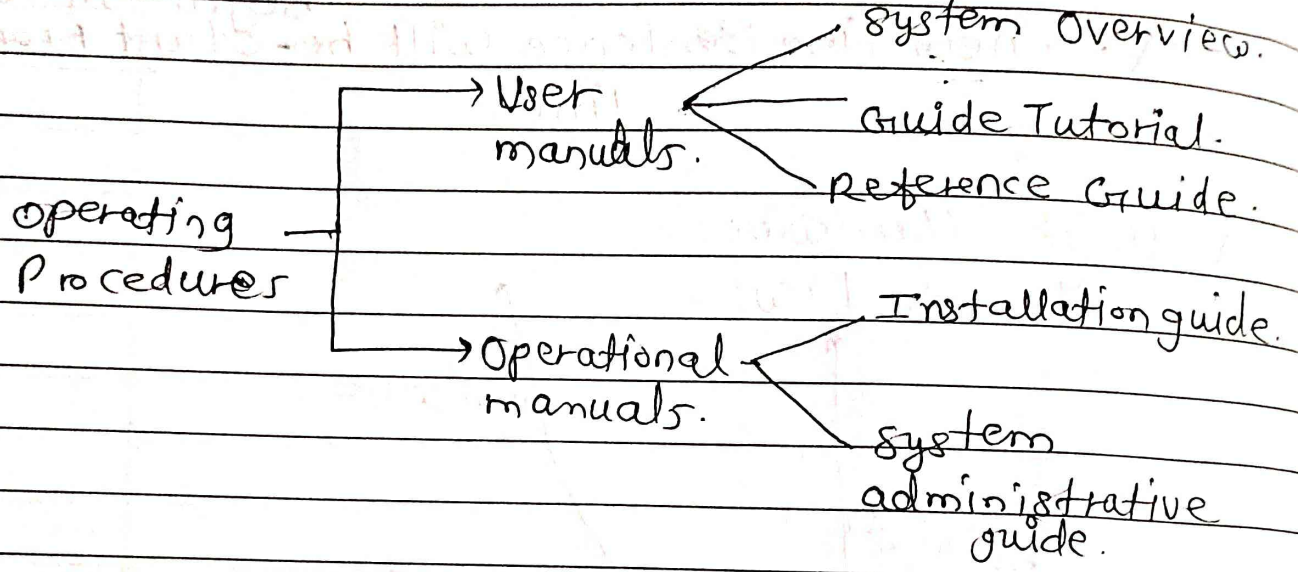
PAGE No.
DATE / /

✱ Some Software failures :-
1) Windows XP
2) Financial Software.

* What is Software?
Set of Computer programs & associated documentation.

Software = Program + Documentation + operating procedures.

✱ Documentation Consists of different types of manuals are -



✱ Software Products :-
1) Generic :-
2) Bespoke :-

Software products is a product designed for delivery to the user.

1) source code
2) reports.
3) document.
4) Object codes.
5) plans.
6) data.
7) prototypes.
8) test results.

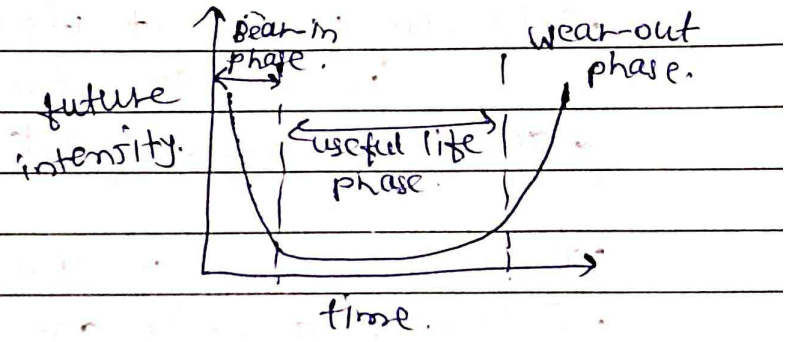
PAGE No.
DATE / /

The establishment and use of sound engg. principles in order to obtain economically developed software that is reliable & works efficiently on real machines.

A discipline whose aim is the production of quality s/w that is delivered on time within budget

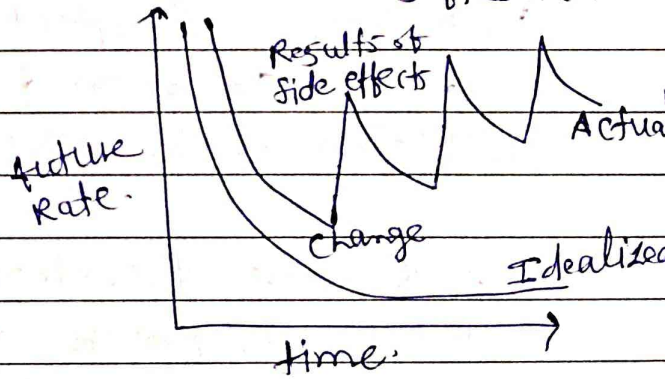
★ Why is diff. to improve software process?
→ 1) Not enough time.
2) Lack of knowledge.

