4/01/24

SE

Submission - Project Report than Viva

Continuous Assessment in Lab

3 Members per group project in Batch

Project - 40 Marks

Assignment - 20 Marks

End Sem - 40 Marks (Theory)
(2hrs Exam)

Project Report in LaTex

Use Git for project individual memembe contribution.

Team & Problem statement - By the end of next week

Book - Pressman R SE A Pratitioners Approach.

3 Marks

iven below:
6, 5, 4, 3, 2, 1)
trix

Doc 2

1 (A)

2

1

...term) - 30 ...
...onal Assignments/Projects - 50 Marks.

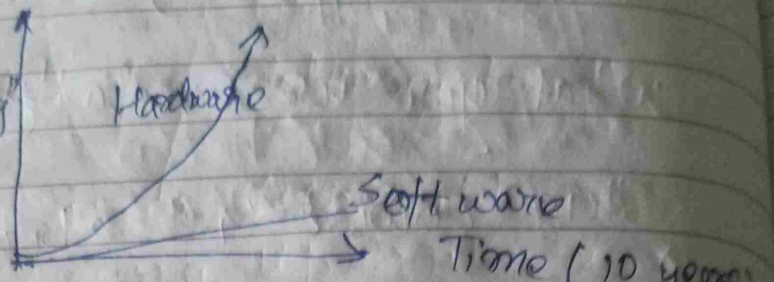
SE

...e Software failure
...ne 5
... problem
... Patriot Missile

...e Evolving Role Of Software.
... project cancelled befor completed.
... over-run cost estimate
... / 100 . restart.

Project failure Percentage Pie Chart

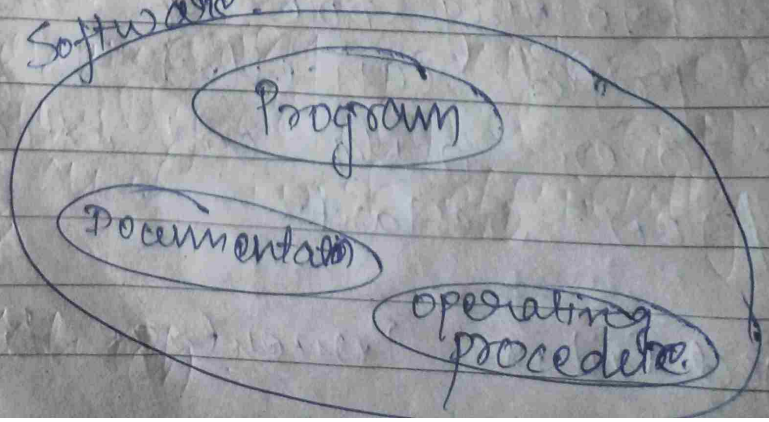
- Unlike Hardware.
- Moore's law: processor speed/ memory capa doubles every two years



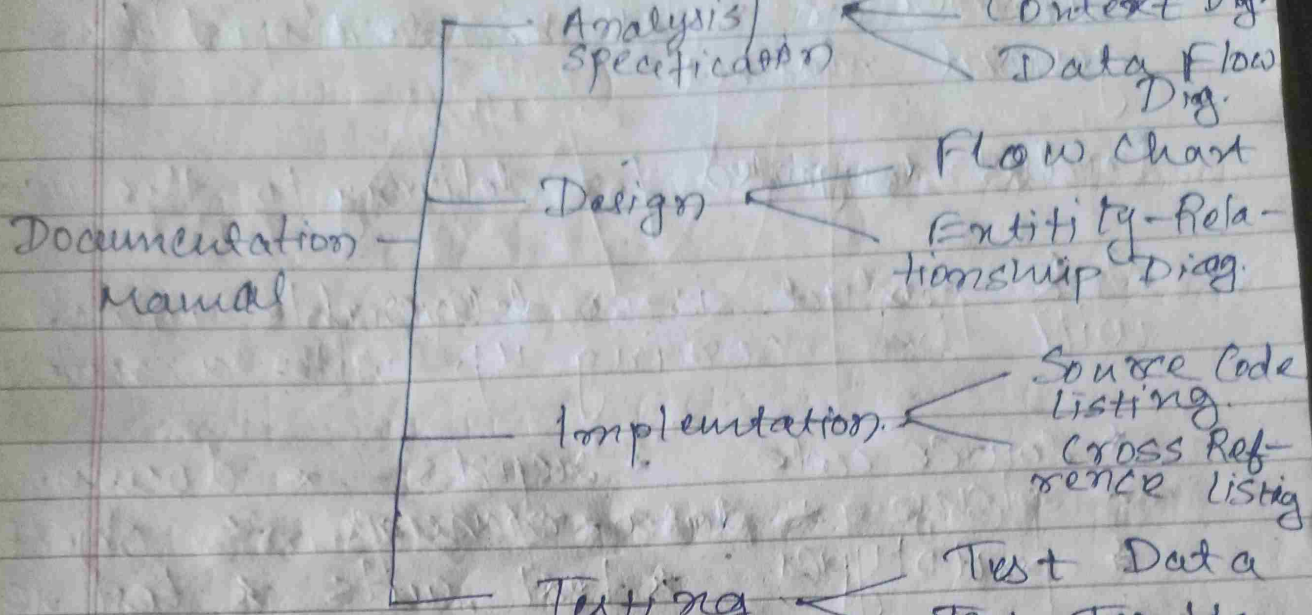
- Mangers and Technical Role of Software:
  - * why long to get prog. finished?
  - * why cost high?
  - * why we can't find all err befor release
  - * why difficulty in measuring prog. of SE development?

- factors contr. to software crisis.
  - * Larger Problems.
  - * Lacks of adequate training in SE.
  - * Increasing Skill Shortage.
  - * Low productivity improvement.

Software - Computer programs and asso documentation.



Documentation consists of diff types of manuals are.



List of Documentation manuals

11/1/24

SRS - Software/System Requirement Specification

* Prepare a project plan
* Gantt Chart

1- Assignment SRS IEEE standard

Operating Procedures

- User manuals
  - System Overview
  - Beginner's Guide Tutorial
  - Reference Guide
- Operational Manuals
  - Installation Guide
  - System Administration Guide

Software Product

It may be developed for a particular customer or may be developed for a general market.

Product designated for delivery to the use

Work product is a deliverable which will be coming up with a milest

* source code * reports * docume
* plans * data * manuals * objects
* prototypes * test suits * test res

Software Engineering - Engineering discipl which is concerned with all aspects of software production.

Software Engineer should
- adopt a systematic and organised approach to their work.
- use appropriate tools & techniques depend on • the problem to be solved
  • the development contraints and
- use the resources available.

* 1968, Fritz Bauer.
Sound engineering pricipe → obtain economically developed software → reliable and work efficiently on real machine

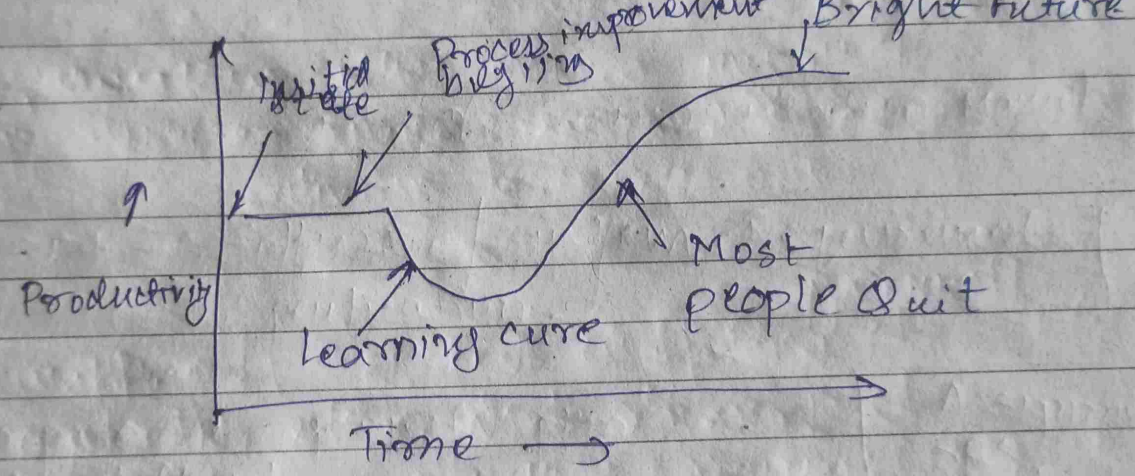
* Stephe Schach.

