Module III

Modelling with UML

⇒ UML : (unified modeling lang):

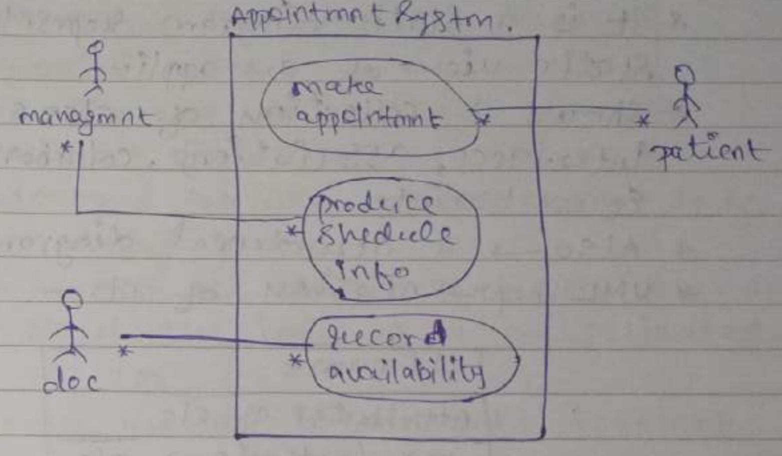
- It is a modeling lang used to visualize, specify, construct & documnt the artifacts (byproduct of (S) development) of a (S) systm.
- provides a set of notations (eg: ▭, lines, ellipse, etc) to create a visual model of a systm.

I Use case Diagram =

- To model a systm, the most imp aspect is to capture the dynamic behaviour. Dynamic behaviour means the behaviour of the systm whn it is operating.
- only static behaviour is not sufficient to model a systm rathr dynamic behaviour is more imp than static behaviour.
- In UML, there are 5 diagrams to model the dynamic ntre & use case diagram is 1 of them.
- usecase diagram is used for—
  - model the contxt of a systm.
  - reverse & forward engneering.

→ actor.

- eg: u.case d. for appointmnt systm.



1) actor role name → is a person | systm that derives benefit frm & is enternal to the systm.

2) (usecase name) → Represnts a major piece of systm functionally

3) [systm name] → include the name of the systm inside / on top

4) *————* → links an actor with the usecase with which it interacts.

Dynamic
use case

static
class

II class Diagram =

- * It is, a Static diagram reprsnts the Static view of an appli~
- * Shows a collection of clses, Interfaces, associations, collaborations & constraints
- * Also → a Structural diagram.
- * UML representation of cls —

cls name
Attributes of cls
opr / methods of cls

* eg → (module II UML diagram)

III Interaction Diagrams =

- * from the term interation, it is clear that the diagram is used to describe Some type of interactions amoung the diffrnt elemnts in the model.
- * This interaction is a part of dynamic behaviour of the systm.
- * This is reprsnted ~~by~~ in UML by 2 diagrams

a) Sequence Diagram :

- * They capture the interaction b/w obj in Sequencial order.
- * This diagrams are used by software developers & business professionals to understand requiremnts for a new systm.
- * S.D useful to:
  - Reprsnt the details of a UML use case
  - Model the logic of a sophisticated ()/opr.
  - see hw obj & components interact with each othr to complete a process

* eg:

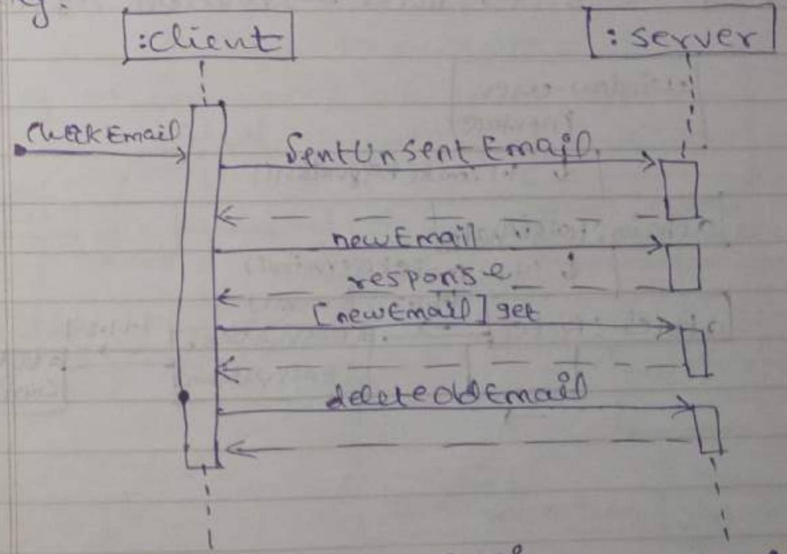
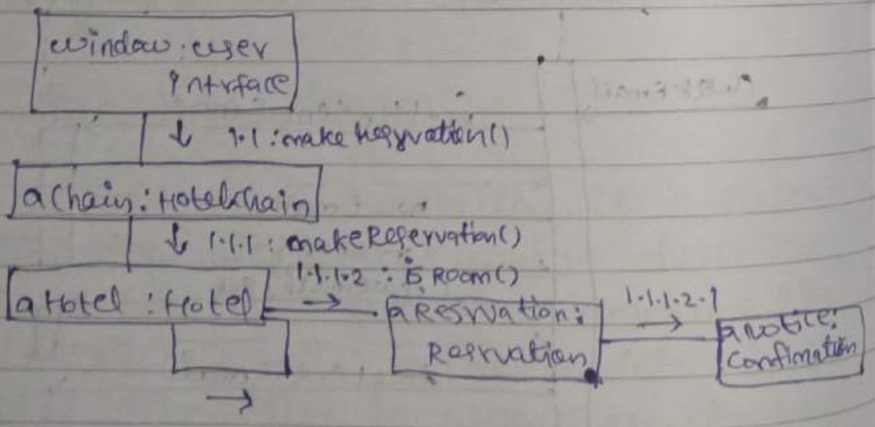