★ Software Characteristics:-
- Software does not wear out.



- Software is not manufactured.
- Reusability of Components.
- Software is flexible.

Failure curve of software.



★ S/w Myths:- Management may be confident about good Management standards & clear procedures of the company.
perspectives:-

used to assertain the status of the project. going chart.

Deliverable & Milestones.

PAGE No.
DATE / /

Company has latest computers & state-of-the-art s/w tools, so we shouldn't worry about the quality of the product.

Addition of more s/w specialist, those with higher skills & longer experience may bring the schedule back on the track!

Customer prespectives:-

- A general statement of objectives is sufficient to get started with the development of software Missing

- Software ~~can work~~ right with more features is better s/w.
- s/w can work right first time.

- Once the s/w is demonstrated, job is done.
- Software quality can not be assessed before testing.

- Developer Prespectives:-
  - The only delivered for a software.
  - Aim is to develop working programs.

☆ Software Development Process & Process Models:-

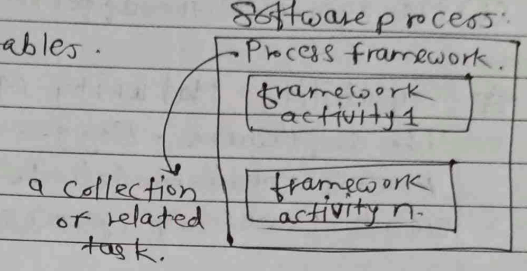
A software process - as a framework for the tasks that are required to build high quality s/w.

Who? - managers, software engineers & customers.

Generic Activities

Why:- Provides stability, control.

* Umbrella Activities:- It is applicable for any process.
  - followed by organization.
  - framework activities.
  - work task
  - work products.
  - milestones & deliverables.
  - QA checkpoints.



* Process framework:-
  - Why s/w process:-
  - To build complete s/w process.

* - Each action is divided into task, called work task.

Work products → What we are delivered.

* Umbrella Activities:- (CCPMCD).

Generic Activities:
- Communication.
- Planning. (resources will require, work produce to be produced & a work schedule).
- Modeling (Analysis & requirement).
- Construction. (code generation: either manual or automated or both).
- Deployment (Delivery to the customer for evaluation).

* Umbrella activities:-
1. s/w project tracking & control.
2. Formal technical reviews.
3. s/w quality assurance.
4. s/w configuration management.
5. Document preparation & production.
6. Reusability management.
7. Measurement.
8. Risk management.

~~gang chart~~.
giant chart.

PAGE No.
DATE / /

① Assessing progress against the project plan.
② Accessing s/w work products in an effort to uncover & remove errors before goes into next action or activity.
③ Define & conducts the activities required to ensure s/w quality.
④ Management the effects of change.

★ Capability Maturity Model Integration (CMMI):-
The Software - Engg. - Institute (SEI) developed process meta-model to measure organization's different level of process capability & maturity.

- goals are associated with key process. (KP[illegible])
- goals are related to tasks.

★ ~~Software Development Process & Process Models:-~~

★ CMMI levels:-
1) Level 1 (initial):-
- processes are poorly managed or controlled.
- lowest quality & highest risk.
- unpredictable outcomes of processes involved.

2) (Manage) Level 2:-
- requirements are managed.
- processes are planned & controlled.
- projects are managed & implemented according to their documented plans.

3) Level 3 (Define
- Medium quality
- processes are
described using
methods, tools

4) Level 4 (Quant
- quantitative
& quality
- quantitative
requirements
- higher qual
- lower ris

5) Level 5 (O
- Continuous

- higher qu

★ Software P
- Build &
- Prescriptiv
- Waterfall
- Incremento
- Incr
- RAD
- Evolutio

- Why M
- poor q
- user e

3) Level 3 (Defined) :-
- Medium quality & medium risk involved.
- processes are well defined characterized & described using standards, proper procedures, & methods, tools, etc.

4) Level 4 (Quantitatively Managed) :-
- quantitative objectives for process performance & quality are set.
- quantitative objectives are based on customer requirements organization need, etc.
- higher quality of processes is achieved.
- lower risk.

5) Level 5 (Optimized) :-
- Continuous improvement in processes & their performance.
- higher quality.

★ Software Process Models :-
- Build & Fix Model :- partially modeled. (suitable for minimum sized projects)
- Prescriptive Model.
- Waterfall model or linear sequential.
- Incremental Process Model.
  - Incremental model.
  - RAD Model.
- Evolutionary Process model.

- Why Models are needed ?.
- poor quality
- user expectations not met.

PAGE No.
DATE / /

Prescriptive model :- Workflow.