Production of quality software → delivered on time within budget → satisfies requirement

With time, maintainability is also an import factor.

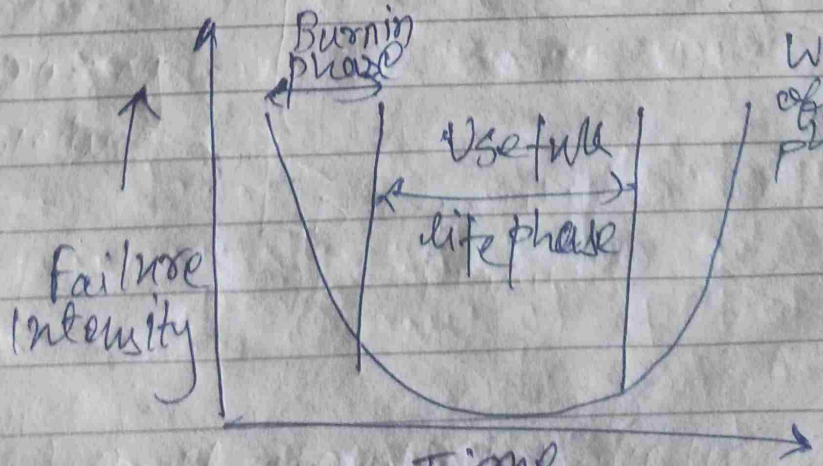
* Software Process – Way in which we produce software.

Difficulty to improve software process
- ↳ Not enough time
- ↳ Lack of knowledge
- ↳ Wrong motivation
- ↳ Insufficient commitment



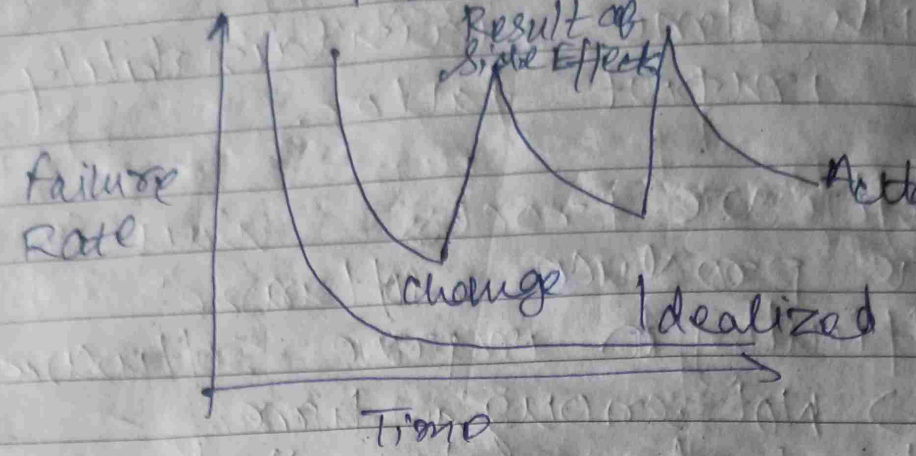
* Software Characteristics

- Software does not wear out



- Software is not manufactured
- Reusability of components
- Software is flexible

Failure Curve for Software



The changing Nature of Software.

Overview

Software Myths (Management Perspectives)

Software Development Process and Process Models:-

Overview

Software Process → framework for the tasks that are req' to build high quality s/w.
Process may comprise a lot of activities
Activities → Actions → Tasks

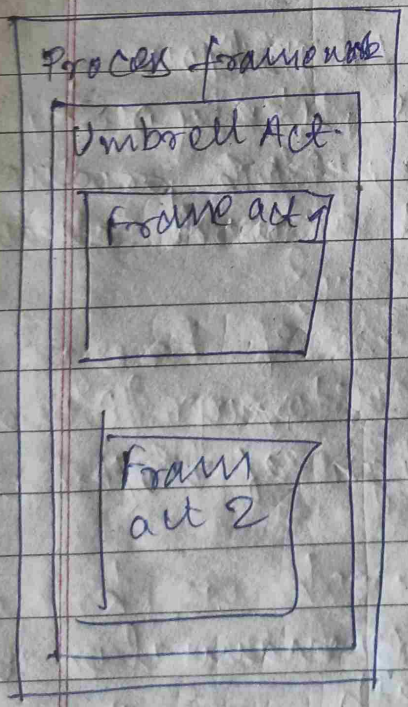
Who? → Software Eng, Manager, Custo.

Why? → Provides stability, control and org to an otherwise chaotic activity.

What? → Handful of activities are common to all s/w process, details vary.

Work product? Program, documents etc

Process Framework



Frame Activities
↳ work tasks
↳ work products
↳ milestone & deliverable
↳ QA checkpoints

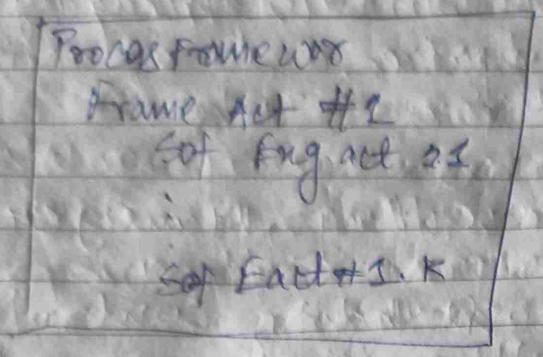
↓

Quality Assurance

→ Formal Technical Review (FTR)

Informal
Formal
Informal

- Each frame activity is populated by a set of software engineering actions — a collection of related task
- Each action has individual work task



* Generic Process Framework Activities

Communication:
↳ Heavy comm with customer, stakeholders
↳ Encompasses requirements gathering and related activities

Planning
↳ Work flow that is to follow
↳ Describe tech risks, likely risk, resource will require, work products to be produced and a work schedule

Modeling
↳ Help developer and customer to understand requirements & Design of software

Construction
↳ Code generation: either manual or automated
↳ Testing - to uncover both error in the code

Deployment → Delivery to customer for evaluation
↳ Customer provide feedback

Risk → Unwanted probable event which may occur in future.

The Prags Model : Adaptability

Umbrella Activities

- Software project tracking and control
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Document prep & production
- Reusability management
- Measurement
- Risk Management

15/4/24 VLSI

25/1/24

SE

→ CMMI — Capability Maturity Model Integration
given by SEI
Key Process Areas — KPAs (Chart See)
↳ to achieve goals.

