner Go

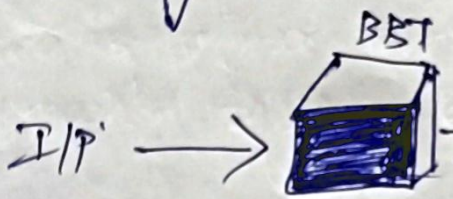
# Black Box Testing

→ It is a method of s/w testing that examines the functionality of an application based on the specifications.

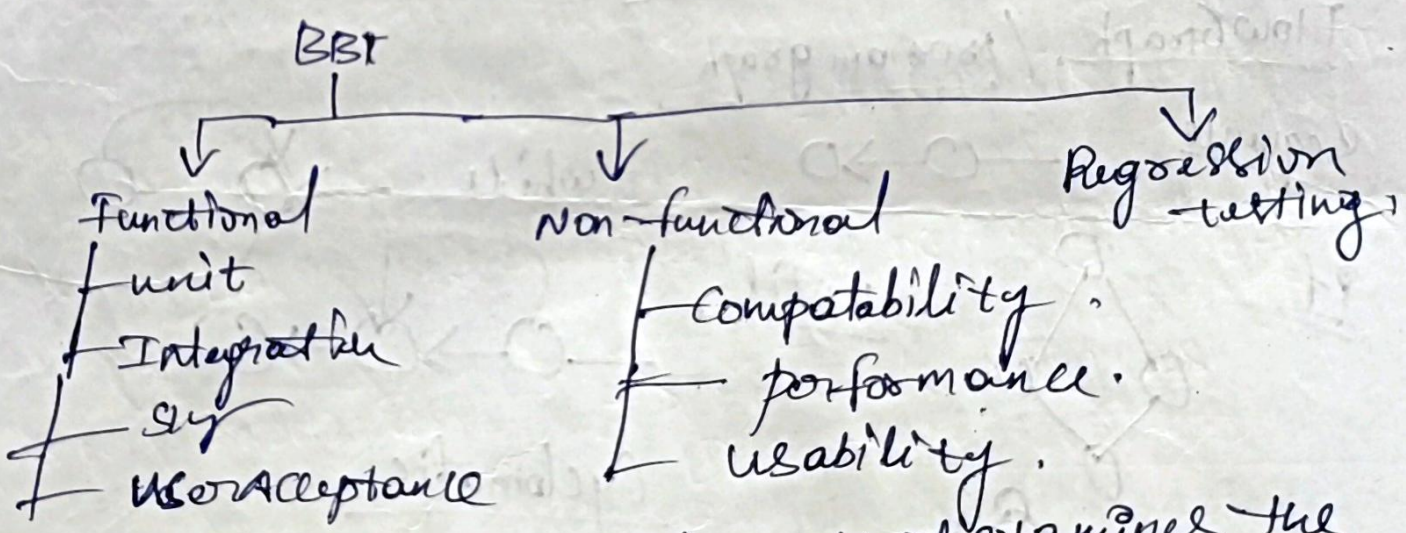
→ It is also known as specifications based testing.

→ Independent testing team usually performs this type of testing during the s/w testing life cycle.

→ The primary source of BBT is a specification of requirements that is stated by the customer.



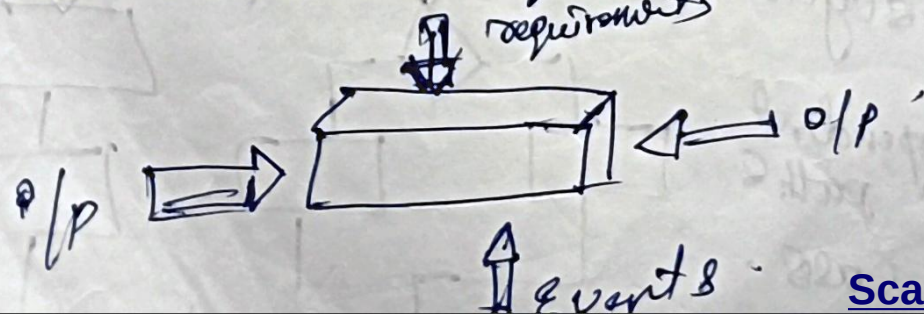

⇒ completely focuses on functional requirements of the s/w




→ It is a method of s/w testing that examines the functionality of an application without looking into its internal structure s(or) workings.

→ This method is applied to every level of s/y testing (i.e functional testing).

→ Tests are based on requirements & functionality




Scanned by Scanner Go

→ It is carried out to test functionality of the program (or) also called as "Behavioral testing".

⇒ The tester in this case has a set of i/p values and desired results. On providing i/p, if the o/p matches with the desired results, the program is tested o/k.

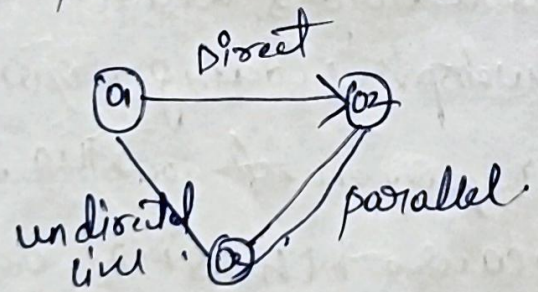
⇒ In this testing method, the design & structure of code are not known to tester and testing engineers and end users conduct the test on s/w.

⇒ It will mainly concentrate on i/p & o/p.

**Black Box Testing Techniques**

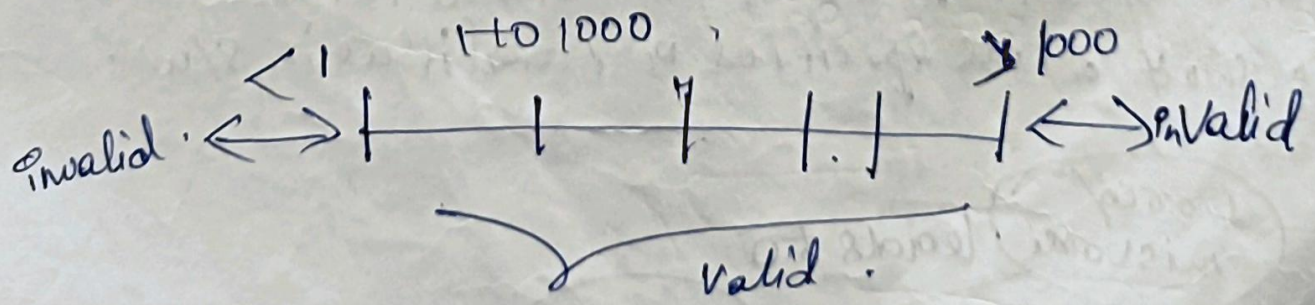
1) Graph Based testing

⇒ All objects (↳ nodes) have the expected relationship to one another.




2) Equivalence partition

⇒ It divides the input domain into various




What kind of errors will we find (or) tested

→ missing functions (or) incorrect functions
→ Interface errors.
→ errors in DS.
→ performance errors

Scanned by Scanner Go

3) Boundary Value Analysis (BVA).

→ It is used to check the errors at the boundaries of an i/p domain.

→ The name comes from the boundary, which means the limits of an area.

→ So, BVA mainly focuses on testing both valid and invalid input parameters for a given range of a S/W component.

Scanned by Scanner Go

## Black Box Testing

→ It is used to test the s/w without knowing the internal structure of code or program.

→ This type of testing is carried out by testers.

→ Implementation knowledge is not required to carry out BBT

→ External, functional.

→ programming knowledge is not required.

→ It is the behavior testing of the s/w

→ It is the least time consuming.

→ It is also known as
1. Data-Driven testing
2. Functional testing
3. Closed box testing.

→ Not considered for algorithm testing.

Fig.

## White Box testing

→ WBT is the s/w testing method in which internal structure is being known to tester who is going to test the s/w.

→ This type of testing is carried by s/w developers.

→ Implementation knowledge is required to carry out WBT.

→ Internal, structural.

→ programming knowledge is required.

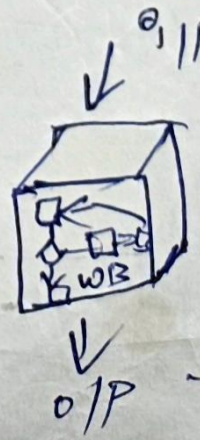
→ It is the logic testing of the s/w.

→ It is most time consuming.

→ It is also known as
1. Clear box testing
2. Code based "
3. Structural "
4. transparent ".

→ It is well suitable for algorithm testing.

Fig:




Scanned by Scanner Go

→ It can be called as high-level testing. (Sly & acceptance testing)

→ Testing reports ⇒ detects on features and performance

→ It can also called low-level testing (Unit & Integration testing)

→ Testing reports ⇒ Both product & code detects

Scanned by Scanner Go

function being tested.