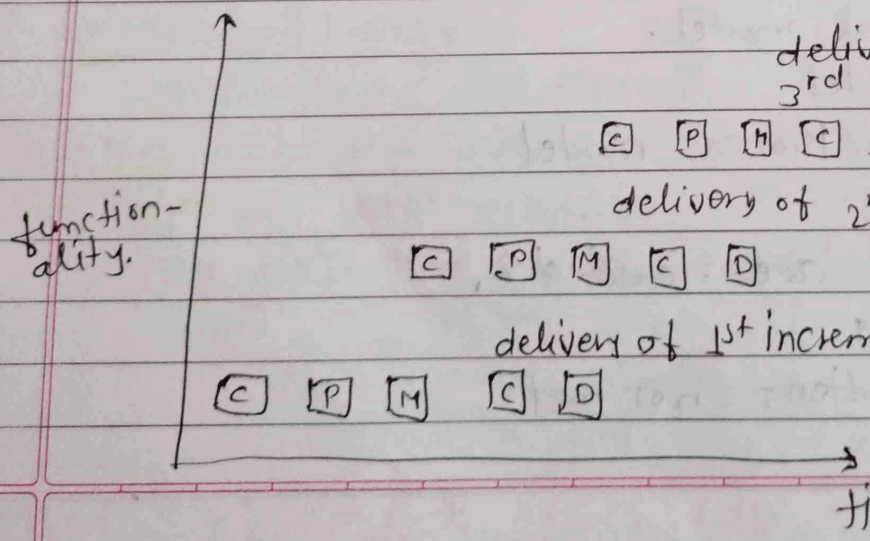
- software engineer choose process framework that includes activities like -
  - modeling
  - Communication.
  - developing deployment.
  - Construction.
  - planning.

Waterfall model or classic life cycle :-

Communication (project initiation requirements gathering.) → planning (estimating scheduling tracking.) → modeling (analysis design.) → Construction (code test.) → deployment. (delivery support feedback.)

~~Prescriptive~~ Incremental Process Model :-

CPMCD should be applicable.

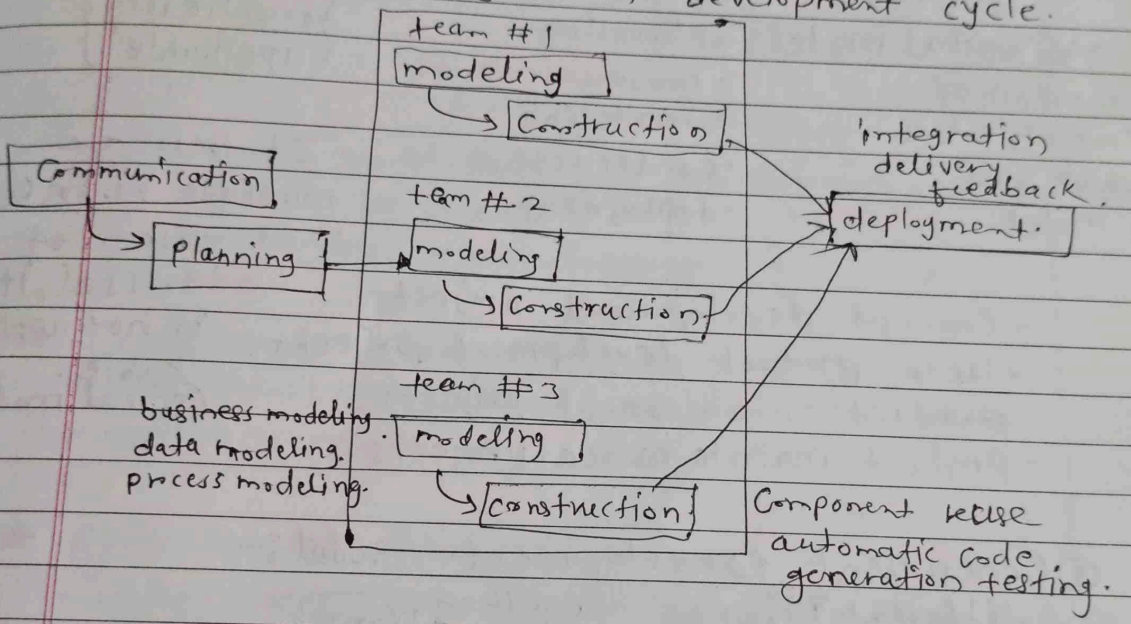


elicit is ~~to~~ to obtain.
~~through~~.

- first increment is often core product.
- unknown & unknown features remain undelivered.
- useful (when) enough staffing is not available for the whole project.
- Early increments act a prototype to help elicit requirements for later increments.

✡ Rapid Application Development (RAD) model:-

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



- High speed edition of linear sequential model.

✡ Evolution process Models:-

process is black box.

Process is white box. ← prototyping. (version of build & fixed model).

- spiral model.
- Concurrent Development model.
- Fourth Generation techniques (4GT).
- Evolutionary models are iterative.

Undesirable event in future is occured.