CMMS levels

Levels (Initial)
" 2 (Managed)
" 3 (Defined)
" 4 (Quantitatively Managed)
" 5 (Optimized)

L1 → poorly managed or controlled
↳ outcomes unpredictable
↳ adhoc & chaotic proces
↳ No KPA
↳ Low Quality High Risk

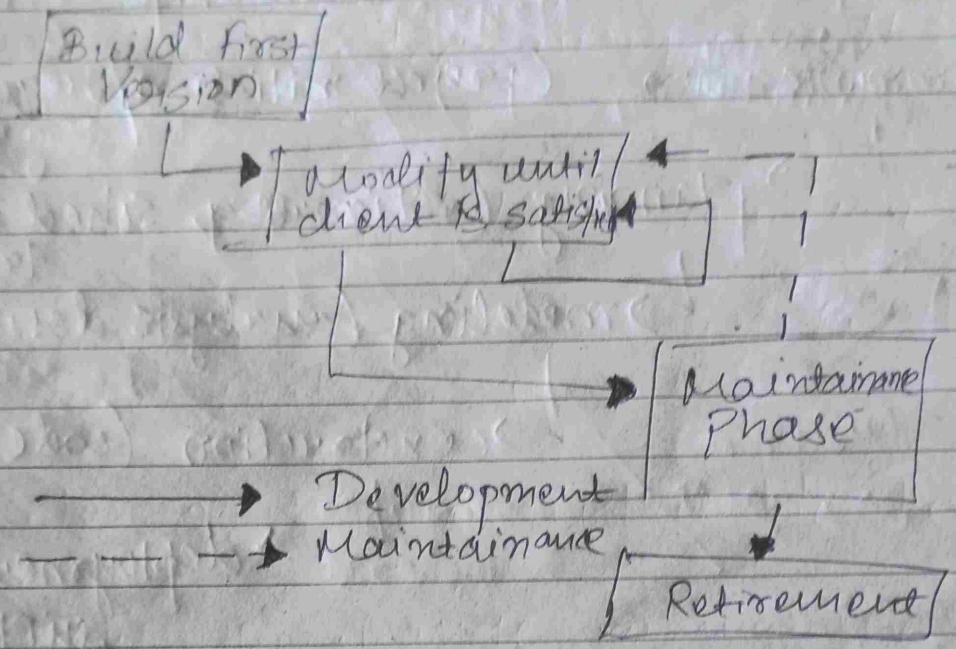
L2 → req are managed
↳ planned & controlled proc
↳ documented plan
↳ lower risk
↳ Better Quality

L3 → process characterised using stored proper procedure & method tools
↳ medium quality & risk
↳ process standardization

L4 → Qualitative objective → based on
↳ performance & Quality set
customer org

Software Process Model:-

★ Build & fix model:-



★ Process as a black box:-
★ Process as a white box:-

★ Prescriptive Model
★ Waterfall Mode or Linear Sequential

★ Incremental Process Model
↳ Incremental Model
↳ RAD model

★ Evolutionary Process Model.
↳ Prototyping → Concurrent Develop Mode
↳ Spiral Model → Model.

- Fourth Generation Techniques (4GT)
- Component Based Development

* Agile

29/1/24

Prescriptive Model:-

- orderly approach to software engineering

Waterfall Model or Classic Life Cycle:-

Communication (project initiation/requirement gathering)

↳ planning (estimating, scheduling, tracking)

↳ modeling (analysis design)

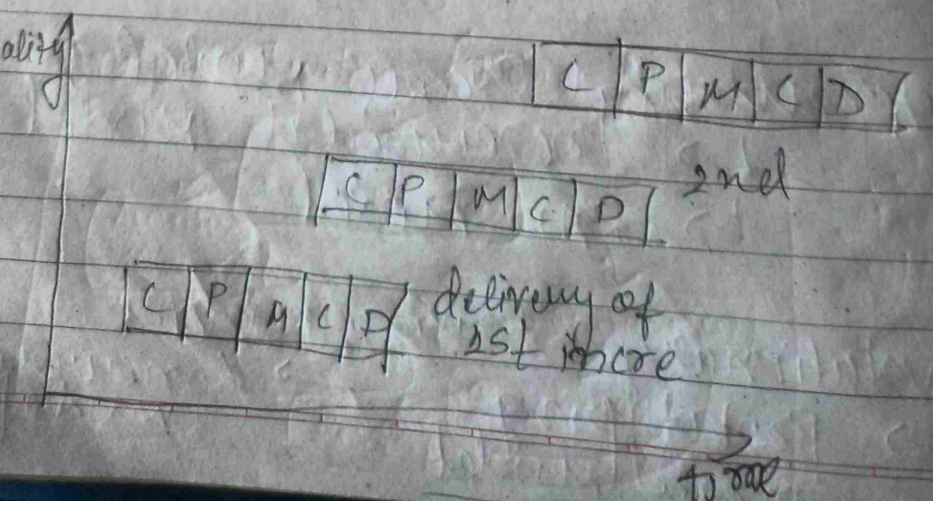
↳ construction (code test)

↳ deployment (delivery support fee

Limitations of Waterfall Model:-

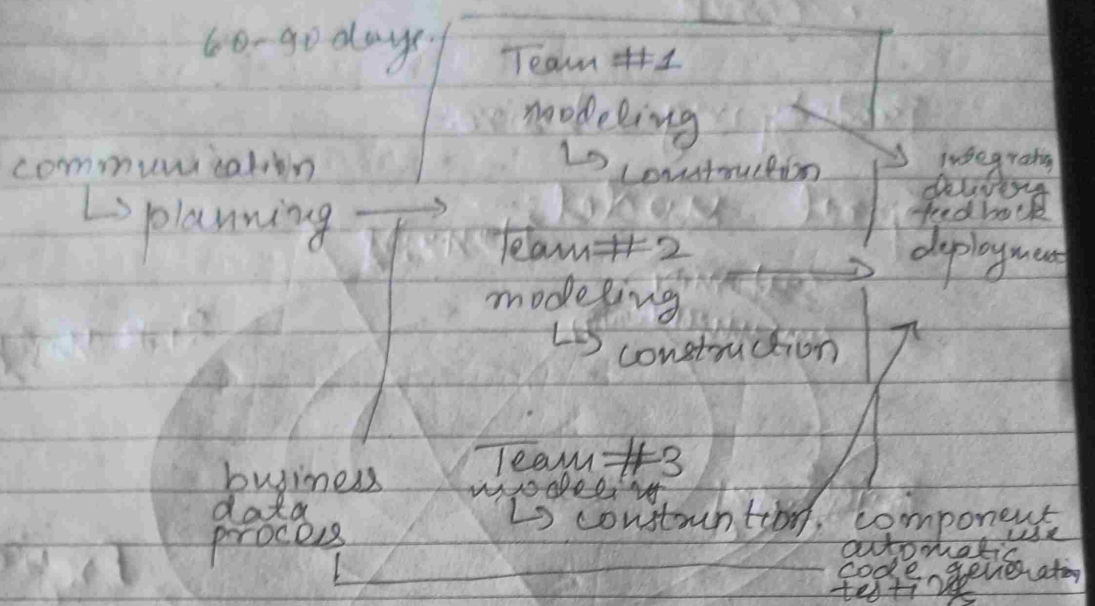
Problems:-

Incremental Process Model:-



Deliver software in small but usable pieces, each piece builds on pieces already delivered.

Rapid Application Development (RAD) model –



Makes heavy use of reusable software components with a extremely short development cycle.

5/2/24 ⇒

Modeling ⟶ Business – info flow among busi.
↳ Data – info refine in set of data obj
↳ Process – Data obj tranform to info flow.

Communication – to understand business prob

Planning – multiple s/w teams work in parallel on dif system

Construction – use of pre-existing s/w comp.

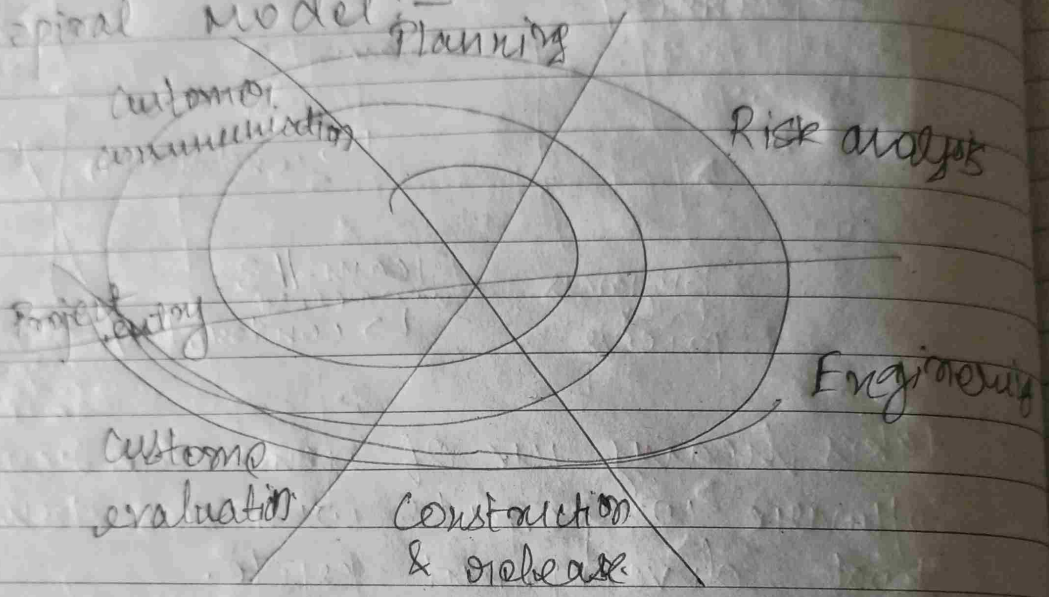
Deployement – delivery basis for sub. iterate

- RAD model emphasizes on short development cycle
- High speed edition of linear seq model.
- Enable construction in short time if req are understood & project is constrained.

