Date:- 13/Feb/19

Activity diagram:-

These are normally employed in business process modeling. This is carried out during initial stages of requirement analysis and specification. They are useful to understand complex processing activities, involving many components. Later these diagrams can be used to develop interaction diagrams which helps to allocate responsibilities to classes. They can be used for describing either use cases or complicated methods. Activity diagrams are similar to the flow charts. The main difference is that it supports parallel activities. These parallel activities are represented by using swimlanes (.... line).

Activity diagram tools:-

1. Activity 

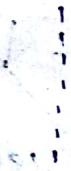
Ex:- Login activity.

2. State 

3. Decision box 

4. start state 

5. end of the state 

6. Swimlane 

7. synchronization

8. Transition 

En:- Login

Form verification

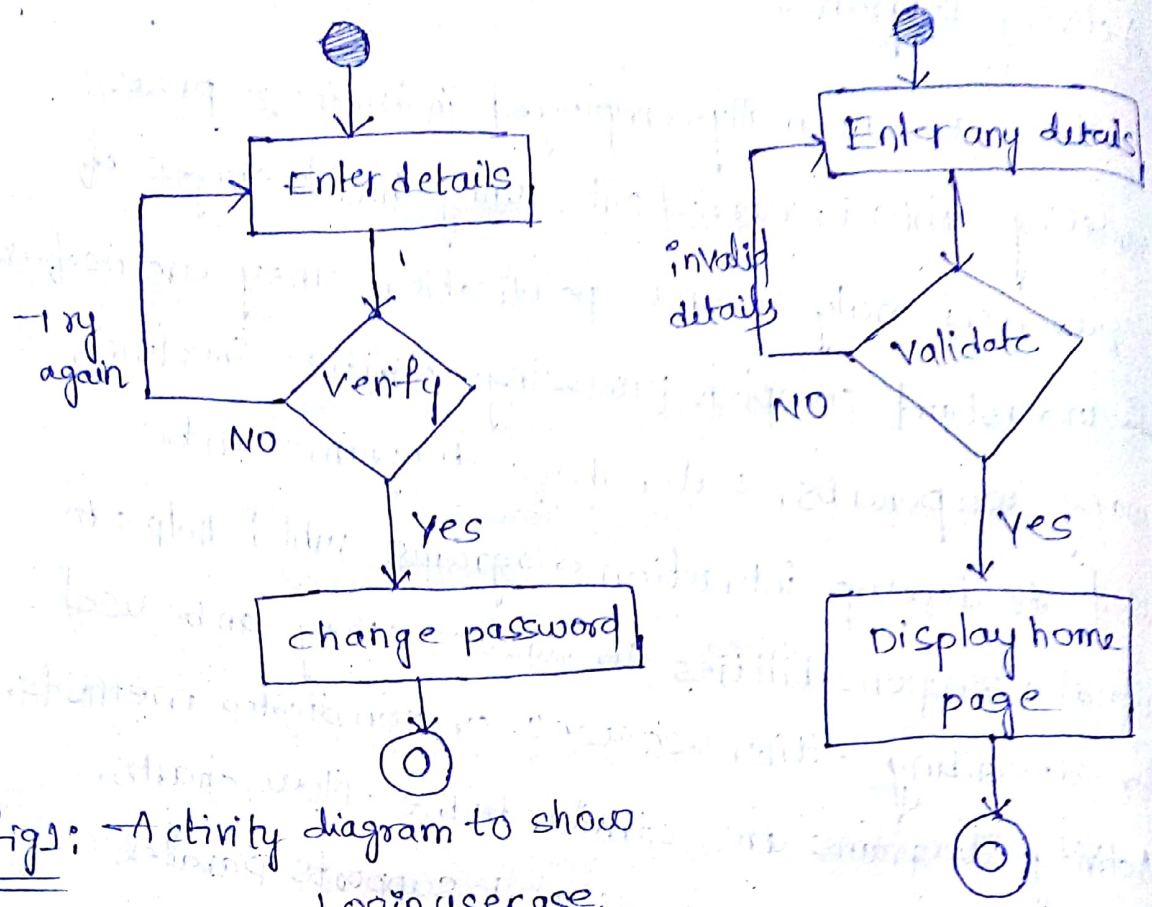
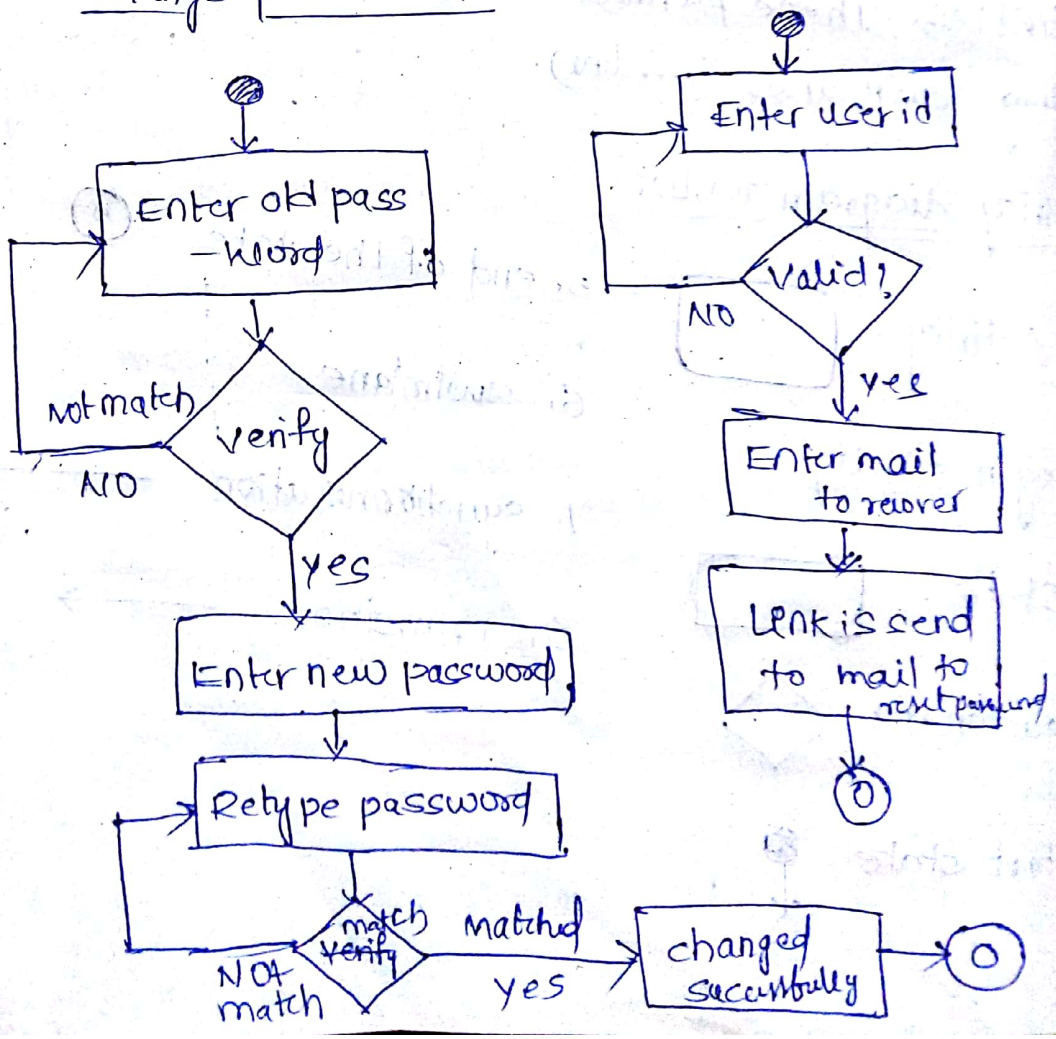


fig:- Activity diagram to show Login usecase.