Risk Analysis :- - sufficient testing e.g.
- requirement changing.

PAGE No.
DATE / /

① Prototyping :- - Communication.
- delivery & feedback.
- quick plan.
- quick design.
- prototype construction.

• Prototyping starts with communication. betn a customer & software engineer to define overall objective. Identify requirements & make a boundary.

② Spiral model :- planning.
- Communication.
- modeling.
- construction.
- deployment.

Combination of Prototype & water fall model.

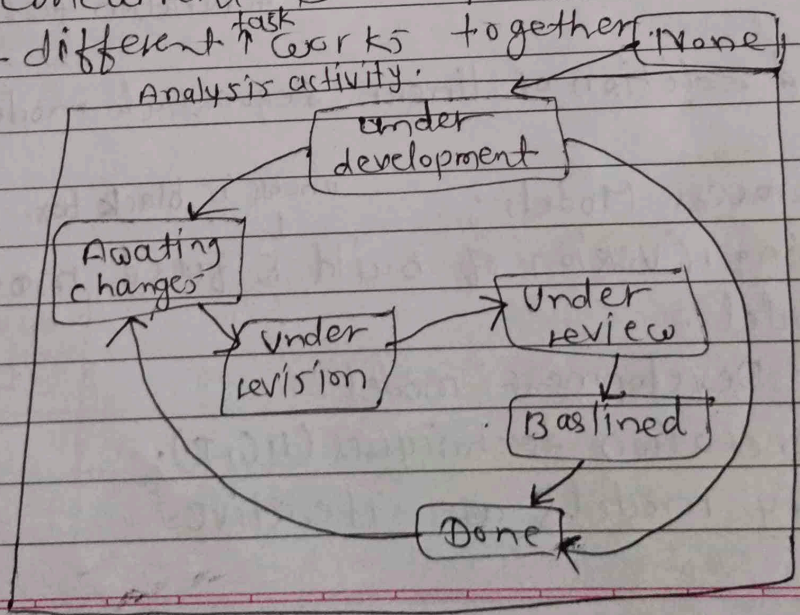
{practically applicable}.

- Concept development projects.
- New product development projects.
- product enhancement projects.
- Product maintenance projects.

- If there are multiple risks & that are not addressed, it is not applicable. Spiral model

③ Concurrent Development Model :-

- different task works together.

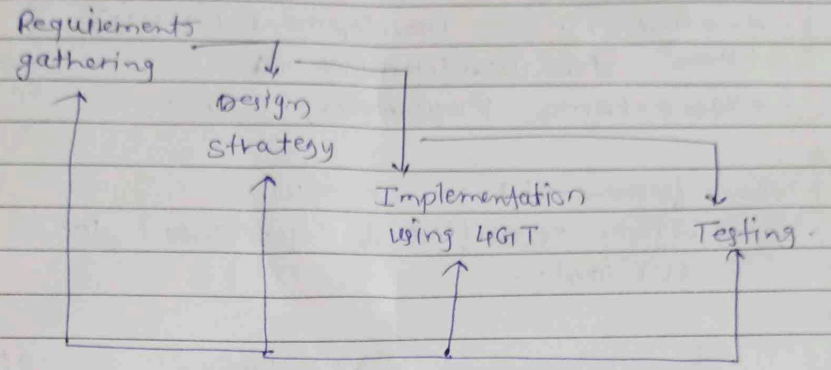


4GTL:- 4 generation Technique language.

non-procedural procedural

9/2/23

★ Fourth Generation Techniques (4GT):-
- SQL, API, python.



Commonly used 4GT in development models are
- Report generation.
- Data base query language.

★ Component Based Development:-
- CBD model leads to software reusability.
- productivity is very high.

★ Agile Model:-(something which is quick).
- Agile process model refers to a software development approach based on iterative development.
- Something which changing not constant.
- This method breaks tasks into smaller iterations.
- Each Iteration is considered as short time "frame" in the Agile process model, which typically lasts from one to four weeks.
- Plans regarding the no. of iterations.

PAGE No.
DATE / /

Agile Testing Methods :-

- Scrum (Roles - 1) Scrum master, 2) Product owner, 3) Scrum team)
- Crystal (chartering, Cyclic delivery).
- Dynamic S/w development method. (DSDM).
- Feature Driven Development (FDD)
- Lean S/w development.
- ~~ext~~ extreme Programming. (XP).

- When frequent changes are required
- When highly qualified & experienced team is available.

} Agile model

Requirement Engineering 16/2/23

Requirement :- A fun^n, Constraint or other property that the system must provide to fulfill the needs of stakeholders.

RE is S/w engineering actions that start with communication activity & continues into the modelling activity.

Characteristics of a good Requirements :-

- Clear & Unambiguous
- Correct.
- Understandable.
- Verifiable
- Complete
- Consistent
- Traceable.

to obtain.

