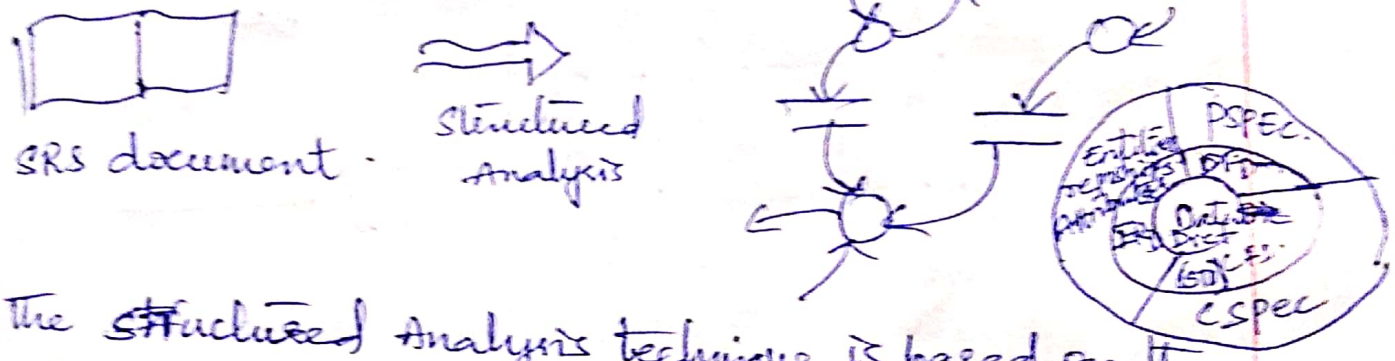
Analysis model: Structured Analysis model :-

Structured-Analysis :-

- The structured Analysis activity transforms the SRS document into a graphical model called the DFD model.
- During structured analysis, functional decomposition of the system is achieved. ↳ modularity.



- The structured Analysis technique is based on the following underlying principles:— (newers property)
  $c(a+b) > c(a) + c(b)$
  $E(a+b) > E(a) + E(b)$
  * TOP-down decomposition.
  * divide & conquer principle
  * It uses graphical representation i.e. DFDs

Top-down decomposition :-

- In this, it starts at high-level view of the system, each high-level fn is successfully refined (decomposed) into more detailed fns.

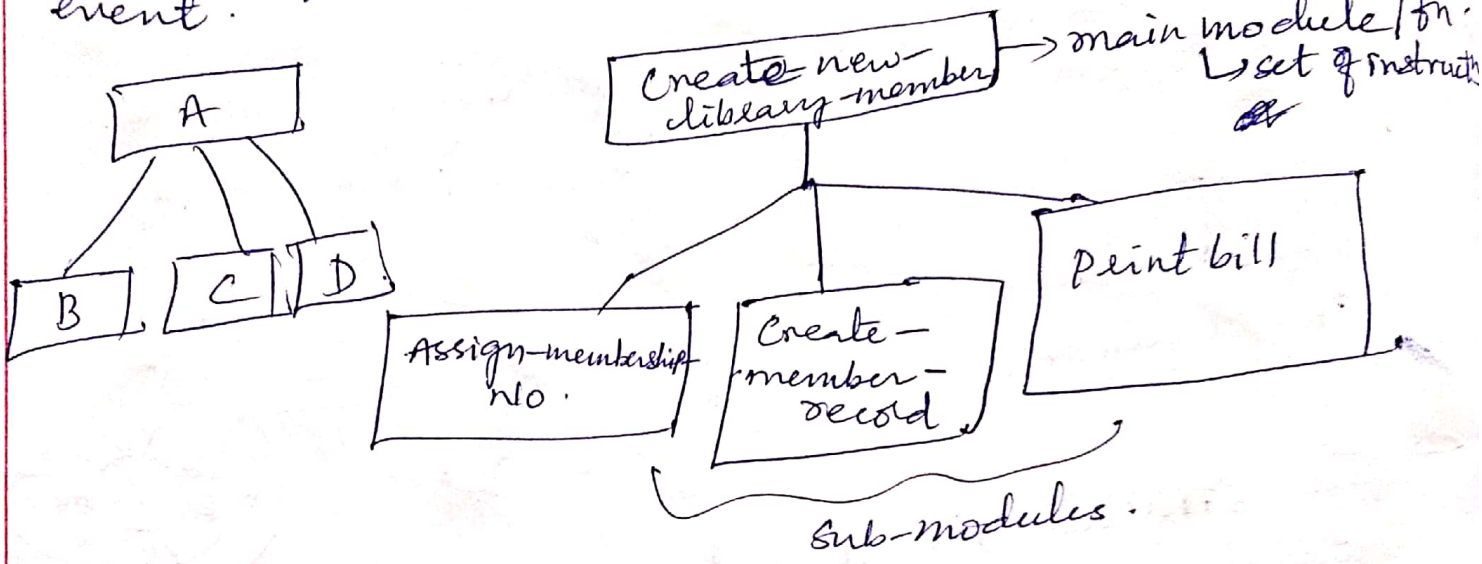
Ex: A function create-new-library member:
create-new-library( ).

