Date 10/1/24

Unit - I

complexity - to evaluate the complexity of a C program

Engineering

→ create, invent new things.

*Software Engineering is defined as the systematic approach to the development, operation, maintainance of Software.

⇒ the framework encompasses a process a set of methods, and an array of tools.

Software is the Bundle of Instructions

Encompasses

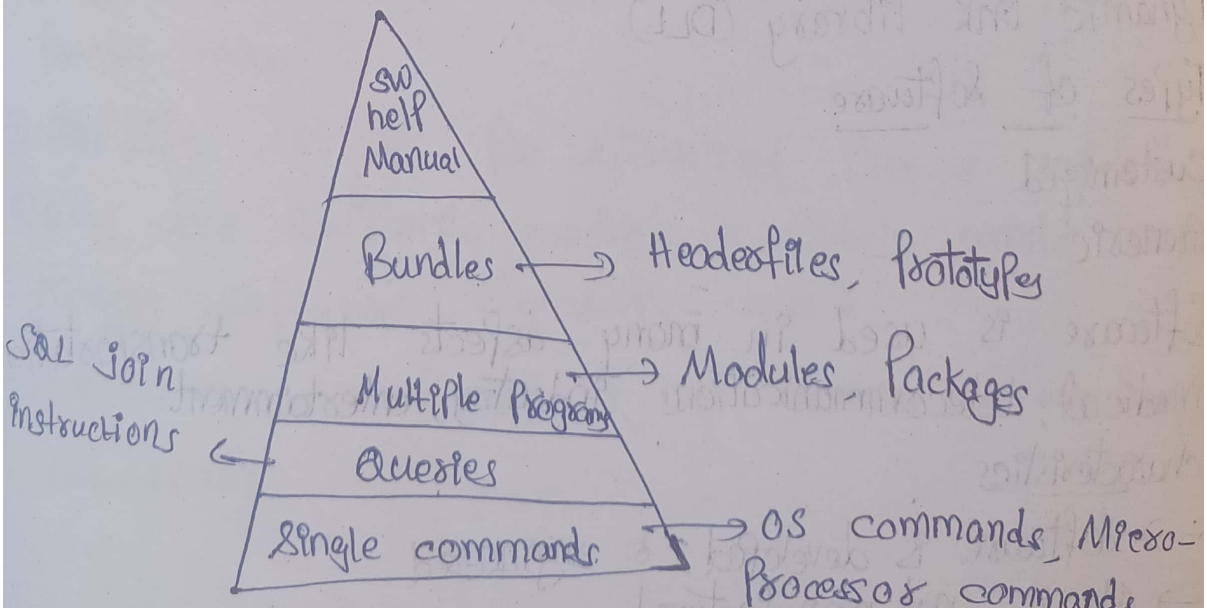
→ Instructions

→ Data Structures

→ Documentation

display data is used by namespace.

components of Software



⇒ Software is information transformer.

Exp:-1

Understanding Study of RUP, Study of Rational Suite

Rational Administrator
Rational Requisite Pro
Rational Rose
Rational Test Manager
Rational Robot
Rational Purify
Rational website scanner
Rational quantify
Rational Purecoverage

Aim

Title

Procedure

Sample I/O

Research.

Date
12/1/24

Dynamic link library (DLL)

Types of Software

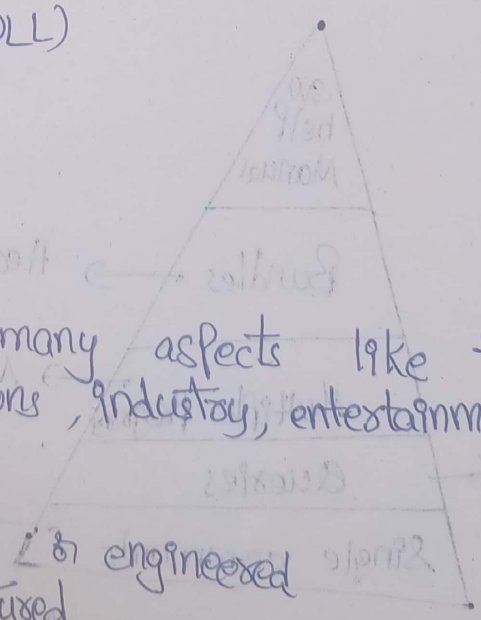
Customized

Generic

Software is used in many aspects like transportation, medical, telecommunications, industry, entertainment.

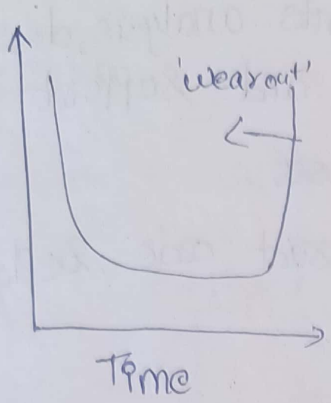
characteristics

→ Software is developed & or engineered
→ It is not manufactured
→ It doesn't wear out.
→ Industry is moving toward component-based construction

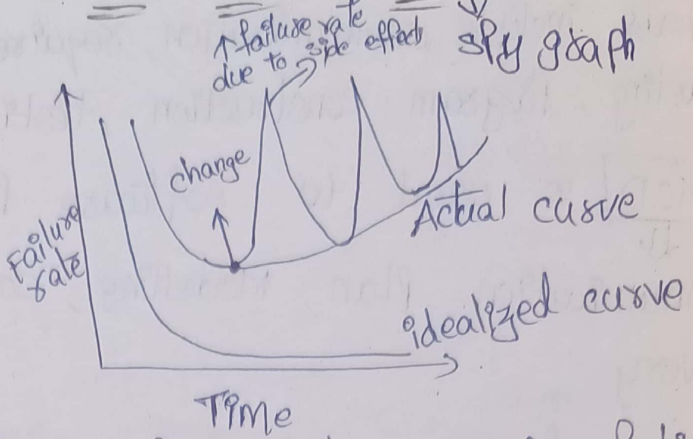


→ Software costs dominate computer system costs.

Differences b/w hardware and software



failure rate as a fn of time hardware (Bath-tub curve).



failure rate as a fn of time for software.

Date 17/1/24

Components of software, how to define SE
Differences between Hardware and Software Engineering
1) what is software process 2) what are generic framework activities that are present in every software process
3) what are process pattern?
4) what are strength and weakness of prescriptive process model?
5) Identify set of ~~pse~~ specialised process models?
6) why does ~~Edi~~gility consider a watch world in modern software Engineering work?
7) what is scrum, spy ~~g~~; in modern software process?
~~In mode~~

Date 19/1/24

Engineering Process

Methods include communication, requirements analysis, design modeling, Program construction, testing and support

CPMCD is used to software process.

↓

Communication, Plan, Modelling, Construct code, Deployment delivery

Essence of Practice

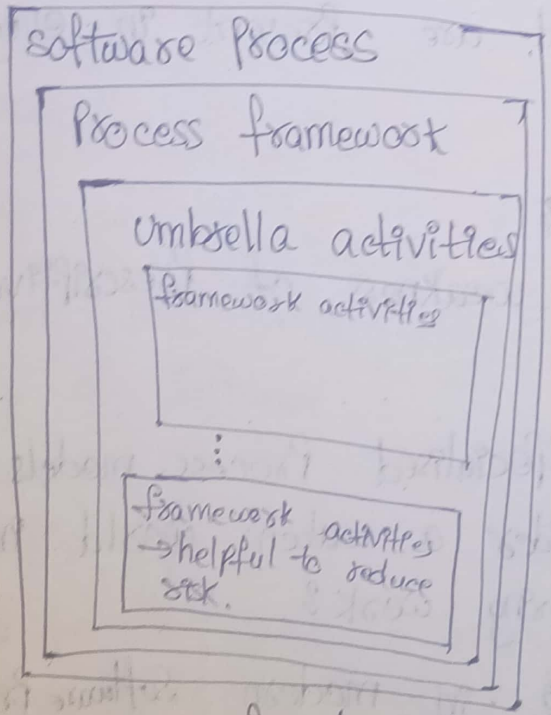
1) Understand the Problem.

CPMCD. Generic Process activities

α-level → engineers

β-level → by Public } construction of software

Software Process Model



Process is of two types

⇒ Linear ⇒ com → Planning → Model → Construct → Deployment

⇒ Iterative ⇒

com → Planning → Model → construct → Deployment

win xp

My computer → right click

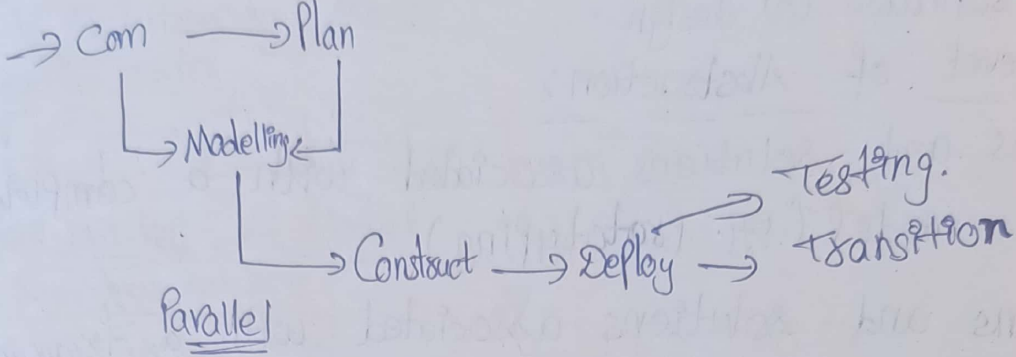
↓

map net work drive

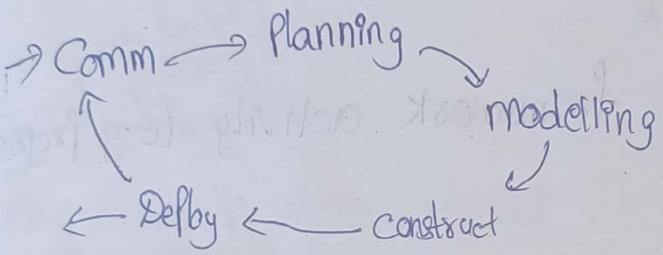
→ \\192.168.72.2\csbstaff

Process flow

1) Evolutionary
2) Parallel



Parallel



Umbrella activities

1) Software Project tracking and control
2) Risk Management
3) Software quality assurance
4) Technical reviews
5) Measurement
6) Software configuration management
7) Reusability management
8) Work Product Preparation and Production

1) layered tech different process flows

Date 20/11/24 Project : IQ Games

Pattern

It is a software (s/w) design which is used to develop another software (s/w) design

Three level of Abstraction :

1) Problems and solutions associated with a complete process model (eg. Prototyping)

2) Problems and solutions associated with a framework activity (e.g. Planning) or

3) an action with a framework activity (e.g. project estimating).