Evolutionary Process Model.
→ increasingly more complet version of
software with each iteration & are de
- Prototyping
- Spiral Model
- Concu

Prototyping cohesive:-

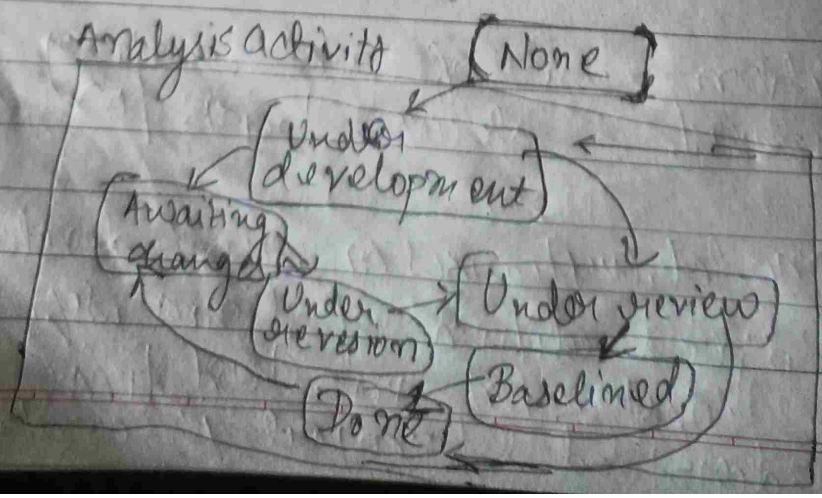
Spiral Model:-



Product maintainance project
enhancement
New project development projects
Concept development project

in to out

Concurrent Deployment Model:- (Not for project)



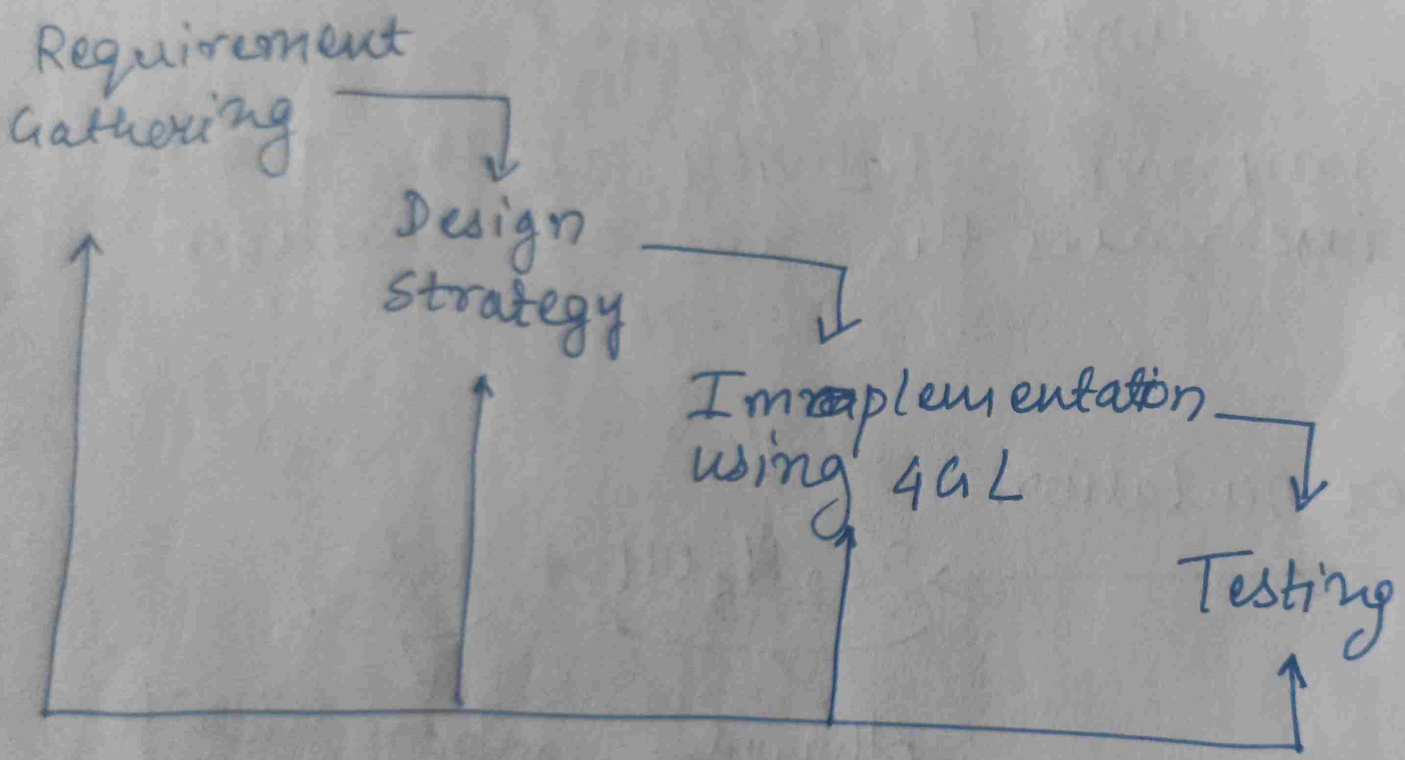
(Server)

React.

23/02/24

SE

Fourth Generation Technique (4GT)

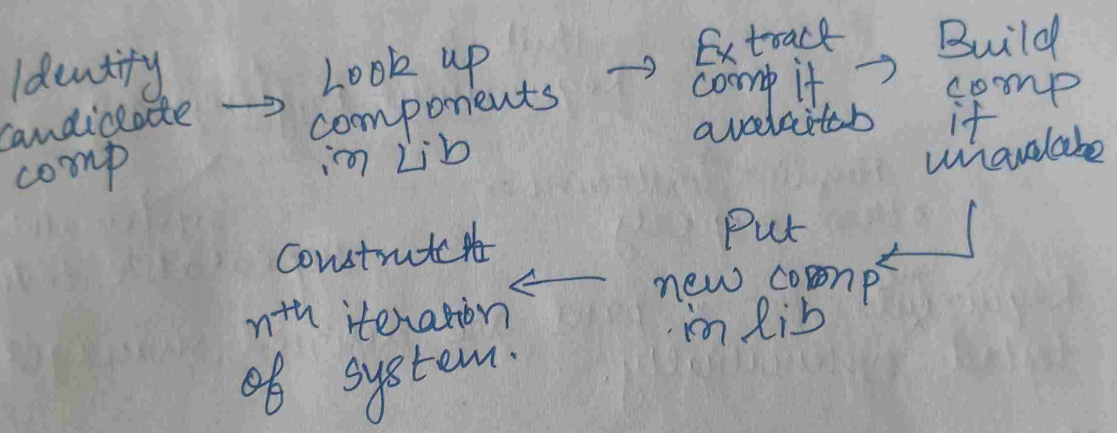


Merits: Reduce SD time
Improve prod

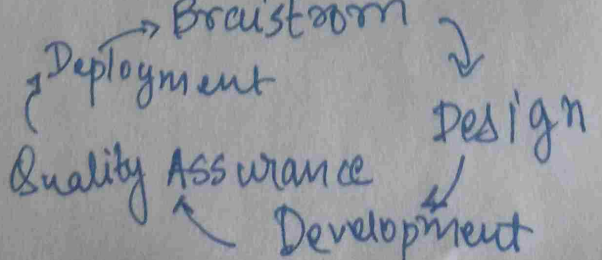
Demerits: Not much easier to use
Large Maintainance

Component Based Development:-
- incorporates spiral model characteristics
- evolutionary
- iterative approach to creat s/w
- creation using prepackaged comp (classes)

Add int Engineering & construction & release phase



Agile Model
- Based on iterative development
- Break task into smaller iterations.
- number of iterations planned
- Each iteration → frame → one to four weeks.
- smaller part → minimize risk → reduce delivery time.
- each iteration → a team working through a full software development life cycle.