En:- change password

En:- Forgot password

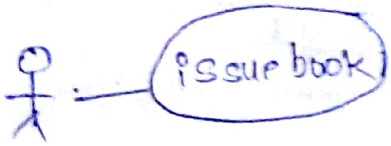


Aim:- Activity diagram for student Admission

Date:- 18/feb/19

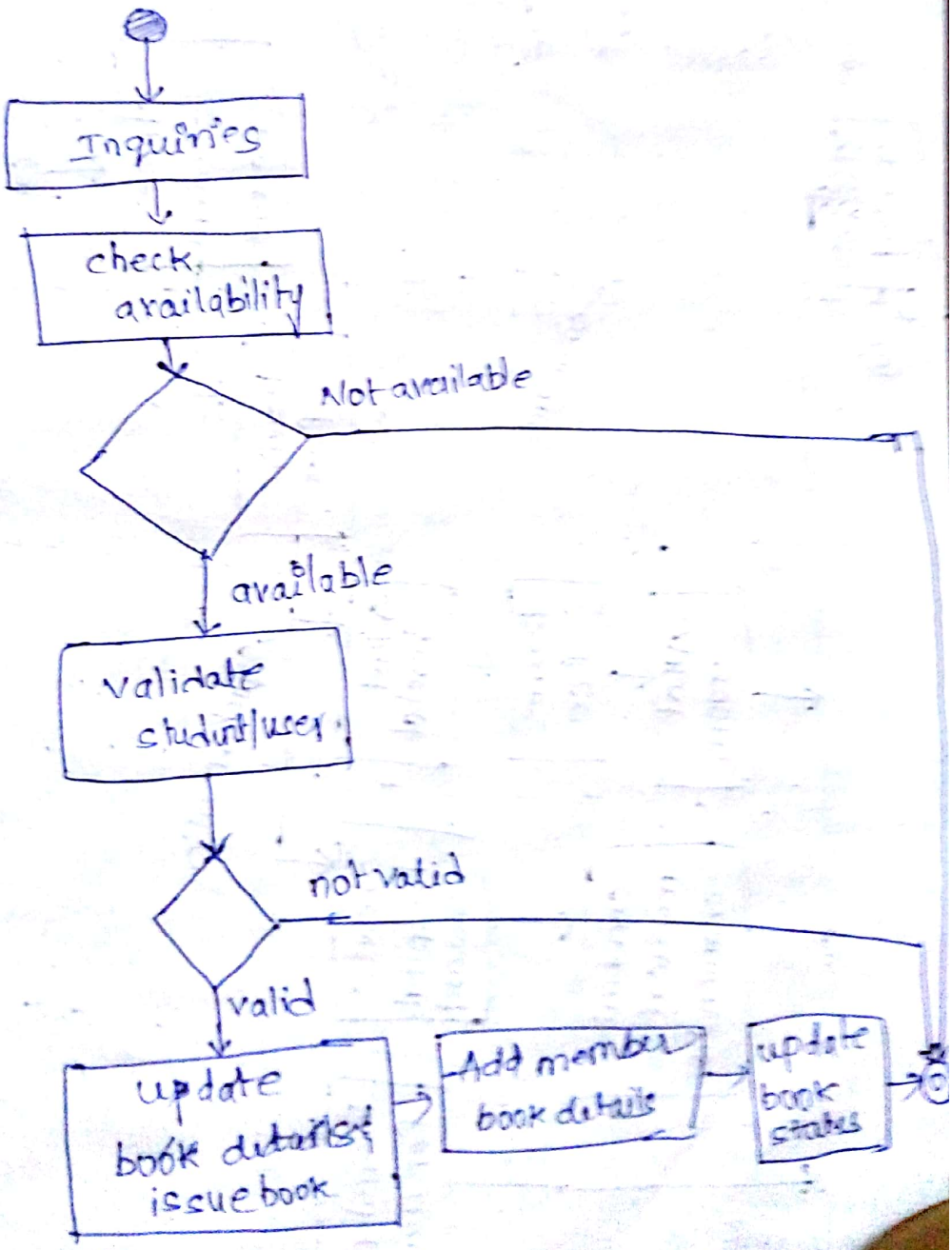
Eg:- Library Management system

(Librarians)



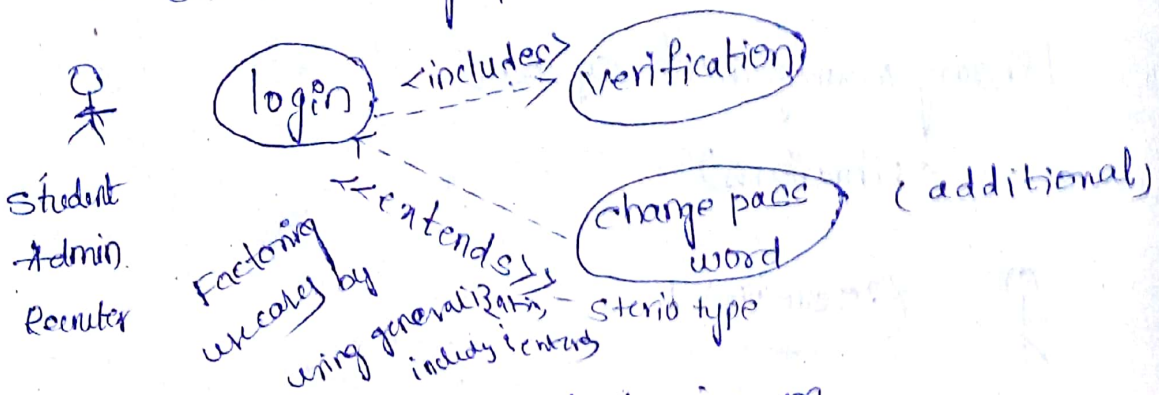
Librarian

Activity diagram to show issue book process activities



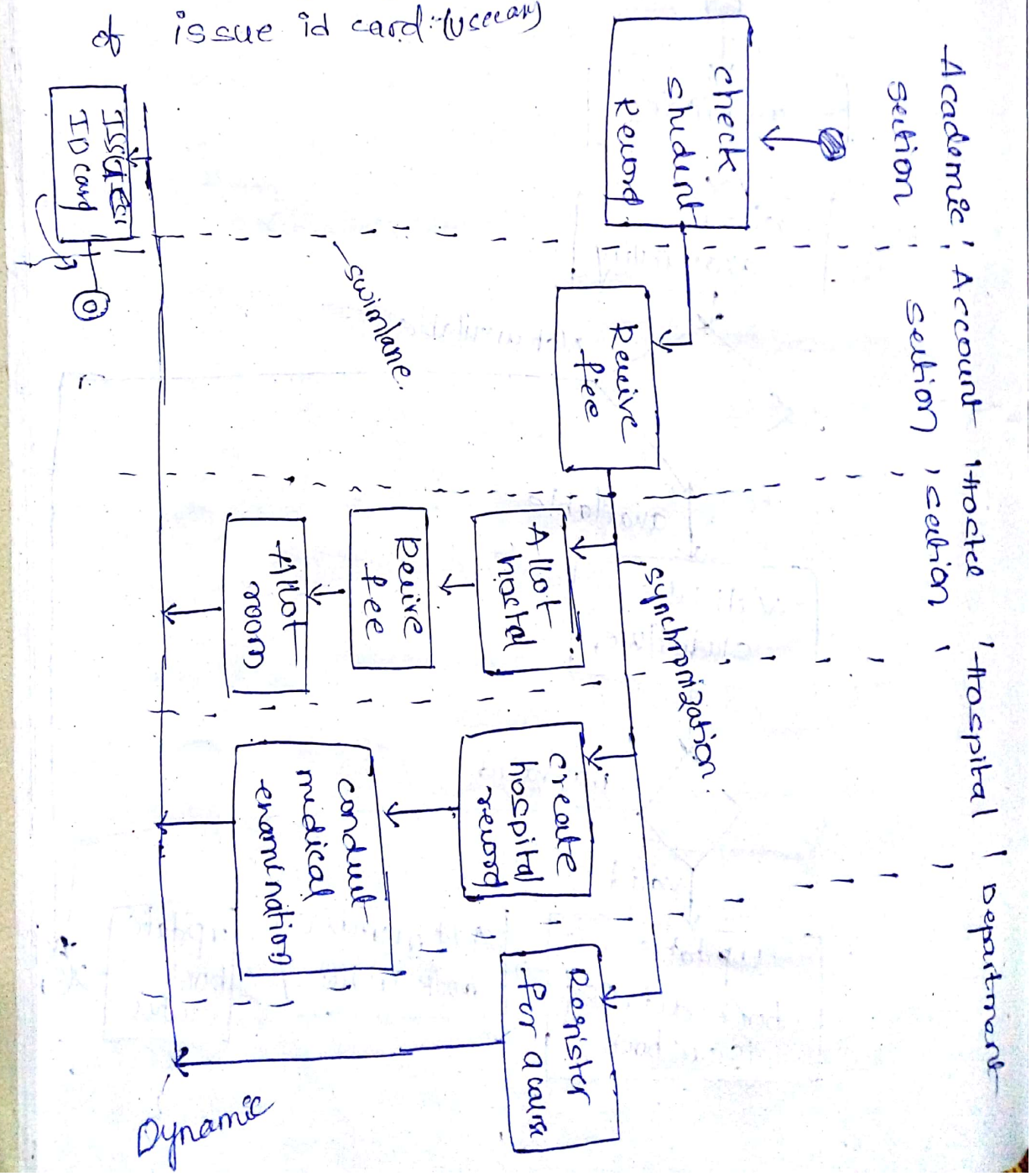
Use case diagram

eg:- Online training & placement



→ Parallel activities in Activity diagram

eg:- Activity diagram to show the processing activity of issue id card: (use case)



Date:- 19/Feb/19

State chart Diagram:-

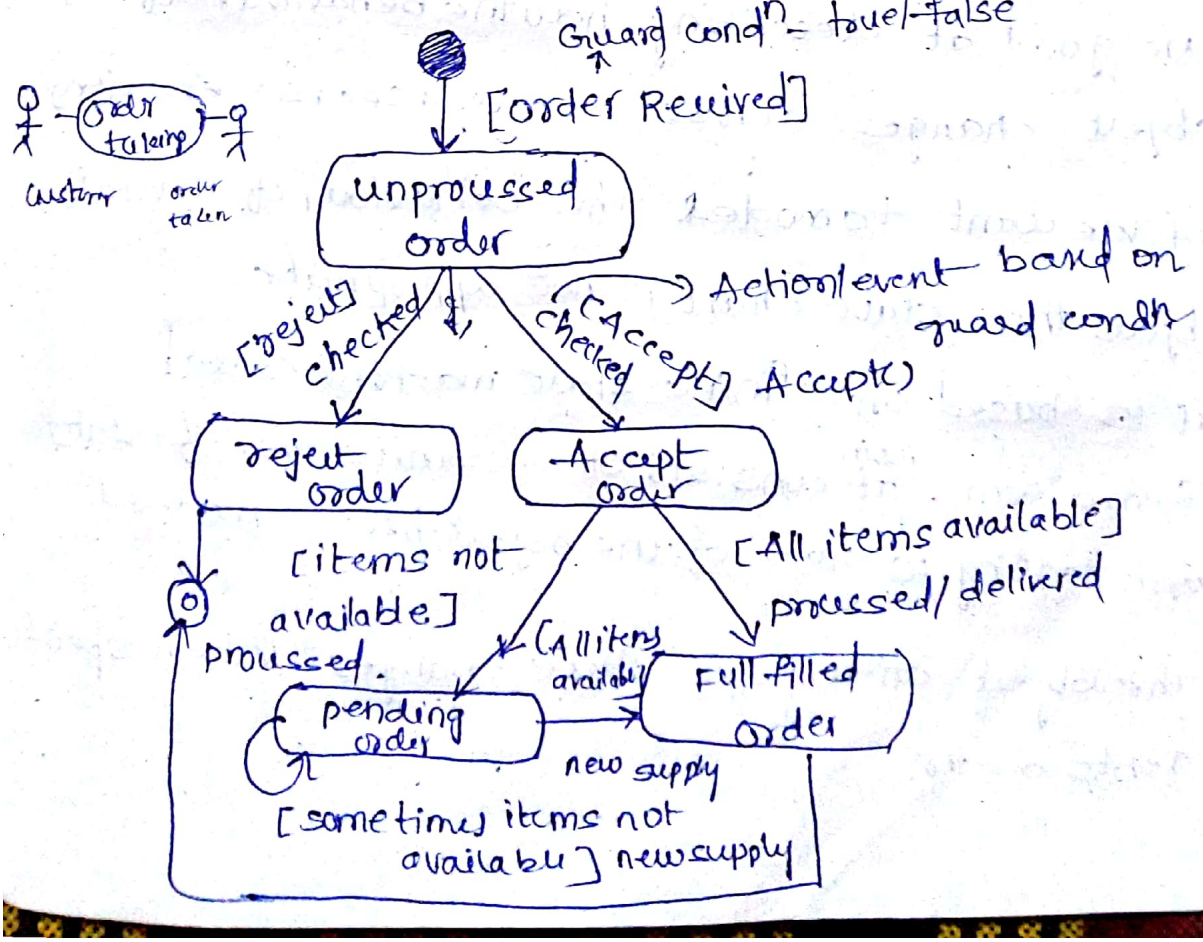
↳ Explains/shows the life cycle models of an object which is executing the usecase.