Process Patterns

Stage } specific
Task }

Phase (Phase1, Phase2)

Date 23/11/24

Software ~~Requirement~~ Requirement

Subtitle :

Purpose

Prblm stmt

3) Solution

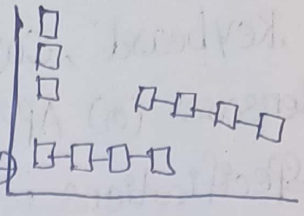
4) Front end, back end, Network Software Specifications & Hardware Specifications

Processor, GPU, CPU, VGA, Mouse, Keyboard, Camera, Scanner, Printer, if any Sensors are applicable.

Date 23/1/24

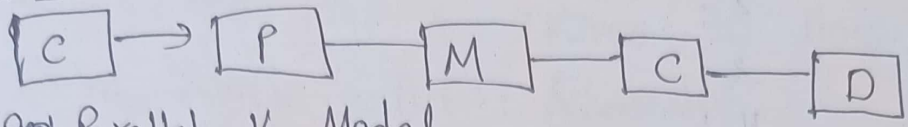
Prescriptive Process Models

1) Linear Waterfall model
2) Parallel V-Model
3) Iterative Incremental model

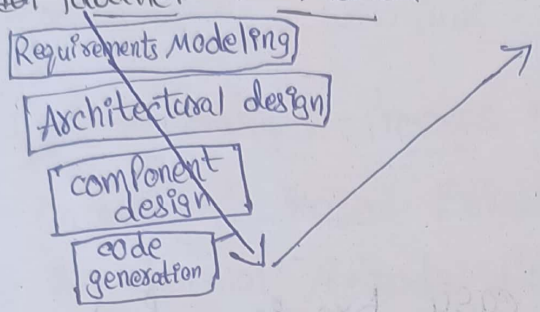


4) Evolutionary Model

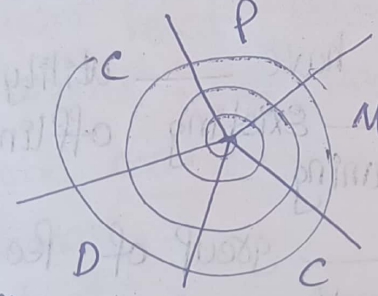
waterfall model



Parallel V-Model



Evolutionary Spiral Model → concurrent Process model



Software Requirement Specification (SRS) for the IQ Games e-app

1) Purpose
2) Problem stmt
3) Solution
4) front end and back end
5) Network, Software Specifications
6) OS

Hardware Specification : Configurations

1) Processor
2) GPU and CPU

Mouse, keyboard, Camera, Scanner, Printer, Haptic only
Any sensors (or) Applicable
Module specifications :
Module /web page 1 :
" 2 :
" 3 :
" 4 :
" 5 :
Expected sample output :
Reference page :
⇒
Purpose : ____ e-app provide easy browing fast delivery of ____
Prbm stmt : Common people doesn't have ____ utility. they are affected by ____ existing offline systems are time consuming
Soln : this application targets ____ group of people to get business / social welfare of ____
SW specification :
Hardware specifications :-
CPU : ____ speed, Processor name : ____ and
ARM / Intel / ____
GPU : ____ speed
VGA : ____
Sensors any : temperature ____
Haptic way : Gloves, 3D eye glass
Camera : ____ HD, ____ Pixel
Scanner : ____ HP/Hawlet
Printer : HP/Cannon color / BW Printer

Module Specifications :-

Module / web page 1 : Home

Module / web page 2 : Menu catalog

Module / web page 3 : Purchase / Place order

Module / web page 4 : Compute bill

Module / web page 5 : Delivers to Home / SMS / Alert msg

Expected Sample output : Screenshot

Reference Page : web link

Date 25/1/24

Specialized Process models

1) Component-Based Development → collecting & Gathering ~~Ingestion~~
2) The Formal Methods Model → Mathematical → formula. Mensuration
3) Aspect-Oriented Software Development
4) Team based Process Model
5) Personal based Process Model

Activates control

⇒ Calendar

Disadvantages of formal methods

1) Time consuming and expensive
2) few Software developers, extensive training is required
3) Difficult to use models of communication mechanism for technically unsophisticated customers.

Ex: Developers of aircraft avionics and medical devices.

Sample features of system

1) Easy browsing
2) Ability to add new user
3) Ability to add new items
4) Secure payment
5) Ontime delivery
6) Sending alert ~~offers~~ messages on each transaction
7) offer messages at festival time
8) Easy way of getting review stars
9) New users are allowed to introduce.
10) Aadhar based authentication
   & Phone number
11) Duplication avoided, existing mail / phone checked

Vs

Use cases

1) Login / sign up
2) Sign up
3) Show catalog
4) Place order
5) Delivery of output
6) Send ~~alt~~ alert msg 6.1 to email, 6.2 to phone
8) Get feedback 8.1 stars 8.2 text comments
9) Introduce friends
10) Authenticate users (Aadhar / Phone / ID)

Date 27/1/24

Aspect-Oriented Software Development

→ Implement set of localized features, fn's
→ ARVR as example.

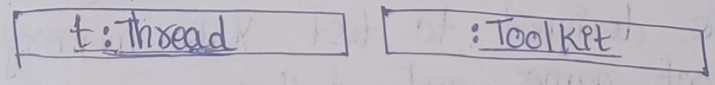
AOCE

Aspect Oriented Component Engineering.

→ User Interfaces
→ Collaborative work
→ Persistency
→ Transaction Processing
→ Security

→ click use case and select new.
→ use case specification.

Sequence diagram



5:

1) class diagram
2) one student class input class out/ code
3) code → model

Date 29/1/24

Software Requirement Specification (SRS) for IQ Games

Purpose: Formally, the main Purpose of creating this app is to avoid playing video games, 3D games etc. By Playing this games, these will be ~~increase~~ ing booring for the Older People. The main Aim is to not spread the lazy People across the world.

Problem Statement: Now-a-days children, Adults are more ~~addicting~~ addicted in Social media such as Instagram, facebook etc. People are not utilizing their brain in the useful things.

Solution: By using this app, children are more confide in studies. They IQ games help diagnose intellectual disabilities & measure someone's intellectual Potential. These games are designed to test your memory with a range of different games and challenges.

Software Specification :-

Front end : HTML, CSS, Javascript

Backend : C++ & other language to handle the game's Server side.

OS : Shapdragon, Android 5.0.

Network : 6.00 GB / 8.00 GB

Hardware Specification :

CPU : 2840 MHz Speed, Processor name : Media Tek Dimensity 1200 ~~and~~ ~~inte~~

GPU : Mali-G68 MC4 Speed

VGA : JUA16IC Android USB to VGA Display Adapter.

camera : 64 Mega Pixels

Module specifications :-

Module /web Page 1 : Home

Module /web Page 2 : Menu catalog

Module /web Page 3 : choosing the level

Module / web Page 4 : complete the level

Module / web Page 5 : Score board

Expected sample output : Screen shot

Reference Page : Web link, ~~In~~ Google, Youtube

Date

30/1/24 *** Agile Software development

Agility is adapting something new.

⇒ Effective communication among stakeholders.

Agility Principles - I

1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2) Welcome changing requirements, even late in development. Agile Processes harness change for the customer's competitive advantage.

3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4) Business People and developers must work together daily throughout the project.

5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6) the most efficient and effective method of conveying information to and within a development team is the face-to-face conversation.

Agile Principles - II

7) Working software is the primary measure of progress.

8) Agile Processes, promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.

9) Continuous attention to technical excellence and good design enhances agility.

10) Simplicity - the art of maximizing the amount of work not done - is essential.

11) the best architectures, requirements and designs emerge from self-organizing teams.

12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Extreme Programming (XP) is one of best Agile Process. Planning

PDCT

Scrum → Packets

XP Design

XP coding

XP testing

Planning, design, coding, test

Date 2/8/24 Task 1:

1a) Estimate the preparation of single cup of coffee?
b) estimate the preparation of 100 cups of coffee by catering?

Task 2:

a) Estimate cost of Pizza?
b) estimate cost of preparing 100 pizzas by catering?

Software Project Estimation

1) Requirements
2) cost
3) Effort
4) Time
5) Greedy Approach
6) D & C approach

Decomposition — $D = f(V_i)$

1) Strategies
→ Top-Down
2) Bottom-up
3) Heuristic method

SCAMPI – SEI CMMI
CBAIPI – SEI CMM
SPICE (ISO/IEC 15504)
ISO (9001:2000)

Approaching of sizing problm

1) Fuzzy logic
2) fn Point
3) Standard component
4) change

Date 1/2/24 Decomposition Techniques

1) Top down Approach (Analogy approach) previous and similar project
2) Bottom up Approach → well known project

PERT, CPM
↓ ↳ Critical Path Method
Project estimation and Review Technique

Software Sizing

The accuracy of Software Project ~~ae~~ estimate
→ the degree which you properly estimated size of Product
→ Ability to translate size estimate into human effort, calendar time and dollars.

Problem Based Estimation

→ LOC and FP data used.

$$S = S_{opt} + 4S_m + S_{pess}/6$$

S_{opt} → optimistic, S_m → most likely, S_{pess} → pessimistic

→ when give LOC

Ex: Range of LOC 3D geometric analysis fn is optimistic, 4600 LOC ; most likely, 6900 LOC and the pessimistic, 8600 LOC.

$$S = 4600 + 4(6900) + 8600/6$$
$$= 4600 + 27600 + 1433.33$$
$$= 33633.3$$

$\frac{69 \times 4}{276} = 36$

FP Based

FP(estimated) = count total $* [0.65 + 0.01 * \Sigma(F_i)] = 375$

Process Based estimation and Based on Use cases

→ Use cases are not having standard form.
↓
Users scenario
→ It is external view.
→ It donot address complexity of fn's.
→ It describes complex behaviour.