PAGE NO.
DATE / /

A Tasks of RE! –
1) Inception. 2) Elicitation 3) Elaboration
4) Negotiation 5) Sepecification 6) Validation. → (Formal Techniques)
7) Requirements management.

(1) Basic understanding of the problem.

(2) - Find out from customers, users & others what the product objectives are.
- Wh questions.
- Why elicitation is difficult ?
→ Problem of scope.
→ Problem of understanding.
→ Problems of volatility.

(3) Focuses on developing a refined technical model of s/w functions, features & constraints using the information obtained during inception & elicitation.

(4) - Requirements are categorized & organized into subsets.
- Relations among requirements identified.
- Negotiation about requirements, project cost & project timeline.

(5) - written document.
- graphical model.
- mathematical model.
- collection of usage scenario.
- A prototype.
- Combination of above.

(6) FTR- formal technical reviews.

2 techniques → structured analysis → objected oriented → Uml-diagrams.

PAGE No. DATE / /

★ 6/3/23. logical DFDs & physical DFDs.

Data flow Diagrams:-
- A graphical tool, useful for communicating with users, managers & other personnel.
- Used to perform structured analysis to determine logical requirements.
- Focus on movement of data betn external entities & processes.

★ DFD elements:-
- source/sink (external entity).
- Process.
- Data store.
- Data flow.

• DeMarco & Yourdan Symbols

Name

Name

D1 Name

Name

• Gane & Sarson symbols.
• Source sink (external entity). [Name]
• Process. (Name)
• Data store. [D1] Name
• Data flow. Name

① Process:- models what happen to the data. ie. transforms incoming data into outing data.

② Data store:- represents permanet data that is used by system.

★ Symbols naming:-
- external entity → noun.
- Process → verb phrase.
- Data store → ~~Noun~~. Noun
- Data flow → Names of data.

* External entities:-
Source:- Entity that supplies data to the system.
Sink:- Entity that receives data from the system.

* Decomposition of DFD:- Top-down approach.

Levels	Description.	Explanation.
Level 0.	Context diagram.	Contains only 1 process.
Level 1.	Overview diagram.	Utilizes all four elements.
Level 2.	Detailed diagram.	A breakdown of a level 1 process.

* Rules of data flow:-

Data can flow from	Data can't flow from.
- external entity to process.	- external to external.
- process to external entity.	- external entity to store.
- Process to store & back.	- store to external entity.
- Process to process.	- store to store.

* 9/3/23.

Unit-3.

* System Architecture & Design Overview:-
- software architecture of a program or computing system is the structure or structure of the system. which comprise the s/w components, the externally visible properties of those components, & the relationship ~~bet~~ among them.

* Data Design:-
(1) Component level:- design of data structures & associated algorithm.

(2) Application level:-

- Component level focuses on the representation of data structures that are directly accessed by one or more S/W components.

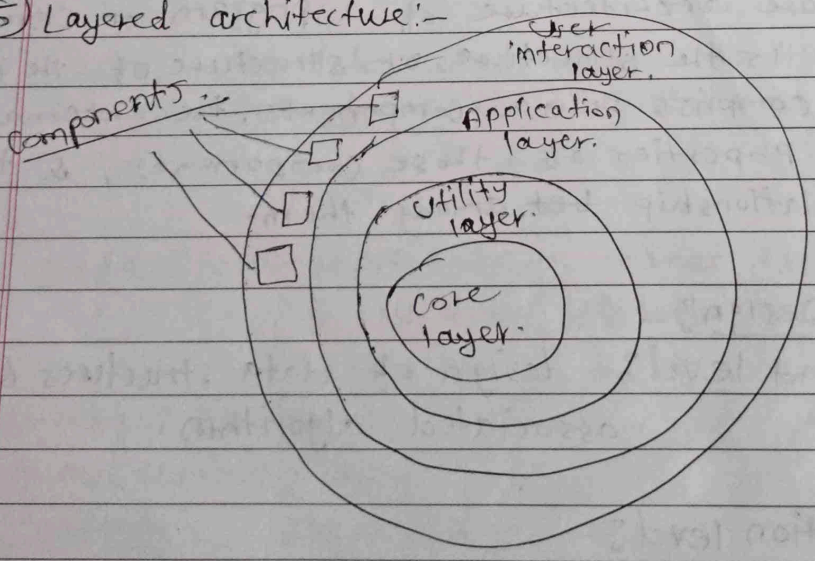
★ Architectural style:—
- A set of components (e.g. a database, computational modules) that performs required by a system.
- It can be represent by client is active → blackboard. repository is passive, then it is data centered.
① Data Centered architecture:-
② Data flow architecture.
③ Call & return architecture.
④ object Oriented architecture.
⑤ Layered architecture.

② Data flow architecture:—

1) filter → filter → Batch sequential.

2) Pipes & filters.

⑤ Layered architecture:—



explicit. requirements → told by Customer.
implicit → one which are not told by Customer.

