Unit - III Building analysis model & Design engineering

→ Analysis model:

→ Analysis model uses a combination of text & diagrammatic forms to depict requirements of data, functions & behaviour so that it will be easy to understand overall requirements of the software to be built.

→ The software engineer also called analyst builds the model using requirements stated by the customer.

→ Analysis modeling approaches:

i) Structured approach (Data modelling)

ii) Object oriented approach.

I) Structured approaches:

Analysis is made on data & processes & transform the data as a

separate entity

→ Data is modeled in terms of only attributes & relationship (not operations)

ex: ER diagrams. (in Data modeling)

→ Data modeling concept:-

In data modeling data objects are examined independent of process. ex ER diagr:

i) Data objects:

→ Data object is representation of any composite information (many attributes/ properties).

ex: Car

Name (company)	Color	Price
lexus	Silver	x dac
BMW	blue	y dac
Ford	black	z dac

Here car is a data object & name, color & price are properties / attributes

ii) Data Attributes:

each data object is having of properties those properties are the attributes.

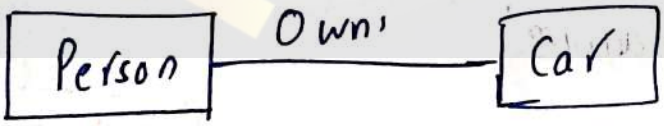
ex: in previous table name, color & price are attributes.

(we also have multiple types of attributes & their representation check in DBMS Unit 1). (For complete ER model.

iii) Relationships:

relationship indicates how data objects are related to one another.

ex: customer purchases the car.



iv) Cardinality & Modality:-

Cardinality specifies number of occurrences of one object related to number of occurences of another object.

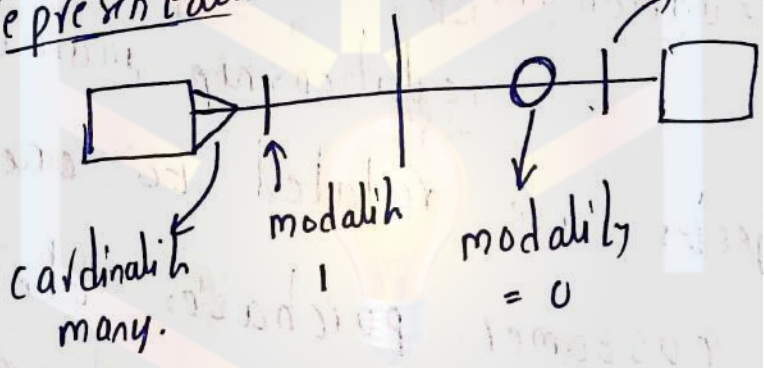
(or)

The maximum number of objects relationship is represented by cardinality

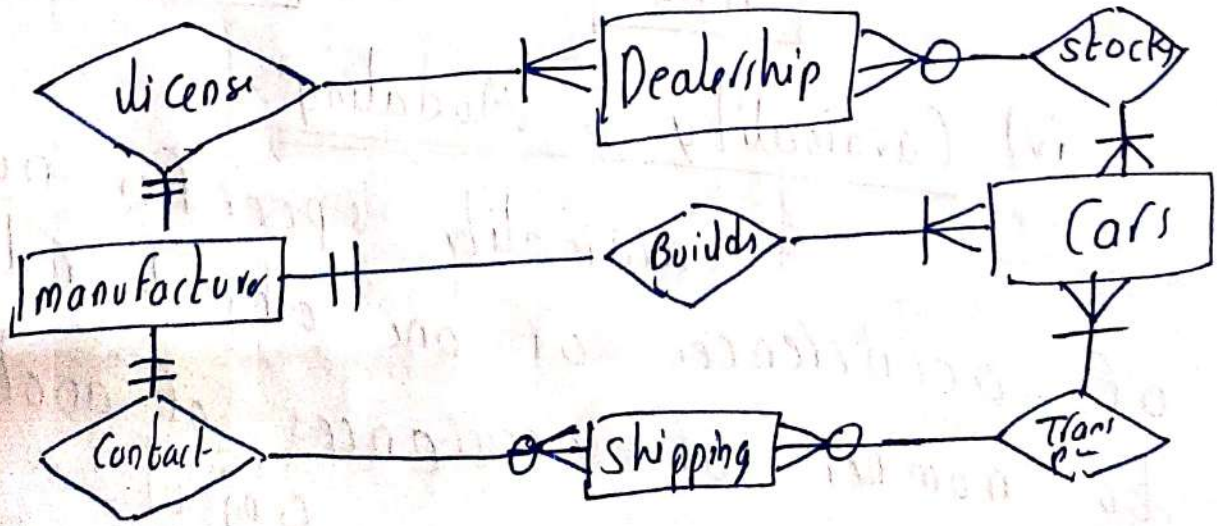
→ One to one (1:1)
→ one to many (1:n)
→ many to one (n:1)
→ many to many (n:m)

modality :- is the least number of row connections (can be 1 or 0)

representation



E-R Diagram:



II) Object Oriented analysis:

→ In object oriented approach, analysis is made on the classes & interaction among them in order to meet customer requirements

→ Object oriented approach views information-system as collection of interacting objects that work together to accomplish tasks.

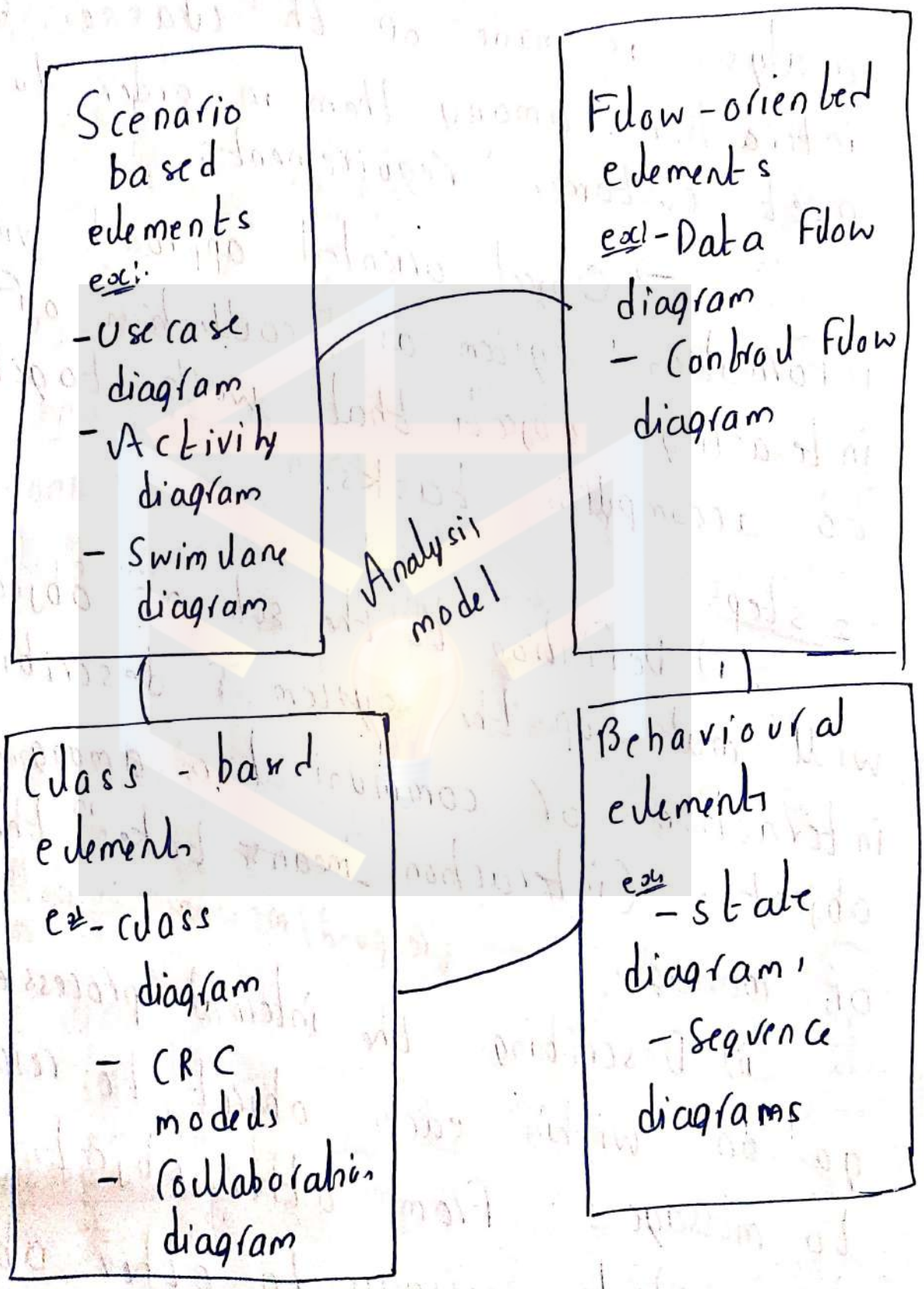
2 steps

i) Definition to the set of objects that will make up the system & describing the interaction or communication among various objects. (interaction ~~means~~ takes in the form of messages (respond/message is nothing but calling)

ii) Describing the internal processes that go on within each object to respond to messages from other objects & to initiate messages to other objects.

→ elements of the analysis model :-

@ with UML



→ Scenario based modeling :-

→ Analysis modeling with UML begins with the creation of scenarios (we have multiple softwares for designing UML diagrams, rational rose enterprise edition, Umbrello, Argo UML etc..)

SBM includes i) Use case diagrams
ii) Acitivity diagrams
iii) Swim lane diagrams

I) Use case diagrams :-

Use case models use to identify & define all elementary business processes that system must support.

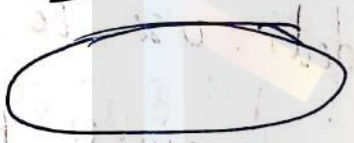
use case : usecase is a scenario by which user can use the system & pictorially we represent these scenarios by usecase diagram.
(single system can have multiple use cases).