fig: S.D for Snding an Email.

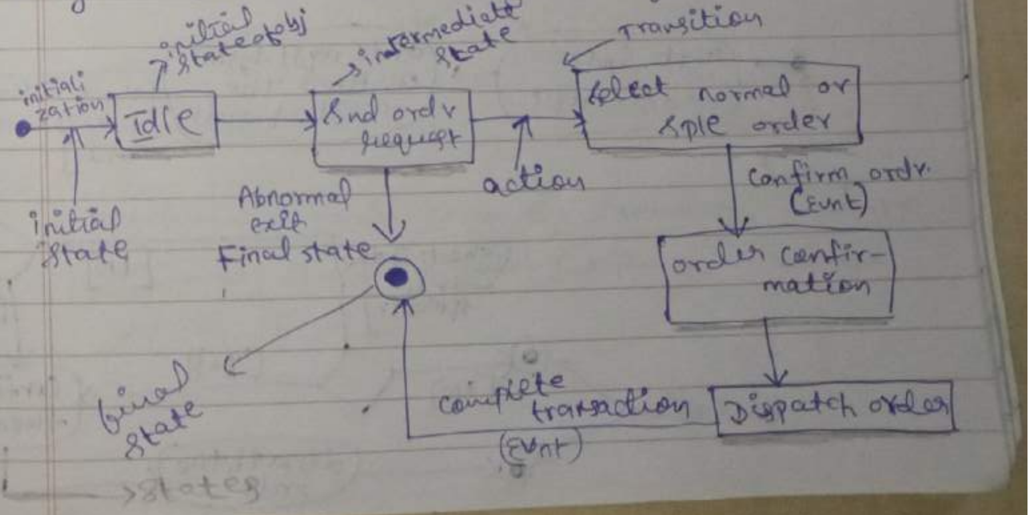
3) collaboration Diagram :

* Here, the method call sequence is indicated by some numbering technique.
* The no. indicates hw the methods are called 1 after anthr.
* Method calls are similar to that of a sequence diagram, diffrnce being the sequence diagram does not describe obj ~~orien~~ organization & the collaboration diagram shows the obj organization.
* eg: C.D for hotel resrvation systm.



IV State chart Diagrams :

* Describes the flow of control frm 1 state to anthr state.
* States are defined as a condition in which an obj exists & it changes whn some evnt is triggered.
* most ~~#~~ imp purpose of this diagram is to model lifetime of an obj frm creation to termination.
* Bfre drawing a statechart diagram we should clarify the following points —
  a) identify the imp obj to be analyzed
  b) identify the states
  c) identify the evnts.
* eg → for order managmnt systm:—



V Activity Diagram :