PAGE No.
DATE / /

★ ① Design Engineering:-
- design model provides detail about s/w data structures, architecture, interfaces & components that are necessary to implement the system.

★ Why design is so important?.

★ Design process & Design Quality:-
- s/w design is an iterative process through which requirements are translated into a "blueprint" for constructing the s/w.

★ Goal of design process:-
The design should provide a complete picture of the s/w, addressing the data, functional, & behavioral domains from an implementation perspective.

★ - The design process should not suffer from "tunnel vision". Designer should consider alternative approaches.
- The design should be traceable to the analysis model.
- The design should not reinvent the wheel.
- The design should minimize the intellectual distance betⁿ s/w & the problem as it exists in the real world.
- The design should be assessed for quality as it is being created not after the fact.
- The design should be reveiwed to minimize Conceptual (semantic) errors.
- The design should exhibit uniformity & integration.
- The design should be structured to accomodate Change.

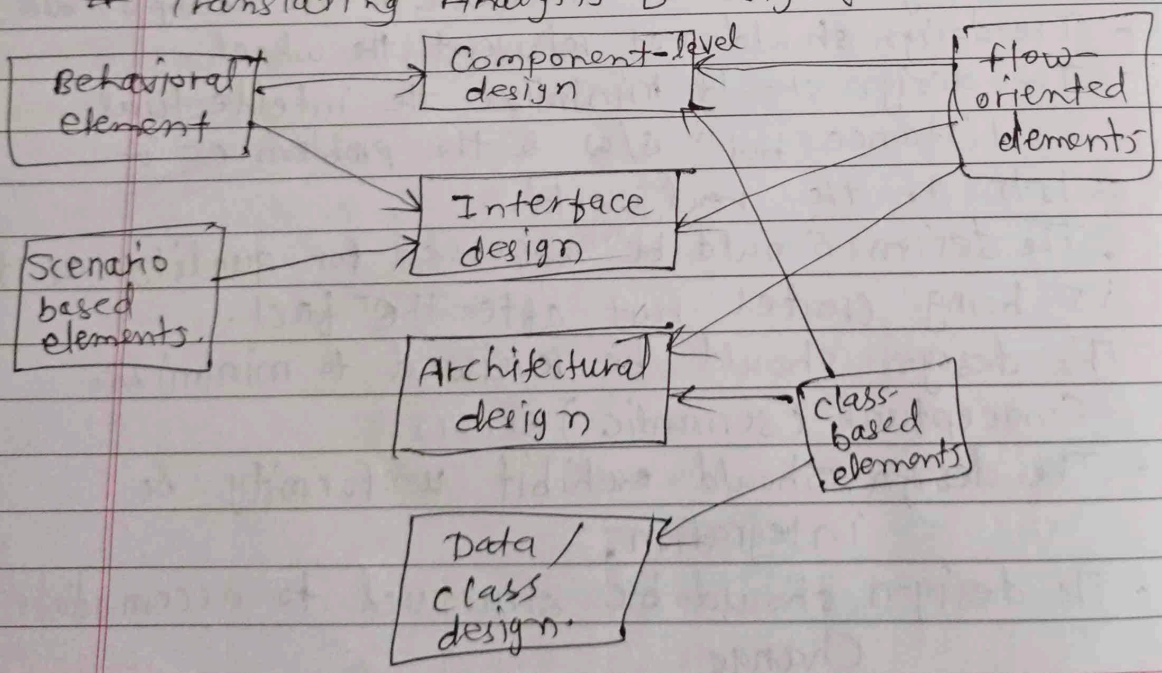
PAGE No.
DATE / /

★ Design Concepts :- Provides the necessary frame work for it to get the thing on right way.
- Abstract, - Refinement, - Modularity, Architecture
- Control Hierarchy.

★ Software Design :-
S/w design model consists of 4 designs :-
1) Data / class design.
2) Architectural Design.
3) Interface Design.
4) Component Design.

★ Elements of Analysis model :-
1) Scenario-based elements. (Activity diagram).
2) Flow-oriented elements. (DFD, Control Flow diagram)
3) Class-based elements. (class diagram).
4) Behavioral elements. (state diagram, sequence diagram).

★ Translating Analysis ⟦ design :-



class diagr
object diagr

★ 23/3/25 :-

★ UML Views
- structural
- Implement
- Behavior
- User vie

★ class diag
- depicts
- describing
- Provide terms
- Used f

- Each c
divid
- nam
- attrib
- oper

★ OO Re
- Gene
- Asso

★ Associ
Agg

Eg.
- V
d

class diagram - structural.
object diagram -

PAGE NO.
DATE / /

★ 23/3/22 -

★ UML View :-
- structural View (class diagram), component, deployment.
- Implementation View.
- Behavioral View (sequence diagram) (activity,
- User View.

★ class diagram :-
- depicts classes & their interrelationship.
- describing structure & behavior in the use cases.
- Provide a Conceptual model of the system in terms of entities and their relationship.
- Used for requirement capturing, end-user interaction.