→ Components of use case diagram (3)

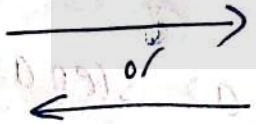
i) Actor : User who interact with the system

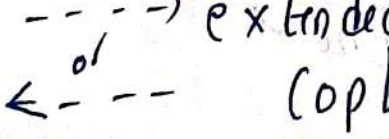
Primary actor	Secondary actor
- who initiate/ interact directly	- who support the system (involved)
eg: customer, maintnance	eg: server

ii) Use case : Functionality or service provided by the system.



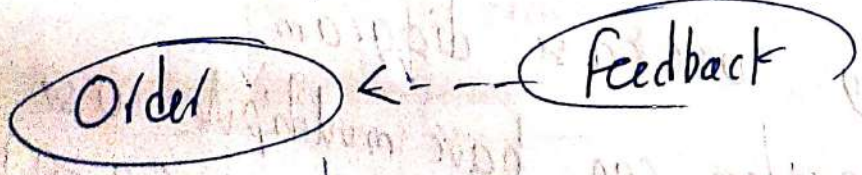
iii) Relation : It shows relation b/n actor & the system. (use case).

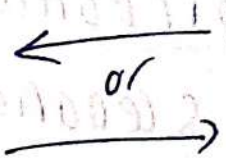


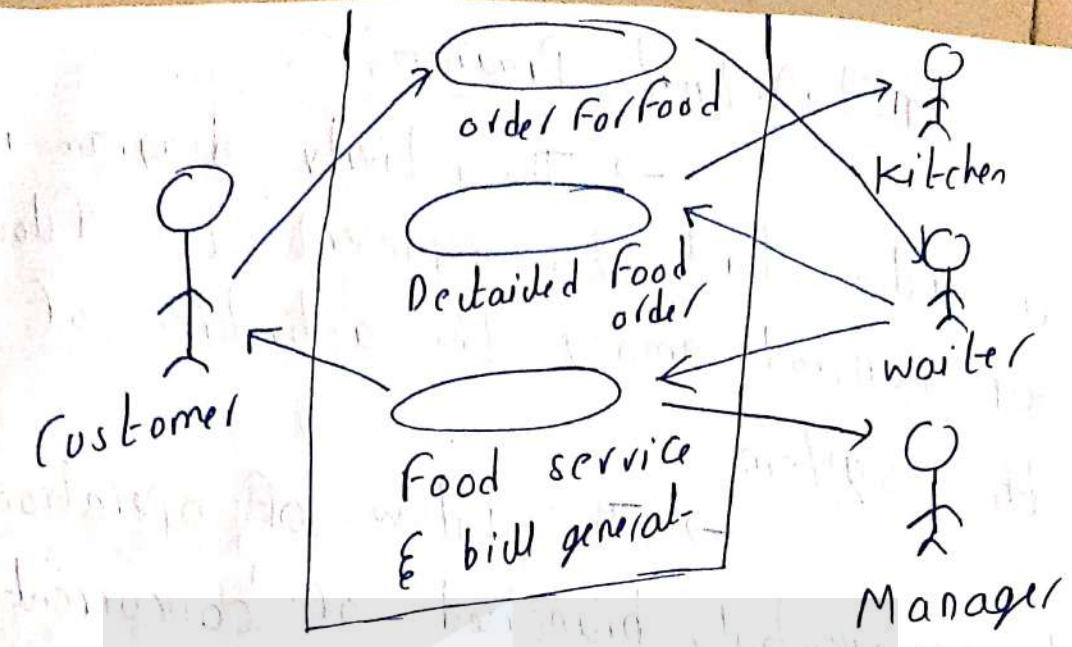


extended relation (optional)

included relation (mandatory)







Path / Flow:

→ Customer orders for the food
→ The waiter receives this order
→ The waiter then forms detailed order
→ The detailed order is submitted to kitchen & service is provided to the customer
→ Finally bill is produced.
→ The bill is submitted to customer & bill report is sent to manager.

II) Activity Diagram:-

→ The activity diagram is a flow chart to represent the flow of control among the activities of a the system.

→ The flow of operation can be sequential, branched or concurrent. The activity diagram is sometimes considered as the flow chart. Although the diagrams look like a flow chart, they are not.

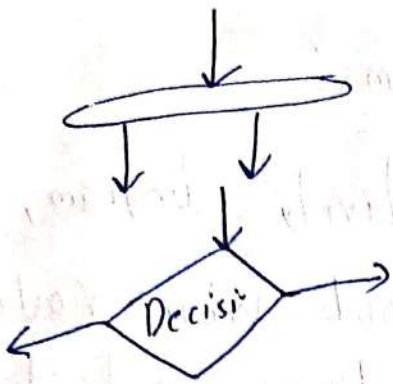
Components / symbols

- start
- Final activity node.

Activity — Activity (with short description)

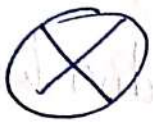
→ Control flow / state transition

Join (2 to 1) conversion joint.

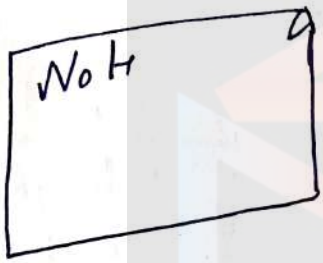


Fork (1 to 2) (divider)

Decision (condition)

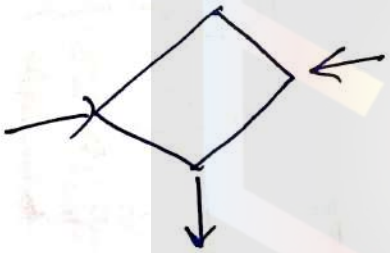


Final flow node (end of only that flow.

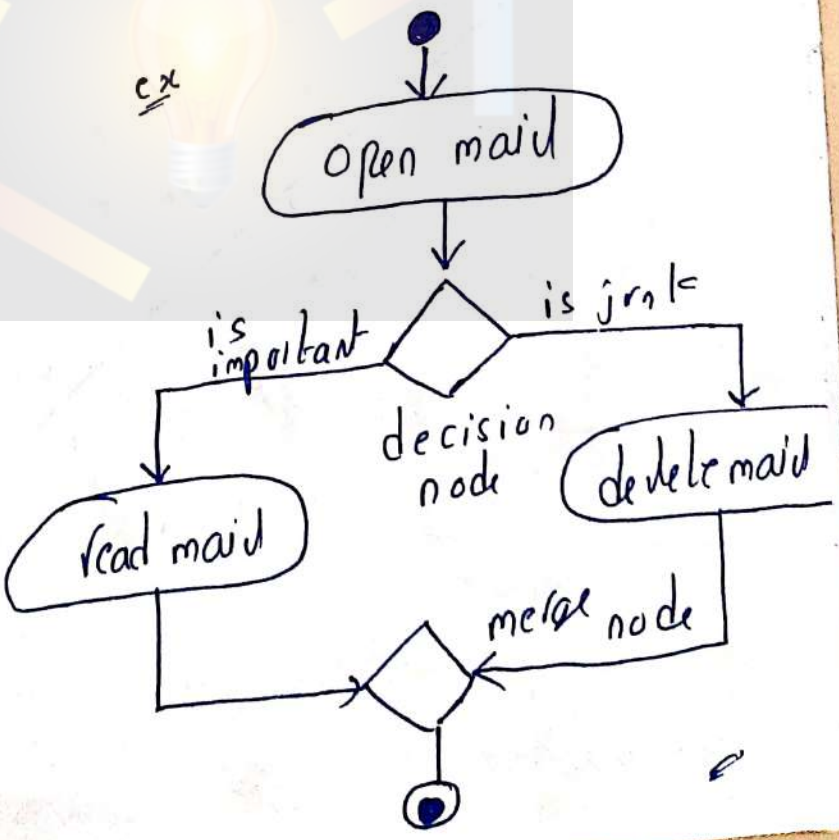


Note (for comments).

merge (2:1)