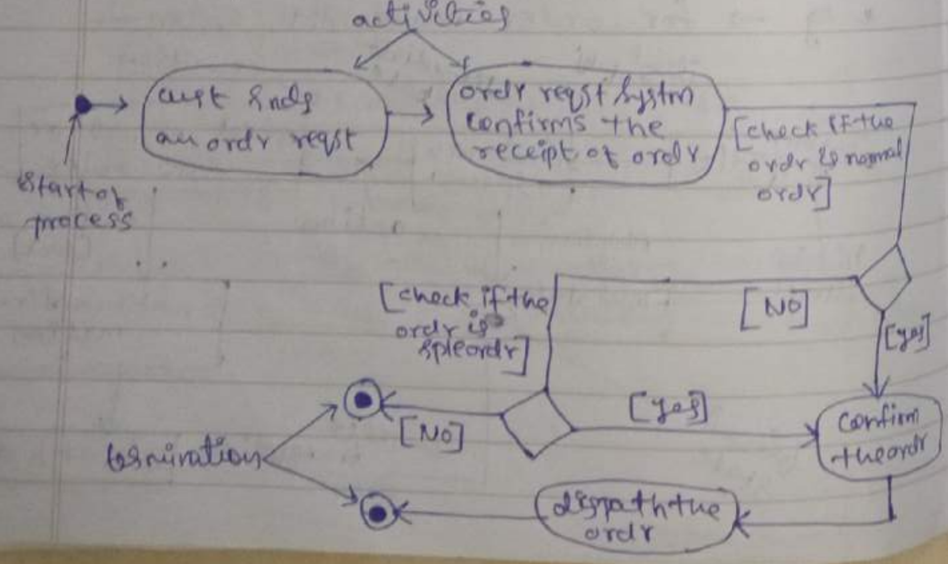
* It is basically a flowchart to reprsnt the flow frm 1 activity to anthr activity.

* The activity can be described as an opr of the system.

* control flow is drawn frm 1 opr to anthr. This flow can be sequential, branched or concurrnt.

* 4 main activities for order m'systm-

- Snd ordr by the cust
- Receipt of the ordr.
- Confirm the ordr
- Dispatch the order.



VI package Diagram :

* To organize complex cls diagrams, you can grp clses into packges.

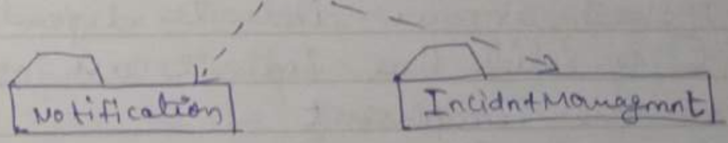
* A package is a collection of logically related UML elemnts.

* Notations —

a) packages appear as ▭ with small tabs at the top.

b) packge name is on the tab / inside the ▭.

c) dotted arrows are dependencies, 1 package depends on anthr

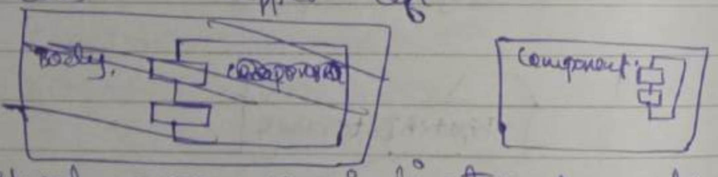


(notifi-ക്കോ IncidntM-നോ എന്തെങ്കിലും change വന്നാൽ അത് DispatchI-ൽ ഒന്നും change ആവില്ല).

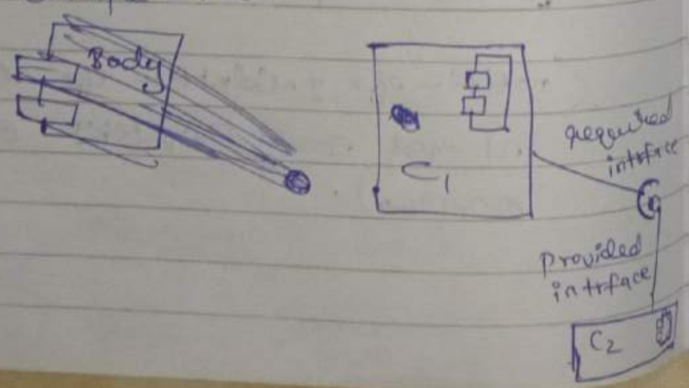
(D) -> diagram

VII Component Diagrams :

- * Shows various components in a system & their dependencies, intrfaces.
- * Explain the str of a systm.
- * Similar to package diagram in that both are used to grp elemnts into logical str.
- * with component (D) all of the model elemnts are private whereas pakage (D) only display public items.
- * components are shown as ▭s with 2 tabs at upper left —



- * Dashed arrows indicates dependencies.
- * ○ & solid line indicates an intrface to the component



* eg:

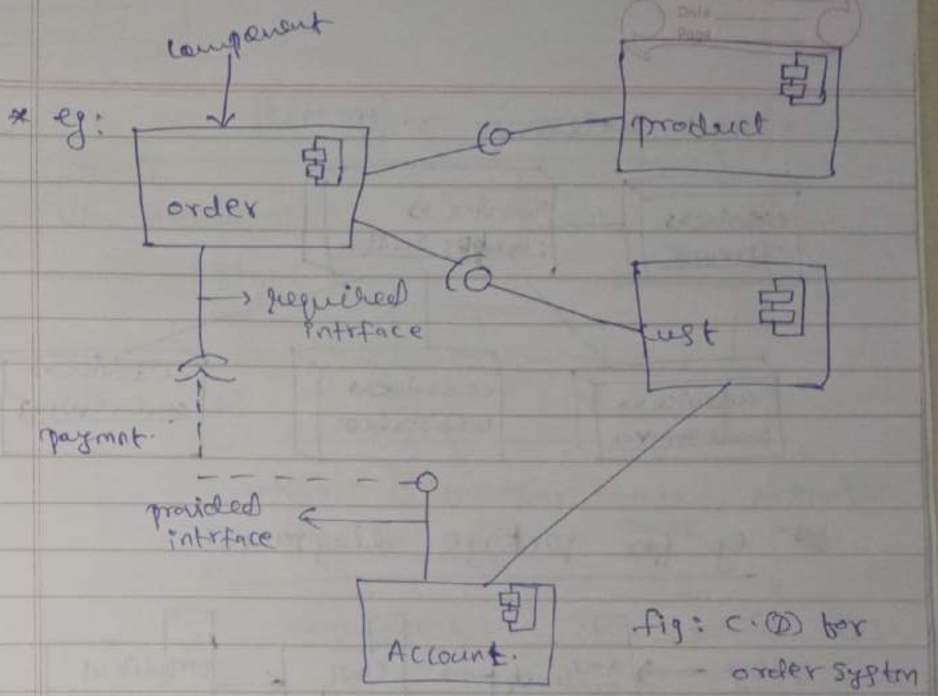
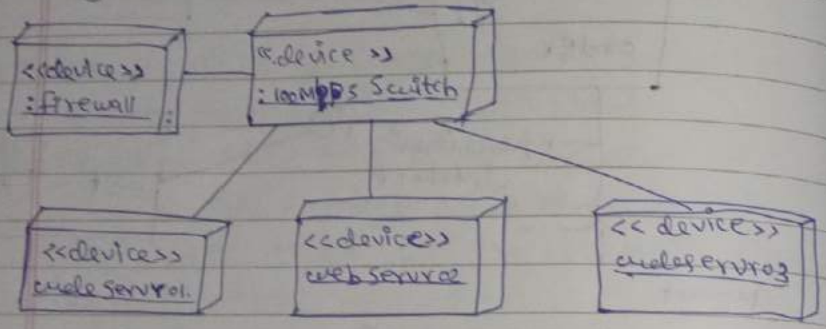


fig: C.D for order systm

VIII Deployement Diagram :

- * used to visualize the topology of the physical components of a systm, where ~~as~~ the software components are deployed ([illegible]).
- * used to describe the static deploymnt view of a systm.
- * consist of nodes & their relationships.
- * used to describing the hardware components, where software components are deployed.

* eg: webserver D. (D).



☆ eg for package diagram :

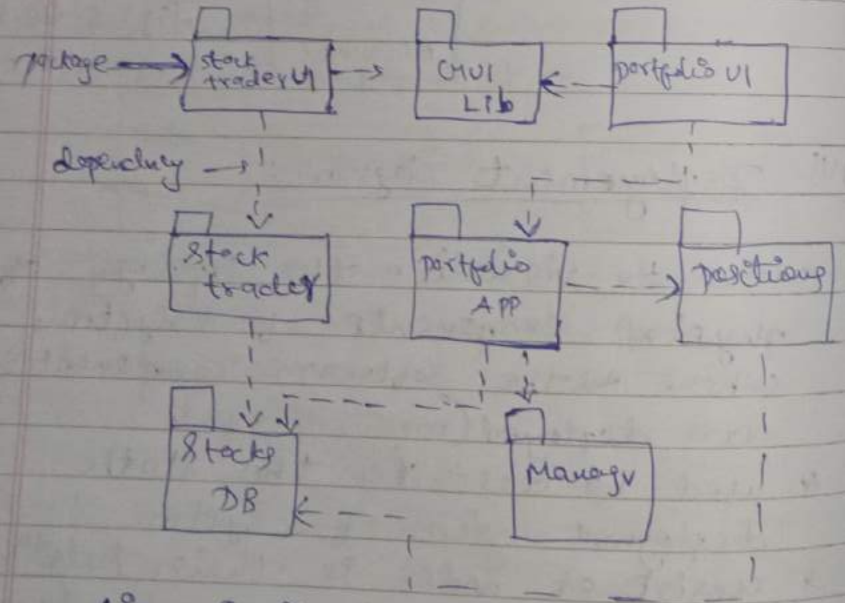
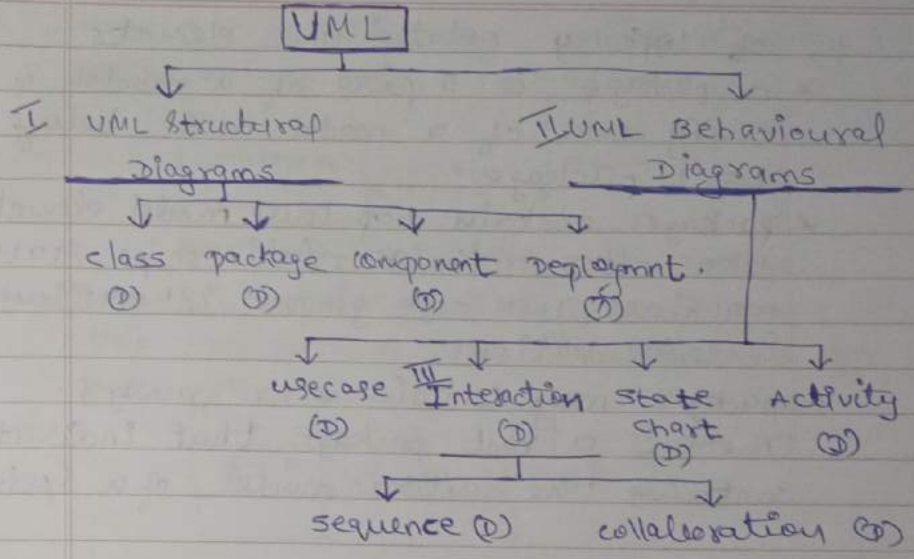