→ How the object is changed during its lifetime while executing the usecase.

Basic symbols:

1. Initial state → ● → object will not be created actually
2. Final state → ⊙ → object is destroyed in this state
3. state →  → shows the state of an object which is changed after transition from 1 state to other.
4. transition → ⟶ → shows change in state of an object

Example:-



Swimlane :- seperates the activity diagram into vertical zones by dashline (- - -). Each zone represents the responsibilities of a particular class or department.

When to use Activity diagram

→ use them for analyzing use cases and understanding the work flow across many users. It can deal with multithreaded applications.

→ Don't use them!

→ To see how objects colloborate.

→ To see how an object behaves over its lifetime.

Statechart diagram :-

→ It is normally used to model how the state of an object changes in its lifetime. These diagrams are good at describing how the behaviour of an object changes across several usecase evntions.

→ If we want to model the behaviour of several objects then state chart is not appropriate.

→ It is based on finite state machine called formalism. FSM It consists of a finite no. of state corresponding to those of the object being modeled.

→ The object undergoes state changes when specific events occur.

→ Too many no. of states in FSM causes too complex model.

→ This can be avoided by state chart diagram by using nested state or composite state.

+1/10 Basic elements with description :-

1. Initial state :- The initial state represents the source of all objects.

→ It is not a normal state, because objects in this state do not yet exist.

2. State :- The state of an object always determined by its attributes and associations. states in state chart diagrams represent a set of those value combinations in which an object behaves the same in response to events.

3. Transition :- A transition represents the change from one state to another →

4. ~~External~~ Guard condition :- A guard condition is a condition that has to be met in order to enable the transition to which it belongs.

[Guard condition]

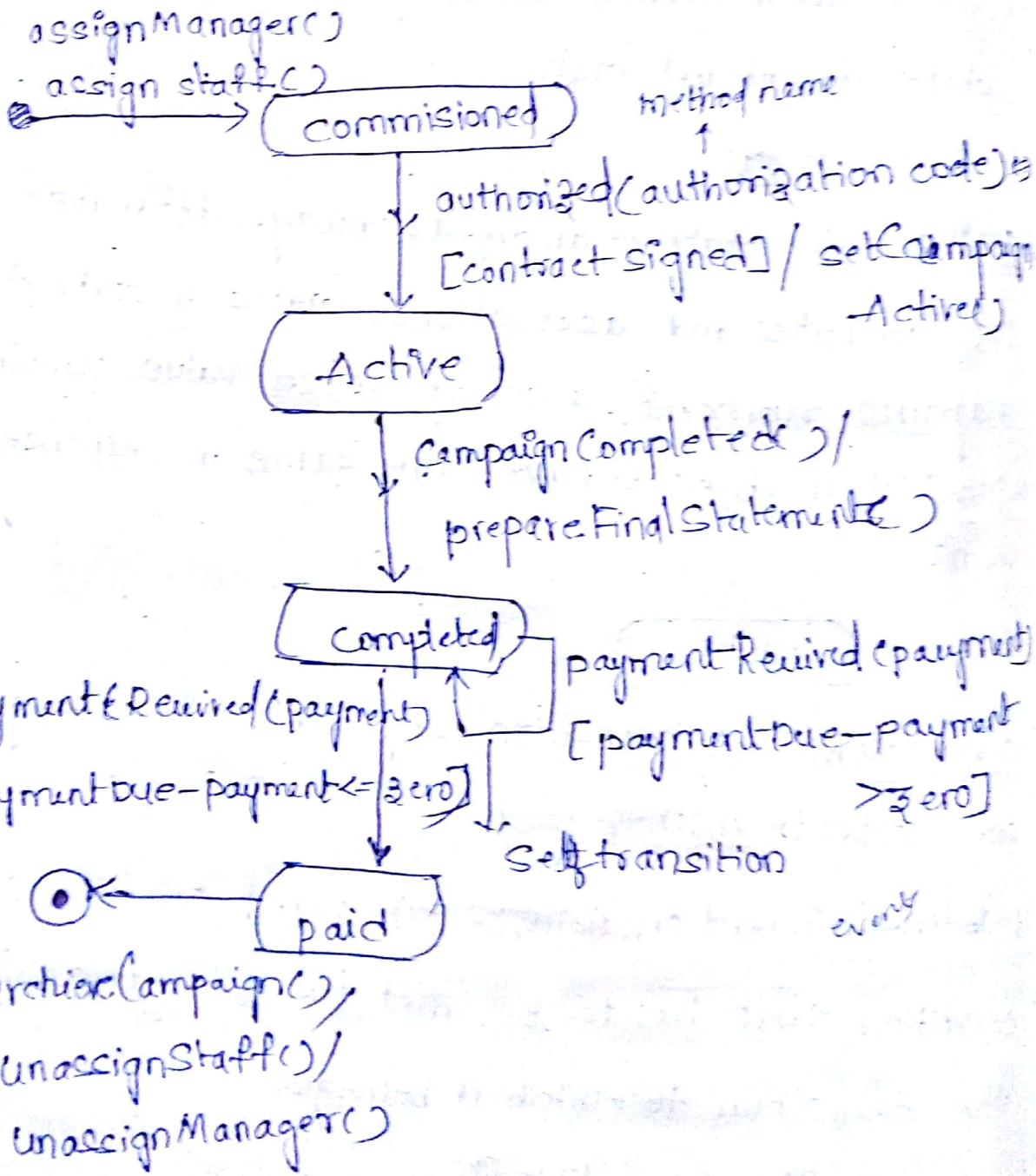
Guard condition can be used to document that a

different transitions.

5. Final state:- The final state represents the end of an objects existance.

The final state is not a real state, because objects in this state donot exist anymore.

Eg:- State chart diagram for campaign class



Date: 22-Feb-19

* capturing the usecases:- During the requirements work flow we identify user and customers needs as requirements. Functional requirements are expressed as usecases in a usecase model, other requirements are either attached to the usecases that they concern or kept in a seperate list or described in someother way.

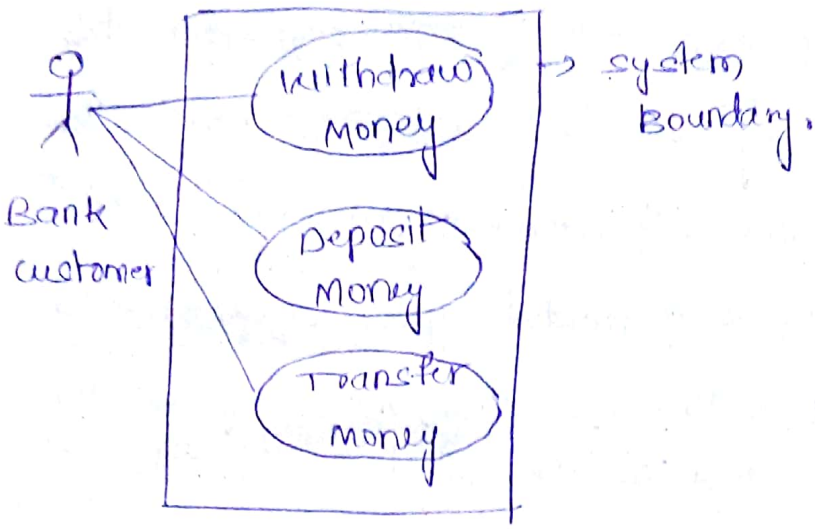
* usecase model represents functional requirements

The usecase model helps the customers/users and developers agree on how to use the system. most systems have many types of users. each type of user is represented as an actor. Actors use the system as they interact with usecase. All the actors & usecases of a system makeup usecase model. A usecase diagram describes part of the usecase model and shows a set of usecases and actors with an association blw each interaction pair of actor and usecase.

Ex:- A usecase model of the atm system.

the bank customer, actor uses an ATM system to withdraw and deposit money from an account and to transfer money between accounts. this is represented by the 3 usecases, showed in the below diagram that have associations to the actor to indicate that they interact.

-ATM system



* Actors are the environment of the system:-

All actors need not represent humans. Actors can be other systems or external hardware that will interact with system. Each actor takes a coherent set of roles when it interacts with the system. A physical user may ~~interact~~ act as one or several actors playing the roles of those actors as they interact with the system. Several individual users may act as different occurences of ~~and~~ one and same actor. For Example There may be thousands of people who act as the bank customer actor. Actors communicate with the system by sending messages to and reciving messages from the system as it performs use cases. As we defined what the actors do and what the usecase do we make a clear seperation b/w responcibilities of actors and those of the system.

* Usecases specify what the system can perform:-

-A usecase specifies a sequence of actions, including variance that system can perform and that yields an observable result of value to a particular actor. We find the usecases by looking at how the users need to use the system to do their work. Each such way of using the system that adds value to the user is a candidate usecase. These candidates will then we elaborated on changed, divided into small usecases or integrated into more complete usecases.

-The usecase model is almost finished when it captures all functional requirements correctly in a way that the customers, users and developers can understand.

The sequence of actions performed by a usecase during operation is a specific path to the usecase.