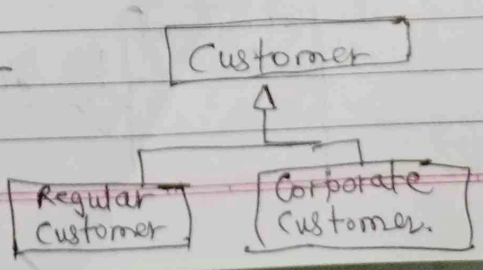
- Each class is represented by a rectangle sub-divided into 3 compartments -
  - name.
  - attributes.
  - operations.

+ => public
# => (friends & inherited)
- => (no one).

★ OO Relationships :-
- Generalization (parent - child relationship).
- Association (student enroll for a course).

★ Association can be further classified as - Aggregation, Composition.

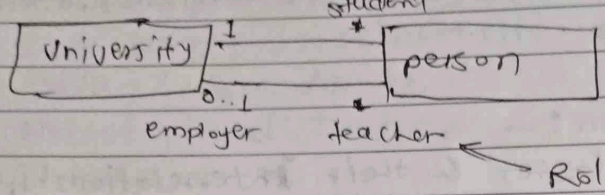
Eg. of Generalization :-
- Used for abstracting details in several layers.



PAGE No.
DATE / /

* Association :- Used to show instances of classes.
- Has two ends :- Multiplicity.
- Role

① Multiplicity & Roles :-

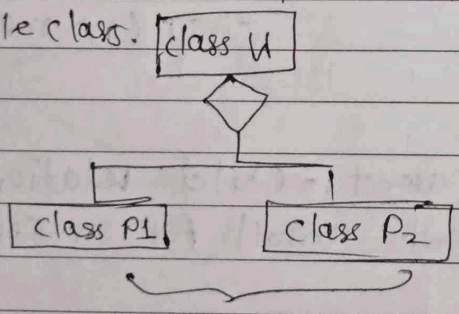


Multiplicity :-

	Symbol	Meaning.
1)	1	One and only one.
2)	0..1	Zero or one.
3)	M..N	From M to N.
4)	*	From zero to any positive integer.

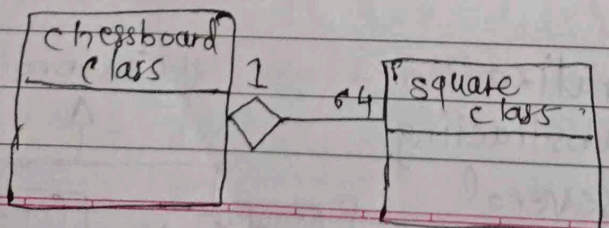
* OO Relationship :- Composition.

Whole class.



- Models the part-whole relationship.
- Every part may belong to only one whole, & if the whole is deleted, so are the parts.
- Expresses the formal relationship.

e.g.



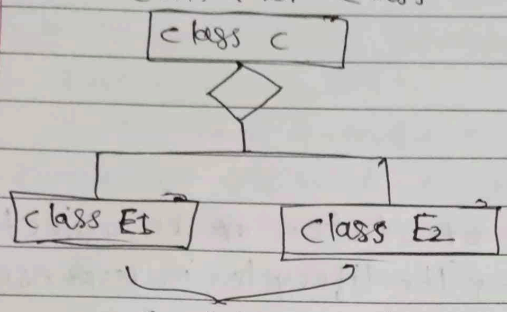
e.g. ABC ⇒ object:

ABC ⇒ class.

PAGE No.
DATE / /

★ OO Relationship:- Aggregation:- Expresses a relationship among instances of related classes.

Container class



Containee classes.

- Expresses a more informal relationship than Composition expresses.

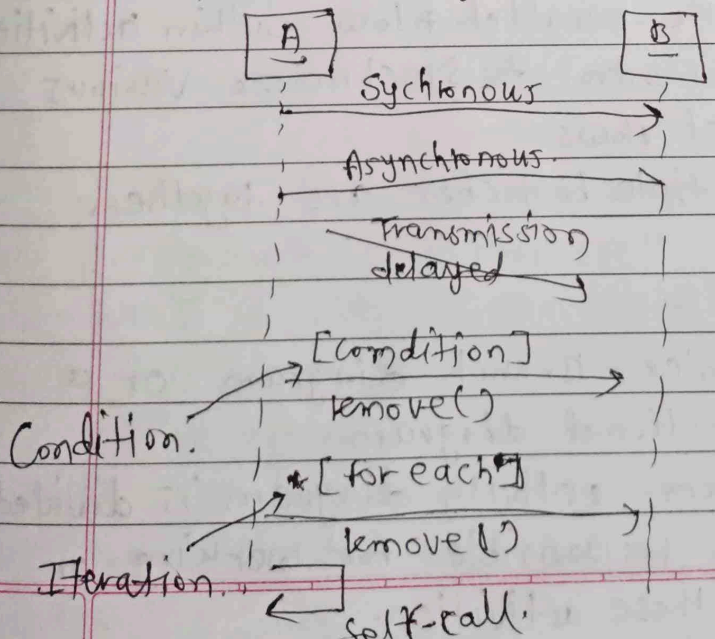
★ Interaction Diagrams:-

→ UML supports two types of interaction diagrams -

1) Sequence diagrams.
2) Collaboration diagrams.