Loc estimate = N x Loc_{avg} + [(Sa/Sn-1) + (Pa/Pn-1)] x Loc_{adj}

for usecases

actual avg, actual avg

6/2/24

1) Identify list of process flows, process models, Prescriptive models and Agile Process.

[CMMI] decides Reputation

capability Maturity Model Integrating model.

WP (work Product)

Emphirical Methods

Marketing MIS : Unadjusted FP count

fn Description	Transaction Type	Row FP (avg. complexity)
Monthly Sales report		
→ Monthly Sales report	EI	4 (4x5x3x5x2, 20x5x2, 40x5, 600)
Sales summary I	EO	
" II	EO	
" III	EO	
" IV	EO	
" V	EO	
Sales enquiry		
Sales file		
Product file		
		57

→ Add functional Points

Value Adjusted factors

$VAF = (AF * 0.01) + 0.65$ → Emphirical

TFPC = ~~Total~~ functional Point * VAF

3 Modes of Software Development

1) Organic
2) Semi attached
3) Embedded (3GL) with AI (3rd General Language)

Types of cocomo Basic ↓ referred (1), Intermediate ↓ (2), Advance level ↓ high

$$E = A + B \times (e_v)^C$$

E → Vm (effort by Persons)

Loc by KLoc

A, B, C empirically derived constants

ev is estimation variable either LOC (&) FP.

LOC Models

1) Walson-Felix → $E = 5.2 \times (KLOC)^{0.91}$
2) Bailey-Basili → $E = 5.5 + 0.73 \times (KLoc)^{1.16}$
3) Boehm simple → $E = 3.2 \times (kLoc)^{1.05}$
4) Doty model for KLOC > 9 → $E = 5.288 \times (KLOC)^{1.047}$

FP Models

$E = -91.4 + 0.355\ FP$ Albrecht and Gaffney

$E = -37 + 0.96\ FP$ Kemerer

$E = -12.88 + 0.405\ FP$ Small Project regression

COCOMO-II

Application composition models
→ Early Design stage Model
→ Post-Architecture stage model

Ex: AI, MLP, OS, cryptography

Number object Points (NOP) = OP × [100 − %.reuse/100]

$$PROD = \frac{NOP}{Persons/month}$$

$$\text{estimated effort} = \frac{NOP}{PROD}$$

Note:

1) Effort e always calculated by emphirical model

$$E = \frac{LOC \times B^{0.43}}{P} \times \frac{1}{t^4}$$

E ↓ effort, t ↓ time

B → Special skills
P → Productivity Parameter

⇒ avg 138 LOC/Pm
Labour Rate = 5800 P/m
cost = $14

Total Effort = Total Loc / Productivity
Developers Effort = Total Effort
Total cost = Total Effort * Labour Rate

FP = count total $\times$ 10.65 + 0.01 $\times \Sigma(f_i)$
Functional Point

Software eqn

$$E = \frac{LOC \times B^{0.33} \times I}{P^{\beta} \times t^4}$$

E → Effort
t → Time
B = 13

P = 2,000 → real time embedded Software
P = 10,000 → Telecommunications
P = 28,000 → Business appli
P = 12000 → Scientific (CAD)

1) $t_{min} = \frac{8.14 \times LOC^{0.43}}{P^{0.43}}$ $t_{min} > 6$ months

$E = 180\ B t^3$ P/m $E >= 20$ Pesson_months

B = 0.25
t = 1.05

18000
15500
12300
7000

$t_{min} = \frac{8.14 \times 52800}{(2000)^{0.43}}$ $8.14 \times \left(\frac{52800}{2000}\right)^{0.43}$

10.692 4297.9½

$\frac{8.14 \times 33200}{(12000)^{0.43}}$ 12

210248

Formulas

1) $d = f(V_i)$ → independent LOC & FP
↓
effort / cost / duration

2) $S = \frac{S_{opt} + 4S_m + S_{pess}}{6}$
↓
S "variable"

3) FP (estimated) = count total $* [0.65 + 0.01 * \Sigma(F_i)] = 375$

4) $LOC_{estimate} = N \times LOC_{avg} + [(Sa/Sn-1)+(Pa/Pn-1)] \times LOC_{adjust}$

LOC_{adj} = n% of LOC_{avg}

Sa → actual Scenario

Pa → Pages

Sn, Pn → avg scenario / Page of Subsystem.

Estimation Models

$E = A + B \times (e_v)^C$

↓ effort in P/M months

estimate value (Loc) or FP

(COCOMO)

constructive cost Model

⇒ NOP = (object Points) * [(100 - % reuse)/100]

$PROD = \frac{NOP}{Person\text{-}Month}$

$estimated\ effort = \frac{NOP}{PROD}$

Software eqn

$E = \frac{LOC * B^{0.333} \times 1}{P^3 * t^4}$

$t_{min} = 8.14 * \left(\frac{LOC}{P}\right)^{0.43}$

$E = 180 Bt^8$

Total effort = $\frac{Total\ LOC}{Productivity}$

$Effort = \frac{Total\ effor}{developers}$ ✓

Total cost = Total Effort * Labour Rate

cost Per LOC = $\frac{Labor\ Rate}{Productivity}$

Total cost = Total LOC * cost Per LOC

Total effort = $\frac{Total\ cost}{Labour\ Rate}$

FPA = UFP X CAF

FPA matrix

5 Rows 3 columns

	low	Avg	high
Input	3	4	6
Enquiry	3	4	6
Output	4	5	7
Interface	5	7	10
Userfile	7	10	15

$UFP = \sum_{i=1,j=1}^{i=5,j=3} w_{ij} c_{ij}$ — weight adjustment factor

Date 13/2/24

Principles that Guide Process - I

1) Be agile
2) Focus on quality at every step
3) Be Ready to adapt
4) Build an effective team
5) Establish mechanisms for communication
6) Manage change
7) Assess risk
8) Create work products that provide value for other

Date 8/2/24

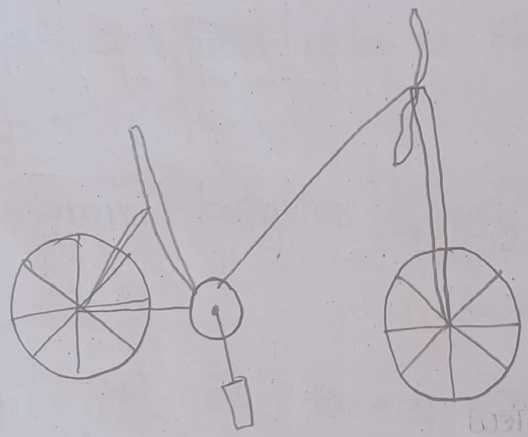
Name : N. Lashya Praneetha

Reg no : 126018032

1) Requirement

Design a bicycle with following features

1) It should have shelter.
2) four people should ride
3) energy converter is used so that either it is semi/ fully automated bike.
4) No Noise, eco friendly.
5) Light weight storage
6) It should be programmable.



Date 14/2/24 UML

SE Process

RUP

Phases :- Inception, Elaboration, construction, Transition

Architecture Models — Diagrams in UML — Hierarchal architect chart

1) Enterprise
2) System
3) Software

1) Structure → component → Deployment → Package → class → object → composite

static models

2) Behavior diagrams / Dynamic models
→ Use case
→ Activity
→ State Machine
→ Communication
→ Sequence
→ Timing
→ Interaction overview

Pest chart

Architecture Modeling - Perspectives

conceptual Physical

Efficiency by Time and Space complexity

No. of Tangents input and output in numbers is scalability

No. of times is executed is throughput.

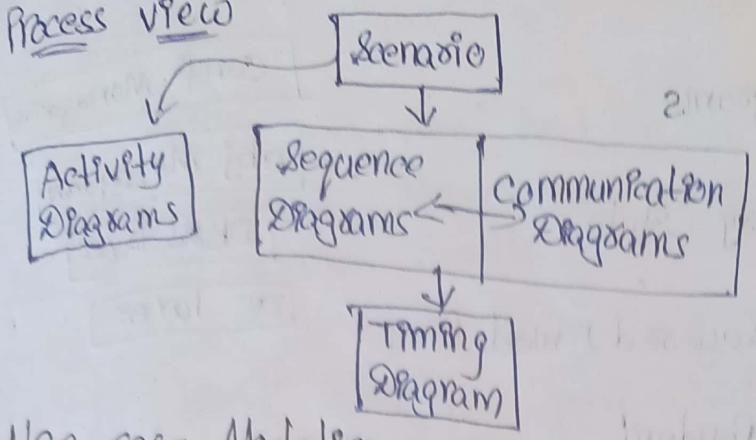
Logical View

Package Diagrams ⟹ composite structure Diagrams

class Diagrams

Obj diagram

State diagrams



Use case Modeling

Use case Specification

name, short purpose, contact and change information
pre, post conditions
basic flow is the good day scenario
alternate scenarios
Inclusion, extension points ~~of~~ to be identified

Activity Diagram

→ captures dynamic behavior (activity - oriented)
→ Purpose

date 29/2/24

1) Write down list of famous buildings in India?
A) Taj Mahal, charminar, Golconda fort, Agra fort, Lotus Temple, Golden Temple

Software architecture

Design process for identifying the subsystems making up a system and framework for sub-system control and communication is architectural design.
O/P is software architecture.

Characteristics

1) Performance
2) Security
3) Safety
4) Availability
5) Maintainability

Architectural styles

1) Box and Line Diagrams

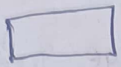
System organisation

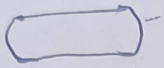
1) Shared data repository style.
2) client - server
3) Abstract machine (layered) model

conf Management
obj Management
DBMS layer
os layer

sub components → independent

Modules → Dependent

objects are 

Process are 

1) Identify fn req
2) non fn req
3) Special req and use case req and to ~~too~~ trace traceability matrix
3) Identify set of obj, entities coming under boundary and business entities.
4) Identify set of attributes and operations of given classes.
5) Primary, 2nd and off stage actors. and find each actor use cases and communications inclusions & exclusions.
6) Find collaborators of entities using CRC card
7) find activities of all real time ~~act~~ entities and develop activity diagram and use case da
8) Prepare SRS of given Prblm stmt.
9) Develop seq, collaboration of BCE class of scenario.
10) Perform forward and reverse engg of code
11) Perform memory, Pure coverage, Performance of sample code.