Ex:- the withdraw money usecase

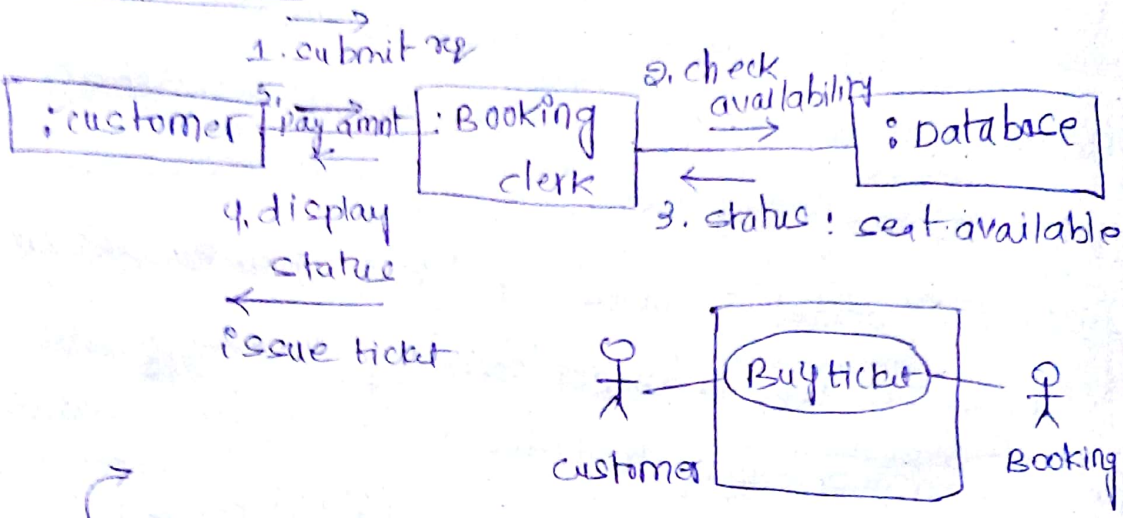
the sequence of actions for a path through this usecase is: 1. The bank customer identifies himself/herself. 2. The bank customer chooses from which account to withdraw. 3. The system diverts amount from the account and dispense the money.

→Usecases are also used as placeholders for non-functional requirements such as performance, availability, accuracy and security requirements that are specific to a

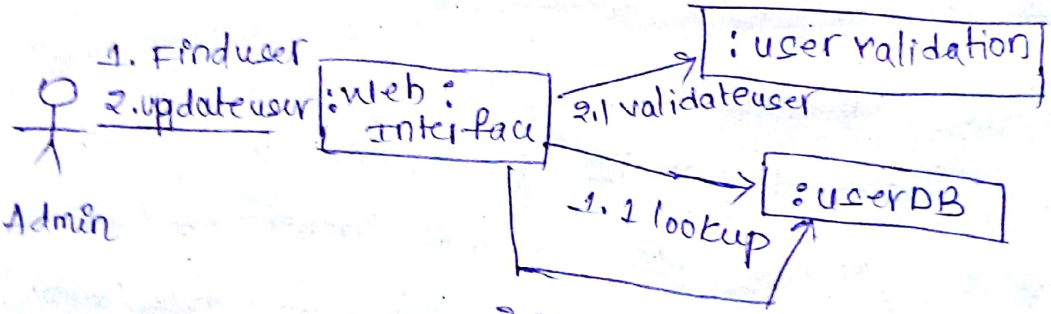
usecase. For Example the following requirements can be attached to the withdraw money usecase:

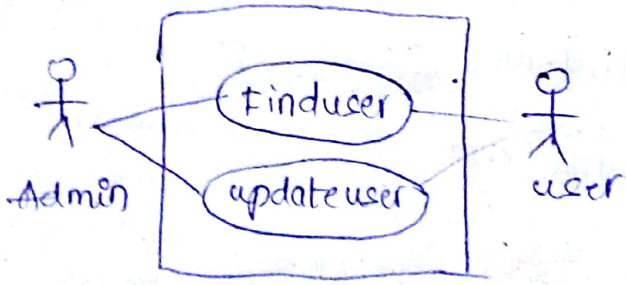
The response time for a bank customer measured from selecting the money to withdraw to the delivery of the bill should be less than 30 sec in 95% of all cases.

Date: 29/Feb/19



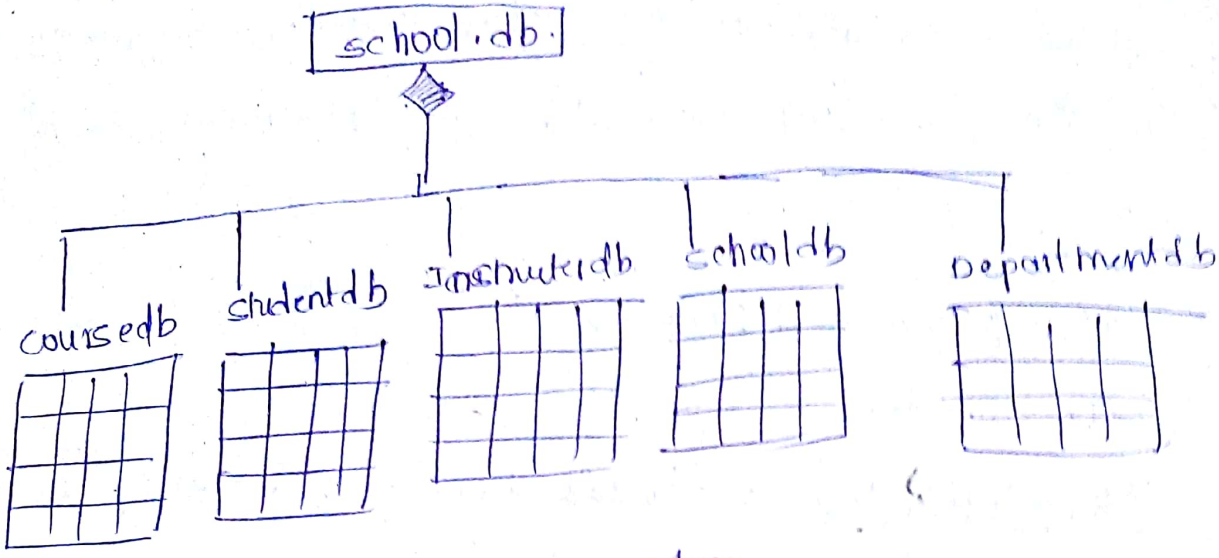
Eg:- collaboration diagram for Railway reservation



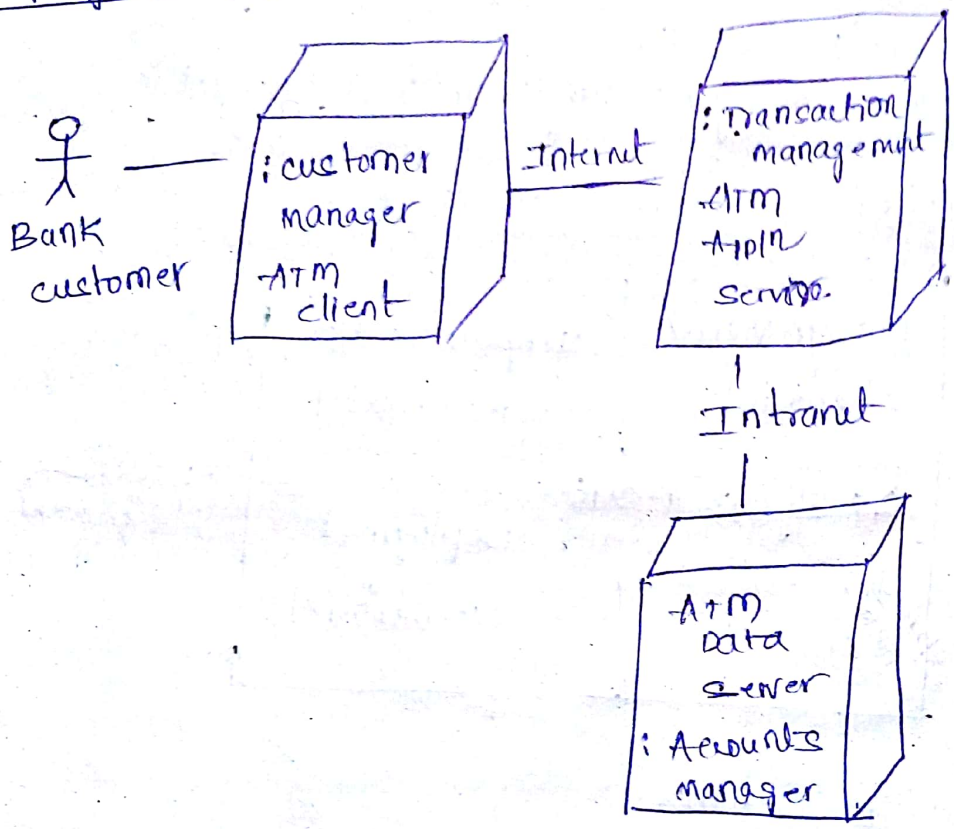


Eg:- Nesting messages in collaboration diagram

Dat Component Diagram :- to show system database



Deployment diagram: for ATM system



Date:- 26/02/19

Unified process Model :- / Rational unified process

Model → representation of a system

process → set of activities needed to transform users requirements into s/w system / product

S/w development process

→ It uses component-based s/w development process

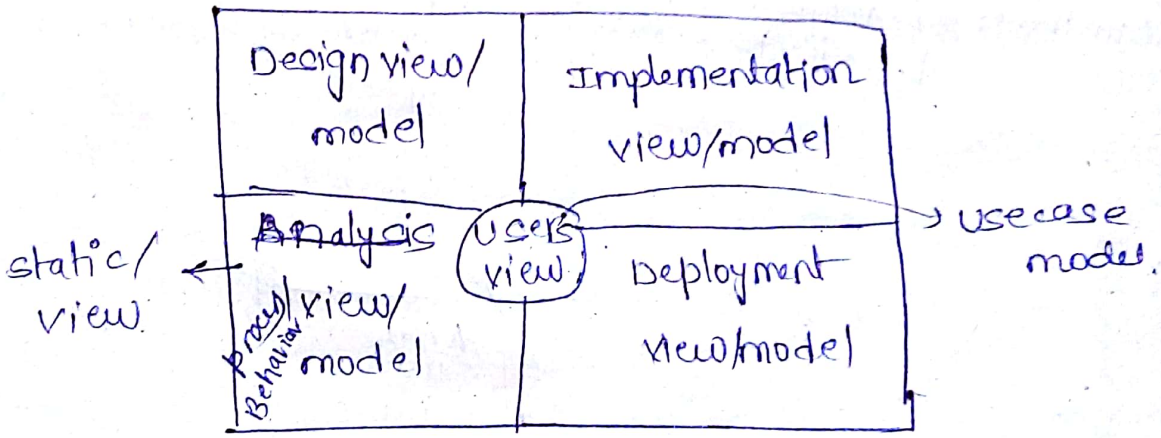
→ It also uses UML to build representation/model of the s/w system.