fig: P. (D) for a business.



→ UML Diagram organization =

* Models of complex Systms quickly become complx as developrs refine thm.
* UML model managmnt provides a mechanism for diagram organization that consists of packags & depndncy relationships.

I packages :

* packages in UML mechanism for grping

of logically related UML elemnts.

- A package is a piece of a model & every part of a model must belong to 1 package.
- packages contain top-level model elemnts like clses & their relationships, state machines, use case graphs, interactions & collaborations.
- packges may contain othr packages. There is a root package that indirectly contains the entire model of a systm.

II Dependncies on packages :

- Dependncies amoung packages summarize dependencies amoung elemnts in thm.
- The top-down approach reflects the overall systm architecture.
- The bottom-up approach can be automatically generated frm the individual elemnts.
- packages are drawn as □s with tabs on thm.
- A package cannot access the contnts of anthr packages.

- The access depndncy applies directly to packages & othr containers.

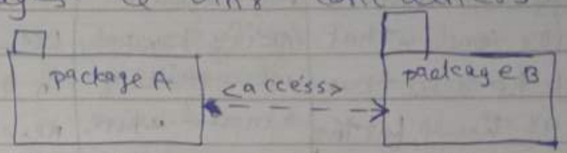
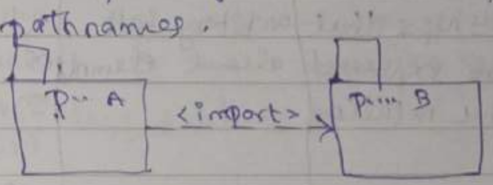


fig: Access depndncy b/w packages.

- import depndncy used to add names to the namspace of the client packagè for full pathnames.



→ UML Diagram Extension =

- UML promotes a usecase driven, architure-centric, iteractive & incremntal process which is obj oriented & component-based.
- The extensions are intended for particular appli- domains / prgming envrmnt.
- 3 extensibility mechanisms in UML –

a) constraints	(b) Tagged values	(c) stereotypes
• It is a semantic restriction represented as a	Are used to extend the (pro) of UML building block.	Allow you to extend the vocabulary

a txt expression.		of UML.
• Specify semantics and/or condi that must be held true at all times for the elements of a model	Allows you to Specify keyword valuepairs of a model where keywords are the attributes	used for classifying UML building blks in ordr to introduce new building blks
• Express restrictions & relationships that cannot be expressed using UML notation	used to store arbitory info about elemnts	The info contnt of the stereotyped element is the same as the existing model element.

☆ Constraint :-

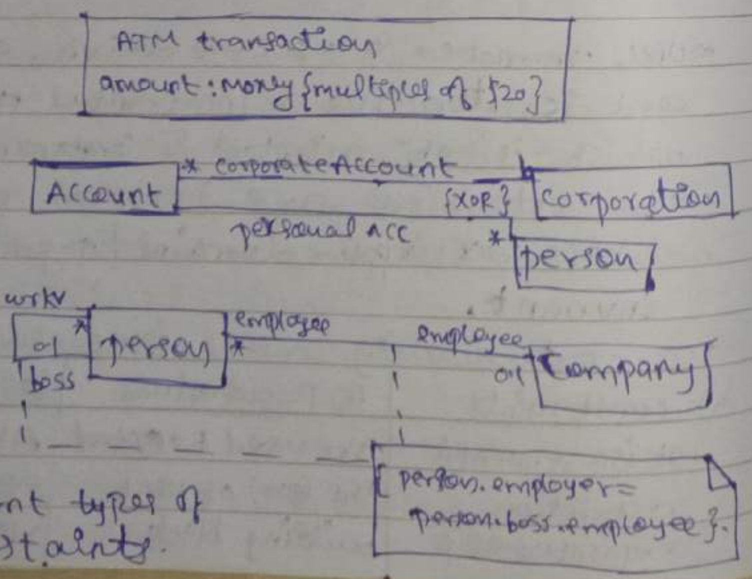


fig: diffrnt types of constaints.