## II) Integration Testing

→ Components are integrated together step by step to form a complete s/w

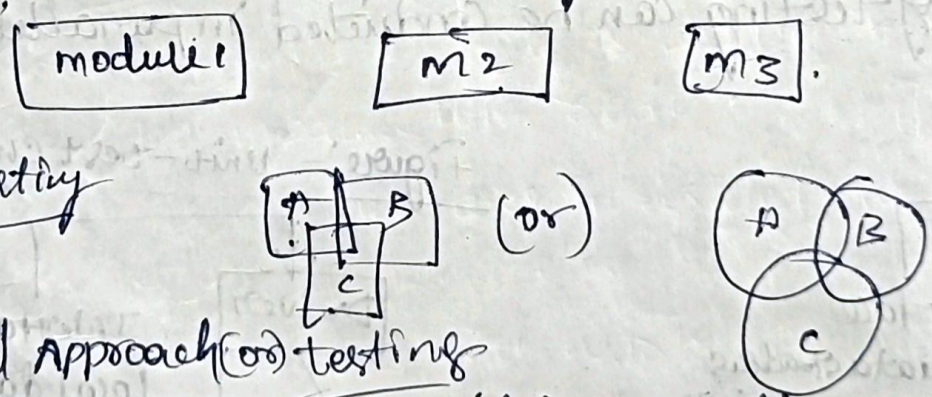
→ It is the second level of the s/w testing process comes after unit testing. → purpose of It is to find module Interfacing errors.

→ In this testing, units (or) individual components of the s/w are tested in a group.

→ It is a process of testing the interface b/w two s/w units (or) modules.

→ The focus of the integration testing level is to expose defects at the time of interaction b/w integrated components (or) units.

Unit testing.




### * Incremental Approach (or) testing

→ In this, modules are added in ascending order one by one (or) according to need. → The selected modules must be logically related.

→ Generally two (or) more than two modules are added & tested to determine the correctness of functions.

→ Incremental testing approach is classified into two ~~parts~~ method

1. Top-Down approach.
2. Bottom-up approach.

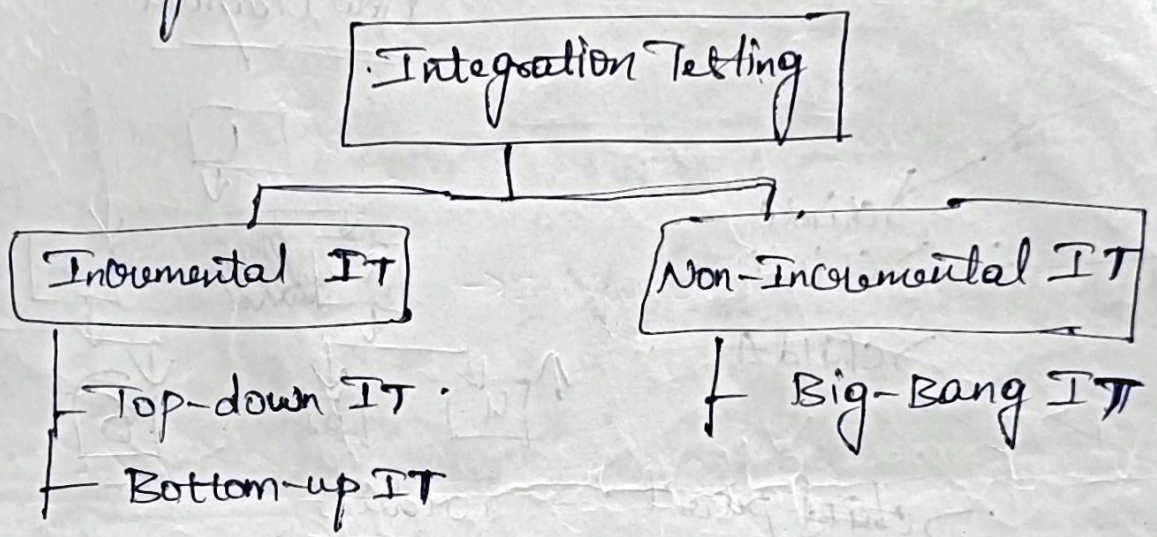
Scanned by Scanner Go

## Non-Incremental Integration testing

We will go for this method, when the data flow is very complex and when it is difficult to find who is a parent and who is a child.

→ In such case, we will create the data in any module bang on all other existing modules and check if the data is present.

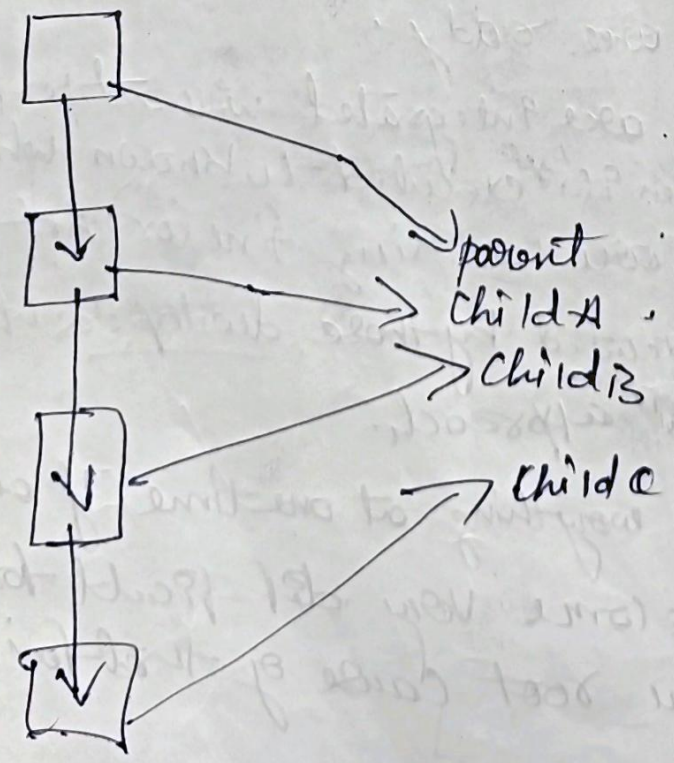
→

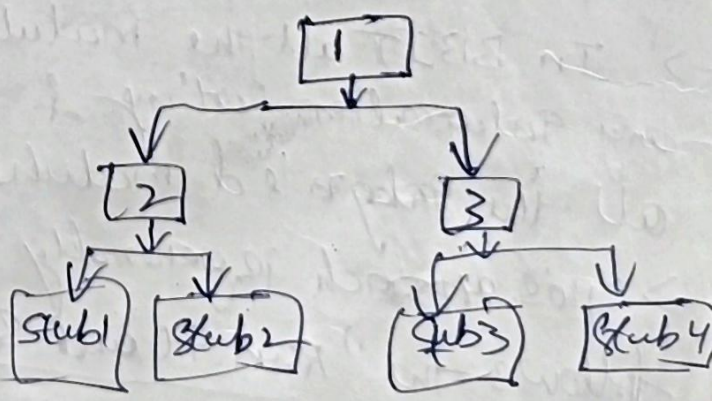



### 1) Top-down Integration Testing

→ It is the process in which ~~testing~~ higher level modules are tested with lower level modules until the successful completion of testing of all the modules.

Flow Diagram




Order of integration will be:

1, 2
1, 3
2, stub1
2, stub2
3, stub3
3, stub4

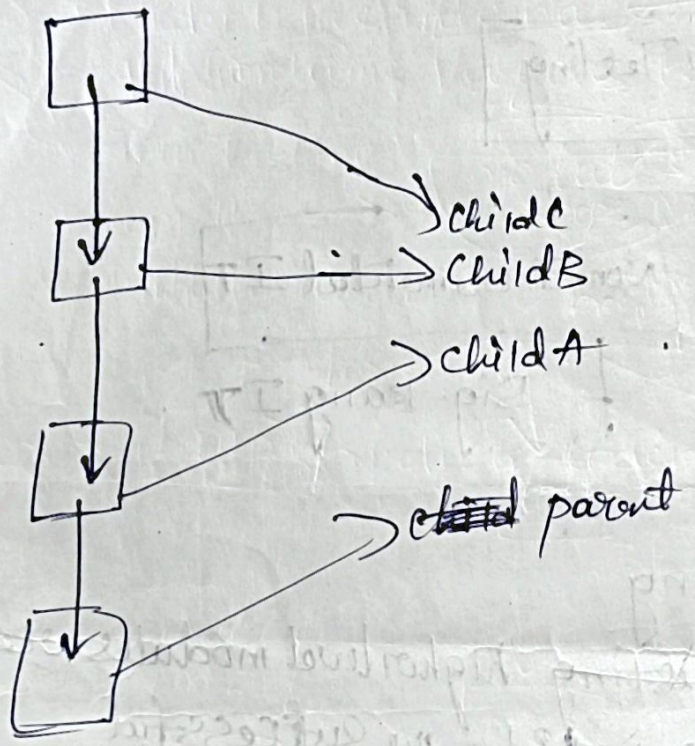
Scanned by Scanner Go

2) Bottom-up Integration testing.