→ UPM involves 4 phases

(1) Inception
(2) elaboration → Architecture view of the system
(3) construction
(4) transition

(1) Inception :- Rough estimation of the model is done. (usecase model)

(2) elaboration :- Architecture view



(3) construction :- coding

complete s/w will be installed on the machine.

(4) Transition :- product will be released. (β-release)

It involves training the users and correcting errors.

Note

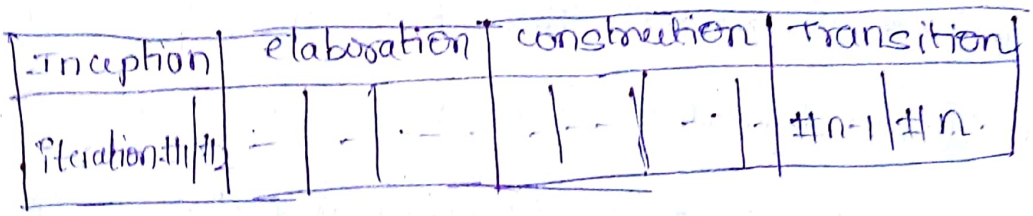
UPM :- structure/ characteristics

(1) It is a Nutshell/ Framework

(2) It is a usecase driven process

(3) It is a Architric-centre

(4) It is incremental & iterative in nature.



→ s/w development process is the set of activities needed to transform the user's requirements into a s/w system.

→ However, the unified process is more than a single process, It is a generic process framework that can be specialized for a very large classes of s/w systems, for different application areas, different types of organizations, different competence levels and different project sizes.

→ Unified process is component-based, which means that the s/w system we built is made up of s/w components interconnected via well defined interfaces.

→ It uses UML for preparing all blueprints of the s/w system

→ However, the real distinguishing aspects of the unified process are captured in 3 keywords.

(1) usecase driven

(2) Architecture-centric

(3) iterative and incremental

→ This is what makes the unified process unique.

(4) Unified process is a usecase driven

usecase driven means that the development process follows. It proceeds through a series of workflows that derive from the usecases.

→ usecases are specified, designed and at the end usecases are the source, from which the testers conduct the usecase test cases.

* Unified process is Architecture-centric

A sw Architecture contains the most significant static and dynamic aspects of the system. The architecture grows out of the needs of the enterprise as send by users and other stackholders and as reflected in the usecases.

→ Architecture is a view of the whole design with the important characteristics made more visible by leaving details aside.

→ the Architecture that must be design so, as to allow the system to evolve not only through sw it's initial development but threw future generation.

→ In simplified terms the Architec contains/does the following

(i) creates a rough outlines of the architecture, starting with the part of architecture that is not specific

to the usecases (En- platform). Although it is a part of architecture is independent usecase independent the Architec must have a general understanding of the usecases prior to the creation of Architectural outline.

(ii) Archiitec works with a subset of the identified usecases, the once that represents the key functions of the system under development. Each selected usecase is specified in detail and realized in terms of sub-systems.

(iii) As the usecases are specified and they mature, more of the architecture is discovered. This in turn leads to the maturation of more usecases.

Interaction diagrams

- Sequence
- collaboration

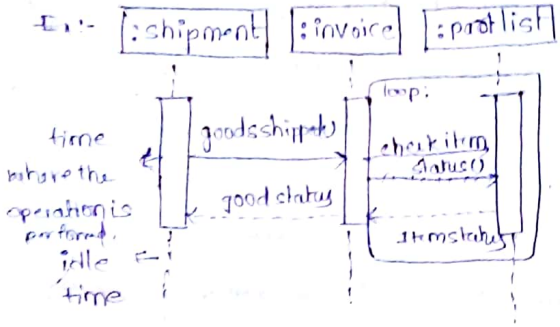
⇒ Interaction shows an act consisting of a set of objects and their relationship, including the messages that may be dispatched among them.

→ Interaction diagrams addresses the dynamic view of the system.

(1) sequence diagram

It emphasizes the time-ordering the messages.

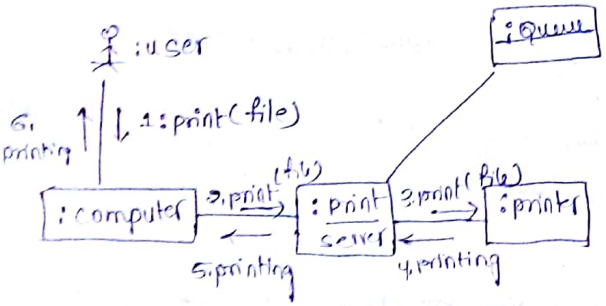
Ex:-



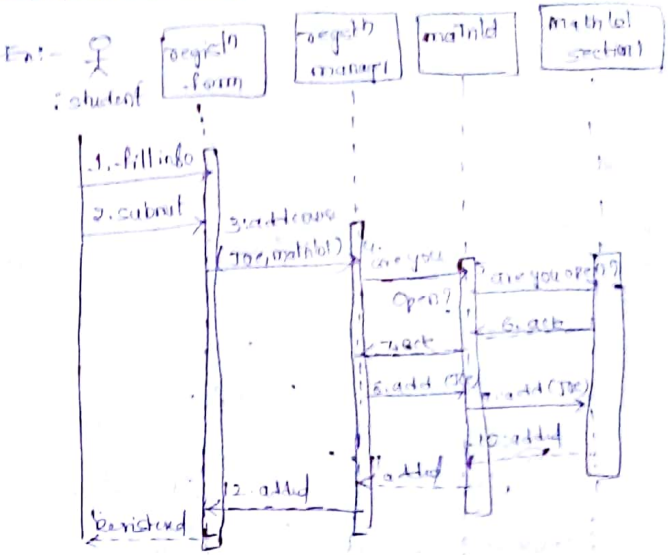
(2) collaboration diagram

Emphasizes on structural organization of the objects that send and receive messages.

Note:- sequence & collaboration diagrams are isomorphic, means, that you can take one and transfer it onto other.

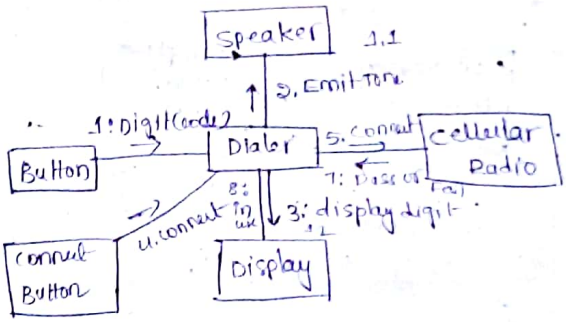


Ex:-



Sequence diagram

Ex:- Telephone system collaboration diagram



Date:- 27/Feb/19

process models

1) Requirement Engineering process → Use case model

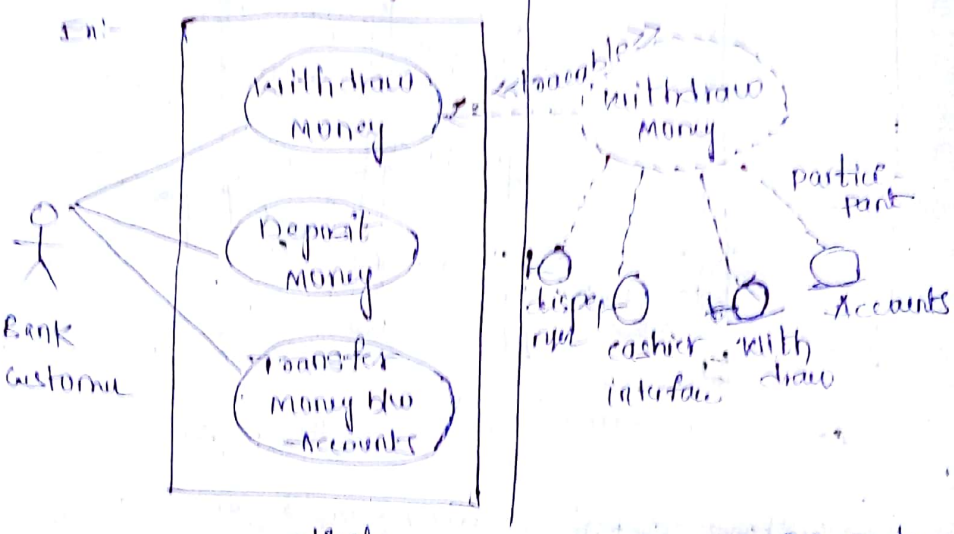
(2) Analysis process → Analysis model

3) Design process → Design model

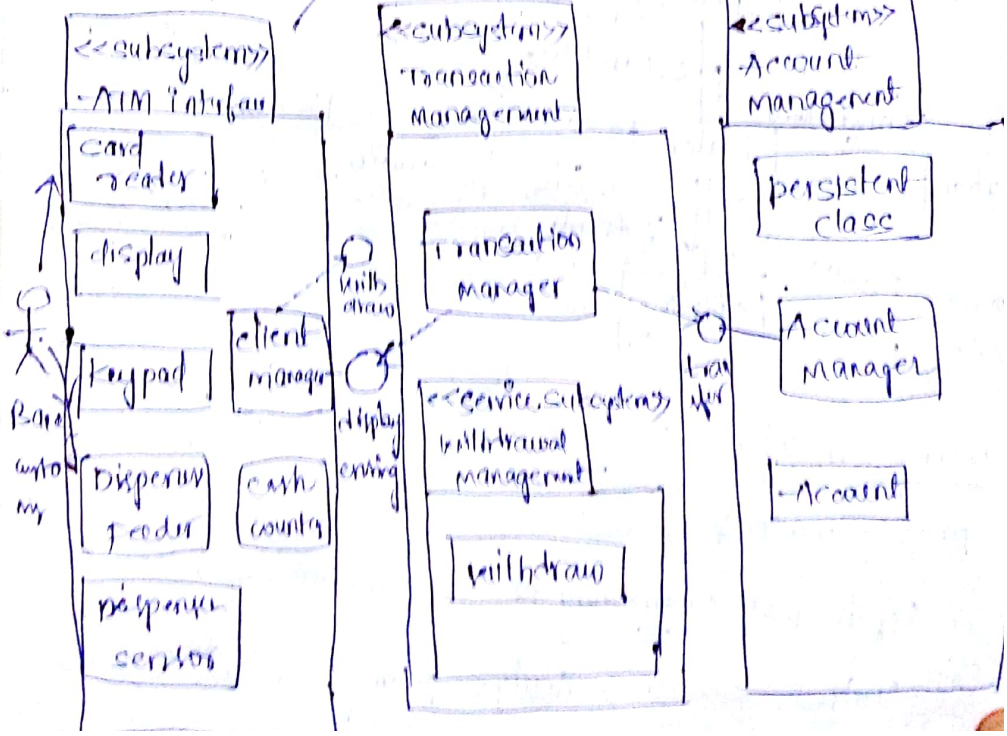
4) Implementation process → implementation model