Agile Testing
Methods
SCRUM
crystal
DSDM FDD
Lean s/w development
XP

When to use:-
- Freq. changes
- ~~less~~ good team
- customer ready to meet all time.
- small project.

Adv
- Freq delivery
- F to F comm
- Efficient design
- anytime change.
- reduce dve tim

Disadv
- shortage of formal document
- maintainance becomes difficult.

Unit-2
Requirement Engineering

Function, constraint or other property

systematic & repeatable techniques

Requirements for a prod are defined, managed and tested systematically.

Characteristics of Good Req.
- Clean & Unambiguous
  * Std. structure.
  * one interpretation
  * one req → one statement.
- Correct
  - req contribute to real needs.
- Understandable.
- Verifiable
- Consistent
- Complete
- Traceable.

Req. Engineering Tasks.

Inception - basic understanding of problem and nature of sol
Elicitation - requirements from stakeholder. (draw)
Elaboration - create analysis model that rep info, func & behavior of the req
Negotiation - agree on deliverable sys that is realistic for developer & custom
Specification - desc the req. formally
Validation - review the req specification for error, ambiguities, omissions & conflicts
Requirement management - manage changing req.

Data Science

Similar Documents - Shingling

SE

~~context free questions asked to establish the problem~~

Analysis Modeling Principles:-

- info domain of prob must be rep & understood. Flow
- ~~Functi~~ responss data flow
- Function - direct benefit & internal support
- Behaviour - interaction with external env.
- stragety of analysis model - partitioning
- essence of prob w/o consideration of implementation

Objectives

- customer req
- basis of creation of s/w
- set of req for validation after s/w built

Analysis rules of thumb:-

- High level of abstraction (only focus on req visibility)
- Ele of AM should give ~~coverages~~ understanding of s/w requirements
- Provide insight into info domain, function & behaviour of the system.
- Consider delay & other non-functional model until design
- minimize coupling - reduce high interconnectedness. (cohesiveness ~~is~~)
- Value to all stakeholder, model used as basis of design
- Keep model simple.

Elements of Analysis model:-

Two approches

Structured

- Data Objects - attributes & relationship
- Process manipulate data in a manner that show how data transform as they move through sys

object oriented.

focuses on classes & how they collaborate

Eg UML

Analysis Modeling Principles:-
- info domain of prob must be rep & understood. then
- ~~Function~~ imposes data flow
- Function - direct benefit & internal support
- Behaviour - interaction with external env.
- stragety of analysis model - partitioning
- essence of prob w/o consideration of implementation

Objectives
- customer req
- basis of creation of s/w
- set of req for validation after s/w built

Analysis rules of thumb:-
- High level of abstraction (only focus on req visibility)
- Ele of AM should give ~~coooooggers~~ understanding of s/w requirements
- Provide insight into info domain, functions & behaviour of the system.
- Consider delay & other non-functional model until design
- minimize coupling - reduce high interconnectedness. (cohesiveness ~~xxxx~~)
- Value to all stakeholder, model used as basis of design
- Keep model simple.

Elements of Analysis model:-

Two approches

Structured
- Data Objects - attributes & relationship
- Process manipilate data in a manner that show how data transform as they move through sys

object oriented.
focuses on classes & how they collaborate
Eg UML

Scenario - based elements
Use case text/diagram
Activity diagram
swim lane "

Flow - oriented element
DFD
CFD
Processing narrative

Class - based element
Class dia
Analysis packages
CRC models
Collaboration diag

Behavioral element
State diagram
sequence diagram

DAA

SE
Unit III

System Architecture & Design Overview

- structure or structures of system, comprise the s/w, externally visible components & relation b/w them.
- not operation s/w but a rep that enables s/w engineer to -
  - Analyze the effectiveness of design in meeting its stated req
  - consider architectural alternative
  - reduce risk associated with construction of software.

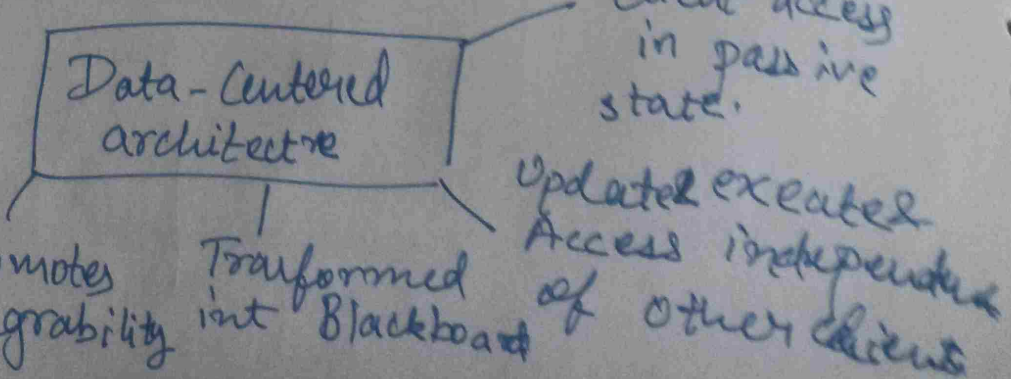
Importance:-

~~Scope of this work~~

Architectural style:- It describes a sys category that encompasse:-

1) A set of component
2) " " connector
3) Constraints.
4) Semantics.

Rep by - Data centered arch
- Data-flow
- Call & return
- Object oriented
- Layered.



Data Flow:-
→ Input data transformed → series of comp & manipulative components to get output data.

Call & Return Architecture:-

Design Engineering:-
- covers set of principles, concepts & practices that lead to the development of a high quality system or product
- Produce a model that depicts:-
  * Firmness
  * Commodity
  * Delight
- Details out → software data structure & architecture

Software Design:-

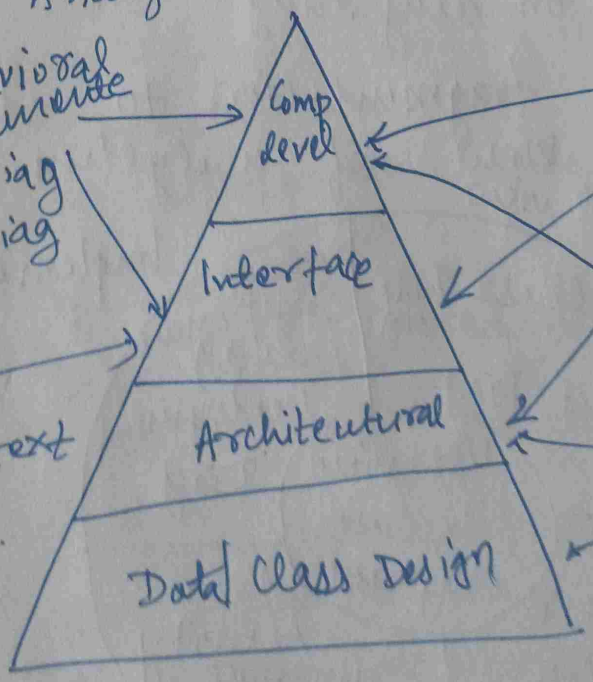
Data/ Class Design

Architectural Design

Interface Design

Component Design.