↓ class diagram

Proverb

19) Laughter is the best medicine

Laughter reduces stress and improves immunity, leading to better health.

Ex: I think the best thing for you right now would be to spend sometime with people you can joke around with. Laughter is the best medicine, after all.

Date
2/2/24

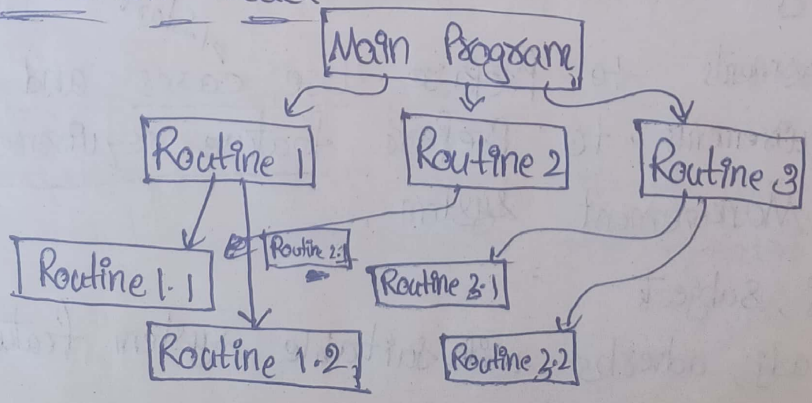
class diagram for Restaurant

i) Registration of consumers
ii) Booking a table, ~~parking car~~ Party hall, restaurant rooms.
iii) Providing Multi cuisine food such as Indian, Italin, chinese, Japanese, african foods.
iv) Managing Home deliveries, Partyhall bookings and event managements, through different Managers.
v) Getting feedback from Consumers ~~at~~ on each event,
vi) complaint register is common to everyone, a maintainance engineer will maintain it.
vii) A Manager who takes role of discount offers at seasonal times and reporting annual income and balance sheet of credits and debits of company owner.

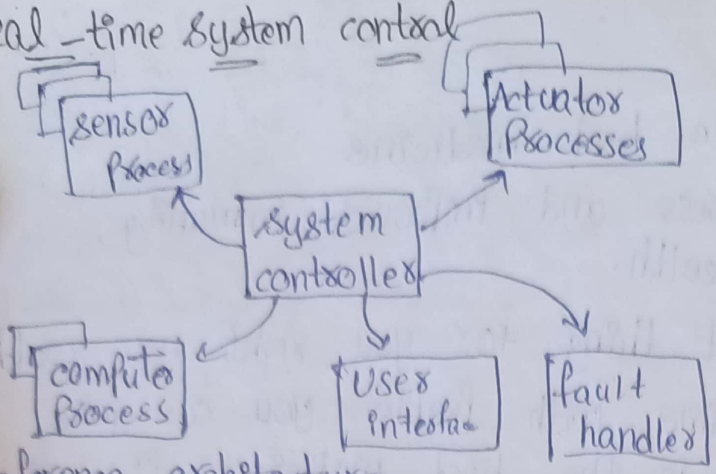
Control Styles

1) Centralised control → call-return, Manager
2) Event-based control

call-return model



Real-time system control

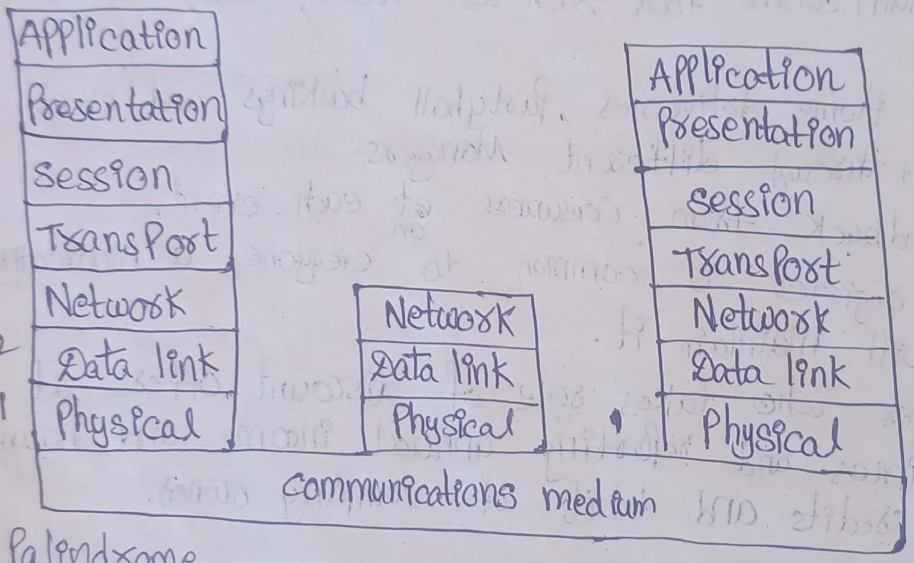


Reference architectures

Generic → Bottom up

Reference → Top-down

OSI Reference Model



1) Palindrome
2) reverse of string
3) Extracting substring
4) Area of rectangle

Date 23/11/22

Find 10 fn requirements to Prepare use cases (start verb) and 10 non-fn requirements to Prepare feature requirements of Restaurant Management System.

use case → obj, subjects

feat req → adj, adverbs with suitable system features

Pure coverage

winxp → start → programs → virtual studio → c++ studio
run click
→ win32 application, enter project name, c++ source file and write code.
→ Build, exe file and extraction

→ Sample fn requirements to prepare use cases.

showcase available facilities
→ choosing their requirements
→ checking availability facilities
→ Payment process
→ Generate Digital Receipt

Primary user → end user
secondary → Home Page
off stage → Bank
intermediate

Non fn Requirement

1) Performance factors
2) Security
3) Safety.
4) Scalability, adoptability, Quality, Mobility

System provides 'n' no. of users

1) Adj
2) Quick Response to Registration
3) Safty authentication methods provided such as password, OTP, captcha

(Quality)
Errors

Graphical error

Bugs
Defects → is a unidentified mistake by developer & tester
failures
Risk

Syntax
Schematic → logical error

A void analyser may identify that they throw away code.

→ may identify when integrating

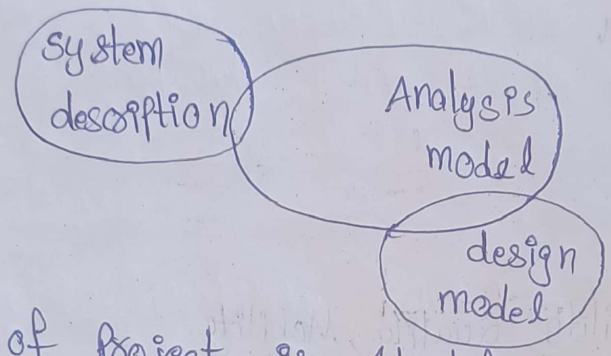
Traceability matrix traces fn & non fn requirements either it is successfully ~~fail~~ facilated by developer

Date
24/2/24 Requirements Engineering

1) Inception
2) Elicitation
3) Elaboration
4) Negotiation

Statechart & Sequence ~~see~~ 2S → behavioral

⇒ Specification
Validation
Requirements management



Scope of Project is Identify b/w boundaries of starting and Ending life span of Project until it move on to the next version.

Analysis classes
External entities, things, Occurrences & events, Roles, Organizational units, Places, Structures

1) Prepare Online Planetorian using AR and VR. Identify Prblm stmt, list of stakeholders, Purpose and beneficiary, Scope of Prblm & Soln, operational goals, fn requirement, non-fn req, at initial Stage then trace feasibility matrix after negotiating and validating stakeholders requirements. Identify special req at least 4 Glossary terms.

2) Find the activities of Travelling SP from home to cover all areas of sales within a given investment amount and within a day how much places visited

and how much saved as pocket money?

3) Take any famous novel and its 5 characters, develop a sequence diagram of a scenario as converted as into multimedia data base for an story telling app?