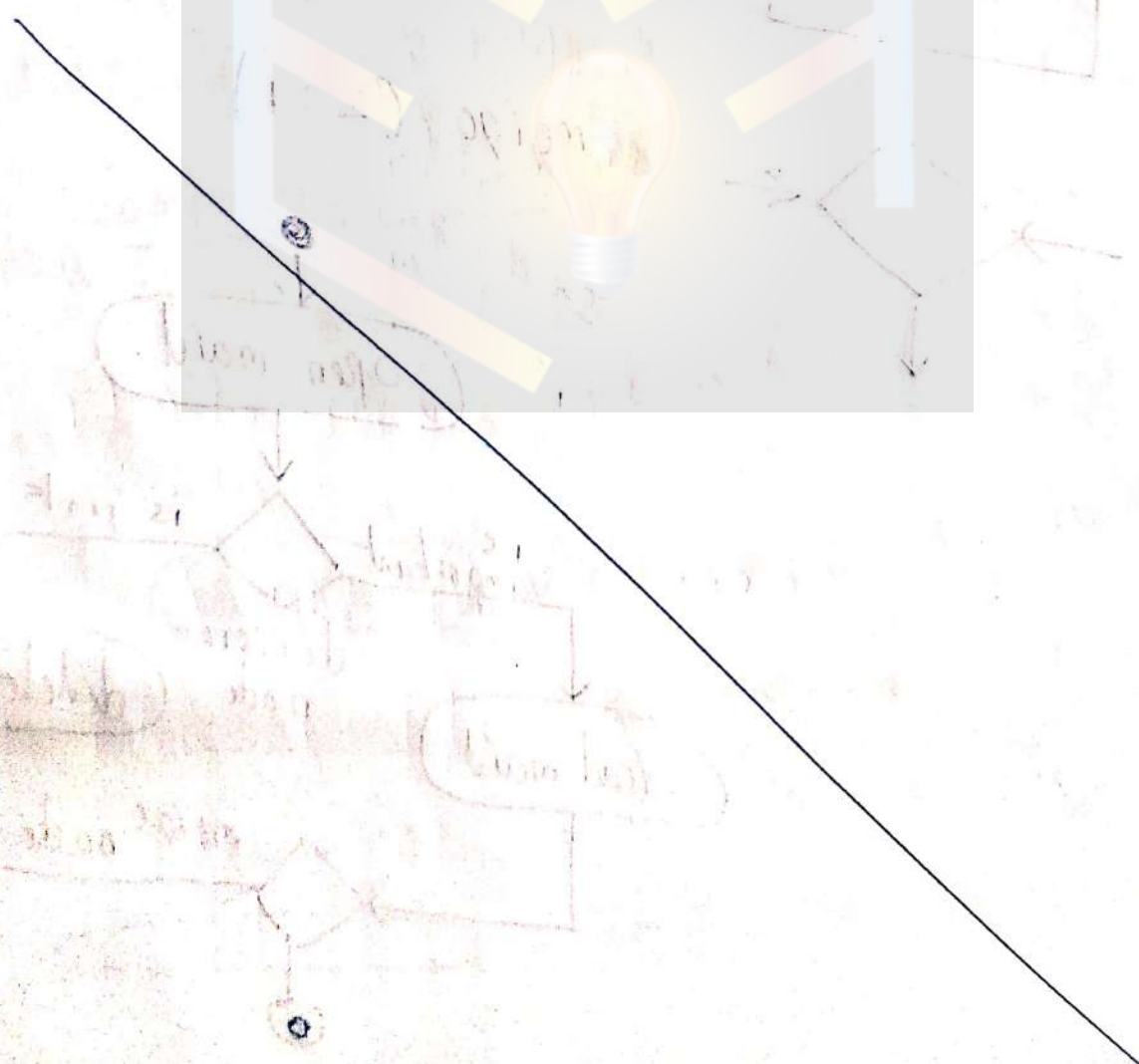
ex

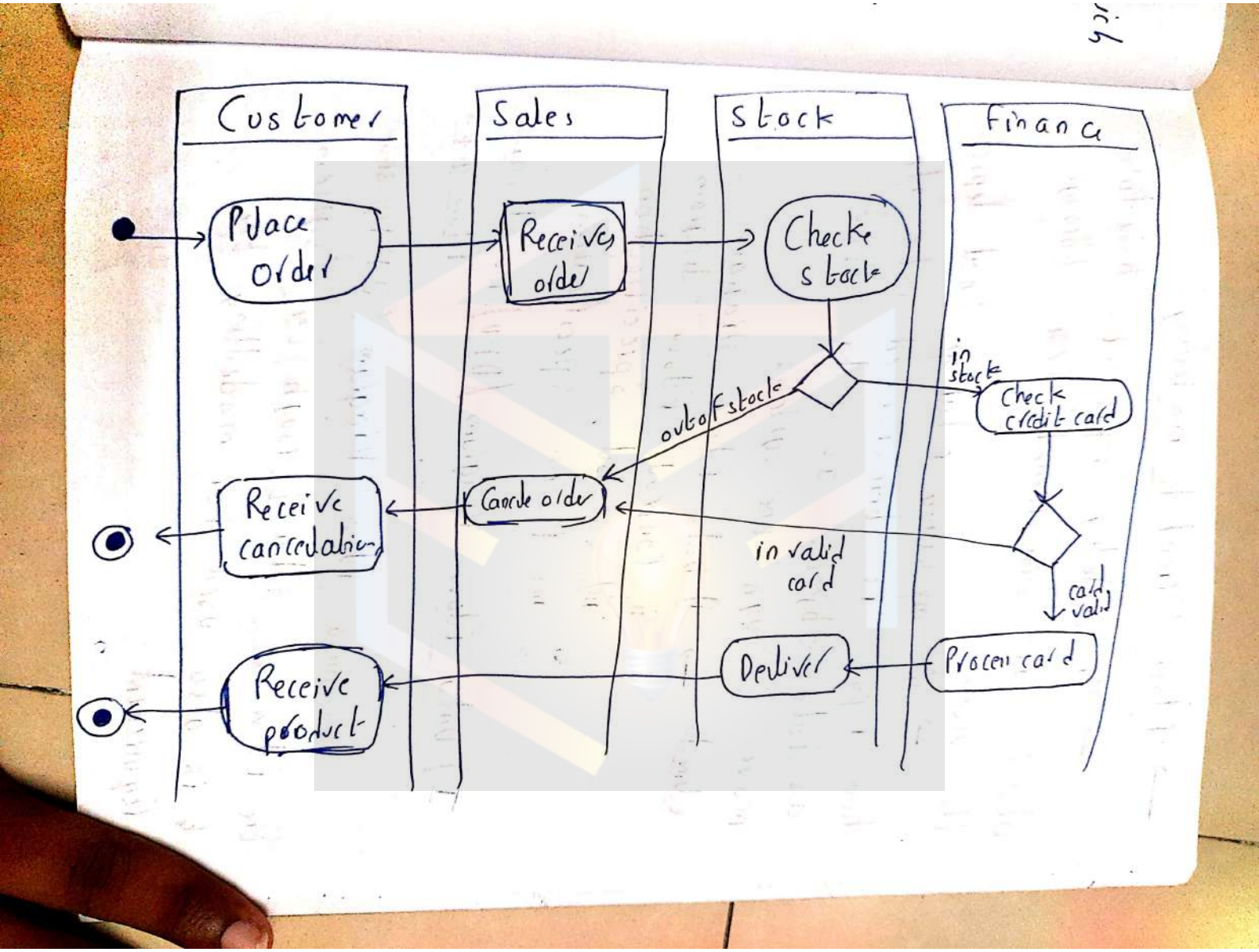


III) Swimlane Diagrams:

→ A flavor of activity diagram, which also give information about which role is performing the underlying activity.

→ Activity diagram is divided into multiple column & each column is dedicated to a particular role.





→ Flow Oriented modelling:

→ This modelling represents how data objects are transformed / moves through the system

→ A modelling notation that depicts how input is transformed into o/p as data objects as data objects move through the systems

elements → i) Data flow Diagram
ii) Control flow diagram
iii) Control Specification
iii) Process specification

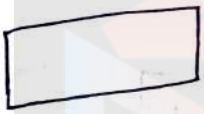
I) Data flow Diagram / DFD / Bubble chart / flow graph

A data flow Diagram (DFD) shows the flow of data through the system & is also used for modelling the requirement.

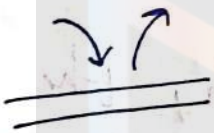
Components / Symbols:

Process (Takes i/p & produces output)

Data Flow (shows flow of data into or out of a process / data store / source.



source or sink (external entity that acts as an i/p to system (source) or o/p to system (sink)



data store (Data repository).

Note:
- → unique names are important
- → DFD's depict flow of data & not order of events like a flow chart.
- → avoid representing of decision nodes. (diamond node).

→ Levelling in DFD:-

→ DFD's can be drawn to represent the system at different levels of abstraction.

lvl - 0 DFD :- represents the entire system as a single bubble with i/p & o/p data indicated by incoming & outgoing arrows.

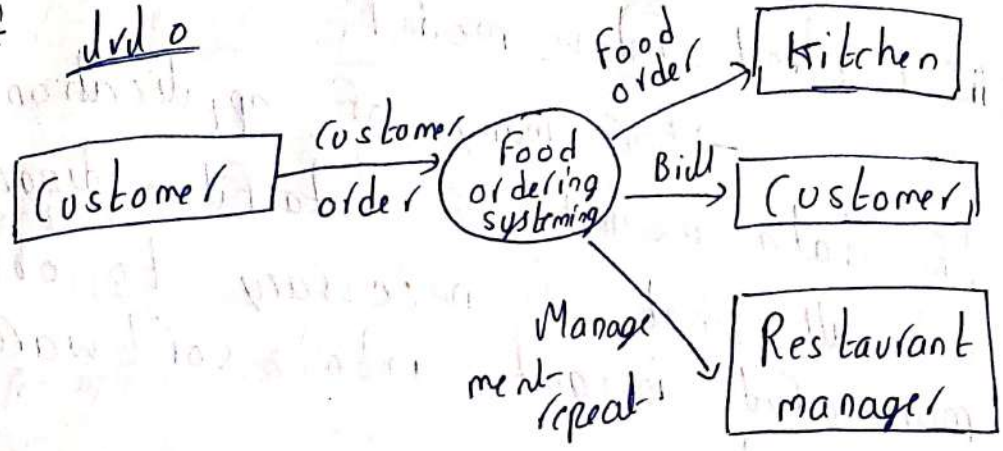
ex:

lvl - 1 DFD:- expanding the above lvl 0 or a bit depth representation is done

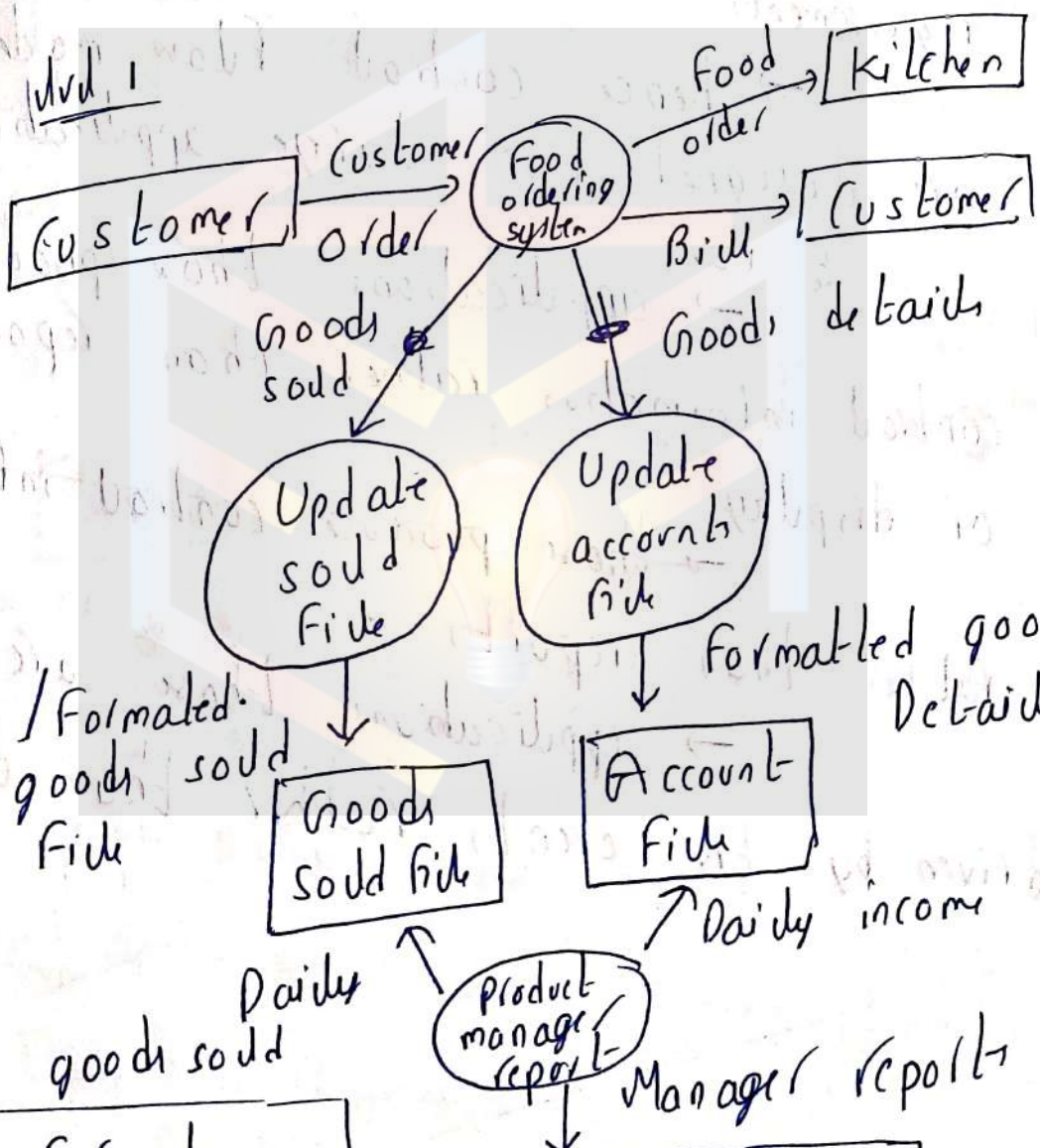
Note! that the no. of i/p / o/p should be kept constant

lvl - 2 DFD:- each individual module will have it own DFD (full depth & 100%. ideal).

ex: lvl 0



lvl 1



lvl 2 Foreach process we will design

III Control Flow model :-

→ For many of applications the data model & data flow diagram, are all that is necessary to obtain meaningful insight into software requirements

→ hence control flow models are designed for large applications & for

→ applications those produce control information rather than reports or displays

~~→ those produce control information rather than reports~~

→ applications those are driven by the events rather than data.

III) The control Specification:

The control specification represents the behaviour of the system. The behavioural of the system can be represented using state transition diagram or state chart diagram.

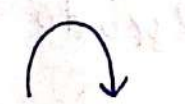
state chart diagram: A state chart diagram shows the state machine that consists of states, transitions, events & activities.

state. (is a situation/ condition/ activity)

start state

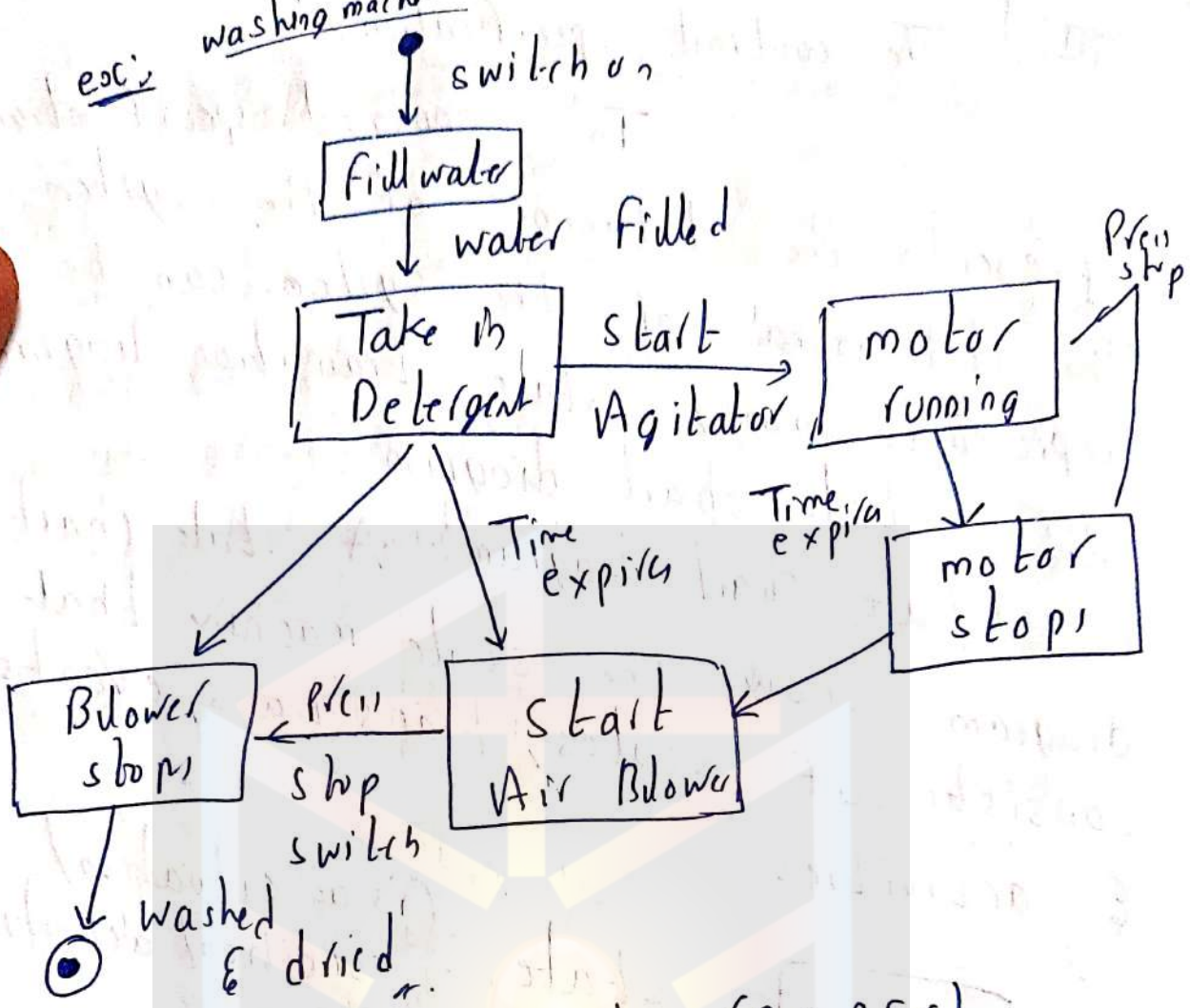
End state

state transition



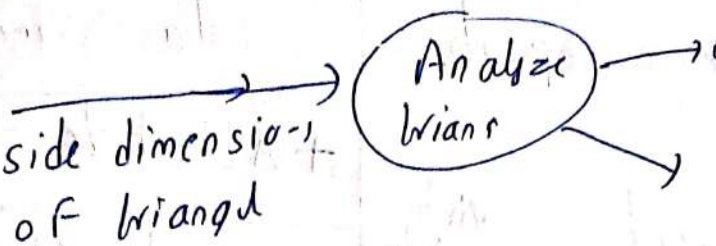
transition to itself

ex:- washing machine



IV) Process Specification (PSPEC)

The process specification is used to describe all flow moded processes. The content of process specification include Program design language (PDL) It is nothing but Pseudo code.



Pseudo code is written to analyze the triangle.

→ Class Based modeling:

→ Class diagram include identification of candidate classes and it show their responsibilities & collaboration

attributes & operations

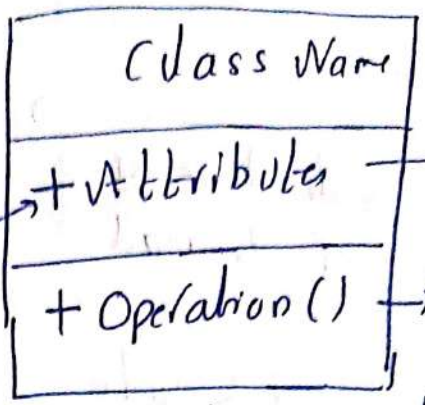
relation bln one obj to another.

→ The main idea / it describes the static structure of the system.

Class is a templdate or formal.