☆ Tagged value :-

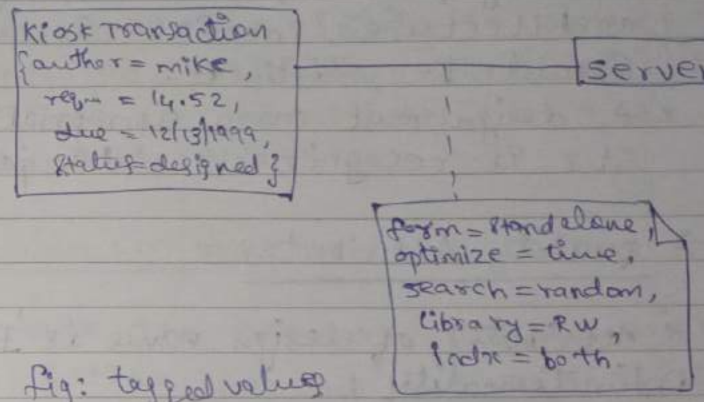


fig: tagged values

☆ Stereotypes :-

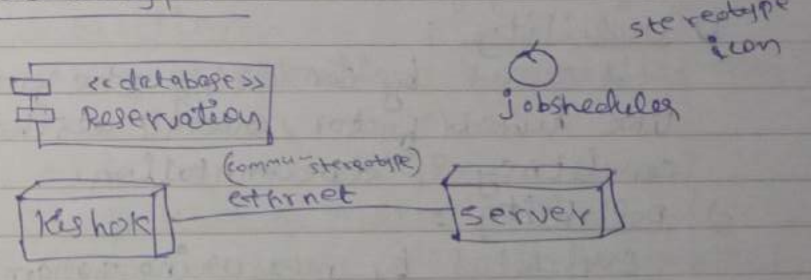


fig : stereotypes.

⇒ Design process =

* A design of the Software must be modular (i.e) Software must be logically

partitioned into elemnts .

* In design, the represntation of data, architecture, intrface & components should be distinct.
* A design must carry appropriate data str & recognizable data, patterns.

→ quality Attributes =

* Attributes of design name as 'FURPS' –

a) Functionality :
Evaluates the feature set & capabilities of the prgm.

b) usability :
Accessed by considring the factors like human factor, overall aesthetics, consistncy & documntation.

c) Reliability :
Evaluated by measuring paramtrs like freq & security of failure, o/p rslt accuracy, the mean-time-to-failure & the prgm predictability.

d) performance:
measured by considring processing speed, response time, resource consumption, throughput & efficiency.

e) Supportability :
combines the ability to extnd the prgm, adaptability, serviceability.
These 3 terms defines the maintainability.

→ Design Concepts =

provide the software designer with a foundation frm which more sophisticated methods can be applied.

I Abstraction :

* process of generalization by reducing the info contnt of an obser.
* 4 types of Abstraction –

a) Highest level of Abstraction :-
Refers to most abstract & general concepts in a (S) System, related to high-level architecture & design pattrns.

b) ~~S~~lowest level of A) :-
most detailed & specific aspects of a (S) System related to low-level

Implemntation details like machine code

c) procedural (A) :-

- * provides mechanisms for abstracting well defined procedures / op^r as entities.
- * The implemntation of procedure requires a no of stps to be performed.
- * eg :- delete opr.
- * used extensively by req analysts as well as designers & prgmrs.

d) Data (A) :-

- * process of creating of data type, that hides the details of the data reprsntation in ordr to make the datatype easier to work with.
- * Involves creating a reprsntation for data that separate the interface frm implemntation.

II Software Architecture :-

- * The complete architecture of (S) → (S) architecture.
- * Str provides conceptual integrity for a Systm in a no. of ways.
- * The architecture is the str of prgm modules where they interact with each othr in a specialized way.
- * The components use str of data.
- * The aim of (S) design is to obtain an architectural framework of a System.
- * Architecture models —
  - a) Str models
  - b) Framework "
  - c) Dynamic "
  - d) process "
  - e) Functional "

III Design patterns :-

- * It is general, reusable sol^n to a commonly occuring prbm with a gvn contxt in (S) design.
- * It is a description for how to solve a prbm that can be used in many diffrnt situations.
- * patterns are templates that are to be implemented in the crt situation

* 3 basic kinds of D. patterns—

a) str patterns deal with relationships b/w entities.

b) creational patterns provide instantiation mechanisms.

c) Behavioural patterns used in commun b/w entities.