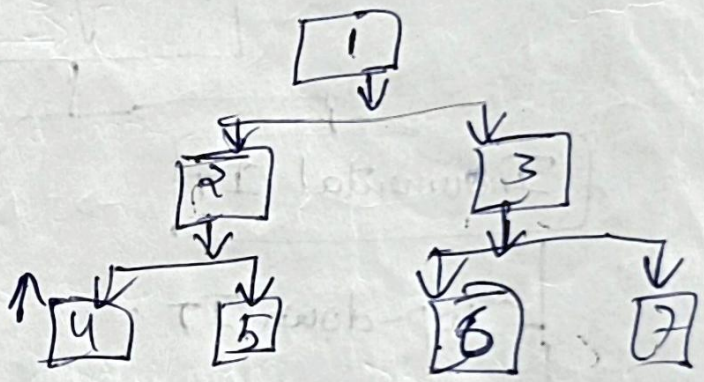
→ It deals with the process in which lower level modules are tested with higher level modules until the successful completion of testing of all the modules.

(or)

Adding modules from bottom to the top.




Flow Diagram




order:-

4, 2
5, 2
6, 3
7, 3
2, 1
3, 1

3) Big-Bang Integration Testing

→ In this approach individual modules are not integrated until and unless all the modules are ready.

→ In BBIT all the modules are integrated without performing any integration testing & then it's executed to known whether all the integrated modules are working fine (or) not

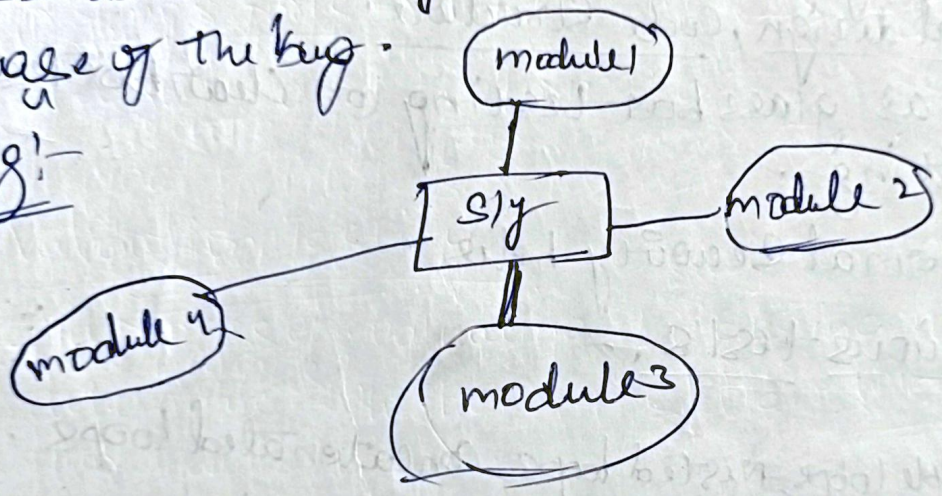
→ This approach is generally executed by those developers who follows the "Run it and see" approach.

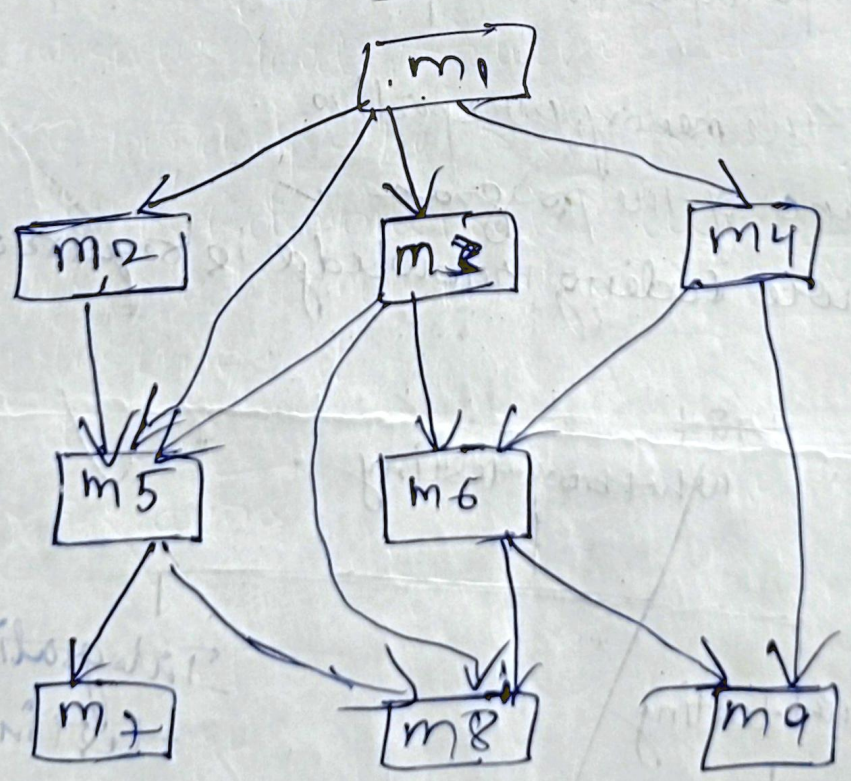
→ Because of integrating everything at one time if any failures occurs then it become very difficult for the programmers to know the root cause of that failure.

Scanned by Scanner Go

→ In case any bug arises then the developer has to detach the integrated modules in order to find the actual case of the bug.

fig:-




Scanned by Scanner Go

## Validation Testing

-> The process of evaluating s/w during the development process to determine whether it satisfies specific business requirements.

-> It ensures that product actually meets the client's need

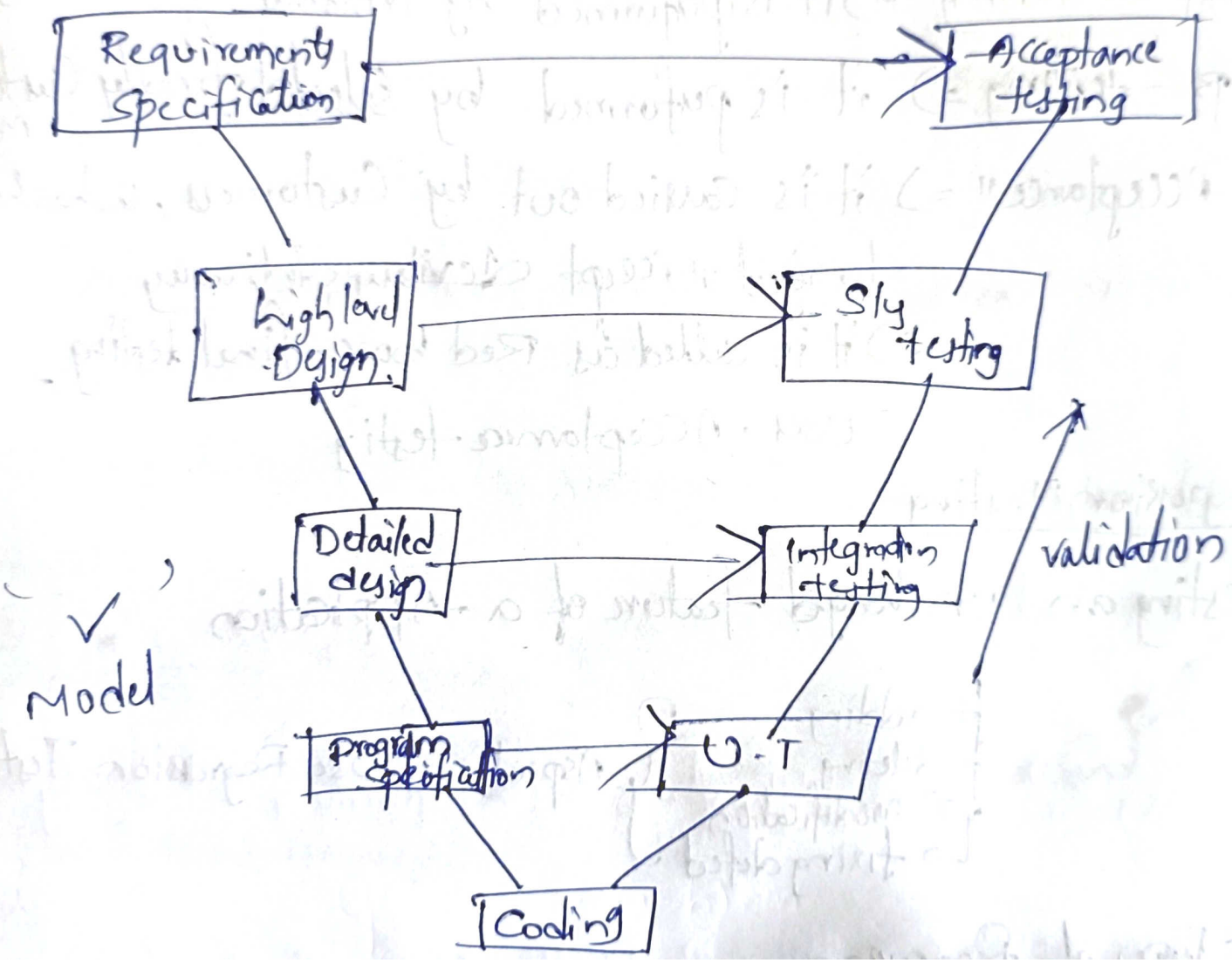
-> "Are we building the right product?"