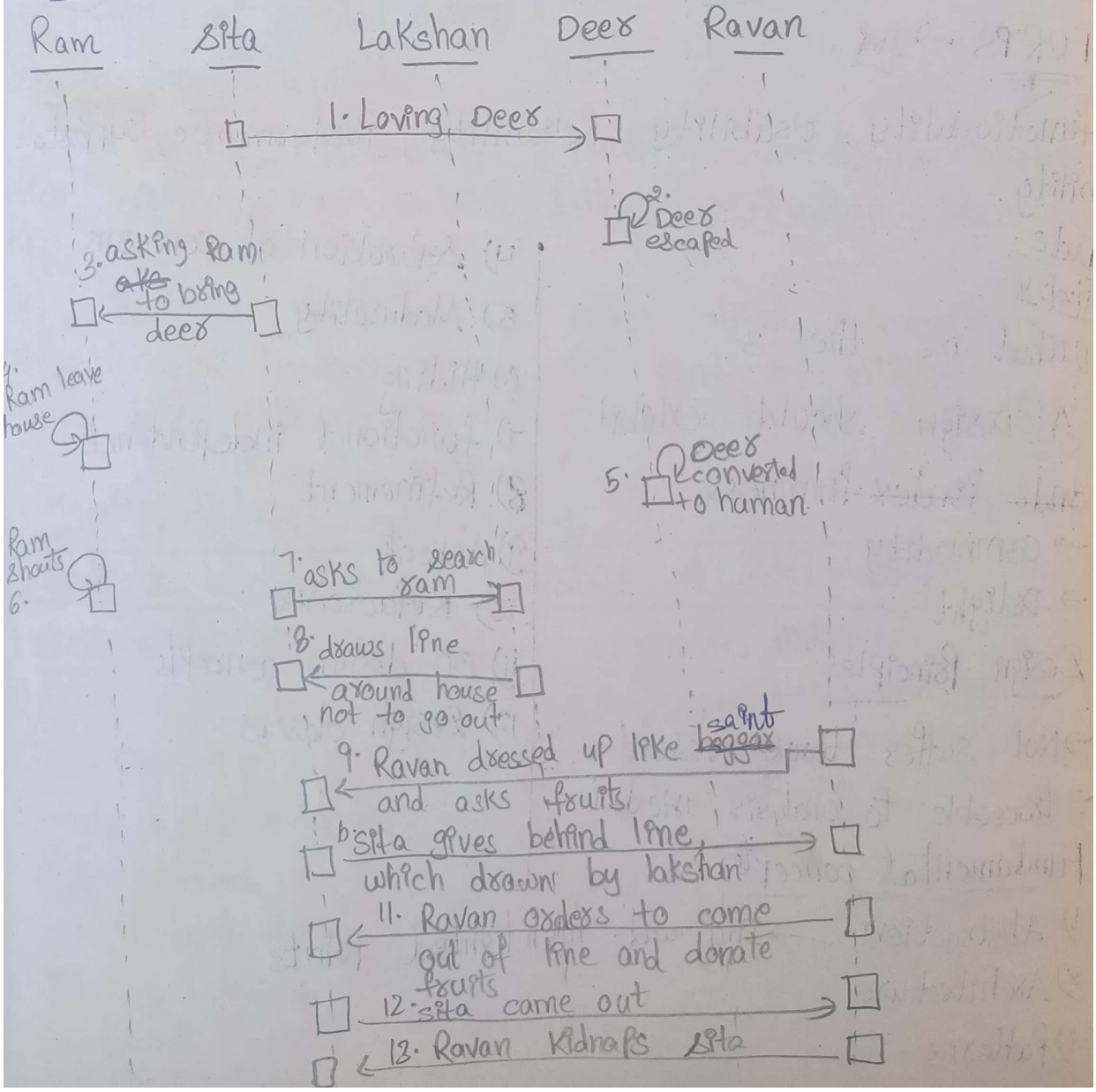
Solutions

1) Pblm stmt: we can't visit planets directly.

sensors
camera
sound system
non-fn req → developers
fn-req → use case for user point of view

Hostel, Library, Play ground, Canteen, classroom.

characters :- Sita, Ram, Lakshan, Deer, Ravan.



29/2/24

1) Find online, offline collaborator of following entities:

i) Library, books, Journals, magazines, articles, E-book, audio-book, Musical lessons, Videos

ii) Food, Serving bowls, mess hall, chef, cook, Manager, waste bins, donator, orphan home.

iii) Play ground, Players, Playing rules, court, amphise, refer, Scoreboard, Teamleader; Pla

iv) Transport, types of transport, transport manager, source destination, list, ticket, Price.

v) classroom, student, Professors, department, office, courses, Session, duration, online class, mentor, online monitor, complaint system, feedback.

Integrated Alaxam App

FURPS → IMP

functionality, Usability, Reliability, Performance, Supportability.

Date
1/3/24

1) What are the 3

A Design should exhibit

~~Inde~~ ~~Index~~ firmness
→ commodity
→ Delight

Design Principles

→ Not suffer 'tunnel vision'
→ traceable to analysis model

Fundamental concepts

1) Abstraction → Imp and high lighted points
2) Architecture
3) Patterns
4) Separation of concerns
5) Modularity
6) Hiding
7) functional independence
8) Refinement
9) Aspects
10) Refactoring
11) OO design concepts
12) Design classes

Date 5/3/24 Page no:-

1 ⇒ get phone no
2 ⇒ find ISP
3 ⇒ show offers
4 ⇒ check validity
5 → Recharge mobile

find suitable architectural design. find useful components
⇒ Low coupling is Principle that has min level of dependency.

over Design → obj → Data abstraction
→ class → Inheritance
→ encap → Polymorphism
→ Data hiding → Abstract classes

Architecture
→ Structural Properties
→ Extra-functional Properties
→ Families of related Systems

Design Pattern Templates
Pattern name, Applicability, Structure, Participants, Collaborations, Consequences, Related Patterns, Motivation, Also-Known-as, Intent

*. Patterns are templates ex: class, object, Polymorphism, Separation of ~~concern~~ concern are kind of Patterns.
*. Any New Pattern may be decided by its responsibility if it is not available yet
* Singleton, Abstract friend class

concern is a feature of behaviour of software.

Functional Independence
1) cohesion
2) coupling

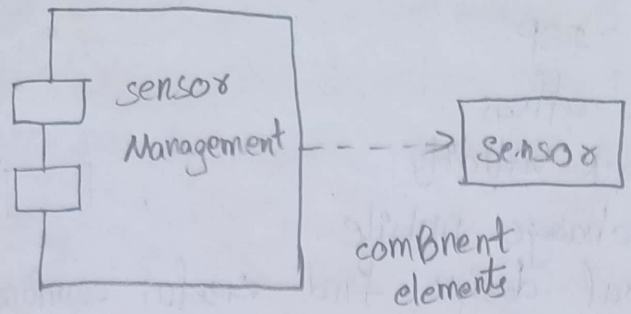
Design Model elements
1) Data elements
2) Architectural elements
3) Interface elements
4) Component elements → .jar, .exe, .dcom, .del, activex
5) Deployment elements → Proxi server, Junction Point

5) Nodes, links

Architectural elements: Sports, entertainment, games, non-profitable, embedded, AI, DL, ML, NLP, etc

Timer, calendar

active, Port no, client



Date
6/3/24

Interface Design

→ GUI
command based, Alexa, Siri

Golden rules
1) Place the user in control
2) Reduce the user's memory load
3) Make the interface consistent

User Interface Design Models
1) User Model
2) Design Model
3) Mental Model
4) Implementation model

Interface Design steps
→ Design interface obj's
→ Define events
→ Depict each interface state
→ Indicate how the user interprets the state of the system

Design Issues

→ Response time
→ Help facilities
→ Error Handling
→ Menu & command labeling
→ Application accessibility
→ Internationalization

Date
7/3/24

Testing

components :- Sample app, a code Module
2) Test Plan
3) Test case
4) Testing Tools
5) Test Script
6) Test input
7) Test results
8) Test report
9) Test log file

Testing -

1) Black box → Testing on external environments how screen appears about input data related bugs (BVA)
2) white box
   - clear box testing
   - logical flow of system

Two ways of conducting testing

→ Manual testing
→ Automated testing

Date
2/3/24 Component

→ Multiple modules, deployable & replaceable part of a system.

Views

→ object oriented
→ Conventional view
→ Process - Related

Design Principles

Open - closed Principle (OCP)
LSP → Liskov Substitution Principle
DIP → Dependency Invession Principle
ISP → Interface Segregation Principle

Packaging Principles

REP → Release Reuse Equivalent Principle
CCP → common closure Principle
CRP → common ~~Resue~~ Reuse Principle

Cohesion → unity

~~CCSDRTIE~~

CSDRTIE → coupling

content, Common, control, stamp, Data, Routine call, Type use, Inclusion or Import, External

coupling and cohesion measures quality of software system design.

Types of coupling

→ Data coupling → good (customer billing system)
→ Stamp coupling
→ control coupling → sort fn
→ external coupling → protocal, external file, device format
→ common coupling → global data structures
→ content coupling → worst modify
→ Sequential
→ Temporal
→ Communicational
→ functional
→ Data structured (Block chain structure)
→ Interaction (Recursive calls)
→ Component (Inheritance)

Artificial Intelligence

Date 13/3/24

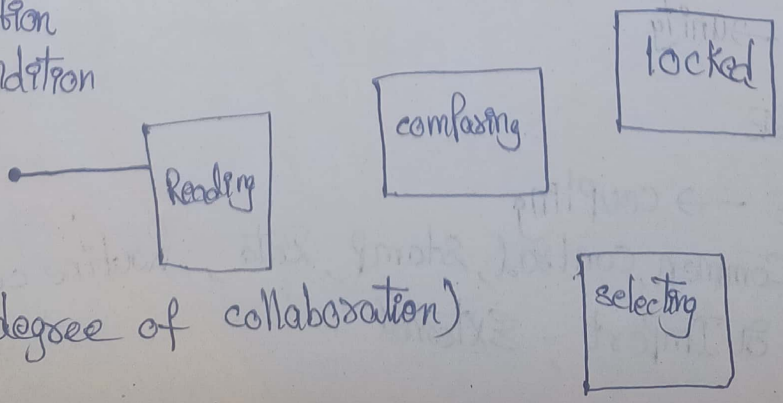
Identify domain, analysis, potential, design, classes of story telling app?

Component levels

→ Identify application & infrastructure of domain classes.

Statechart is rounded rectangle box
→ It has 2 compartments

Constraints
→ Invariant
→ Precondition
→ Post-condition
→ Guard



cohesion (degree of collaboration)

Types of cohesion → internal glue FSCPTLC

1) Function → High → calculating Total amount of ID (transaction)
2) Sequence → Purchase() → stock() → sell() → deliver()
3) Communicational → update record in db and send to printer
4) Procedural → calculate GPA, print student record, not change order
5) Temporal → checkOTP()
6) Logical → logically related not functionally
7) Coincidental → bcoz ; print next line and reverse the characters of a string

Informational cohesion → Linked list, Blockchain
Layer cohesion :- User view / DB

Components of Software Process

Testing

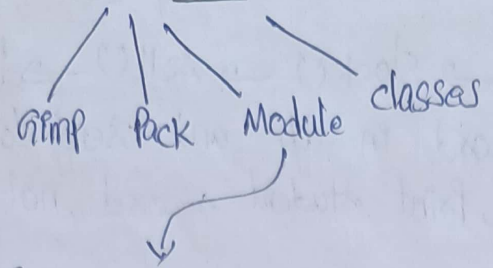
Objective : Finding uncovered errors
2 types of Testing : blackbox testing, white box testing
two ways / approaches of testing : Manual, automated testing
Testing components are :-

1) Testing tools
2) Sample applications
3) Test plan
4) Test cases from use cases
5) Test inputs
6) Test results
7) Test log

exit → end

Date
15/3/24

System features



UI - View
event trigger - controller
DB - Model

Online food Ordering System

Living
Hotels, foodie, user, Delivery boy, APP developer, Admin
↑ upto designing application
→ web
→ DB
→ system