5) Testing process → test model

* Use-case model — ATM system — Analysis model



Design model — client machine — ATM ATM server — ATM DB system server



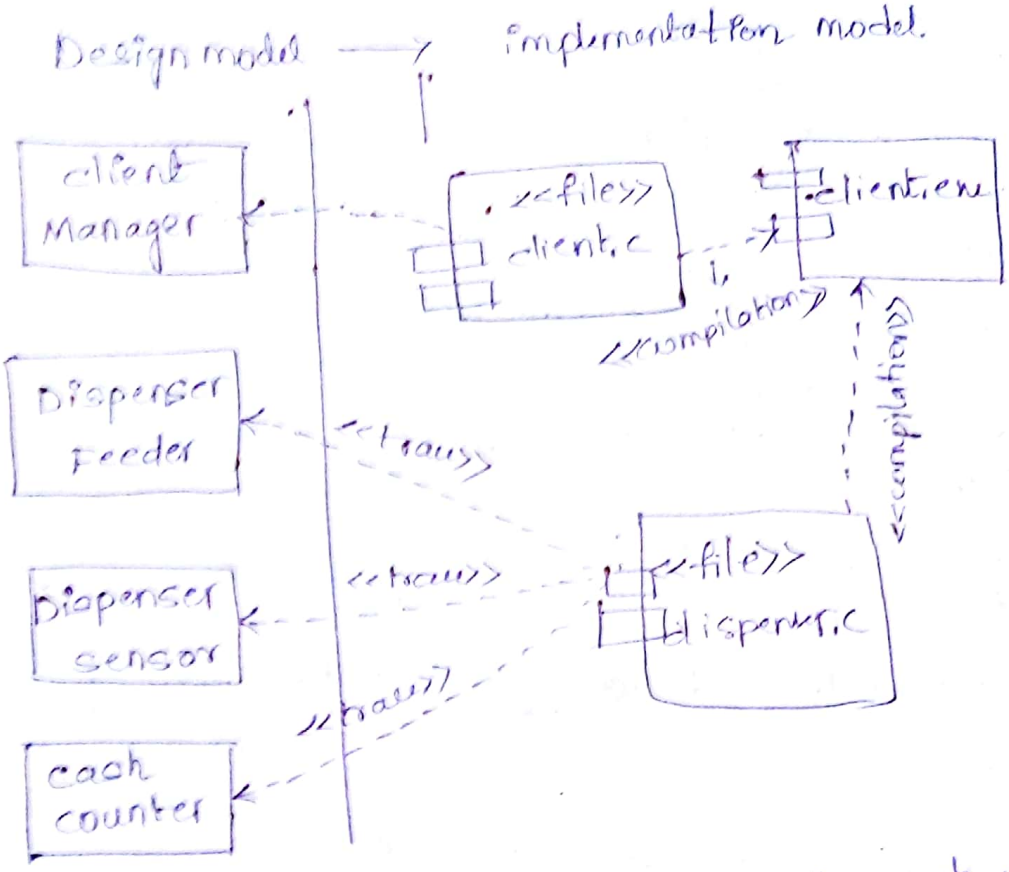
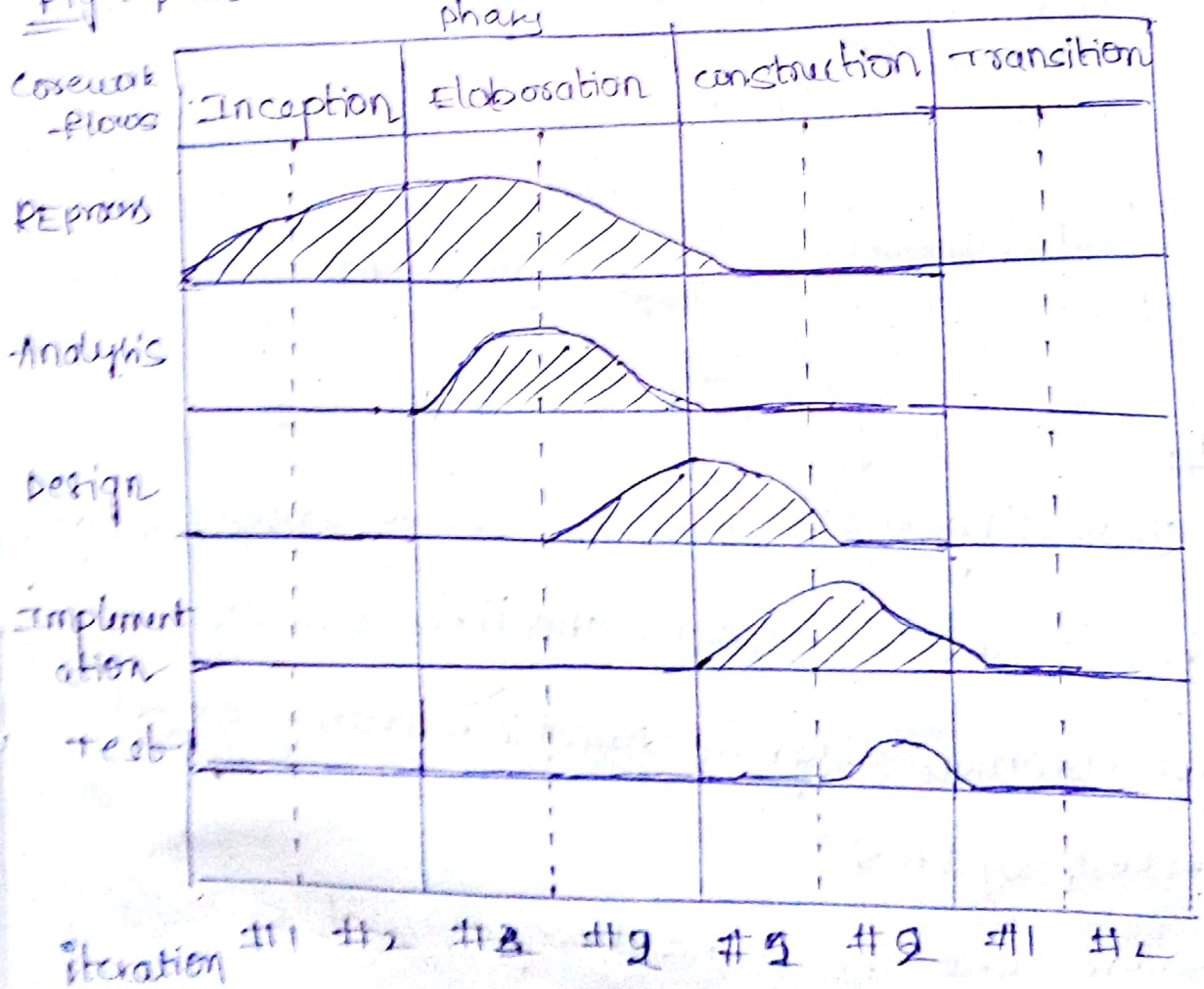


Fig:- phases of unified & work-flows in each process



Date:- 1-march-19 UPM

5 work-flows :- 4 4 phases

① Requirements
⑤ Analysis
③ Design
④ Implementation
⑥ Testing

① Inception
② elaboration
③ construction
④ transition

Date:- 5th march 19

* Whole Architecture of the system will be given by 5 views

(1) User view → usecase model
(2) Design view → Design model & Analysis model
(3) process view / Behavioural view
(4) Implementation view → component diagram
(5) Deployment view → Deployment model

UML :- things + relationships ⇒ diagrams

(1) class diagrams :- It shows set of classes, interfaces and collaborations and their relationships

→ most common diagram found in modeling object oriented systems

→ It addresses the static design view of system.

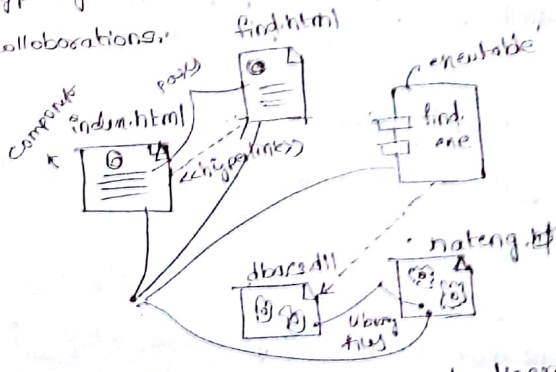
Ex:- Shape

(2) component diagram

→ A component diagram shows the organization and dependencies among a set of components.

→ component diagram addresses the static implementation view of a system.