-> It can be best demonstrated using V-model.

-> The s/w / product under test is evaluated during this type of testing.

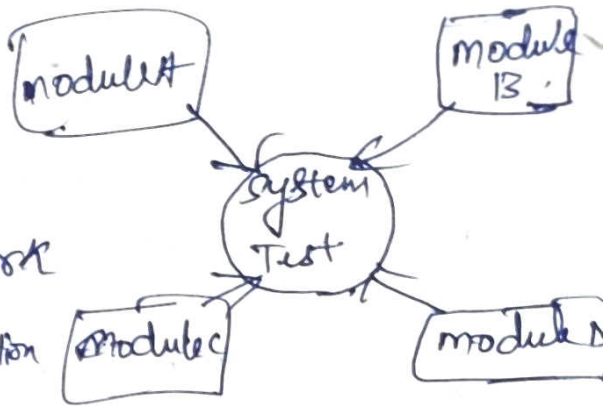
Scanned by Scanner Go

## validation Testing Figure




Scanned by Scanner Go

## System Testing




→ The system as a whole is tested to uncover requirement errors.

→ Verifies that all sys's elements work properly and that overall sys's function & performance has been achieved

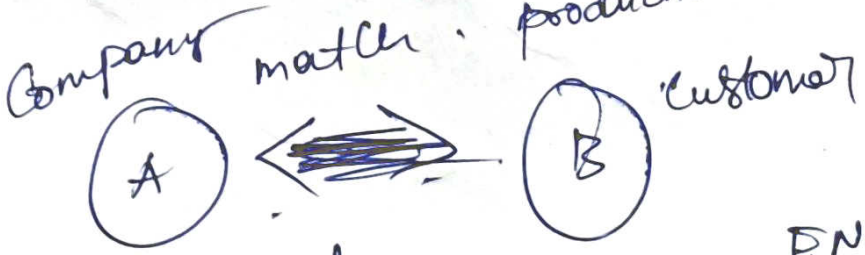
### Types

1) Alpha testing :- It is carried out by the test team within the developing organization.

2) Beta testing :- It is performed by a selected group of friendly customers.

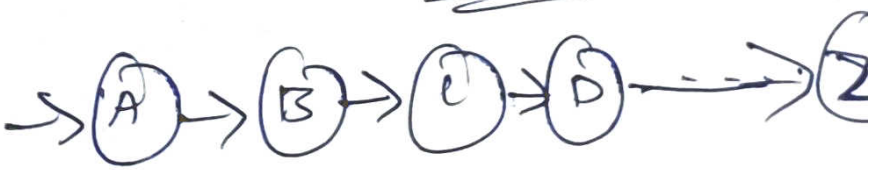
3) Acceptance testing :- It is performed by customer to determine whether to accept (or) reject the delivery of sys's

4) performance testing :- It is carried out to check whether the systems meets the non functional requirements identified in the SRS document.




QA Environment Testing

END to END




Scanned by Scanner Go

- sly testing is a level of testing that validates the complete fully integrated s/w product.
- The purpose of sly testing is to ~~[illegible]~~ evaluate the end-to-end sly specification ⇒ navigate through all features
- Sly testing is a black-box testing.

Sly testing.

- Based on who is doing testing.
  - Alpha.
  - Beta.
  - Acceptance.
- performance / Nonfunctional
  - Volume
  - load.
  - Stress.
  - Security.
  - configuration
  - compatibility
  - Recovery.
  - Install.

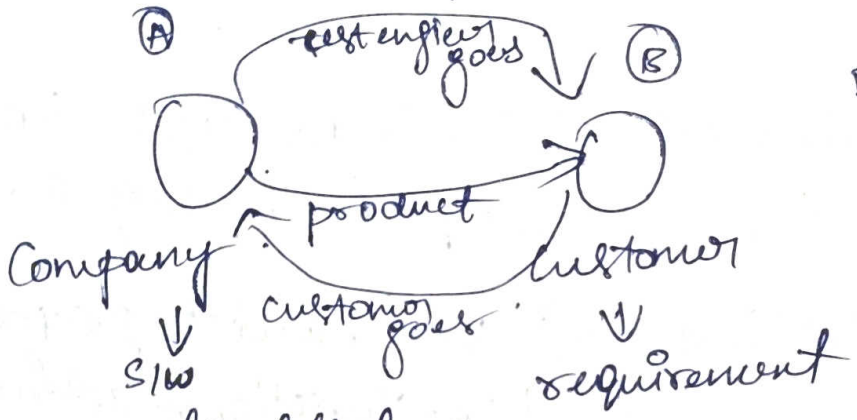
~~Test case :- It is a sdg actions performed on a sly to determine if it satisfies s/w requirements and functions correctly.~~

Scanned by Scanner Go

## * Acceptance Testing

→ It is a method of S/w testing where a s/w is tested for acceptability.

⇒ It determines whether the S/w is conforming specified requirements & user requirements (or) not.




End-to-End testing

⇒ It is allocated as.

1) User Acceptance Testing (UAT)
2) Final Acceptance Testing (FAT)
3) Red Box Testing (last stage of testing) (RBT)

⇒ They test the S/w by real time scenarios

⇒ In this customer direct goes to company with specification sheet.

⇒ Test engineer goes and collect the proper test (or) from the customer.

Scanned by Scanner Go

## Regression Testing :-

⇒ Testing the unchanged features of application to make sure that changes adding a feature, deleting feature, modification & fixing defect.

Unchanged feature

- Adding feature
- Deleted feature
- modification
- fixing Defect.

**3 Types.**

1) unit RT
2) Regional RT
3) Full RT.

1) **Unit RT :-** It is ~~Know~~ nothing but testing only the changes (changed part) (or) modification.

2) **Regional RT :-** Testing the changed part & also impacted part.

3) **Full RT :-** Testing the change feature & remaining part of application.

⇒ Regression Testing done whenever the code is changed, and you need to determine whether the modified code will affect other parts of the s/w application.

Scanned by Scanner Go

When can we perform Regression Testing ?

⇒ We do RT whenever the production code is modified.

1) When new functionality added to the application.
2) When there is a change requirement.
3) When the defect fixed.
4) When there is a performance issue fix.
5) When there is an environment change.

Scanned by Scanner Go

## * Functional Testing

1) It focuses on customer's requirements.
2) Helps to validate the behavior of the application.
3) Objective is to carry out to validate s/w actions.
4) done by manual testing
5) It describes what the product does.
6) Example.

   Check login functionality of
   - Instagram,
   - Facebook.

Types

1) Component (or) unit testing.
2) Integration testing
3) Sly testing.
4) Acceptance.
5) Smoke.
6) Regression
7) Retesting.

## Non-functional testing

1) It focuses on customers expectation.
2) Helps to validate the performance of the application
3) It is done to validate the performance of the s/w.
4) Very hard to perform non-functional testing manually.
5) It describes how the product works.
6) Example

   The dashboard should load in 2 seconds.

Types

1) performance Testing.
2) Volume testing
3) Scalability,
4) Usability testing.
5) portability "
6) Load testing
7) Stress testing.

Scanned by Scanner Go

# * Art of Debugging

## Introduction

⇒ Debugging is the process of fixing a bug in the software.

→ It refers to identifying, analyzing and removing errors.

→ This activity begins after the s/w fails to execute properly and concludes by solving the problem and successfully testing the s/w.

→ It is considered to be an extremely complex and tedious task because errors need to be solved at all stages of debugging.

Tools for debuggers

1) Radare 2
2) WinDbg (windows debugger)
3) Valgrind.

## Debugging process

### Steps.

⇒ Problem identification and report preparation.

⇒ Assigning the report to s/w engineer to the defect to verify that it is genuine.