Sequence diagram:- When communication beth two objects is there.

· Sequence diagram: object interaction.



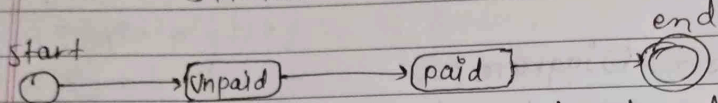
(who is responsible for flowchart) → Swimlane ⇒ variation of activity diagram. ↓ not UML.

PAGE No.
DATE / /

★ Sequence Diagram :- Object Life Spans.
- Creation.
- Activation.
- Deletion.

★ 27/3/23 :-

★ State Diagrams :-
- shows the sequences of states of an object goes through the during its lifecycle in response to stimuli.



Invoice created — paying — invoice destroying.

★ Activity Diagram :- Used for represting the flow of interaction from one activity to another in the form of graphical representation.

Elements - 1) Fork :-
It is used to represent the multiple parallel flows.

2) Branches :- allow the parallel flow within activities.

3) Join :- Used to control & synchronize various parallel flows.

4) Merge :- when multiple branches are together.

★ Swimlane Diagram :-
- also called as Rummler-Branch diagram or a cross-dio-functional diagram.
- In swimlane diagram, activity diagram is divided according to the class responsible for working or performing out these activities.

KPI - key p
- used to
★ Component
- rectangle
- Interface
- Provide
circle at
that
- Required
circle at
that
★ Compon
★ Deploymer
★ 03/04/2
★ Softwa
- i
are qu
• Three
1) Produ
• KPI i
- S/w
- goal

Kpi - key performance indicator.

PAGE No.
DATE / /

- used to capture the flow of dynamic diagram.

☆ Component Diagram :— contain aggregation, association, Componets, etc.
- rectangle with component's name.
- Interfaces :—
  - Provided interface symbols with a complete circle at the end. "Lollipop" represent an interface that the component provides.
  - Required interface symbols with only a half circle at the end. "Sockets" represent requirements that the component requires.

☆ Component diagram basically has components, ports & interface.

☆ Deployment Diagram :— Contains nodes & association.

☆ 03/04/2023 :—

☆ Software Metrics : Unit-4 :—

- is a measure of s/w characteristics that are quantifiable or countable.

- Three types of software metrics :—
  1) Product. 2) Process. 3) Project.
- KPI is project, process & product.

- s/w process & project are quantitative measures; used a management tool.
- goal is to determine whether the quality & improvements.

PAGE No.
DATE / /

★ Reasons of Measure -
1) To characterize in order to gain an understanding of processes.
2) To evaluate in order to determine status with respect to plans.
3) To predict in order to gain understanding of relationship among processes & products.
4) To improve in order to identify roadblocks, root causes, ineficiencies.

Imp ★ product characteristic will you to assets the process metrics.

★ Process Metrics :-
Set of metrics based on outcomes of the process such as
- errors uncovered before release of the S/w.
- Defects delivered to & reported by the end users.
- work products delivered.
- Human ~~effects~~ efforts expended.
- Calendar time expended.
- Conformance to the schedule.
- Time & effort to Complete each generic activity.

★ Project Metrics :-

- used to minimize the development schedule by making the adjustments necessary to avoid tasks & mitigate potential problems & risks.

★ S/w Mana
Two Categ
1) Direct m
- process
- product
2) Indirec
- product

★ 16/4/23

★ S/w Met
eg for
set
e.g. Ag
→ lea
→ velo
→ cycle
→ effici
→ Mean

★ functi
1) Unit
2) what
- fu

★ Typ
1) Esti
2) He
3) Ana

PAGE No.
DATE / /

☆ S/w Management :-
Two Categories :-
① Direct measures of the -
- process (cost, efforts).
- product (lines of code produced, execution speed, defects reported over time, etc.).
② Indirect measures of the -
- product (functionality, quality, complexity, efficiency, reliability, maintainability).

☆ 16/4/23 :-

☆ S/w Metrics :-
e.g for a project process planning, for goal setting we use metrics.
e.g. Agile process metrics.
→ lead-time → time taken to deliver.
→ velocity → no. of o/p per splits.
→ cycle time → time to implement cycles.
→ efficiency.
→ Mean time to recover (MTTR)

☆ function point Analysis :-
① Unit of measurement of s/w.
② what do these measure ?
- functional user requirements.

☆ Types of Estimation Techniques :-
1) Estimation Empirical E.T.
2) Heuristic E.T.
3) Analytical E.T.