Translating Analysis → Design.



Behavioral elements
State Diag
Seq. Diag

Scenario Based
Use-cases - text
Use-case diag.
Activity diag
Swim lane diag

Flow oriented elements
DFD
CFD
processing narrative

Class based elements
Class Design
Analysis packages
CRC models.
Collaboration Design.

Data/Class Design – Transforming analysis model class based elements into classes & data structures

Architectural – relationship among major structural elements.

Interface – communication b/w elements.

Component-level → Structural elements
↓
Procedural Description of the s/w compo

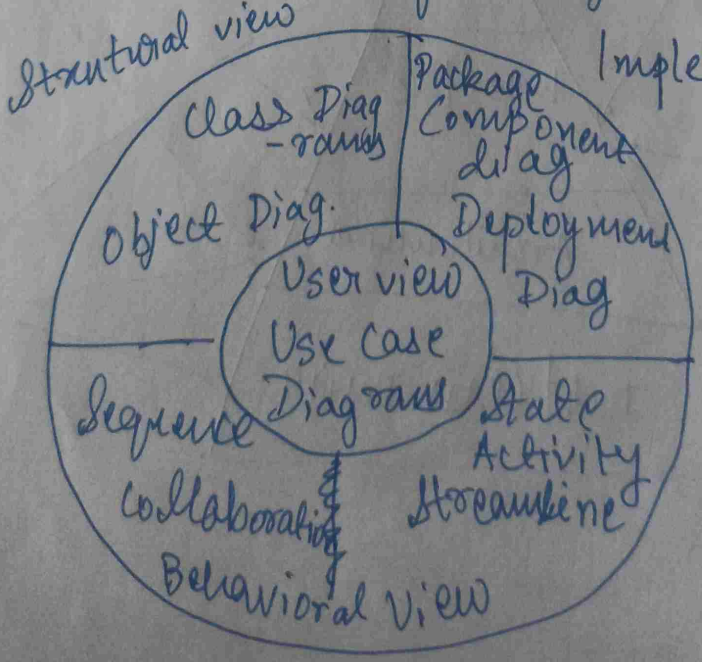
* Goal of design process

What is UML and why we use UML?

UML → Unified Modelling Language.
↓
Describing a s/w system at a high level of abstraction

Industry standard graphical lang for specifying, visualizing, constructing and documenting the artifects of s/w system.

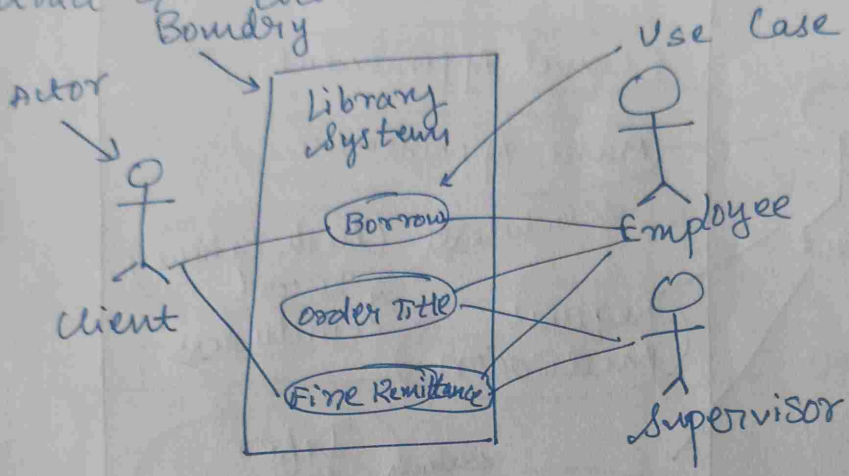
* B/w natural lang (imprecise) & code (too detailed.
* not dependent on any lang
* moves us from fragmentation to standardization.
* acquire overall view of a system..



Use-Case Diagrams

- Set of use case
- Model of interaction b/w
  - * External user of of s/w prod (actors) and
  - * The s/w product itself
  - * More precisely, an actor is a user playing a specific role.

- describe user scenarios
- capture user req.
- contract b/w end user and s/w developers.



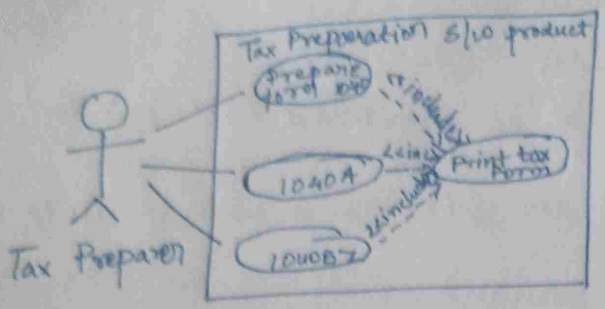
Actors - role that user plays w.r.t systemtem

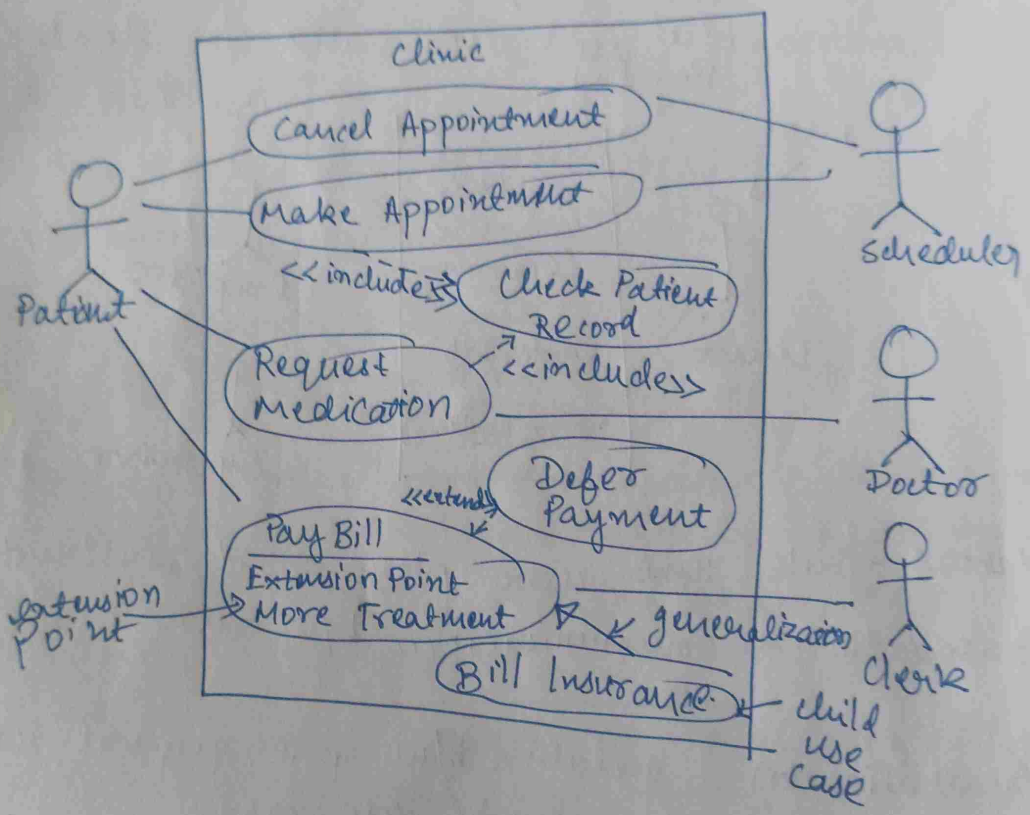
Association - Communication b/w actor & use case

Generalization - relation b/w one general use case & a special use case.