→ represent entities with common characterishics or featura

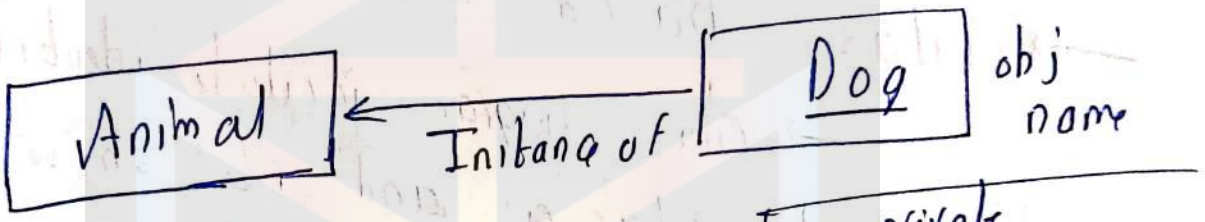
→ attribute & operation

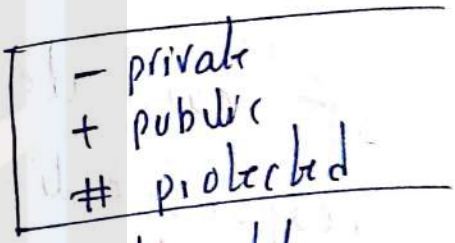


Form a class

→ in object attributes get the values & operations are used by object

object ← with underline



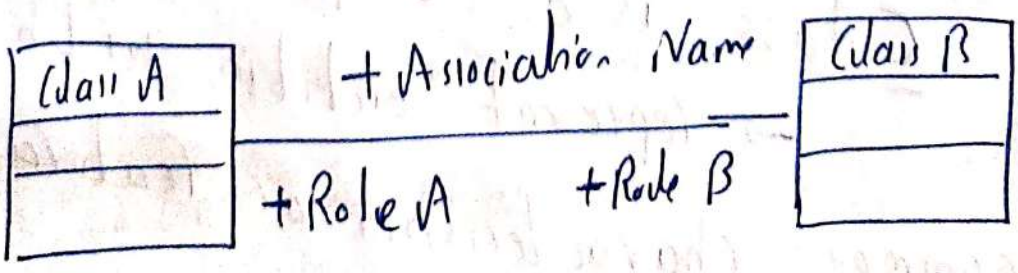


relation b/n classes-

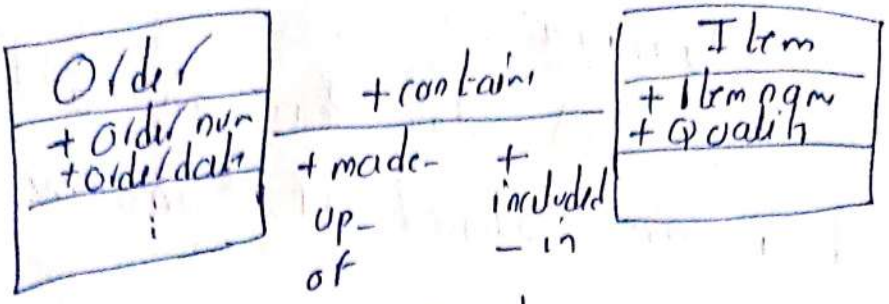
Associations:

→ structural relationship b/n peer classes

→ Association can have a name & direction, or Be bi directional



eg:



order made up of item
or
item is included in order.

Multiplicity / Cardinality :

same above example with

1 *

→ one instance of order is (0 or more) instance of item.

→ one instance of item is connected with 1 instance of order

Aggrigation :-

A class has an attribute which is an object of another class. (Part) whole

→ here class a is an object of class B (in attribute)

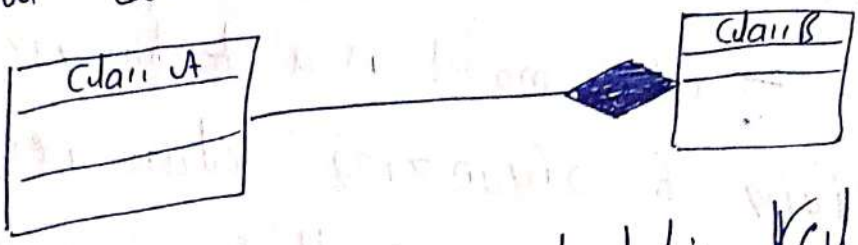
→ This is not strong relation (hence deleting ~~dealing~~ I will not effect other)

Composition :-

A class as attribute which is an object of another class

→ here also class a is an object of class B (in attribute)

→ here it's strong (hence deleting one will delete other)



Dependency: to establish relation depending on another

ex: class diagram

Sub classes →

Class responsibility Collaboration (CRC)

→ CRC model is a technique for identifying & organizing class responsibilities

→ It is a collection of (3 x 5) that represent classes

→ CRC card is a useful technique for identifying class responsibilities.

→ A simple CRC card

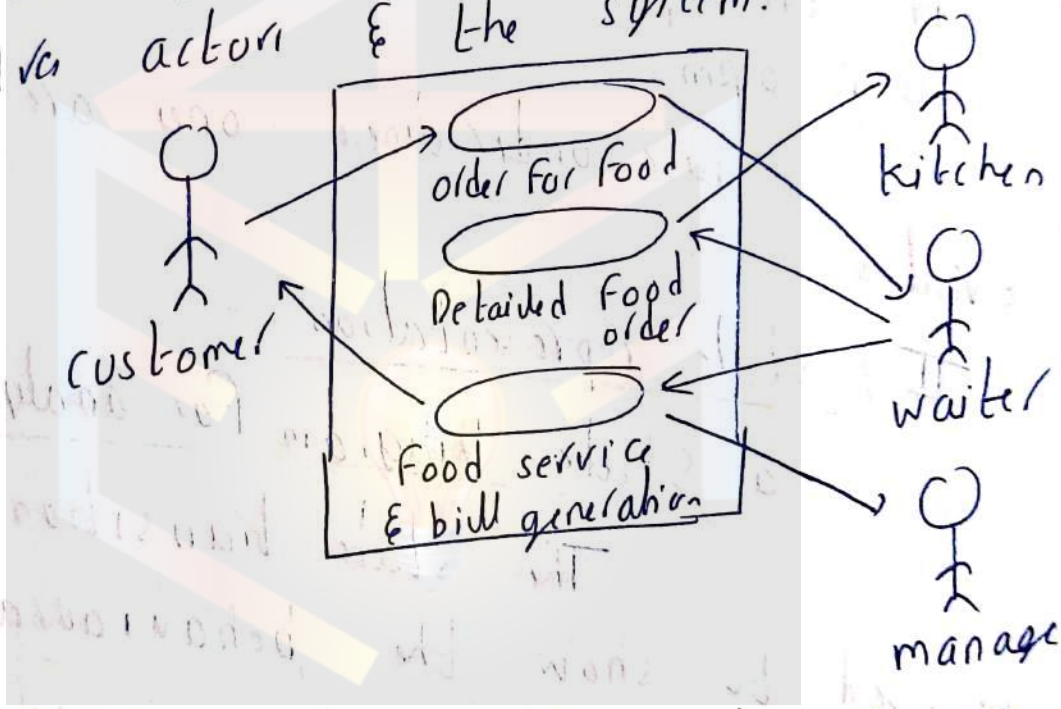
Class : Floor Plan	
Responsibilities:	Collaborations
- Define floorplan	
- Manage floor plan	
- Scale four plan	
- Incorporate walls, doors & windows	walls
- Show position of video cameras	camera

→ Behavioural model:

The behavioural model indicates how software will respond on occurence of external events.

I) Identifying events with the usecase:

A sequence of activities that can be represented by a usecase involves actors & the system.



when ever the system & an actor exchange information an event occurs

ex: The homeowner uses the key pad to enter his four ~~dit~~ digit password

The password is compared with the valid password stored in the system

If the password is incorrect, the control panel will beep once & reset itself for additional i/p.

if the password is correct the door opens.

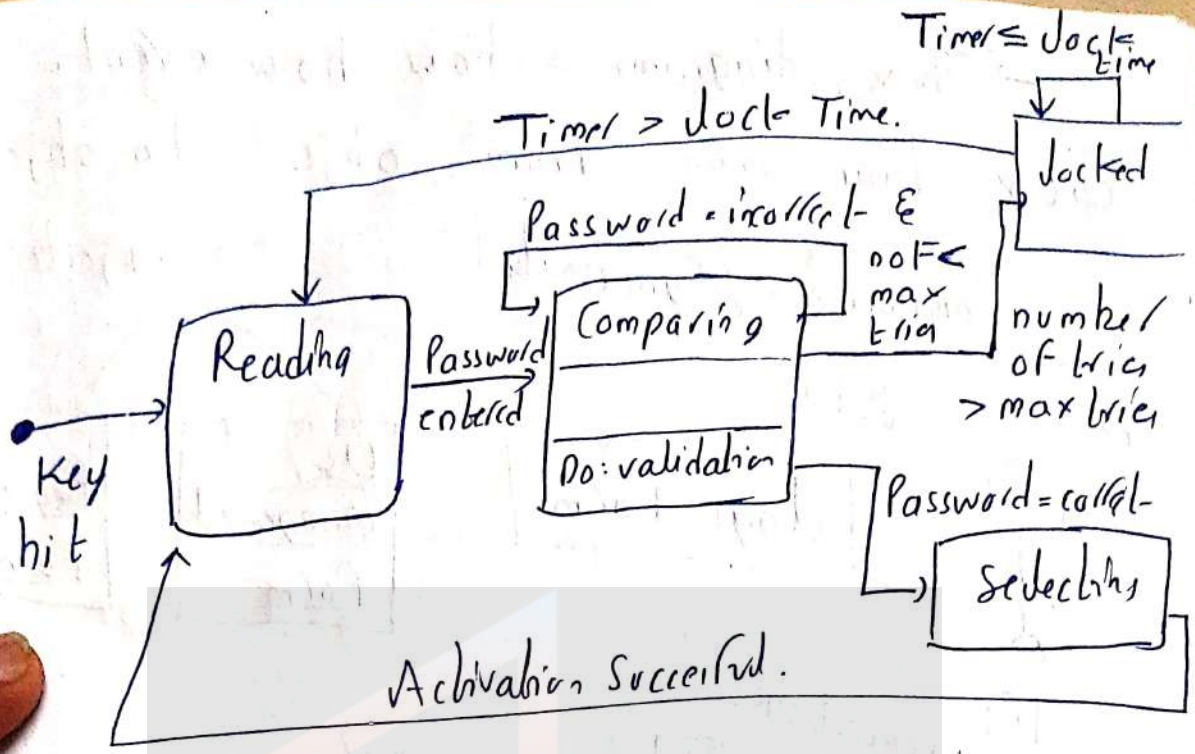
here underlined ones are the events

II) state representation :-

a) State Diagram for analysis classes

The state transition diagram are used to show the behavioural of the system. They show transisions from one state to another.

Ex: same above security's state Diagram



⇒ no of tries < max tries

⇒ no. of tries < max tries

explain above example in words.