Non-Living
food types, order, menu, cancel, reserve, Reschedule

A class may be a noun, things, Place or Process,

Hospital Management System
doctor, nurse, Patient, ward boys, APP developer

Practice
UML - unified Modelling Language
Phases in RUP
1) Inception → scope of Project
2) Elaboration → Plan, specify features and baseline architecture
3) Construction → Build
4) Transition → Transit Product to its users

Requirements → use case Model
Design & Analysis → Design Model
Implementation → Implementation Model
Test → Test Model

Architecture Models → Enterprise
→ System
→ Software

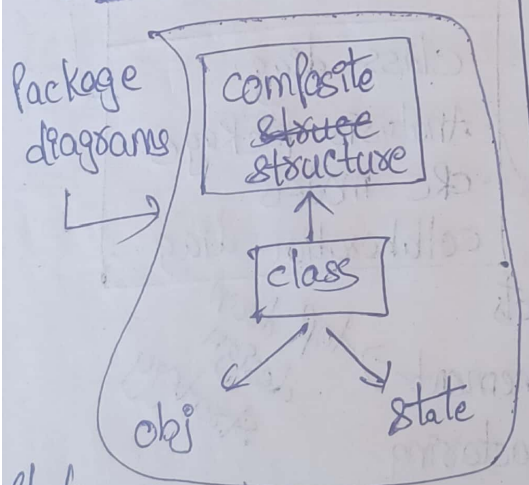
Diagram

Structure Diagram (static models)
1) class Diag
2) component
3) obj
4) Profile
5) composite structure
6) deployment
7) Package

Behaviour Diagram (Dynamic models)
1) Activity Diag
2) use case
3) Interaction
4) State Machine
5) Sequence
6) communication
7) Interaction overview
8) Timing

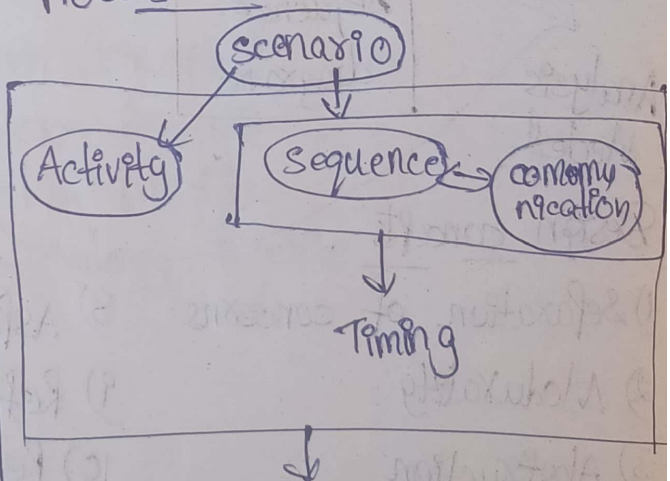
Architecture Modeling
Logical view (end user)
Implementation (developer) view (Programmers & Software managers)
Process view (Integration)
Deployment view (System Physical engineer)

Logical View



State representations — Active, Passive
1) Reading
2) Locked
3) Selecting
4) Comparing

Process view



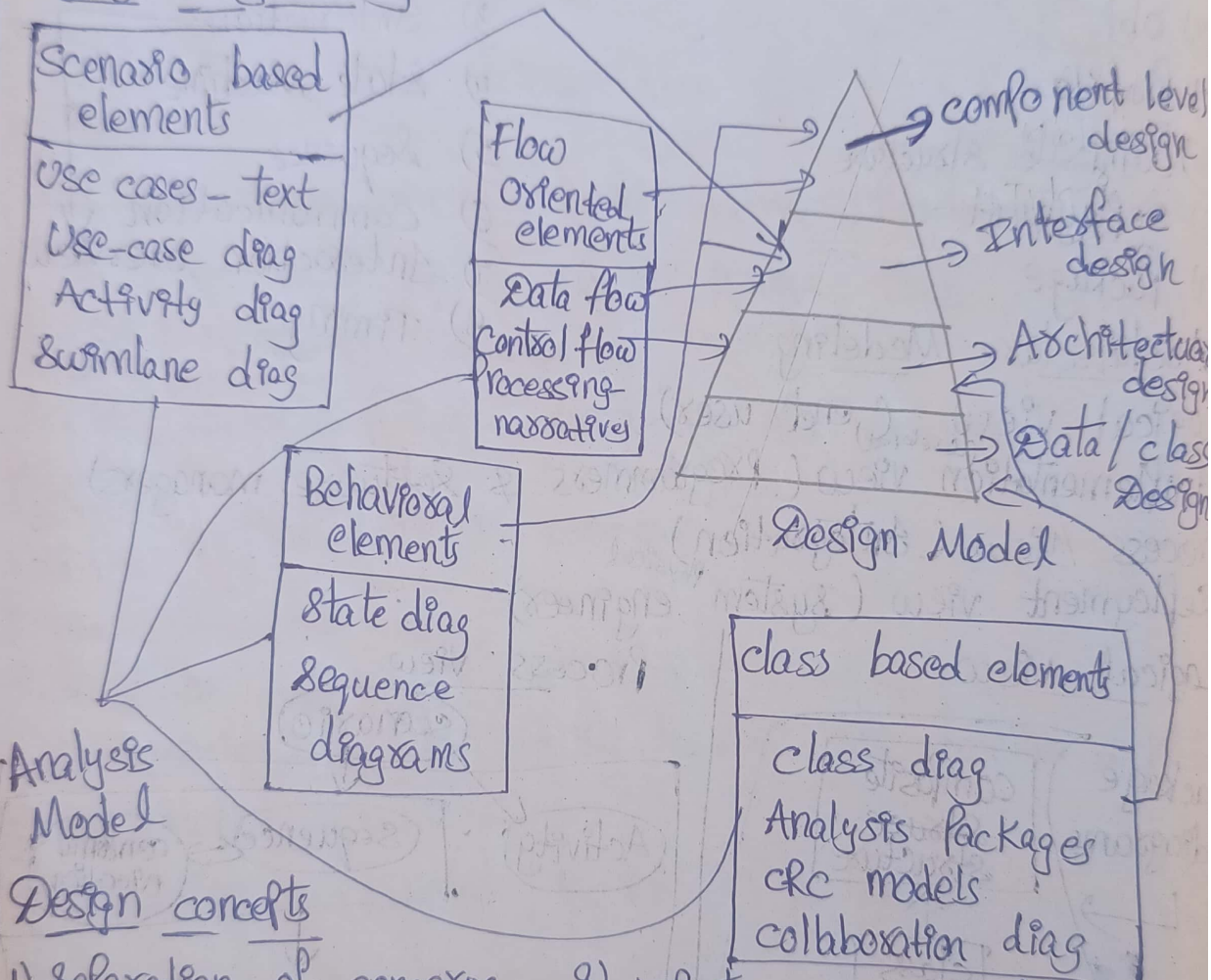
Interaction overview

Object oriented Analysis

1) finding objects
2) Organizing objects
3) Describing object interaction
4) Defining operations on the objects
5) Defining object's internals

UNIT-III

Design Engineering



Design concepts

1) Separation of concerns
2) Modularity
3) Abstraction
4) Procedural
5) Data
6) Information hiding
7) functional dependence
8) Aspects
9) Refinement → top down design strategy
10) Refactoring
11) Architecture

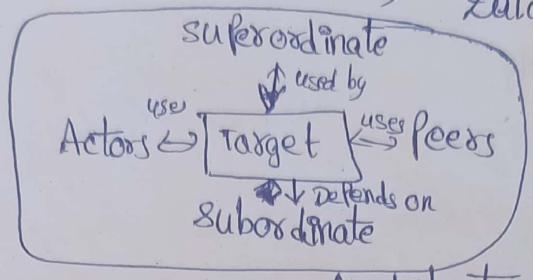
Architectural Genres

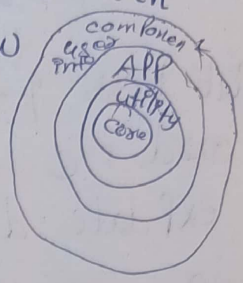
Artificial Intelligence
Entertainment & Sports communication
Financial
Games → SAI (Software Architecture Immersipresence)
Govt
Industrial
Legal
Medical
Military
Scientific
Transportation

Styles Home

1) cape cod
2) A-type
3) Raised Ranch
4) Central hall colonial

→ Data centre arch
Data flow





Architecture Design

Assessing Alternative Architectural Design → Trade off analyses

SEI (Software Engineering Institute)

Component is a modular building block for computer software

Traditional View : 1) control component 2) Prblm Domain Component 3) Infrastructure component

Basic Design Principles

(OCP) → Open closed Principle
LSP → Liskov Substitution Principle
DIP → Dependency Inversion Principle
ISP → Integration Segregation Principle
REP → Release Reuse Equivalency Principle