IV Separation of concerns = (SoC)

* SoC refers to the delimitations & correlation of (S) elements to achieve order within a system.
* SoC promotes the idea that keep yr design as loosely coupled as possible.
* principle of separation of concerns states that system elemnts should hve exclusivity & singularity of purpose
* It is achieved by establishmnt of boundaries.
* eg for boundaries—
  projects, solⁿ, & folder hierarchies.
* various techniques used for S. of C—

a) Horizontal separation.

b) vertical "

c) Aspect "

d) Behaviour "

e) Delegating "

V Modularity =

* refers to the extnt to which a (S) appli- may be divided into smallr modules.
* It is a practical appli- of the principle of "separation of concerns" by ÷ a complx system into simpler & more managble modules.
* Takesplace in 2ways—

a) composition takes modules & put thm togther to form a largr system.

b) decomposition take a complete system & decompose it into its modules.

VI Info Hiding =

Technique of encapsulating (S) design decisions inside modules & othr (S) components.

It is an imp aspects of abstraction specifically, considr that the final (S) system is the lowest level of abstraction.

common use of info. H is to hide the physical storage layout for data so that if it is changed, the change is restricted to a small subset of the total prgm.

concerns → details of hardware for an appli.

VII Functional Independence =

* It is a direct rslt of separation of concerns, modularity i.e the concepts of abstraction & info hiding.

* occurs where modules address a specific & constrained range of functionality.

* It can be judged in 2 concepts –

a) cohesion : It is a measure of functional strength of a module.

b) coupling : It is a measure of the degree of interaction b/w 2 modules.

VIII. Stepwise Refinement

It is relatively old technique of (S) design that has been successfully used in a wide range of structured prgming & modular prgming lang.

It is the stp by stp transformation of an abstract (high level) formal specification into a concrete (low level) executable prgm.

At each stp of S.R, 1/several components of the gvn soln are decomposed into more detailed soln.

IX Aspects =

* A representation of a cross-cutting concern that must be accommodated as refinement & modularization occur.

* eg :- module A cross cut module B means B cannot be satisfied without considring A

X Code refactoring =

Refactoring is a organization technique that simplifies the design of a component without changing its ().

whn (S) is refactored, the existing design is examined for redundancy, unused design elements, inefficient, etc.

⇒ obj- oriented Design concepts = (OOD)

I objects :

* obj are basic runtime entities in an obj-oriented system.

* Have both state & related behaviours.

II classes :

It is a blue print frm which objects are created.

clses are composed frm structural &

	behavioural Constituents. Structural cons- defines data field associated with state variables. Behaviour of cls is defined using methods
* methods operate on an obj's internal state & serve as the primary mechanism for obj-to-obj commu..	methods are subroutines with the ability to operate on obj / clses.
* In oop terms, obj is an instance of a cls. obj take up space in the m/y & hve an associated address. whn a prgm is executed, the obj interact by sndng msges to 1 anothr.	A cls implementation consist of 2 parts — Intrface is the outside view of the cls & this part is visible to evrybody. implemntation is the actual code that implemnts the behaviour of the cls.

III Attributes :	IV Methods :
* Are attached to clses & they are structural constituent that defines (pro) of an obj.	An obj encapsulates data (collection of attributes) & the algorithms that process the data. These (al) → opr, method
* Are individual things that diffrnciate 1 obj frm anothr & determine the appearnce, state / othr qualities of that obj.	obj hve both (pro) & behaviour.. (pro) are represnted as attributes of obj & behaviours are reprsnted as methods of obj.
* Are defined in clses by variables. These vartypes & names are defined in the cls, & each obj can hve its own values for those var.	(pro) & behaviour comprises an intrface of an obj, which specifies hw the obj may be utilized by any of various consumrs of the obj.

* OOD is the process of using an obj-oriented methodology to design a computing system / appli..
* OOD serves as part of the oop process.

=> obj - oriented Analysis & Design Concepts =

* OOAD is a popular technical approach for analysing & designing a (S) system by appling oop.
* Entity clses reprsnt things that are to be stored in a db.
* Boundary clses create the intrface that the user sees & interact with (S).
* Controller cls's are designed to manage the creation / update of entity obj.