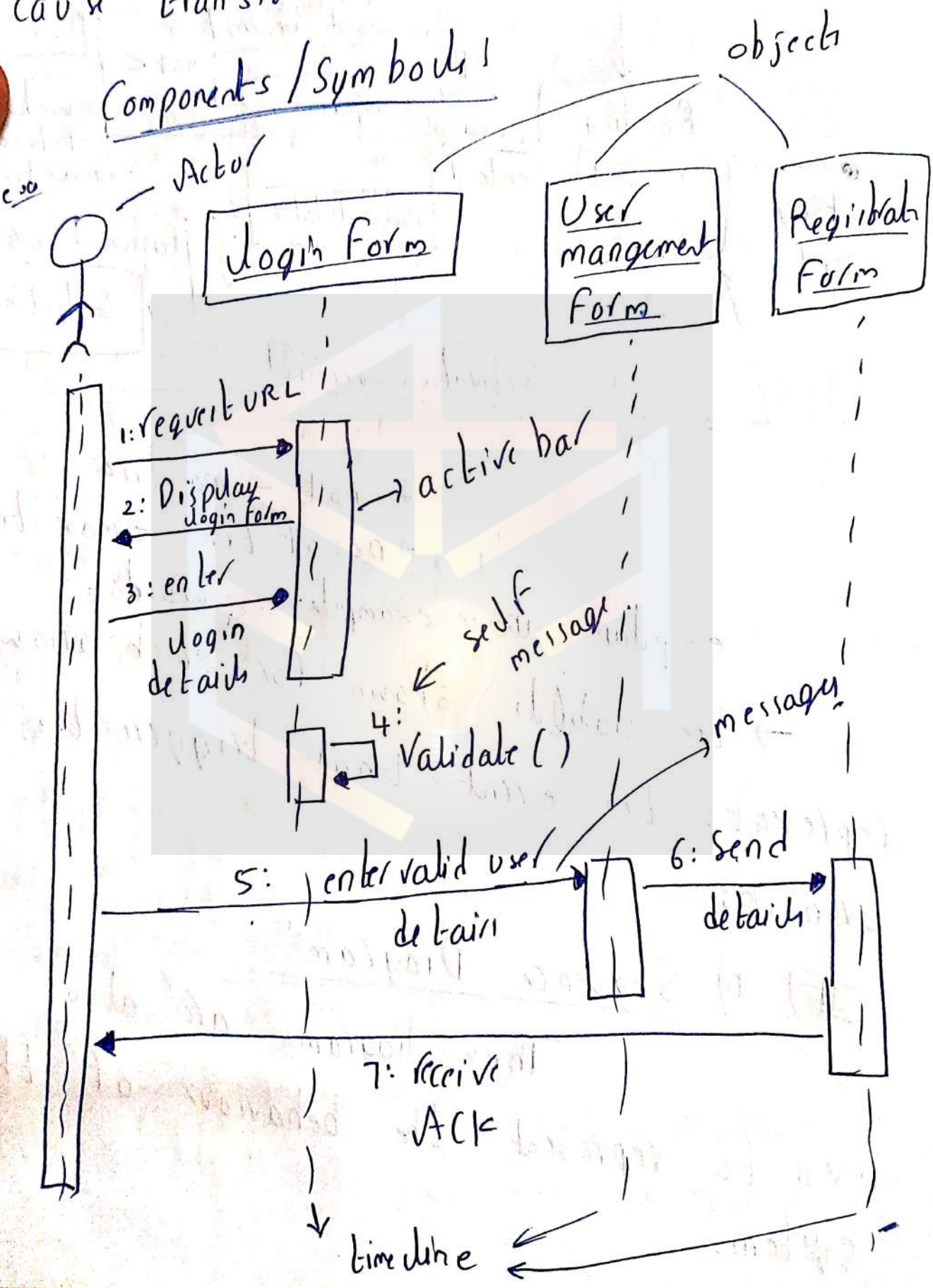
→ the labels shown for each arrow represents the event that triggers the transition.

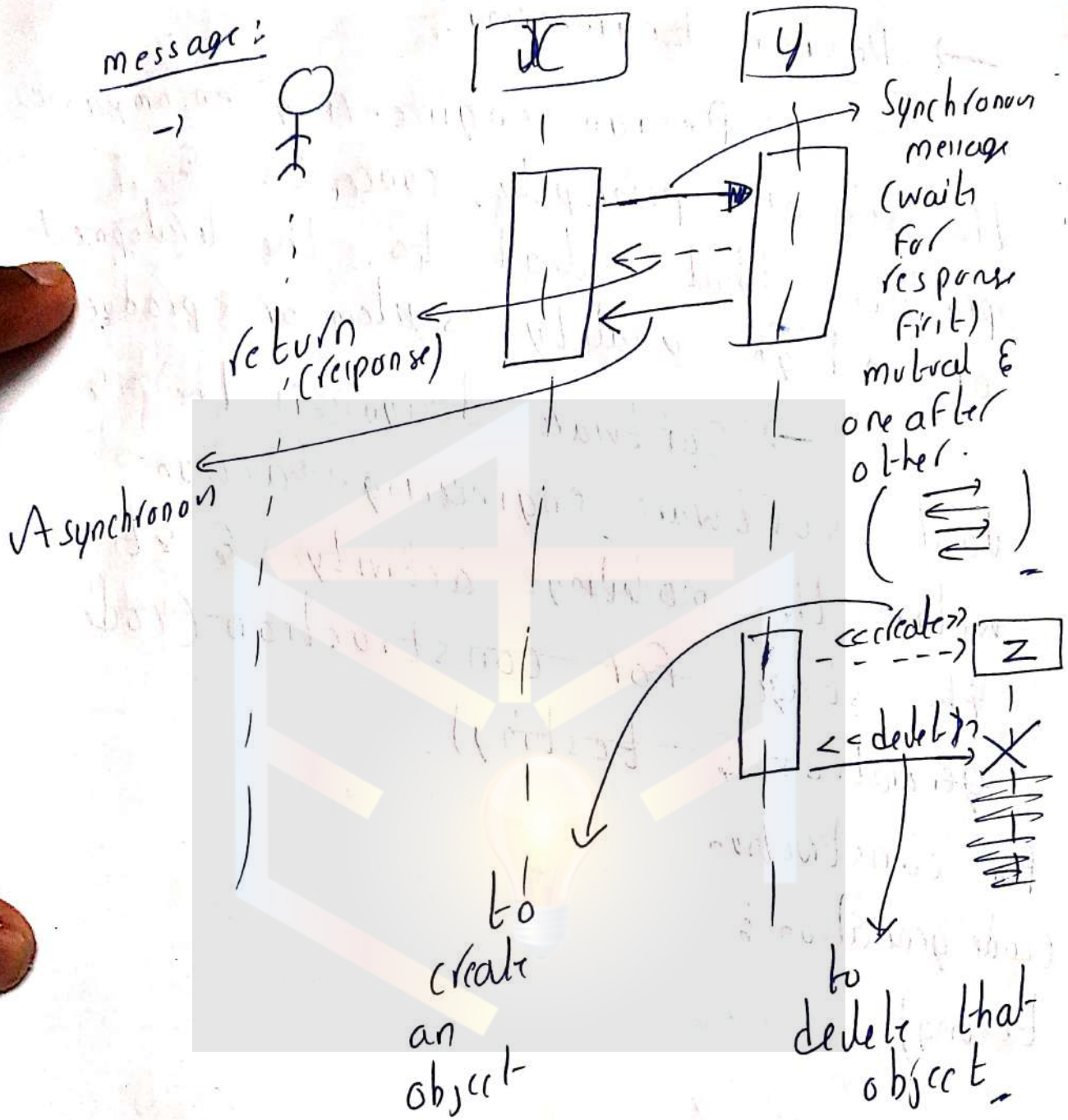
b) Sequence Diagram:

These diagrams are also used to represent the behaviour of the system.

→ These diagrams show how events cause transition from object to object

Components / Symbols :