Include - Base use case <<include>> include use case.
when a chunk of behaviour is similar across more then one Use case

Extend - <<extend>> base case.
Add behaviours to base use case.
Base case declares "extension points"





Class Diagram:-

- Classes & their interrelationships.
- Describe structures & behaviors in use cases.
- Conceptual model of sys in terms of entities & their relationship
- Requirement capture, end user interaction.
- Used by developer

Entity → Classes
Attribute → Properties
Behaviors → Methods

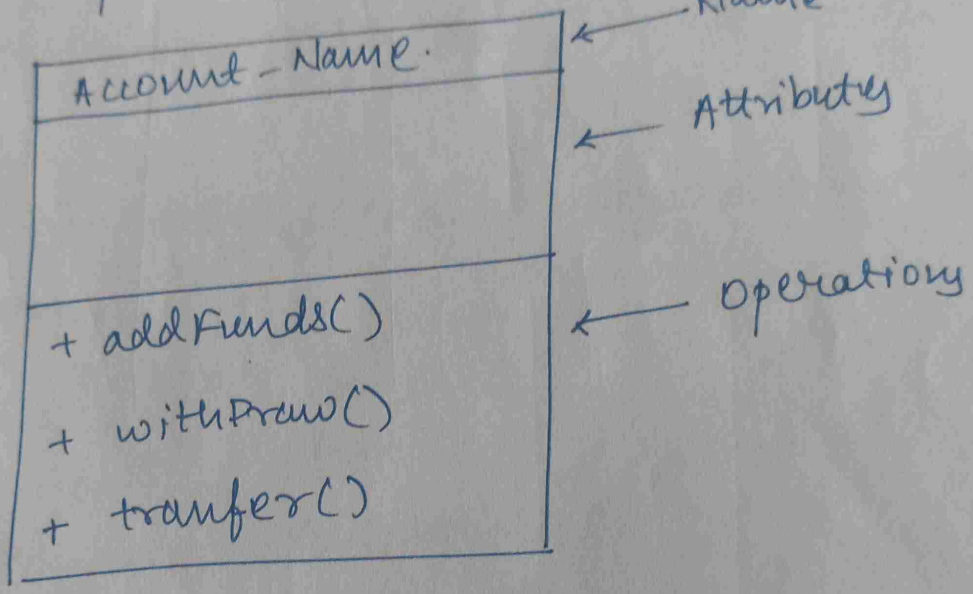
Class — rectangle divided into

- Name
- Attributes
- Operations

Modifiers → indicate visibility of attribute & op.

- \+ — Public (everyone)
- # — Protected (friends & derived)
- — Private (no one)

Default → attributes – hidden
operations – visible



OO Relationships.

- Generalization (parent-child)
- Association (student enrolled in a course)
  - Aggregation
  - Composition.

SEMP

OO Relationships

Association

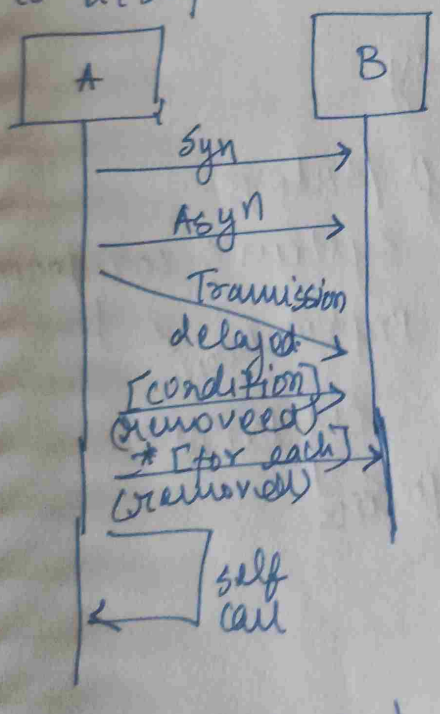
- Aggregation
  relation among instances of related classes
  Container - Containee relationship
- Composition
  strong form of association

Class Diagram

Interation Diagram

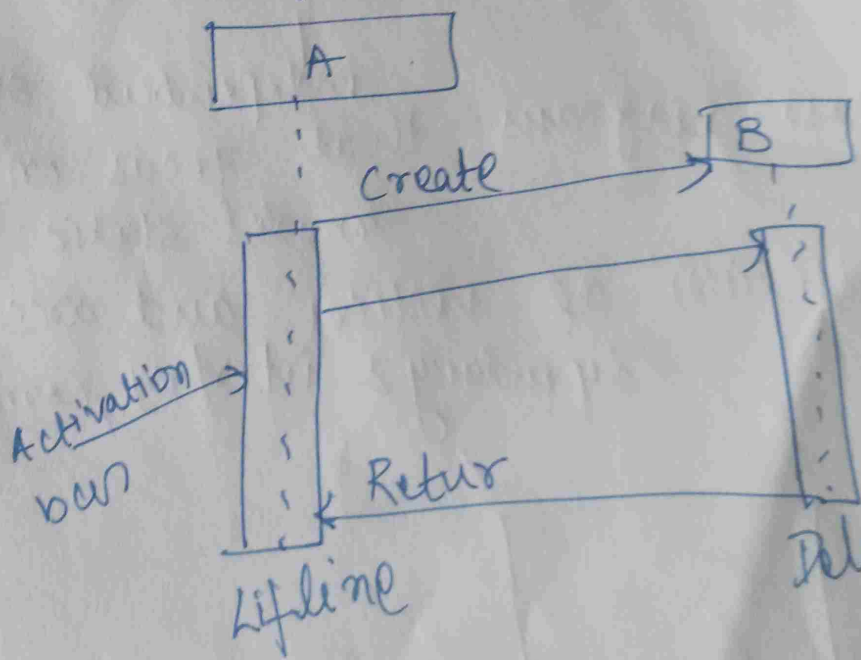
- Sequence Diagram
- Collaboration.

Self call: A message that an Object sends to itself



Condition: indicated message sent. Sent only when condition is true

Object Life Span
Create Message.
Creation → Object Life starts.
Activation →
Symbolize by rectangulas stripes.
Object activated



- Tell behaviour of object

Collaboration Diagram:-

Focus on class

State Diagram:-

Activity Diagram:-

Used for rep. the flow of interaction from one activity to another in the form of graphical rep.

Elements → Fork — rep multiple parllel
→ Branches
→ Merge. — combines multiple branches
→ Join — control and syn parallel flows.

Swimlane Diagram:- (Skip)

Visualy distinguish job & responsibilites.

Rummler-Brache diagram

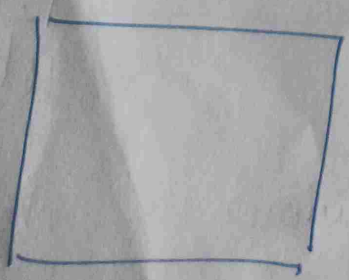
Implementation View:-

Component Diagrams:-

relation b/w component of sy
independent sys/subsys.

class diagrams that focus on system's component model static implementation.

collection of vertices and arcs contains dependency & interface, components



Basic Components.

↳ rectangle → component's name
→ icon
stereotype text and/or icon

Interface

↓ Provided

→ Required.

Ports.

Port
expose req and provided interface of a component

Request Provided.

- - - → dependencies
↕ hollow (compulsary)

One diagram

Deployment Diagram:-

↳ execution architecture of system

nodes

hardware execution env — s/w — middleware.

Visualize → physical h/w and s/w of sys.

node → execution of code & comp interaction

<<artifact>> — in progress component

node

physical machine — environment

SE

Software Metrics

Measure of s/w characteristics quantifiable or countable

Product
Process
Project

Reasons of Measure:
- characterize
- Predict
- evaluate
- improve.

Decision
- strategic
- tactical

Key Process Indicator

Process Metrics — based on outcomes of the process
- error
- defect
- Calendar Time
- Time & effort to complete each generic activity.
- work product
- Human Effort
- Conformance to the schedule.

Project Metrics
- s/w project mange to
  - ass status
  - uncover prob area
  - potential risk
  - adjust work flow or tasks
- Estimat time & effort
- Error uncovered during generic framework activity (CPMLD) are measured

Productions rates.
- models created
- review hours
- function points
- delivered source lines of code

Lines of code
↳ comment & header lines ignored.
↳ simplest

COCOMO (Constructive Cost Model)
↳ By Boehm
↳ efforts in terms of "person-months" of the technical project staff

Three modes of COCOMO

organic
↳ small project team
↳ good application for less rigid req

semidetached

embedded.