I Inheritance :

* process of deriving new cles (sub-clses) from existing ones (spr-clses).
* A sub-cls inherits visible (pro) & methods from its parnt. They can add var & methods to the ones they inherit frm the spr cls.
* using this mechanism, the prgmr can build an cls hierarchy which is an deep as needed.
* OOD sprt diffrnt types of (I) - single (I), multi-level (I), multiple (I), multipath (I), hyrarchical (I) & hybrid (I).

II polymorphism:

means the ability to take more than 1 form.

It plays an imp role in allowing obj having diffrnt internal str to share the same ex intrface.

orride -> അസാധുവാക്കുക.

It is extensively used in implemnting inheritnce.

2 types of (P) —

(1) Static (P):

- Also -> compile time (P)
- (P) that is resolved during compiler time.

(2) Dynamic (P) :

- Also -> runtime (P).
- process in which a call to an orridden method is resolved at runtime.

III Msg passing :

* The obj are made to commu- with each othr with the hlp of a mechanism -> msg p.

IV Design clses :

Req model defines a complete set of analysis clses. Each describes sme elemnt of the prbm domain, focusing on aspects of prbm-

* The msgs are snd & receive by passing various var among specific methods using the signature of methods.	5 diffrnt types of D.C. 1) user intrface cls: define all abstractions that are necessary for human-comp intraction (HCI) 2) Business domain cls: Are refinemnts of analysis clses defined earlier. 3) process cls: implemnt lower-level business abstrations. 4) persistnt cls: represnt data stores that will persist beyond the execution of the (S) 5) System cls: implemnt (S) managmnt & Control ()s.

=> Design Model =

* Design is the 1st phase of transforming the prbm into a soln.
* It use appropriate (S) models for expressing a (S) design.

* It is an iterative process through which req are translated into a blueprint for constructing the (S).
* It can be viewed in 2 diffrnt dimensions—

a) process dimension:

Indicates the evolution of design model as design tasks are executed as part of (S) process.

b) Abstraction dimension:

Reprsnts the level of detail as each elemnt of the analysis model is transferred into design equivalent.

* Elemnts of design model use many of the same UML diagrams that were used in the analysis model.
* 5 designs the design phase produces —

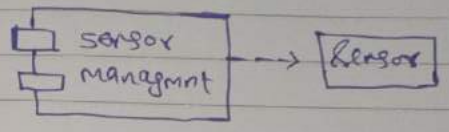
I Data (D) :	II Architectural (D):
* It produced a model of data that reprsnt a high level of abstraⁿ * This model is thn more refined into	provides us overall view of the system. Reprsnted as a set of intrconnted

more implemntation.

* The str of data is the most imp part of (S) design.

Subsystm that are derived frm analysis packags in req model

eg: UML c.(D) for sensor managmnt



III. component level (D):

* It is similar to set of detailed specification of each room in a house
* It completely describes the internal details of each (S) component.

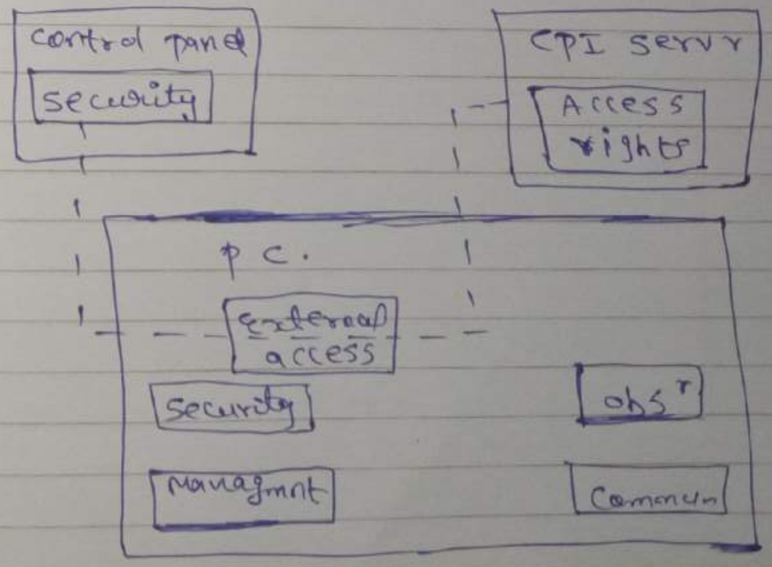
* processing of data str occurs in a component & an Intrface which allows all the component opr.
* UML diagram is used to reprsnt the processing logic.

IV Interface (D):

Reprsnts the info flow within it & out of the systm. They Communicate b/w the components defined as part of architecture.

V Deploymnt level (D):

* Shows the (S) functionality & subsystm that allocated in the physical computing envnmnt which spt (S).
* eg:- 3 computing envmnt - PC, CPI servr & control panel.


control panel
security
CPI servr
Access rights
P C.
External access
Security
obsr
Managmnt
Commun