⇒ Defect analysis using modeling, documentations, finding & testing candidate flaws, etc.

⇒ Defect Resolution by making required changes to the s/y.

⇒ Validation of corrections.

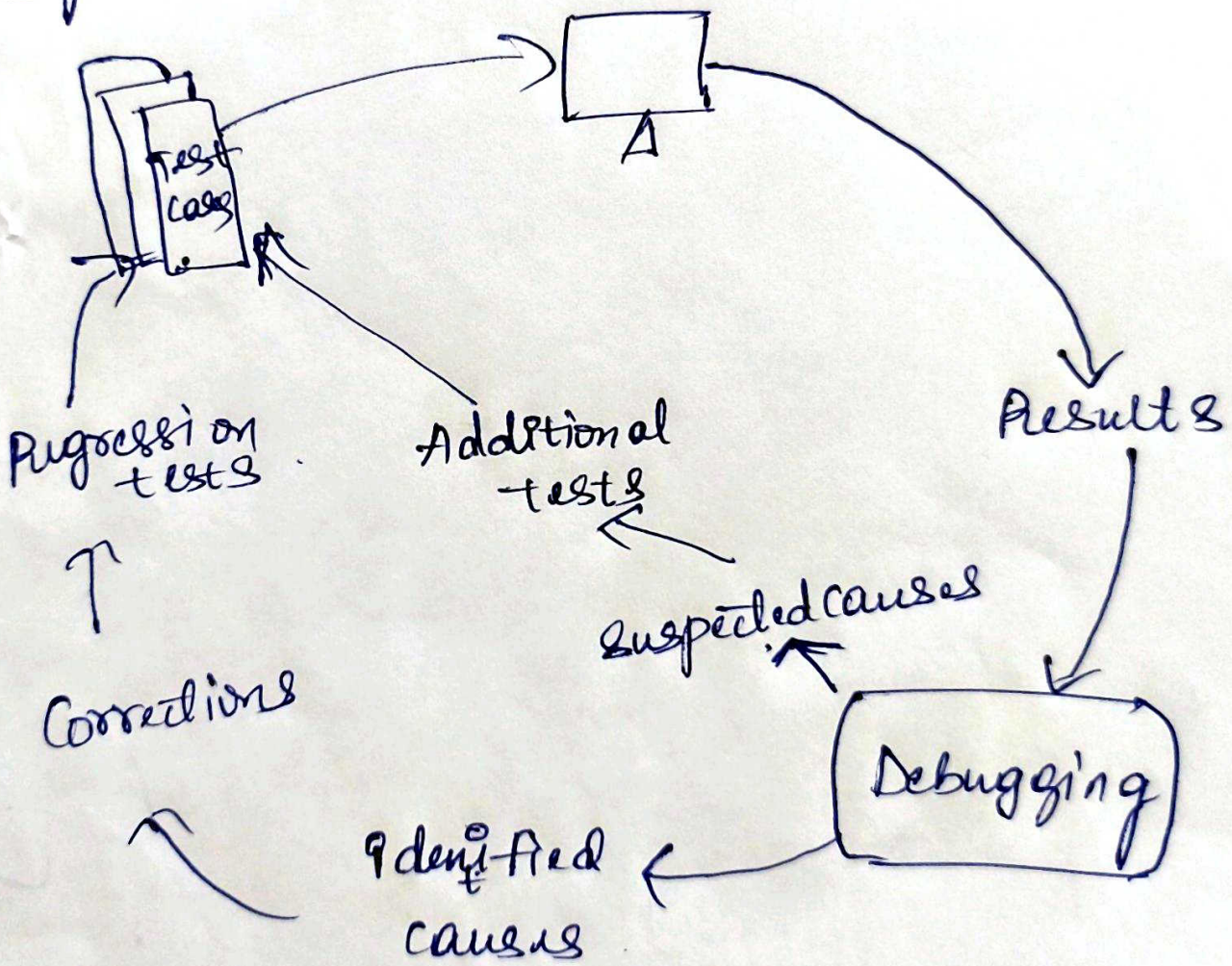
Strategies

1) Study the s/y for the larger duration to understand the s/y.
2) Backward analysis of the problem which involves tracing the program backward from the location of failure.
3) Forward analysis ⇒ ~~that~~ using breakpoints (or) print statements.

Scanned by Scanner Go

## * Art of Debugging

### The Debugging process figure




Scanned by Scanner Go

## * Software metrics

→ A S/w metric is a measure of S/w characteristics which are measurable (or) countable.

→ S/w metrics are important for measuring.

⇒ S/w performance.

⇒ planning work item

⇒ measuring productivity.

⇒ debugging.

⇒ estimating cost.

### Characteristics of S/w metrics

→ Simple & ~~Compatable~~ ~~Compatible~~ Computable

→ Consistent & unambiguous.

→ use consistent units of measurement

→ Independent of programming lang.

* product metrics :- These are the measures of various characteristics of the S/w product.

→ 2 important S/w characteristics.

1) size and complexity of S/w.

2) Quality & reliability of S/w

* process metrics :- These are the measures of various characteristics of the S/w development process. They are used to measure the characteristics of methods, techniques, & tools that are used for developing S/w.

Scanned by Scanner Go

## * Metrics for Software Quality

→ The goal of SE is to produce a high-quality s/w, application, (or) product with in a time frame that satisfies a market need.

→ To achieve this, you must apply effective methods coupled with modern tools within the context of a mature s/w process.

- Measuring Quality :—

There are many measures of s/w quality.

① Correctness is the degree to which the s/w performs its required function.

→ It is measured in terms of defects per KLOC

→ Defects are those problems reported by a user of a program after the program has released for general use.

→ For quality assessment purposes, defects are counted over a standard period of time, typically one year.

② Maintainability consumes more effort.

→ maintainability is all about to correct an error, if encountered, adapt any change in environment.

→ There is no way to measure maintainability directly.

→ It can be measured by a simple time-oriented metric i.e. mean-time-to-change (MTTC)

Scanned by Scanner Go

MTTC maintain the project as follows :-

(a) the time it takes to analyze the change request.
(b) design an appropriate modification,
(c) implement the change,
(d) test it, &
(e) Distribute the change to all users.

(3) Integrity This attribute measures a s/w's ability to withstand attacks to its security.

→ It can be measured in terms of threat & security!

Threat :- is the probability that an attack of a specific type will occur within a give time.

Security :- The integrity of a s/w can then be defined as:

$$\text{Integrity} = \Sigma\left[1-(\text{threat} \times (1-\text{security}))\right]$$

(4) Usability :- If a program is not easy to use, it is often doomed to failure, even if the functions that it performs are valuable.

→ usability is an attempt to quantify ease of use.

— Defect Removal Efficiency (DRE)

A quality metric that provides benefit at both the project & process level is DRE.

DRE is defined as ⇒ $\boxed{DRE = \frac{E}{E+D}}$

E = no. of errors found before delivery of s/w

D = no. of defects found after delivery.

Scanned by Scanner Go

* Various metrics formulated for products in the development process are listed below.

① Metrics for analysis model :- These address various aspects of the analysis model such as s/w functionality s/w size, & so on.

② Metrics for design model :- These allow s/w engineers to assess the quality of design & include architecture design metrics, component-level design metrics, & so on

③ Metrics for source code :- These assess source code complexity, maintainability, & other characteristics.

④ Metrics for testing :- These help to design efficient & effective test cases & also evaluate the effectiveness of testing.

⑤ Metrics for maintenance :- These assess the stability of the s/w product.

* ~~Metrics~~

Scanned by Scanner Go

# I) Metrics for Requirements model (or) ~~Functional~~ Analysis model

→ The technical work in SE begins with the creation of the requirements model.

→ It is the first stage of S/w development.

→ It is the stage that requirements are derived (extension) and a foundation for design is established.

→ These metrics examine the requirements model with the intent of predicting the "size" of the resultant system.

→ Size is sometimes an indicator of design complexity and is almost always an indicator of increased coding, integration, & testing effort.

→ It is useful in estimating the project.

## * Function-Based metrics (or) function point (FP) metric

→ Function point metric can be used effectively as a means for measuring the functionality delivered by a system.

~~Five function units~~ :-

Function point metric can be used to :

1) estimate the cost (or) effort required to design, code and test the software 2) predict the number of errors that will be encountered during testing.

3) Forecast (estimate) the number of components & the number of projected source lines in the implemented s/y.

5 Function points

1) Number of external inputs (EIs)
2) Number of external outputs (EOs)
3) Number of external inquiries (EQs)
4) Number of internal logical files (ILFs)
5) Number of external interface files (EIFs)

Scanned by Scanner Go

Number of Internal logic files (ILF) :- The control information (or) logically related data that is present with'in the system.