(1) BASIC COCOMO

$$\text{Effort} = a_1 \times (\text{KLOC})^{a_2} \text{ PM}$$

$$\text{Tdev} = b_1 \times (\text{Effort})^{b_2} \text{ Months.}$$

↓ development time

		a_1	a_2
Effort	organic	2.4	1.05
	semi-detached	3.0	1.12
	Embeddes	3.6	1.20
Tdev		2.5	0.38
		2.5	0.85
		2.5	0.32

KLOC = 32 Salary – 15000 /-per mnth

Effort = $2.4 \times (32)^{1.05} = 913.31$

Tdev = $2.5 \times (913.31)^{0.38} = 33.34$ months

Cost = 15000×33.34

No. of people = $\frac{913.311}{33.34} = 28$

$28 \times 15000 = 4,20,000$

Prediction ~~– creation date time~~

Project Metrics (Function Point Analysis)

↳ By Albrecht [1983]

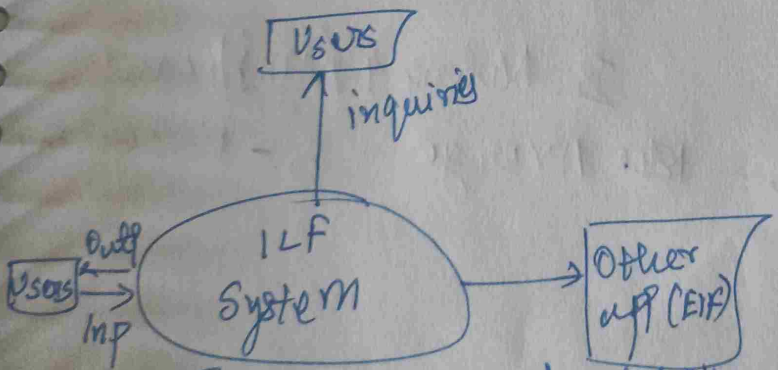
↳ estimate size of s/w project from problem specification.

→ size → no. of functions
→ size → feature

→ funtion ⟹ read data → transform → output

→ size → no. of files
→ size → " " interface.

→ FPM computes size of s/w prod (in FPs)



ILF – Internal Logical files

ELF – External " "

Measurement Param	Example.
① No. of external inputs (EI)	Input screen and tables
② No. of external outputs (EO)	Output Screen and reports
③ No. of external Inquiries (EQ)	Prompt & interrupts
④ No. of internal files (ILF)	Databases & directories
⑤ No. of external files (ELF)	shared databases & routins

(1) Function Points(FP) = UFP * TCF

(2) TCF = (0.65 + 0.01 * DI) .... Technical Complexity Factor (TCF)

Total degree of influence (DI) is summation of total 14 influencing factors

or

(2) CAF = 0.65 + 0.01 * F .... Complexity Adjustment Factor

VAF = 0.65 + 0.01 * DI .... Value Adjustment Factor

F = 14 * scale.

(3) Unadjusted Function Points(UFP)

Function Units	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Values

UFP = Function units . Values.

0 - No influence 1 - Incidental 2 - Moderate 3 - Average 4 - Significant 5 - Essential

SE

FP = ? Productivity = ? Cost per function = ?

PF – 5, 1, 0, 4, 3, 5, 4, 3, 4, 5, 2, 3, 4, 2

EI – 22 (Avg) EO – 45 (Low) EI – 6 (High) ILF – 5 (Avg) ELF – 2 (Low)

UFP – 22×4 + 45×4 + 6×6 + 5×10 + 2×5

= 88 + 180 + 36 + 50 + 10

= 268 + 96 = 364

TCF = 0.65 + 0.01 (45)

= 0.65 × 0.45

= 1.10

364 × 1.1 = 400.4 FPs

Productivity

Unit – 6

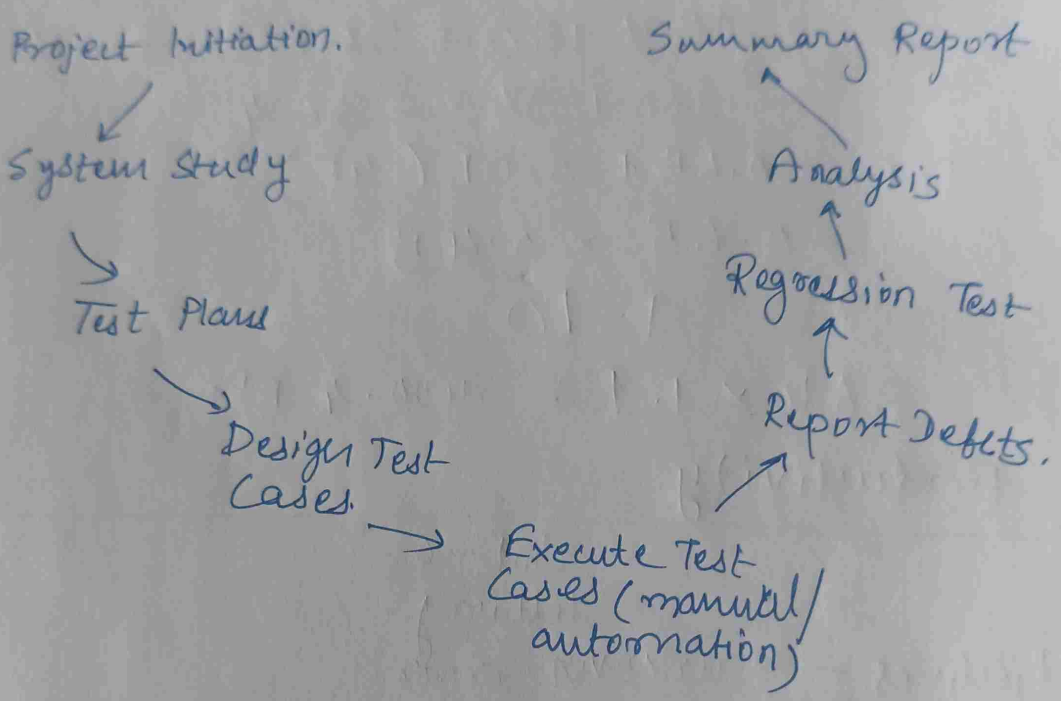
Testing

Objectives →
- uncover error
- demonste req.
- validate quality
- minimum cost & effort
- High quality test cases, perform effective tests & issue correct & helpful problem reports.

Error – Human action → incorrect result
produce fault

Bug – presence of error at time of execution
Fault – state of s/w caused by an error
Failure – Deviation of the s/w from expected result. It is an event

Testing Life Cycle :-

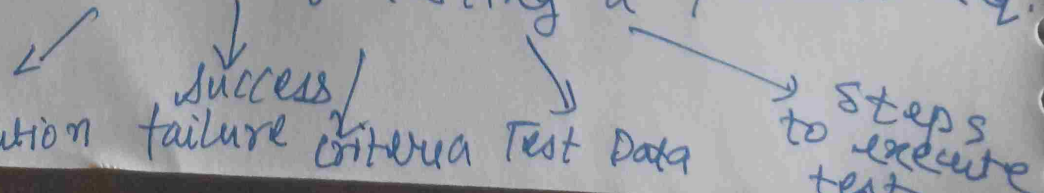


Regression Test – Test ripple effect due to high coupling.

Industry Standard – Cohesion & Coupling
50 50

Test Plan – Systematic approch to test a system.

Test Case – Procedure of testing a particular req.



Verification
↳ confirm to its specification (Building the product right)

Validation.
↳ s/w should do what user requires (Building the right product)

Testing Methodologies.

Black box (Functionality) no knowledg of internal program design or code required.	White box (Structural) Knowledge is there
Test based on requirement & functionality	Test based on coverage of code statements, branches, paths, conditions.

Testing Levels
↳ Unit
↳ Integration
↳ System