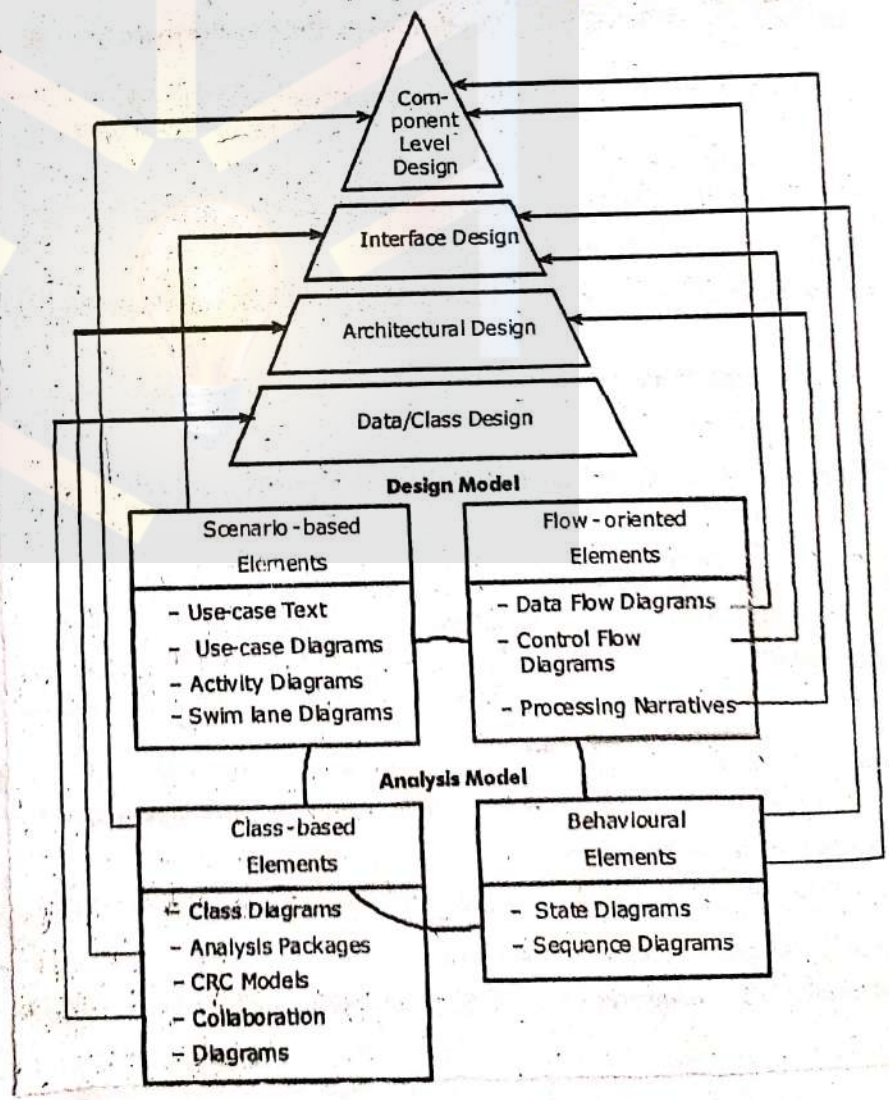




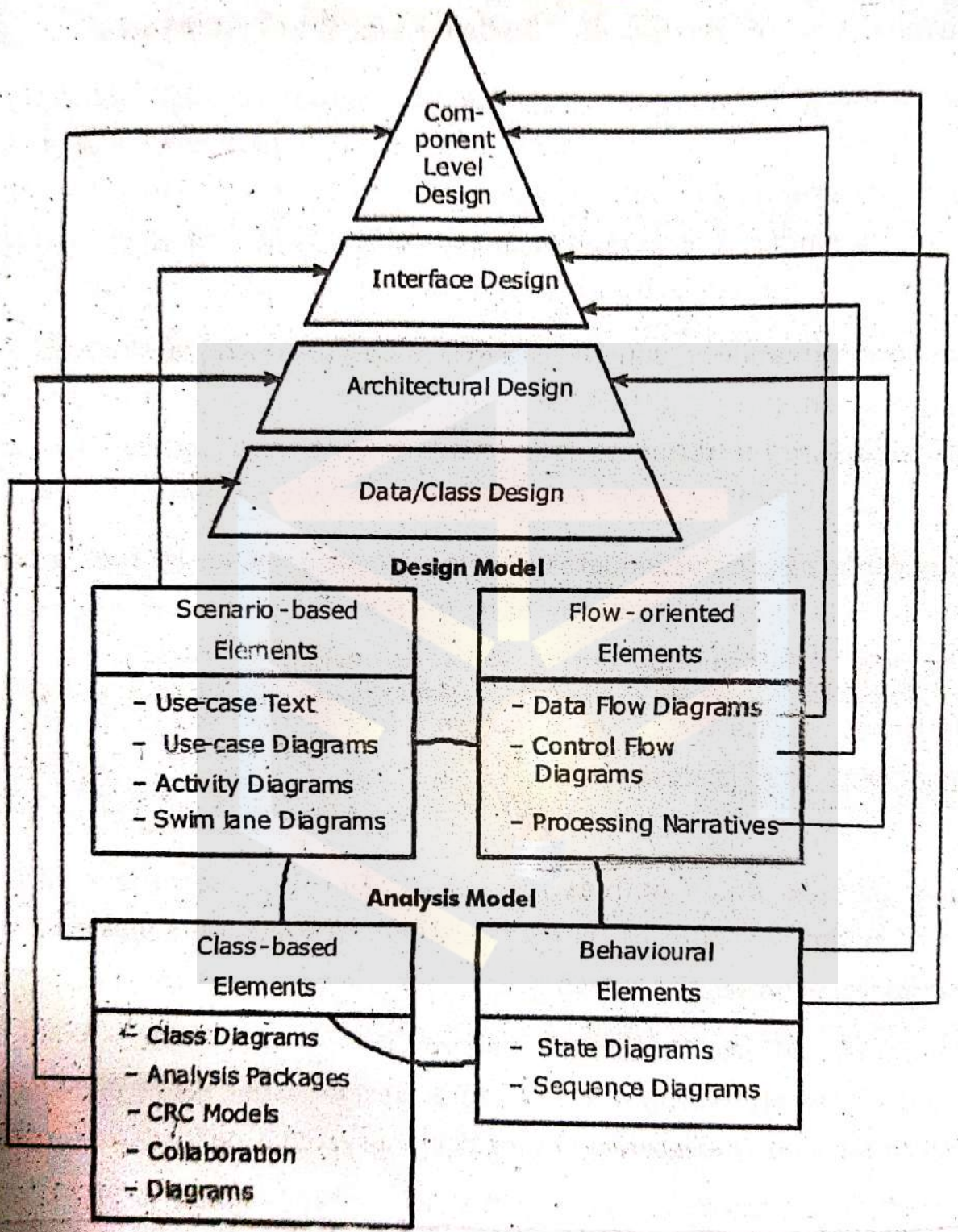
→ Design Engineering :-

Design engineering encompasses the set of principle, concepts & practices that lead to the development of a high quality system or product

→ Software design is the last software engineering action within the modeling activity & it is the stage ~~generation &~~ for construction (code generation & testing)



modeling activity & its



→ Design Process & Design Quality:

→ Design Process:

→ Design process is an step by step & repetitive procedure where software is simulated according the requirements

→ The simulation will be in detailed (high level of abstraction)

→ The design represented by this level can be directly traced to the specific system objective & requirements such as data, functional & behavioural requirements.

→ As said earlier, design process is required for better quality software. (This quality is achived by a series of formal technical reviews.

The 3 main characteristics are

i) Design must consider & fulfill all the customer's requirement that are implicit & must implement explicit requirements contained in the analysis model

ii) The design must be readable, understandable & as a guide for those ~~test~~ who generate code & for those who test & subsequent support the software

iii) Design should provide a complete picture of the software, addressing data, functional & behavioral domains.

→ Design quality :-

→ These are the few guidelines which are the characteristics of a good design.

1) A design should exhib an architecture that has
   a) Creation by recognizable architectural styles or pattern.
   b) Components leading to good design characteristic & evolutionary fasion which facilitates implementation & testing.
2) A design should be module base (partition of elements or subsystems).
3) A design should have different representation of data, architecture, interfaces & components.
4) design should explore the data structures useful for the classes to be implemented.
5) A design should consist of components that have independent functional characteristics.

6) A design should lead to interface that reduce the complexity

7) design should be achive from iterative method.

The above are the mandatory guidelines to be followed while desiging.

→ Characteristics of Good Design Proces

1) The notion of design should refluct its meaning.

2) Design should be implemented based on the information obtained from the requirement analysis phase of the software development.

3) The design should clearly represent the interface application in the system.

4) The design should consist of all the independent modules.

5) The design should exhibit clearly all the modules of the software

6) Various elements such as data architecture, interface & components, should be distinguished clearly.

7) Using the design, a software engineer should directly build data structure & start implementation.

8) A design should describe architecture, should encompass several components & have dynamic behaviour.

→ Quality attributes:

Hewlett Packard (HP) developed a set s/w quality attributes given by acronym FURPS

1) Functionality :-

i) Architecture :- It is a key property of software system (It describes the internal structure of software system) A good, simple structure makes the system easy to understand change, test & maintain.

ii) Correctness :- A design should be created as per the client requirements (collected & analyzed)

iii) Performance :- It is measured by processing speed, response time, resource consumption, thoughput & efficiency.

iv) Functionality :- It is assessed by evaluating feature set & capabilities of the program, generality of functions & security of overall system.

→ Design Concepts :-

Following is a set of Fundamental software design concepts

1) Abstraction
2) Architecture
3) Patterns
4) Modularity
5) Information hiding
6) Functional independence
7) Refinement
8) Refactoring
9) Design classes.

1) Abstraction :- (hiding data/code).

→ Abstraction simply means to hide the details to reduce complexity & increase efficiency or quality.

→ Different level's of abstraction are necessary & must be applied at each stage of design process so errors that are present can removed efficiently.

applied for → Functional abstraction (in functional methods)

→ Data abstraction (for data structure & data representation)

→ Control abstraction (to the flow).

2) Architecture :- (design a structure)

It is the structure or organization of program components (modules), their interaction & structure of data that are used by components.

↑ used to represent major system elements & their interactions.

3) Pattern :- (a repeated form)

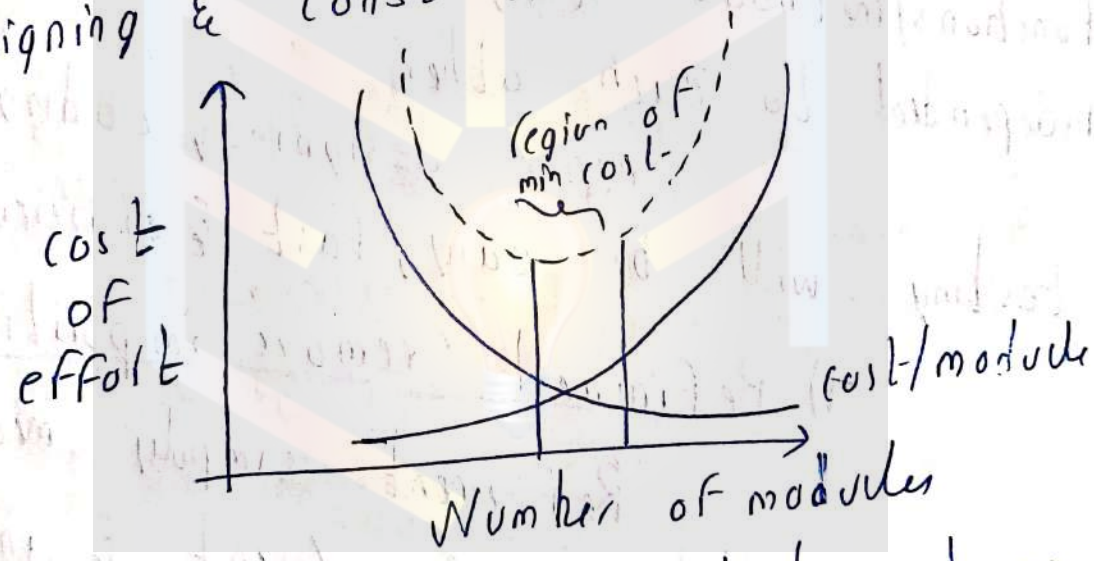
The pattern simply means a repeated form or design in which the shape is repeated several times to form a pattern.

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

4) Modularity (Subdivided the system)

→ modularity simply means to divide the system or project into smaller parts to reduce the complexity of the system or project.