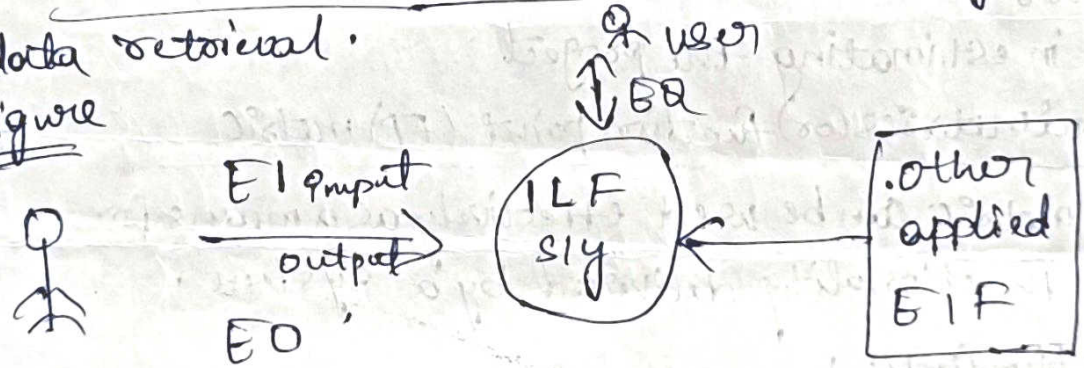
2) External Interface file (EIF) :- The control data referenced by the s/y but present in another system

3) External Inputs (EI) :- Data / control information that comes from outside of our system.

4) External output (EO) :- Data that goes out of the s/y after generation.

5) External inquiries (EQ) :- Combination of i/p & out resulting data retrieval.

Figure




To compute fp, the following relationship is used:

$$FP = \text{Count total} \times [0.65 + 0.01 \times TF]$$

$TF = 1 + 0.14$

Ex

| Information Domain Value | Count | | Weighting factor Simple | Avg | Complex | |
|---|---|---|---|---|---|---|
| EIs | 3 | × | 3 | 4 | 6 | = 9 |
| EOs | 2 | × | 4 | 5 | 7 | = 8 |
| ILFs | 2 | × | 3 | 4 | 6 | = 6 |
| EIFs | 1 | × | 7 | 10 | 15 | = 7 |
| Count total | 4 | × | 5 | 7 | 10 | = 20 |

Scanned by Scanner Go

$FP = UFP \times CAF$

unctional point

Unadjusted functional point

Complexity adjustment factor.

## II) Metrics for design model

### Architectural design metrics

→ It focus on characteristics of the program architecture & effectiveness of ~~the program~~ modules (or) components within the architecture.

→ Card and glass defines three s/w design, complexity measures.

i) Structural complexity.

ii) Data "

iii) S/y "

### i) Structural complexity :-

→ Structural complexity of module is defined in manner as

$$S(i) = f^2_{out}(i)$$

↳ invoked by the module i.

### ii) Data Complexity :-

It provides an indication of the complexity in the internal interface for a module i and is defined as

$$D[i] = \frac{V[i]}{f_{out}(i)+1}$$

Scanned by Scanner Go

V[i] = No. of i/p/o/p variable & then are passed to & from module i.

i) System Complexity :- It is defined as the sum of the structural & data complexity.

$$C[i] = S[i] + D[i]$$

$S[i]$ → structural complexity

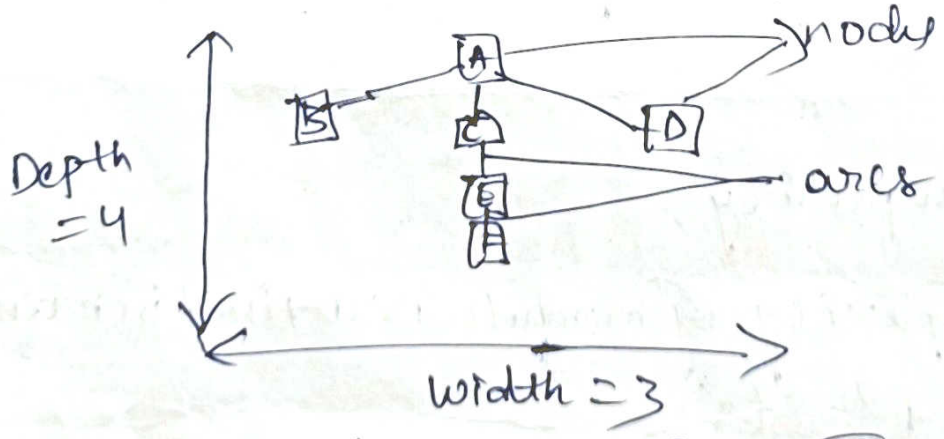
$D[i]$ → data complexity.

Fenton suggest a number of simple morphology (shape) metrics that enable different program architecture to be compared using a set of straight forward dimensions

$$\text{size} = n + a.$$

$n$ → no. of nodes

$a$ → no. of arcs.




size = 6 + 5 = 11

edge to node ratio $= \frac{a}{n} = \frac{5}{6} = 0.83$.

$$\therefore DSI(i) = \sum W_i D_i \qquad \boxed{i = 1 \text{ to } 6}$$

DSI → Design Structure quality index

$W_i$ → weight of each intermediate node.

Scanned by Scanner Go

## III) Metrics for Source Code :-

> This is done after the completion of writing of the required code, or after the completion of designing phase of the s/w projects.

length $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$

program volume $V = N \log_2 (n_1 + n_2)$

Volume ration $L = 2/n_1 * n_2 / N_2$

$n_1$ = no. of distinct operators in a program

$n_2$ = no. of distinct operands in a program.

$N_1$ = total no. of operator ~~occurance~~ occured.

$N_2$ = total no of operands occured.

## IV) Metrics for Testing :-

The metric is derived by considering the implementation of process of testing.

Halsted metrics applied to testing.

$$PL = \frac{1}{\frac{O_1}{2} \times \frac{OC_2}{O_2}}$$

↙ program level

$$e = \frac{V}{PL}$$

↙ (halstead effort)

$O_1$ = no. of operators available in code.

$OC_2$ = sum of occurrence of operators

$O_2$ = no. of independent operands available in the code

$PL$ = program level

Scanned by Scanner Go

V = volume / no. of bits accomodating the program.

e = halstead effort.

I) metrics for maintenance

With regards to metrics for maintenance, IEEE came out with a value referred as s/w maturity index, normally known as ~~sum~~ SMI

$$SMI = [M_t - (F_c + F_a + F_d) / M_t]$$

$M_t$ = No. of units/modules in the current release.

$F_c$ = No. of modules in the current ~~values~~ release that have been changed.

$F_a$ = No. of modules in the current release that have been added.

$F_d$ = No. of modules that are deleted / unavailable in current release.

Scanned by Scanner Go

# Risk Analysis & Risk Management UNIT-5



**Risk:-** A risk is a potential problem. It's an activity (or) event that may compromise the success of a s/w development project. (or) Risks are uncertain things that may occur in s/w & leads to heavy losses.

**Risk Analysis:-** Risk Analysis in project management is a sequence of processes to identify the factors that may affect a project's success.

→ These processes include risk identification, analysis of risks, risks management and control, etc.

→ Proper risk analysis helps to control possible future events that may harm the overall project.

**Risk management:-** process of making decisions so that risks in the s/w can be reduced.

→ Risk is an uncertain event that may have +ve (or) -ve impact on project.

→ RM is the process of identifying & migrating risk.

**Why is it important**

Risk affects all aspects of your project, your budget, your schedule, your scope.

**risk in project management**

⇒ plan risk management
⇒ Identify Risk
⇒ Qualitative risk analysis.
⇒ Quantitative risk analysis.
⇒ plan risk response
⇒ monitor & control risk.



Scanned by Scanner Go

## Reactive vs proactive Risk Strategies

-> To manage risks we have two Strategies

### 1) Reactive Risk Strategy :-

⇒ Corrective measures are taken only when the S/W enters into risk.

(or)

⇒ Reactive risk management tries to reduce the damage of potential threats & speed an organization's recovery from them, but assumes that those threats will happen eventually

-> No preventive care

-> Corrected only after occurence

-> Older risk management approach

-> If they are not solved, the project is in danger.

### 2) Proactive Risk Strategy :-

⇒ proactive Risk management identifies threats and aims to prevent those events from ever happening in the first place.

-> Begins before the risk occurs

-> First identify the risks then we assess the impact of these risks on S/W then risks are prioritised.