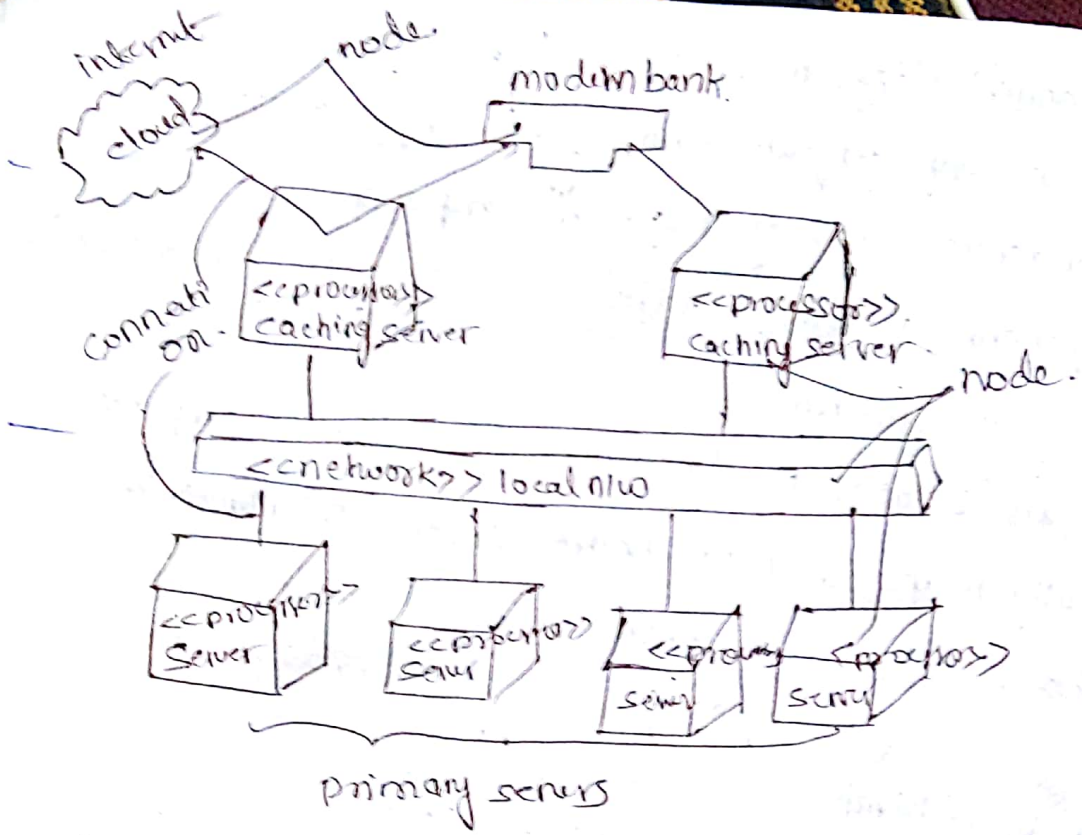
→ they are related to class diagram in that a component typically maps to one or more classes, interfaces or collaborations.



(3) Deployment diagram :- A deployment diagram shows the configuration of run-time processing nodes and the components that live on them.

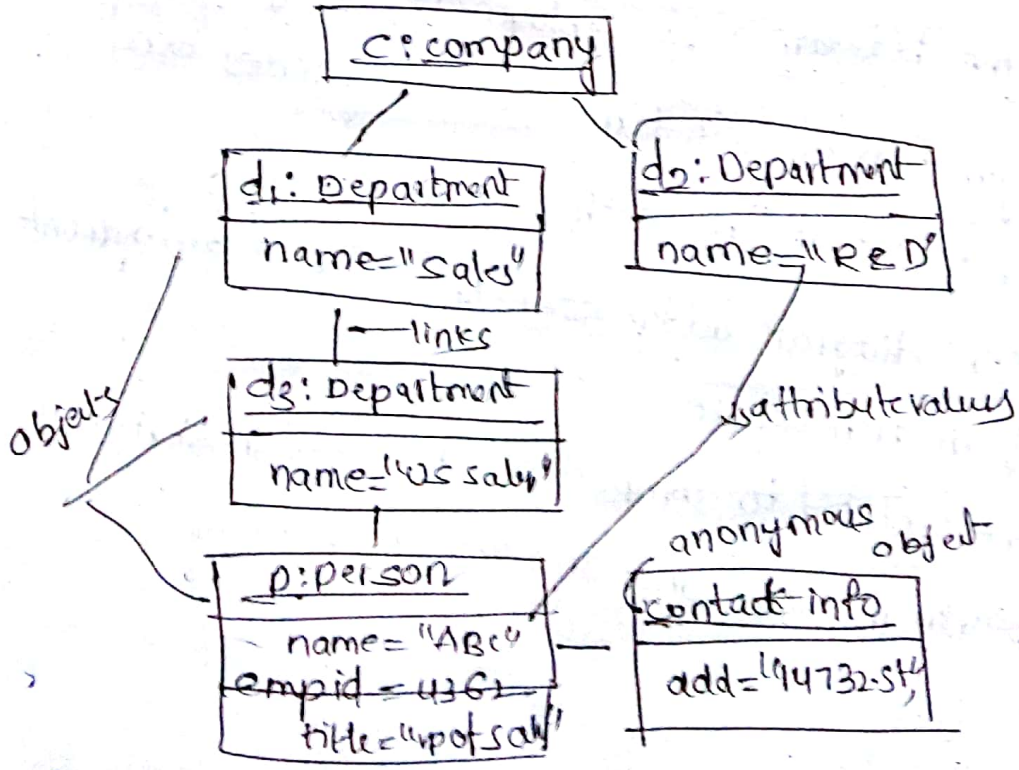
→ Deployment diagram addresses the static deployment view of an architecture.

→ they are related to component diagram in that a node typically includes one or more components.



Object diagram :- It shows a set of objects and their relationships.

→ Object dig represent static snapshots of instances of things found in class dig.



→ To verify completeness and accuracy of class

diagrams object diagrams are used.

object names → rectangle (underlined and colored)

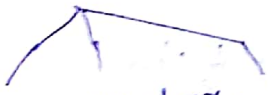
object attributes 

Links → 

Date:- 11th march 2019

Formal specification

Formal methods :- collection of techniques used to prepare a SRS document.



Rules, syntax, semantics

$\wedge, \vee$ → proposotional conjunctions

(a) Analysis model :- 2 types

(1) structured analysis model → functions/modules

(2) object oriented analysis model

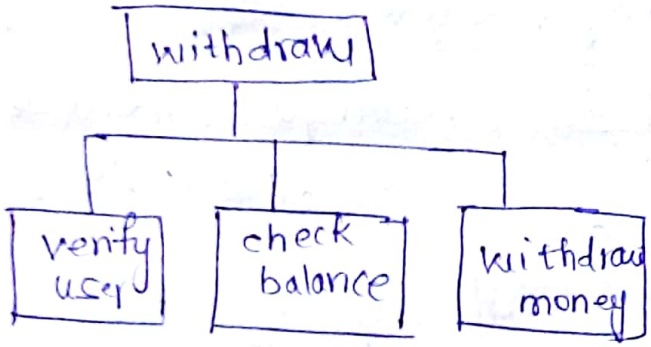
structured analysis → DFD (Data flow diagrams)

↳ It is an activity which transforms SRS document into a graphical model called DFD. contain

↳ It is a "top-down" decomposition which splits the system functionality into subfunctions.

Ex:- withdraw in ATM

↳ function



Key elements/Notations used in DFD:-

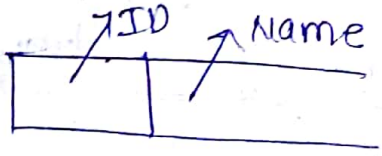
(1) Data source/ Data sink } external agents

Ex- customer who interacts with system

External Agent

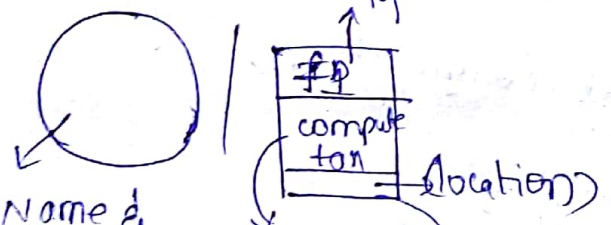
Name & ID

2) Data store.



con → name of data store

(3) process



1) Name &
2) ID
3) location

(4) Label

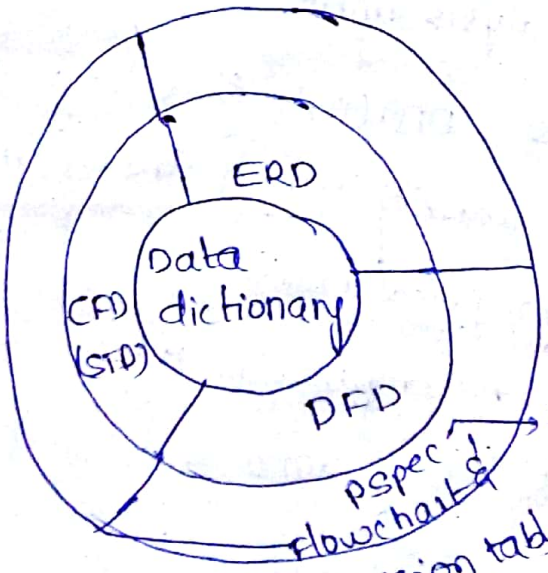
Name → con ← name

shows direction of data flow b/w modules

5) Output

Data dictionary:-

contains system description

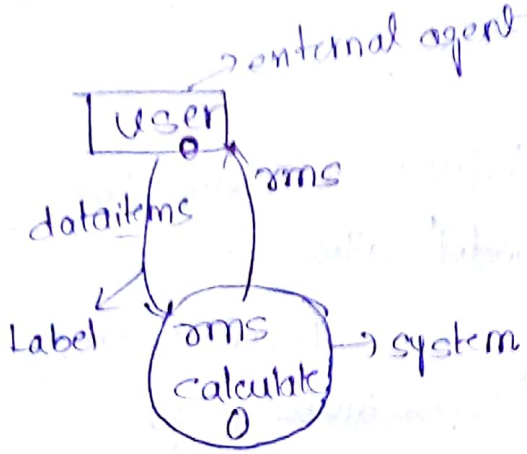


Constructing DFD's

Eg:- Root mean square of a number

$a, b, c \rightarrow rms = \sqrt{\frac{a^2+b^2+c^2}{3}}$

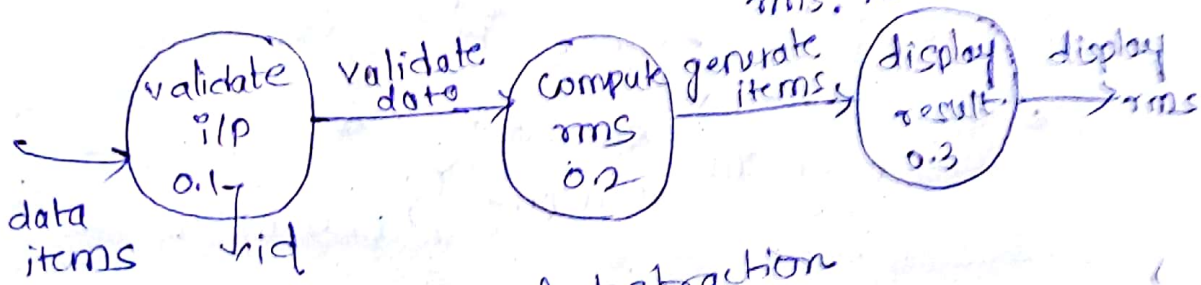
Level-0 DFD (context diagram)