(CCP) common closure Principle
(CRP) common Reuse Principle

Steps for External data source

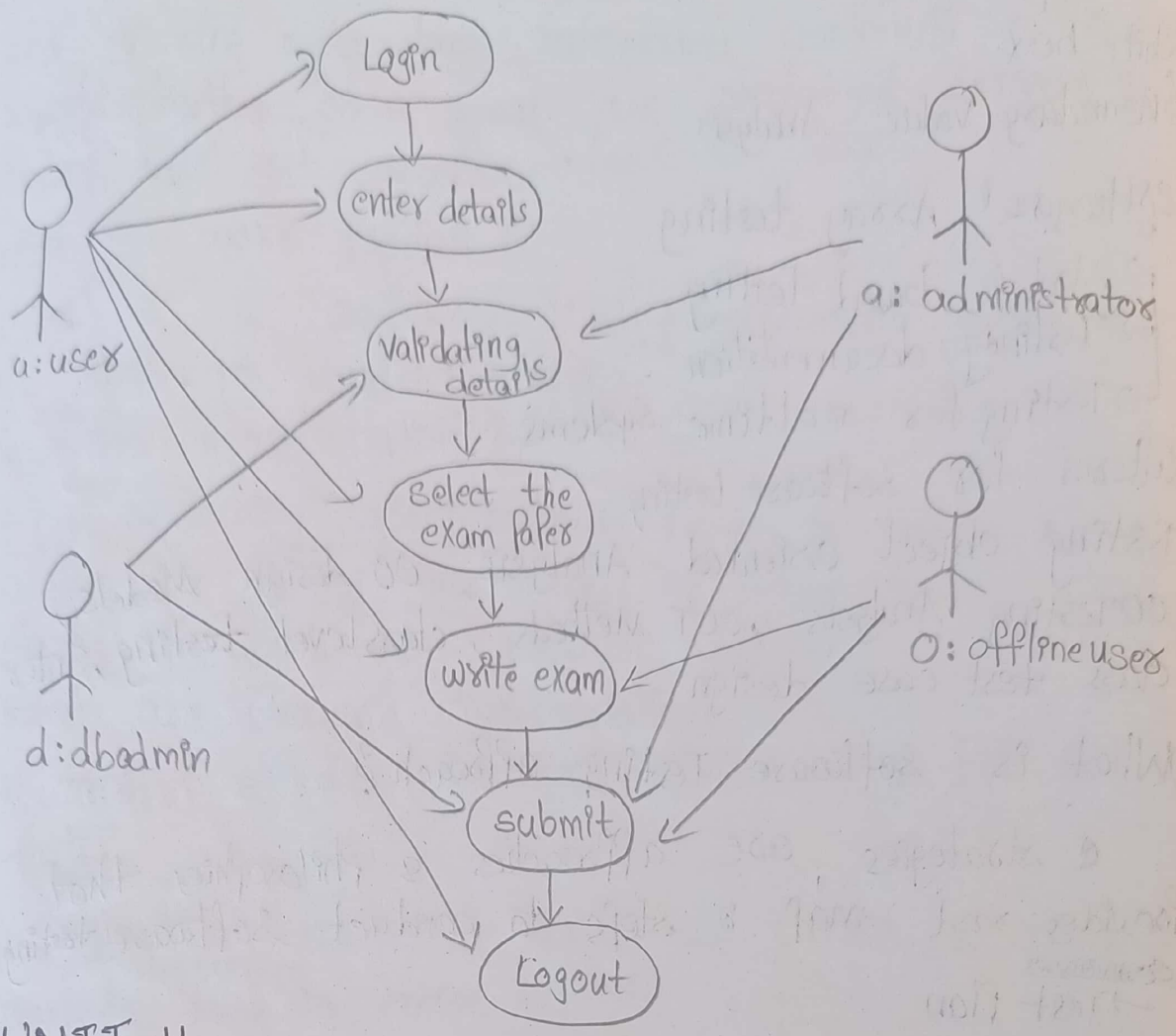
1) Go to Rational Robot

```
Dim q, c, d as Variant
Open "c:\filename.csv" for Input as #1
do while not eof(1)
Input #1, q, c, d

Input Keys q → Quantity
    card :- c
    exp date : d

loop
close #1
end sub
```

11b

```
'$include "sqautil.sbh"
sub Main()

  dim x as integer
  dim dp as long
  dim q, c, d as ~~Static~~ Variant

  After Green
  dp = SQADataPool.Open("sample1")   → file in Test manager
  for x=1 to 10
  call SQADataPool.Fetch(dp)

  call SQADataPool.Value(dp,1,q)
  Edit
  Input Keys q                (dp,2,c)

  Input Keys c                (dp,3,d)

  Input Keys d

next x
call SQADataPool.Close(dp)
```

Draw Use case Diagram of Monitoring examinations of online examination system



UNIT-4

Strategic Approach of testing

Issues and testing Strategies of Software develop, object oriented Software development

Validating → Boolean ope
Loop ope

System

Art of debugging

Software Testing fundamentals

Internal, External use of testing

white box testing

- Basis Path
- Control structure
- Graph matrix

Black box
Loop testing
↳ do while, while loop, nested, concatenated, unstructured loops.

White box

Boundary Value Analysis

Orthogonal Array testing

→ Model based testing

→ Testing documentation

→ Testing for realtime systems.

Patterns for software testing

Testing object oriented Analysis, OO design Models, OO Testing Analysis, OOT Methods, class level testing, Inter class test case design

1) What is Software Testing approach?

Strategies are approaches & Philosophies that provides road map & steps to conduct Software testing

components

→ Test Plan

→ Test case design

→ Test execution using testing tools

→ Resultant data collection / Test input data and what Evaluation

→ finding uncovered errors that were made inadvertently as it was designed and constructed.

2) who does it?

Strategy made by Proj Manager, developer & software engineer and Testing Specialist.

3) why do you Testing strategy?

→ Testing often accounts & more often account than any other Software engg actions. It is conducted hat-hazardly, wasting time, unnecessary effo

even worse, errors sneak through undetector.

what are steps?

→ It begins from small units from single component and growing upto large integrated components which are related together as a group. we conducted series of testing until get acceptance from the user.

what is work product?

→ It is a software testing document which consist of steps to conduct testing.

How to ensure testing?

→ By reviewing specifications, prior to testing, you can access the process. This honour of tests ensure all stages of construction product effort.

What are Generic characteristics?

1) conduct effective technical review using PPt or reports.

2) Begins at component level at works at outward towards integration to entire system.

3) find different testing approaches for diff engg process such as conduction application, OO app, mobile app, web app.

Testing conducted by developer at intial stage then for large projects handet over to the ITG (Integrated Testing Groups)

→ Testing & debugging are different process but debugging encorporated on testing strategies

Strategies provided guidance for practicioner

$ Provides set of milestones to proj manager

→ Verification & Validation of SQA & Software Quality Analysis

1) Verification (Implementation of Testing) & building Product right

2) Validation (Traceability)., & we building right Product

Misconsumptions

1) Develop a new ver do Testing &

2) My Mexciness Test conducted any strangers &

3) Testers enter @ only at the level of testing Bxcess &

Date 23/3/24

Testing Phases

- Unit Testing (white box @ testing)
- Integration Testing
- System Testing (Black box testing)
- Acceptance Testing

→ Smoke Testing (daily build)

→ Regression Testing (Due to changes) → bugs remaining = bug already fixed

→ Stress Testing (under Pressure, overloaded operations, induced errors, checking robustness strength)

→ Volume Testing (Massive input)

Anti-debugging (exception handlers)

Testing vs inspection

↓

Periodically

Ripple Effect: High Probability that the efforts to remove the defects may have actually added new defects.

Independent tester

→ driven by Quality

Developer

→ driven by delivery

first Pass of unit and integration testing is called developer testing

Date 27/3/24

Find use cases of course registration system and develop use case document following details

use case name, ~~for~~ Purpose, flow of events
- → basic flow → Post condn
- → sub. flow → Inclusion
- → Alternative → extension
- → ~~Post~~ Pre condn → 3 types of Actors

Primary, Secondary, off stage and draw use case diagram

Actor – noun

use case – sequence of actions ; start from verb

→ who is interested in certain requirement ?

→ where is organization used ? Platform

student
Professor
Registrar
Billing System

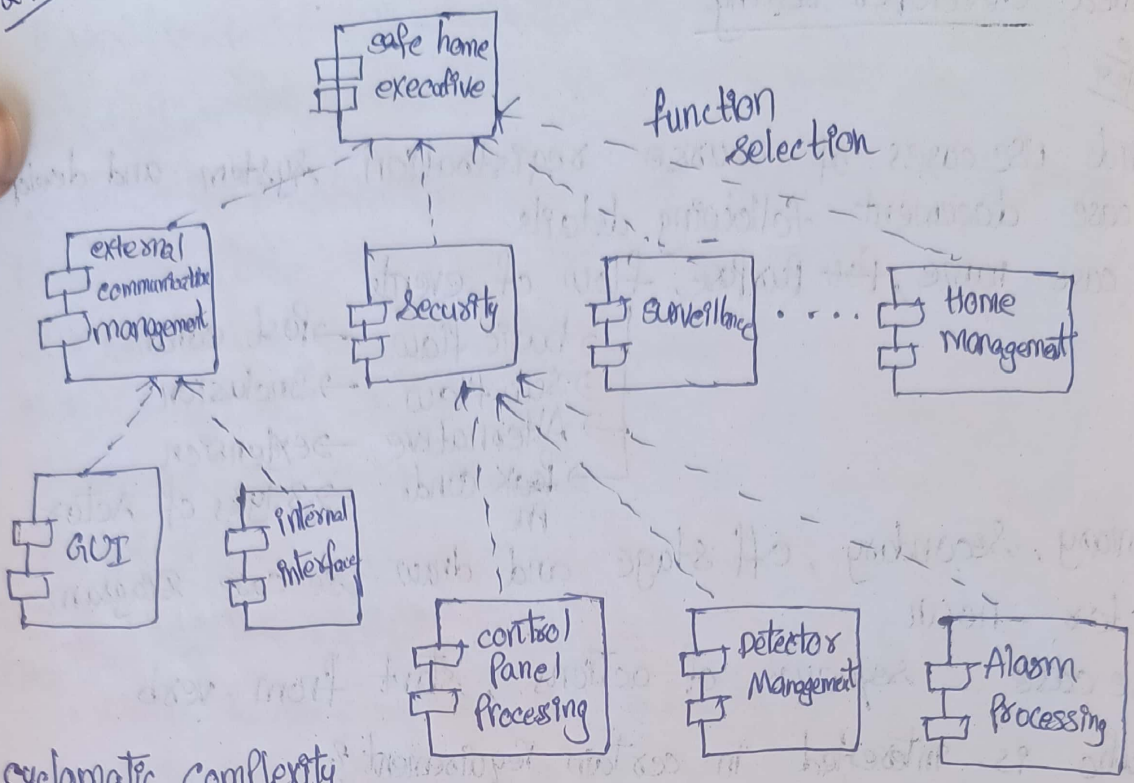
Use cases are tasks

1) Register for courses
2) select courses to teach
3) Request course roster
4) Maintain course information
5) Maintain Professor information
6) Maintain student information
7) create course catalog

Rule of Thumb

→ visibility of system status
→ Match btw system & real world
→ User control & freedom
→ consistency and standards
→ Error prevention
→ Recognition rather than recall
→ Flexibility and efficiency of use
→ Aesthetic and minimalist design
→ Help users recognize, diagnose & recover from errors
→ Help and documentation

Date
27/8/24 Safe-home top-level components



Cyclomatic Complexity

It is counting ~~cycles~~ circuits in the flow chart & regions covered by the Program.