-> High priority risks are managed first

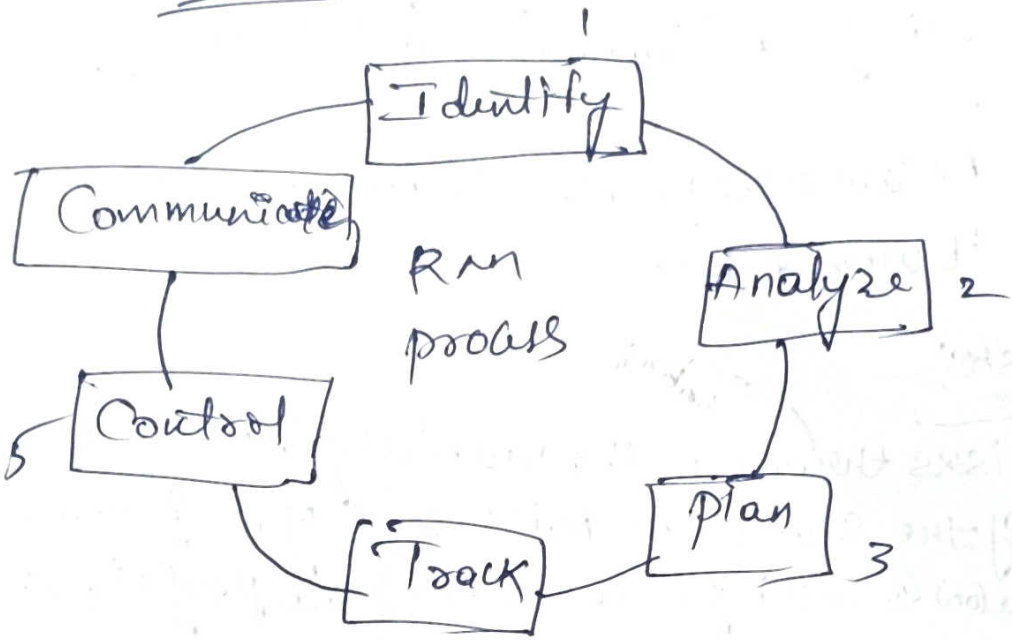
* Types of S/W Risks

1) project Risk - Scheduling, Budget, Staffing, ~~Interface~~ -> module
2) Technical Risk - Implementation, Interface - module.
3) Business Risk - market Risk, Strategic Risk, management risk, Sales risk, Budget risk
4) Known Risk
5) predictable Risk
6) Unpredictable Risk

MR - no customers
SR - not following comp.
MR - when Senior manager want
SR - how to sell
BR - loss in budget

Scanned by Scanner Go

Figure :- Risk mng




* Software Risks Types :- 4

1) Project Risks :-

→ Project Risks are threaten the project plan.

→ That is, if project risks become real, it is likely that the project schedule will slip and that costs will increase.

→ project risks identify potential budgetary, schedule, personnel (staffing & organization), resource, stakeholder, and requirements problems and their impact on a s/w project.

→ project complexity, size, and the degree of structural uncertainty were also defined as project risk factors.

2) Technical Risks :-

→ Technical Risks threaten the quality & time lines of the s/w to be produced.

→ Technical risks identify potential design, implementation, interface, verification and maintenance problems.

→ If a technical risks becomes a reality, implementation may become difficult (or) impossible

Scanned by Scanner Go

→ In addition, specification, ambiguity (doubtfulness (or) unclear), technical uncertainty, & "leading-edge" technology are also risk factors.

→ It occurs because the problem is harder to solve than you thought it would be.

3) Business risks:-

→ Business risks threaten the viability (ability to work sucessfully) of the s/w to be built and often jeopardize (put some one (or) some thing into a situation) the project (or) the product.

Candidates for the top 5 Business Risks are:-

a) Market risks:- Building an excellent product (or) system that no one really wants.

b) Strategic risks:- Building a product that no longer fits into business strategy for the company.

c) Sales risks:- Building a product that the sales force doesn't understand how to sell.

d) management risks:- losing the support of senior management due to change in focus (or) a change in people.

e) Budget risks:- losing budgetary (or) personnel commitment.

4) Known risks:- Known risks are those that can be uncovered after careful evaluation of the project plan, Eg:- unrealistic delivery date, lack of documented requirements (or) s/w scope, poor development environment.

Scanned by Scanner Go

) predictable risks :- PR are extrapolated (extended) from past project experience.

eg:- Staff turnover, poor comm. with the customer, dilution of staff effort as ongoing maintenance requests are serviced).

(b) Unpredictable risks :- They can & do occur, but they are extremely difficult to identify in advance.

* ICS - 1 unit
DMPA - 2 or 3
AI - 1
DM - 1

I) Risk Identification

→ It is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.)

→ we identify the risks and list them based on 2 approaches :-

⇒ Generic risks are a potential threat to every s/w project.

⇒ product-specific risks can be identified only by those with a clear understanding of the technology, the people, & the environment that is specific to the s/w that is to be built.

* Steps in Risk Identification

(1) preparation of risk item check list.

(2) Creating risk components and drivers list.

Scanned by Scanner Go

① Preparation of Risk Item Check list :-

Risk Items can be identified based on below components

* product size :- risks associated with the overall size of the s/w to be built (or) modified.

* Business impact :- risks associated with constraints (or) conditions imposed by management (or) the marketplace.

* Stakeholder characteristics :- risks associated with the sophistication of the stakeholders & the developer's ability to communicate with stakeholders in a timely manner. (Choices of customer)

* process definition :- risks associated with the degree to which the s/w process has been defined & is followed by the development organization. Standards

* Development environment :- risks associated with the availability & quality of the tools to be used to build the product.

* Technology to be built :- risks associated with the complexity of the s/y to be built & the "newness" of the technology that is packaged by the s/y.

* Staff size & experience :- risks associated with the overall technical & project experience of the s/w engineers who will do the work.

Scanned by Scanner Go

Creating risk components & drivers list

⇒ RC & DL are prepared along with their probability of occurrence

(a) Performance risk – the degree of uncertainty that the product will meet its requirements & be fit for its intended use.

(b) Cost risk – the degree of uncertainty that the project budget will be maintained.

(c) Support risk – the degree of uncertainty that the resulting S/W will be easy to correct, adapt & enhance.

(d). Schedule risk – the degree of uncertainty that the ~~product~~ project schedule will be maintained & that the product will be delivered on time.

II) Risk projection

(or)

(Risk estimation)

Risk projection attempts to rate each risk in two ways:-

(1) probability that the risk is real.

(2) Consequences with that risk.

Steps involved

performed ↳ by project planner, technical staff etc.

(1) Building risk table – should have list of all risks, category, probability & impact.

(2) Enlist the consequences of risks – which are associated with each & every risk which are mentioned in risk table.

(3) Estimate the impact of those risks on our project.

(4) maintaining overall accuracy of risk projection (or) estimation.

Scanned by Scanner Go

## III) Risk Refinement

→ process of explaining the risk in detailed way

→ Represented by CTC formats
↳ Condition transition Sequence.

<Condition> then there is concern that <Consequence>

Condition — Subcondition 1
— Subcondition 2
—
Subcondition n

→ If ~~there~~ these condition happens (or) arises then there is concern that ~~con~~ these consequence may happen.

→ Deriving one condition into sub conditions

→ Then we will be assessing the consequences of those sub conditions in the project

⇒ Consequences cannot be eliminated by can be refinement which helps in easy analysis.

Scanned by Scanner Go

## Risk Mitigation, Monitoring, Management (RMMM)

1) Risk Mitigation ⇒ (proactive approach) we use.
↳ Risk Avoidence
⇒ preventing the risks from occurence

1) Communicate with staff & find risks
→ periodically we need talk (or) comm. with the staff & ask them if there are any risks.

2) Eliminate the causes for the risk before project starts.

3) prepare a policy so that project continues even if staff leaves in b/w.

4) Maintain all the relevant documents.

5) Conduct reviews on time.

2) Risk Monitoring ⇒ (project manager)
↳ These will happen after the risk has occur
→ The risk monitoring activies done by project manager & he will monitor the following:

1) Behaviour of team as pressure
⇒ When the pressure increasing what would be the behaviour of team (or) how they are behaving.

2) Spirit of team work

3) Cooperation among team members.

4) problems that may occur

5) Availability of job in and out of organization.

Scanned by Scanner Go

3) Risk management ⇒ (Reactive approach)

→ Done by project manager when risk becomes reality

→ If mitigation is done properly, management becomes easy

Scanned by Scanner Go

## 35.7 The RMMM Plan

A risk management strategy can be included in the software project plan, or the risk management steps can be organized into a separate *risk mitigation, monitoring and management plan*. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan.

Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a *risk information sheet* (RIS) [Wil97]. In most cases, the RIS is maintained using a database system so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily. The format of the RIS is illustrated in Figure 35.4.

Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence. As we have already discussed, risk mitigation is a problem avoidance activity. Risk monitoring is a project tracking activity with three primary objectives: (1) to assess whether predicted risks do, in fact, occur; (2) to ensure that risk aversion steps defined for the risk are being properly applied; and (3) to collect information that can be used for future risk analysis. In many cases, the problems that occur during a project can be traced to more than one risk. Another job of risk monitoring is to attempt to allocate origin (what risk(s) caused which problems throughout the project).

Scanned by Scanner Go

FIGURE 35.4

Risk information sheet

Source: [Wil97].

| Risk information sheet | | | |
|---|---|---|---|
| Risk ID: P02-4-32 | Date: 5/9/09 | Prob: 80% | Impact: high |

**Description:**
Only 70 percent of the software components scheduled for reuse will, in fact, be integrated into the application. The remaining functionality will have to be custom developed.

**Refinement/context:**
Subcondition 1: Certain reusable components were developed by a third party with no knowledge of internal design standards.
Subcondition 2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
Subcondition 3: Certain reusable components have been implemented in a language that is not supported on the target environment.

**Mitigation/monitoring:**
1. Contact third party to determine conformance with design standards.
2. Press for interface standards completion; consider component structure when deciding on interface protocol.
3. Check to determine number of components in subcondition 3 category; check to determine if language support can be acquired.

**Management/contingency plan/trigger:**
*RE* computed to be $20,200. Allocate this amount within project contingency cost. Develop revised schedule assuming that 18 additional components will have to be custom built; allocate staff accordingly.
Trigger: Mitigation steps unproductive as of 7/1/09.

**Current status:**
5/12/09: Mitigation steps initiated.

| Originator: D. Gagne | Assigned: B. Laster |
|---|---|

Scanned by Scanner Go

# UNIT – 5 part – B

## Software Quality Assurance (SQA)

Quality Concepts :– Quality Management

What is Quality?

Quality is a complex and multifaceted concept that can be described from ~~two~~ different points of view.

① Transcendental View argues the quality is something that you immediately recognize, but cannot explicitly define.

② User View sees quality in terms of an end user's specific goals. If a product meets those goals, it exhibits quality.

③ Manufacturer's view defines quality in terms of the original specification of the product.

④ product view suggests that quality can be tied to inherent characteristics of a product.

I. Software Quality

→ It can be defined as an effective s/w process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it.

→ s/w quality is defined as how well the s/w works

→ s/w Quality is of two types

① Quality of Design : characteristics of items used by designer.

Scanned by Scanner Go

② Quality of Conformance:-

Degree to which design specifications are followed in manufacturing the product or S/w

II. Quality control

→ The process in which several activities are conducted to maintain the quality of product.

→ Several activities are in terms of inspection, review & test

→ These activities can be done manually, automatically & semi-automatic.

III. Quality Assurance

→ planned & systematic pattern of activities are done which ~~planned & systematic pattern~~ are needed to provide high degree of reliability in S/w.

IV. Cost of Quality

→ Total cost required to obtain quality and conduct quality related activities

Three types of costs

① prevention cost :- It include

ⓐ - The cost of management activities required to plan & coordinate all quality control & quality assurance activities.

ⓑ - The cost of added technical activities to develop complete requirements & design models.

Scanned by Scanner Go

(c) test planning costs &
(d) - the cost of all training associated with these activities

2) Appraisal cost

-> It include activities to gain insight into product condition the "first-time-through" each process

Examples

(a) cost of conducting technical reviews
(b) Cost of data collection & metrics evaluation.
(c) Cost of testing & debugging.

(3) Failure cost are those that would disappear if no errors appeared before (or) after shipping a product to customers.

-> It is subdivided into
- Internal failure cost
- External failure cost.

* Quality factors

(1) Correctness
(2) Reliability
(3) Efficiency
(4) Integrity
(5) Usability
(6) Maintainability
(7) Flexibility
(8) portability
(9) Reusability
(10) Interoperability.

Scanned by Scanner Go

## * Software Quality Assurance (SQA)

-> The process which ensures developed s/w meets the quality specifications of standardised software.

-> Which is done by s/w engineers group & SQA group.

### Activities of SQA

-> These activities are performed by an independent SQA group.

1. prepares an SQA plan for a project
2. participates in the development of the project's s/w process description.
3. Reviews s/w engineering activities to verify compliance with the defined s/w process.
4. Audits designated s/w work products to verify compliance with those defined as a part of the s/w process.
5. Ensures that deviations in s/w work and work products are documented and handled according to a document procedure.
6. Records any noncompliance & reports to senior management

## * Software Reviews

-> s/w reviews are performed to identify errors & defects that can be removed

-> It also checks parts where
- improvement is needed
- which are already improved.

Scanned by Scanner Go

→ S/w review is an important part of SDLC that assists S/w engineers in validating the quality, functionality, & other vital features & components of the S/w.

→ S/w review is used to verify various ~~S/w~~ documents like requirements, system designs, code, test plans & test cases.

## Types of Software Reviews

① Software peer Review

→ The main purpose of peer review is to provide "a disciplined engineering practice for detecting (or) correcting defects in the s/w artifacts, preventing their leakage into the field operations".

→ It is performed inorder to determine / resolve the defects in the S/w.

Types

ⓐ Code Review: To fix mistakes and to remove vulnerabilities from the S/w product.

ⓑ pair programming: Two programmers work on a single workstation & develop a code together.

ⓒ Informal meeting: Meeting out side the office (Eg:- lunch/dinner).

ⓓ walk through: Here a designer (or) developer had a team of S/w developers to go through a S/w product.

Scanned by Scanner Go

(c) Technical Reviews:

(d) Inspections

(2) S/w management Review

-> The Objective of this type of review is to evaluate the work status.

(3) S/w Audit Review

-> type of external review in which one or more critics, who are not a part of the development team, organize.

-> This is done by managerial level people.

* Formal Technical Reviews (FTR)

-> A FTR is a software quality control activity performed by software engineers.

Objectives of an FTR are:

(1) To uncover errors in function, logic, or implementation for any representation of the S/w

(2) To verify that the S/w under review meets its req's.

(3) To ensure that the S/w has been represented according to predefined standards.

(4) To achieve S/w that is developed in a uniform manner

(5) To make projects more manageable.

Scanned by Scanner Go

4 Steps

1. Review Meeting
2. Review Reporting & Record Keeping
3. Review Guidelines
4. Sample Driven Reviews.

1. Review Meeting

-> Each review meeting should be involved in the review.

* Between 3, 4 & 5 people should be involved in the review.
* Advance preparation should occur but should require no more than two hours of work for each person.
* Duration of the review meeting should be less than two hours.

2. Review Reporting & Record Keeping

-> Done by a reviewer (or) by recorder

-> Reviewer also prepares report

-> When ever review is going on he will record all the issues all errors which are encountered he will prepare a list & final from data list he will generate a report

A review summary report answers three questions:

a. What was reviewed?
b. Who reviewed it?
c. What were the findings and conclusions?

Scanned by Scanner Go

③ Review Guidelines

→ Some review guidelines has to be set & established in advance.

→ Review guidelines are distributed to all reviewers.

→ All the reviewers have to agree, then only we can be implemented.

④ Sample driven Reviews:

→ All s/w work product samples are inspected.

→ Errors prone work products should be identified

→ When we are performing full FTR, should be given more priority.

## * Software Reliability

→ Also called as operational Reliability.

→ It is described as ability of s/y / component to perform its required functions under static conditions for a specific period.

→ It can be measured directly & estimated using historical data.

measures / metrics

① MTBF = MTTF + MTTR

MTBF - Mean Time Between Failure

MTTF - Mean Time TO Failure

MTTR - Mean Time TO Repair.

Scanned by Scanner Go

Metric (2)

$$\text{Availability} = \frac{MTTF}{MTTF + MTTR} \times 100$$

## ✓ ISO 9000 Quality standards

ISO → Internal Standards organisation.

→ It is group of 63 countries which is established to plan standardisation (for Quality).

→ ISO 9000 introduced in 1987 year

→ ISO 9000 determines the guidelines for maintaining the quality of systems.

→ ISO 9000 describes quality assurance elements in generic terms that can be applied to any business regardless of the product (or) services offered.

### Types of ISO 9000 Quality Standards

(1) ISO 9001: An organisations involved in design, development, production and servicing of goods should follow ISO 9001 standards

(2) ISO 9002: The organisations which do not design products, but involved in production.

→ It does not apply to s/w development organisations.

Scanned by Scanner Go

(3) ISO 9003: Organisations that are involved only in testing and installation of products.

⇒ In order to get ISO certification, an organisation should registered.

Registration process steps:—

1. Application submit
2. pre-Assessment
3. Document review & Adequacy of Audit
4. Compliance Audit.
5. Registration
6. Continous Inspections.

Scanned by Scanner Go