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



→ Hence alway select ~~not~~ as medium module based on project cost & complexity

5) Information hiding (hide the info)

Information hiding simply means to hide the information the information so that it cannot be accessed by unwanted paths

6) Functional Indepndt (independent methods)

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

hence designing, coding, testing will be easy, fast & efficient //

7) Refinement (remove impurities)

Refinement simply means to find impurities if present & increase the quality.

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

8) Refactoring : (reconstruct something)

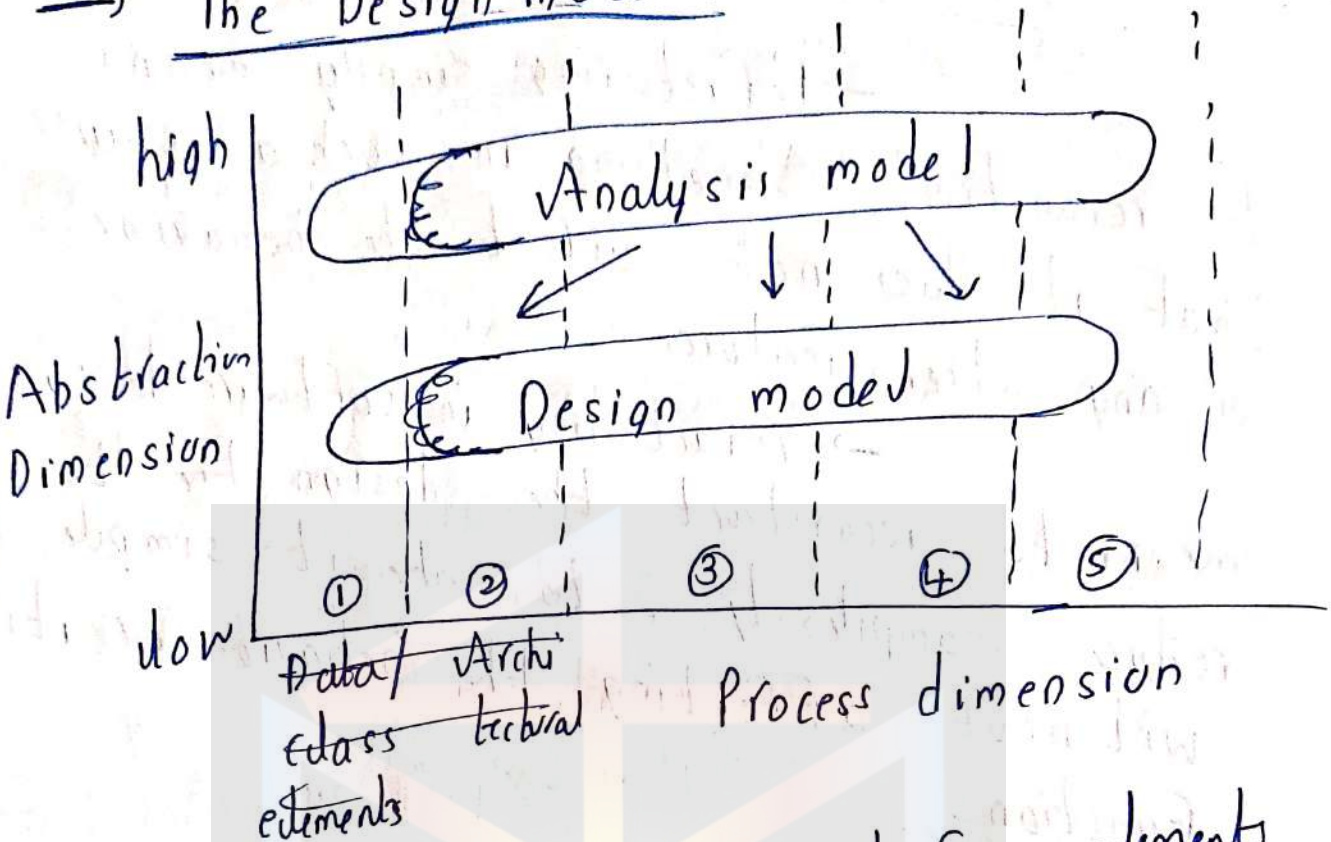
→ Refactoring simply means to reconstruct something in such a way that it does not affect the behavior or any other features

→ refactoring in software design means to reconstruct the design to reduce complexity & to make it simple. without affecting the behavior or its functions.

9) Design classes

→ System classes, Process class, Persistent class, user interface class & business domain classes are refered as design classes.

→ The Design model:



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

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

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

- (vertically) The abstraction dimension represent the level of details as each element of the analysis model is transformed into design model & then iteratively refined.

→ elements of the design model use many of the same UML diagrams used in the analysis model

→ They are only refined & elaborated.

→ more implementation specific is provided.

→ Data design elements :- It is also referred as "Data architecturing". Data design creates a mode of data and/or information that is represented at a high level of abstraction.

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

→ architectural Design element-s:

→ architectural design is nothing but a photocopy of the end software to achive that follow

i) knowledge on the application domain

ii) relationship & uml diagram are drawn

iii) proper architecture & styles.

→ Interface design element-:

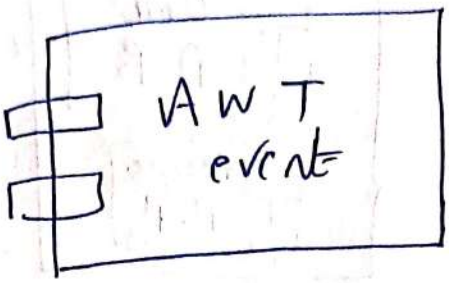
→ Tells how information flow into and out of the system

→ includes user interface, external interfaces & internal interface.

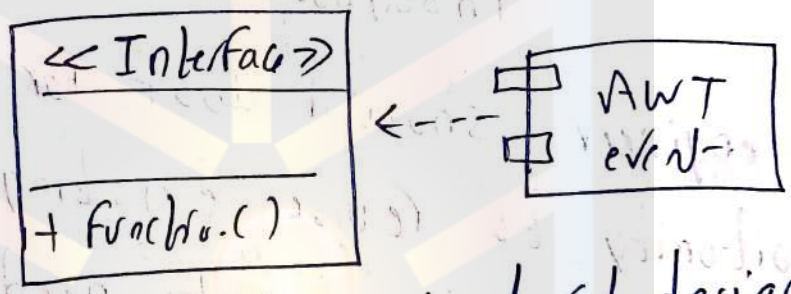
→ Component-level design

As the name suggest component level design provides all details of a given software component-

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

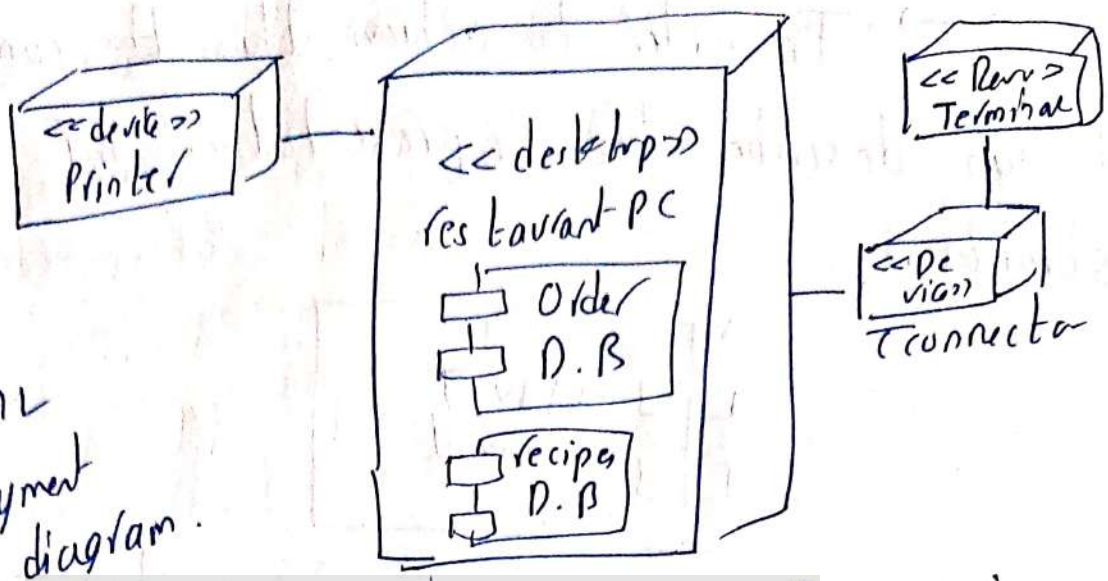


The component is inscribed into the interface (along with).



→ ~~Develop~~ Deployment level design elements

These indicate how software functionality & subsystems will be allocated within the physical computing environment that will support the software.



UML deployment diagram.

→ Pattern Based Software Design :-

Throughout the design process s/w engineer should look for every opportunity to reuse existing design patterns (when they meet need of the design) rather than creating new one.

→ mature engineering disciplines make use of thousands of design patterns

→ Design Pattern Template :-

Pattern name :- Describes the importance of the pattern in short (short & sweet name)

Intent: Describes the pattern & what it does (its use)

Also know as: Lists any synonyms for the pattern.

Motivation: Provides an example of the problem

Application: Note specific design situation in which the pattern is

Participant ~~Structure~~: Describes the responsibilities of the classes that are required to implement the pattern

~~Participants~~: Describes the responsibilities

structure: Describes the classes that are required to implement the patte

Related patterns: Cross references related to design patterns.

→ Using Patterns in design:- Design pattern can be used throughout s/w design. The problem ~~describe~~ description is examined at various levels of abstraction to determine if it related to one or more types of pattern

i) Architectural patterns:- These pattern
- define overall structure of software
- indicate relationship among subsystem & components.
- Define rules for specifying the relationships among the elements (class, components, packages, subsystems) of the architecture

ii) Design patterns:- These patterns addresses a specific element of the design such as an aggregation of components to solve some design problem.

iii) Codding patterns: They are also called idioms, these language specific patterns generally implements an algorithm element of a component, a specific interface protocol, or a mechanism for communication among components

→ Framework: A Frame work is not an architectural pattern, but rather a skeleton with a collection of plug points (also called hooks & slots) that enable it to be adapted to a specific problem domain.