Target for Unit Test cases

1) Module interface
2) Local data structure
3) Boundary condin's (BVA) Boundary Value Analysis
4) Independent Paths (basis Paths)
5) error handling Paths

Problems to uncover in Error handling

→ Error description is unintelligible or ambiguous.

Drivers and Stubs for Unit Testing

- Driver
- Stubs

Integration Testing

Defined as a systematic technique for constructing the software architecture.

Non-Incremental Incremental

Non-Incremental Testing

→ called as Big Bang Approach

→ Tested as whole

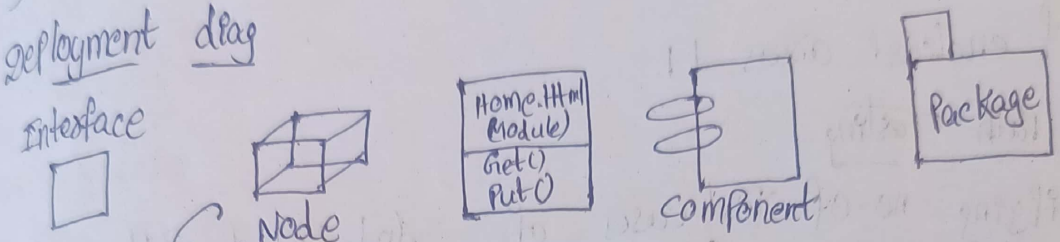
Incremental Integration Testing

1) Top-down → Modules moves downward beginning main module

2) Bottom-up : leaf to root level

3) Sandwich : combine of top-down and Bottom-up

Deployment diag



↳ All servers, devices are nodes

→ capable to computation

Date 2/4/24

1) What are 2 approaches of Integration testing ?

2) Differentiate bottown up & top down Integration ?

3) what is stub ?

4) Lets take order of modules in H_1 : M_1, H_2 : M_2, M_3, H_3 : M_4, M_5, M_6, M_7, H_4 : M_8, M_9, M_{10}. Find order of BFS & DFS Integration ?

5) what is smoke testing, volume, stress, regression and random testing ?

Properties

→ Observability

→ controllability

→ Simplicity

→ Stability

→ Understandability

→ operability

→ Discomposability

Software Testing

White box → all condns executed atleast once

Black box

Basic Testing

Computing cyclomatic complexity :-

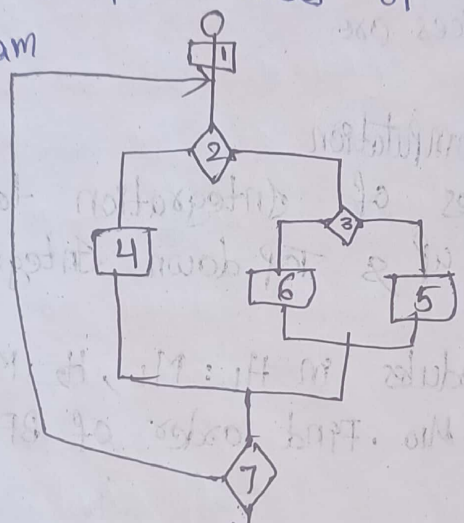
No. of simple decisions + 1

(or)

No. of enclosed areas + 1

Basis Path Testing

Identifying no. of test cases of Independent Paths given program



1 2 3 5 7 8

1 2 3 6 7 8

1 2 4 7 8

1 2 4 7 2 4 7 8 ... 2 4 7 8

Control structure Testing

→ condn testing (do while, go to)

→ Data flow testing

White box testing → Loop Testing

Loop Testing

1) Simple Loops
2) Nested loops
3) concatenated loops
4) Unstructured loops

Black-Box Testing

functionality and mismatching errors

Graph Based Methods

Equivalence Testing

data

FK input

prompts

output formats

mouse picks

user queries

- Write a program Quick sort, Find cyclomatic complexity, loop testing, no. of possible test cases using basic path testing?

Date 3/4/24

OO Testing

Object Oriented Testing

objects, class, encapsulation.

Three Things to be done:

1) Error discovery techniques applied to object-oriented analysis and design models.

Strategies

1) Unit Testing
2) Integration Testing
3) Validation Testing

Integration Testing:
- thread-based Testing (event or operation)
- use-based Testing
- cluster Testing

Testing Methods

1) Fault-based testing
2) class Testing and the class Hierarchy
3) Scenario-Based Test Design

Password recovery using State chart Diag → ~~ERMP~~

OOT Methods : Random Testing

→ Partition Testing
  → state_based partioning
  → attribute_based Partioning
  → category_based Partioning

1) write a c program which consist of 4 Prototypes without using any build-in fn's

Prototype 1 → find length of string

Prototype 2 → Reverse of string

Prototype 3 → find Substring with beginning & ending position

Prototype 4 → replace a letter instead of given letter

Prototype 5 → Palindrome & not

2) Develop a menu driven program with atleast 10 elements of an array & perform linear and binary search of a given data element.

Design a flow graph of program 1 & 2 and find the cyclomatic complexity, no basis path testing and identify no. of possible testcases of conventional software development.

A)
```
#include <stdio.h>
#include <string.h>
int main()
{
  char str[10];
  Printf("Enter the string:");
  scanf("%s", &str);
  Printf("Length of string:";
  Scanf("%d", &str);
}return0;
```

Website Designing

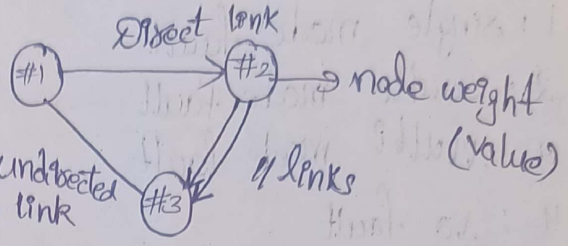
Date 9/4/24 Black box Testing

→ Behavioral and functional Testing

Finds → Incorrect & Missing fn's
→ Interface errors
→ Error in data structures.

Graph based Testing

1) find objects → Object #1
2) Find its relationships (symmetric, uni, bi)
3) Assign weight / state



Test Methods :-

1) Transaction flow Modelling
2) Finite state Modelling
3) Data flow Modelling
4) Timing Modelling

FTW = 0.62XGW

1) Diagram
2) VB
3) web

Equivalence Partitioning

1) Input is range → 1 Valid

Orthogonal assay testing (mass)

When input domain is relatively small but too large in exhaustic testing.

E= edges N=Nodes
P=Predicate Nodes

V(G) = E-N+2
V(G) = P+1
V(G) = R includes start and end boundary.

Predicate nodes &
Cyclomatic complexity &
Basis path testing, selective path testing &
How do you identify paths to calculate no. of testcases in conventional software &
Develop menu based calculate program with 5 options + , - , * , / , exit &
Draw flow graph

Date

2/10/24 Orthogonal array Testing

Applicable to problems.

Used to reduce testcases in inputs.

To find the region faults

Fax Sending scenario

P_1 = send it now

P_2 = sent it one hour

P_3 = sent after midnight

Case 1 : Single mode fault

Case 2 : Double mode fault

Case 3 : Multi mode fault

Case 4 : No fault

4 cal, 3 levels, find array

Calculate OAT = Lruns (levels ^{factors})

Runs (N) - No. of rows

factors (K) - No. of columns

Levels (V) - Max no. of values

Steps

1) Identify independent variable

2) Find smallest array with no. of runs

3) Map the factors to the array.

4) choose the values for any leftover.

Exp:1 No. of factors = 3 (Top, Bottom, middle)

No. of levels = 2 (Hidden, shown)

Array Type = $L_4(2^3)$ = 8 test cases.

reduced as 6 test cases through OAT

Ex 2: Temperature : 100 c, 150 c, 200 c
Pressure : 2 psi, 5 psi, 8 psi
Doping Amount : 4%, 6%, 8%
Deposition Rate : 0.1 mg/s, 0.2 mg/s and 0.3 mg/s

A) Total 4 factors, 3 levels
conventional Test cases = 3^4 = 81 max. test cases
Test case #, Temp, P, DA, DR. → columns

Test case #	Temp	Pressure	Doping Amount	Deposition Rate
1	100	2	4	0.1
2	100	5	6	0.2
3	100	8	8	0.3
4	150	2	4	0.1
5	150	5	6	0.2
6	150	8	8	0.3
7	200	2	4	0.1
8	200	5	6	0.2
9	200	8	8	0.3

Advantages
→ finds regional errors, missed values, suspicious values/ cases.
→ Reduced no. of repeated TC
→ Allowed pair testing
→ Suitable for complex system, large input scenario.

1) start
2) Assignment
3) condn start
4) Multiple condn &
5) No. of branches &
6) end condn &
7) Again assignment &
8) any loop &
9) end

Date 07/04/24 MBT - Model Based Testing

UML diagrams are tested

food ordering system

foodie Primary actor

System secondary actor

⇒ Identifying subuses and subcases.

1) Enter login / Register
2) Search availability of Hostel, food
3) order a food by phone call
4) How far from you (GPS)
5) cancel the order
6) Make Payment
7) Deliver to customer
8) FB (feedback)
9) logout

⇒ Registration of the new user
1) show registration form
2) Get details from the user → Get Name, Get Ph.no, Email, Address / location, favourites
3) Store details to the DB
4) conform with ID

⇒ Search availability of Hotel
1) Search Hotel
2) Search food items → Veg, Non-Veg
3) Search location

⇒ Order food by a Phone call
1) list of Phno: of Hotels
2) Select language
3) Select food quantity
4) Suggestion about ingredients

5) Move to cast

⇒ check location
1) Give your location
2) Giving Hotel location
3) check dist

⇒ cancel
1) ⇒ is delay 2) long away 3) No service

⇒ Make Payment
1) check offer
2) check coupon
3) Bank balance
4) Mode of Payment → Payment @Rcode

⇒ Deliver the food
1) find address
2) track the order
3) suggest delivery points
4) Delivery boys
5) Provide tips

⇒ fb
1) show stars
2) Pick stars
3) show comment box
4) fill fb
5) Suggestion

⇒ logout
1) ⇒ logging out

-----▷ Reti Realization/ Implementation