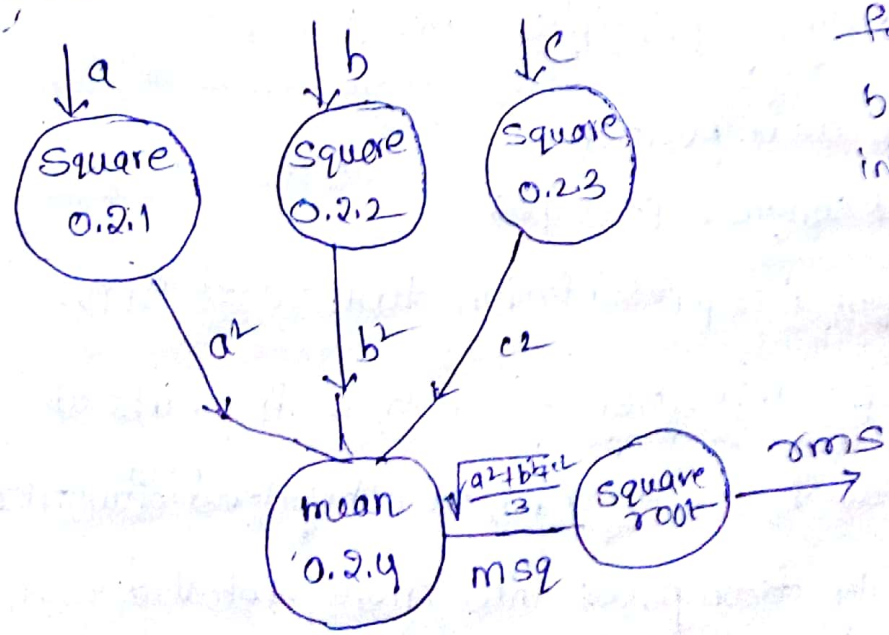
Data Dictionary

a : integer
b : "
c : "
etc. info.
a^2 : "
b^2 : "
c^2 : "
msq : float
rms : float

Level 1 - DFD:



Level2 DFD - High level abstraction



* 1:5 → A single function can be divided into 5 sub functions.

→ also known as bubble chart diagram.

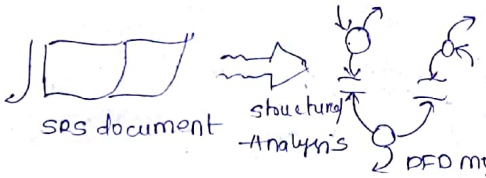
Date:- 18th March 2019

Data Dictionary :- Every bubble description is specified in this data dictionary.

→ It contains 'meta data' (data about data)

Structured Analysis Model :

Structured Analysis:- This activity transforms the SRS document into a graphical model called the DFD model. During structured Analysis functional decomposition (Modularity) of the system is achieved.



→ The structured analysis technique is based on the following underlying principles.

1) Top-Down Decomposition.
2) Divide and conquer principle
3) User's graphical representation tool. called DFD.

1) Top-Down Decomposition :- In this it starts at high level view of the system, each high level function is successfully decomposed into more detailed functions. Eg:- → Function - create_new_Library member()

→ The high level function may be decomposed into following subfunctions.

(a) Assign_membership_number
(b) create_member_record.
(c) print_bill

→ Each of this subfunction may be split into more detailed subfunctions and so on.

→ The system is centralised and shared among different functions. Here, the system state can be defined as the values of certain data items that determine the response of the system to a user action or external event.

Dataflow diagram / Bubble chart :-

It is concerned about external specification of the system only. Dataflow diagram is also known as Bubble chart it is a tool to depict the flow of data thru processes / modules of the system.

→ It is a simple graphical model that can be used to represent a system interms of the input data to the system, various processing carried out on these data and the o/p generated by the system.

→ It is a directed graph which is a collection of vertices and edges and also includes cycles.

Vertices → These are the nodes that specify the processing activities associated with the system.

Edges → An edge specifies the transition of data b/w modules. These are used to inorder to document requirements analysis

→ DFD is related to external specification only that model is it doesn't provide any procedural aspects or control information

Difference b/w Flowchart and DFD

DFD	Flowchart
→ It is used to represent the external specification of the system but not any procedural aspects and it is used to document the requirements analysis.	→ It is used to represent the process specification of the system and for every DFD we need to draw a flowchart to represent the process specification
→ Decision making box is not included in DFD.	→ Decision making box is included in the flow chart.

Data Dictionary:- A data dictionary lists all data items that appear in a DFD model.

→ The data items listed include all data flows and the contents of all data stores appearing on all the DFD's in a DFD model.

→ It contains system functional descriptions or external specifications that is simply it contains system description that is whatever we gathered as requirements from the customer are placed in it.

→ "Composite data items" are defined in the data dictionary interms of primitive data items using the following data definition operators.

(1) + → a+b → denotes composition of 2 data items. Ex:- (a+b) → represents data 'a' and data 'b'.

(2) [,] → represents selection that is any one of the data items listed inside the square bracket can occur. Ex:- [a, b] → represents either a or b occurs.

(3) ( ) → the content inside the bracket represent optional data item which may or may not appear. Ex:- a+(b) → Represents either 'a' or a+b occurs.

(4) { } → Represents iterative data definition Ex:- {Name}5 → it represents 5 name data.

(5) = → Represents equivalence Ex:- a=b+c → means that 'a' is a composite data items comprising of both 'b' and 'c'.

Comments:- /* */ → Anything appearing /* */ is considered as comment.

Developing / construction of DFD model of a system.

→ A DFD model of a system graphically represents how each i/p data item is transformed to it's corresponding o/p threw a hierachy of DFD's.

→ DFD model of a system consists of many of DFD diagrams and a single data dictionary.

→ The toplevel of DFD is called Level 0 DFD or the content diagram. This level is easy to draw and understand.

→ At Each successive lower level ~~and~~ DFD's, more and more details are gradually introduced.

→ To develop a higher level DFD model processes are decomposed into their subprocesses and the dataflow among these subprocesses are identified.

→ For any given diagram, content diagram is ~~used~~ drawn using single process diagram

→ only at level 0 and final level the external agents are represented.

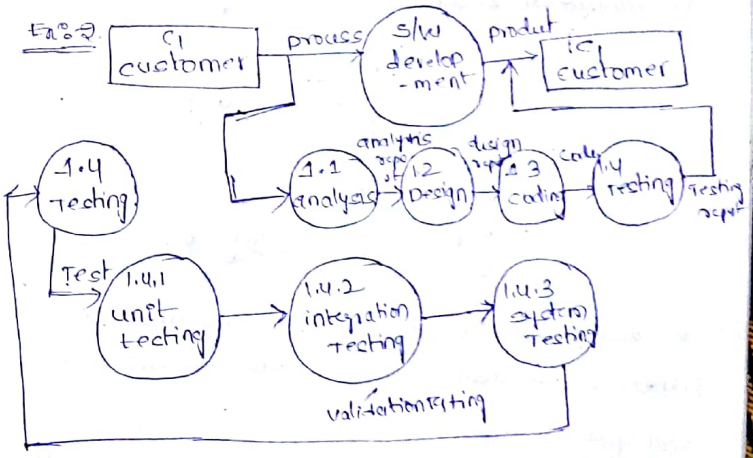
Ex:- Level 0 : ucer0

→ Identify candidate key and refine (decompose) them into smaller activities. It should be appropriately approximately equal to 1:5.

→ Individual bubble should be decomposed or refined at a given time.

* → Names or labels are compulsory for every notation in a DFD.

Ex:-



Guidelines for constructing DFD's :-

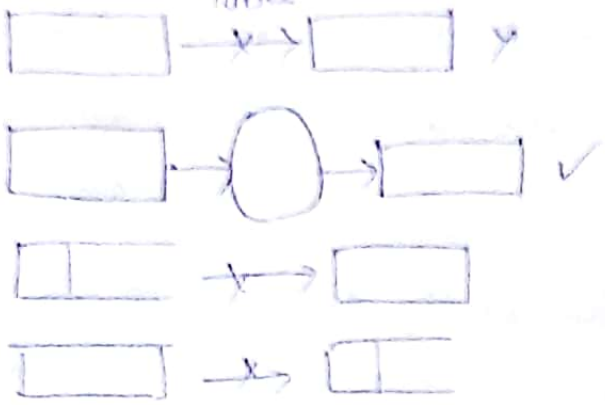
→ All the notations in DFD should have proper labels

→ DFD doesn't provide control information.

→ It is not required to show starting and ending of every process.

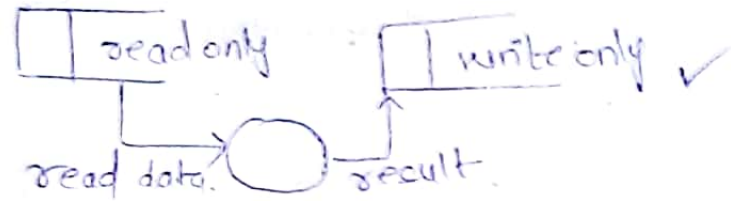
→ Two external agents cannot be connected directly in a DFD always there should be a process which connects them.

Ex:- label



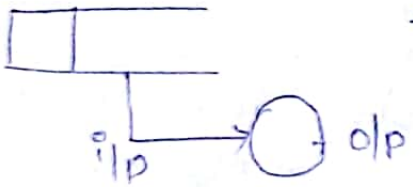
→ There must be a No notation in the DFD should be isolated.

→ Be aware of readonly / write only data stores.

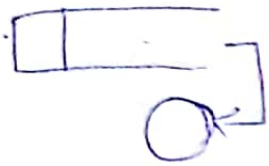


→ Be aware of the process that takes input but doesn't produce o/p and also the process that doesn't take i/p and produce o/p.

case1:-



case2:-



?