The highlevel fn may be refined into the following sub fns.

(a) Assign-membership-number( )
(b) create-member-record( )
(c) print bill( ).

- Each of these sub fns may be split into more detailed sub fns & so on.

→ The system state is centralized & shared among diff 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.



Data flow Diagram (DFD's) :- (bubble chart).
It is concerned about external spec of the system only.

Data flow diagram is also known as the "bubble chart".
It is a tool to depict the flow of data between the processes (functions) modules of the systems.
It is a simple graphical model that can be used to represent a system in terms of the i/p data to the systems, various processing carried on those data, & the o/p data generated by the system.
It is a directed graph which is a collection of vertices & edges and may also include cycles [ie, cycles are possible if any].
Vertices → are the nodes that specify the processing activities associated with the system.
Edges → an edge specifies the transmission of data b/n modules.
They are used inorder to document requirements analysis.
DFD is related to external spec i.e it doesn't provide any procedural aspects.

Difference b/n Flow chart & DFD :-

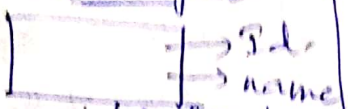
DFD	Flow Chart.
① DFD is used to represent the external specification of the systems but not any procedural aspects. i.e it is used to document the requirements analysis.	① It is used to represent the process spec of the systems. & for every DFD, we need to draw a Flow chart to represent the process spec.
② Decision making box is not included in the DFD.	② Decision making box is included in the DFD.

Key Elements of DFD:

→ It includes Data source, Data sink & Data store.

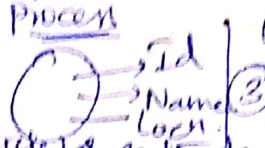
Symbols / Notations used for constructing DFD's:

① External Agent:

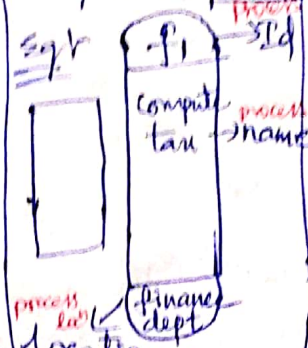


external entity (or) external Agent. can be any Data source, Data sink.

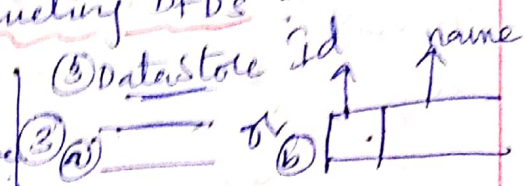
② Process



Eg:



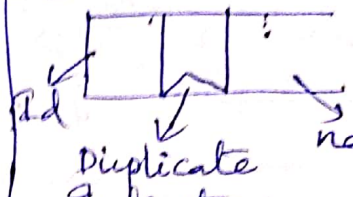
③ Datastore Id name.



Datastore.

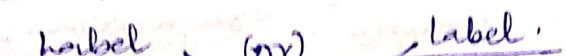
Eg: File can be an example for Datastore.

④



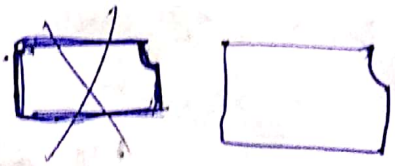
* used to indicate this datastore is a duplicate store eg: duplicate file.

④ Data flow: It is represented by the following symbol with a label:



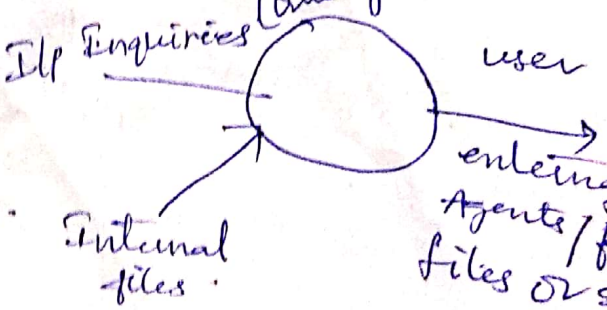
Note: Labels are important in DFD, otherwise they (DFD's) are rejected.

⑤ O/P: symbol is



* Process Information: includes:

No of I/P.
No of O/P.
No of Inquiries.
No of files
No of external Interfaces.



A process can take the above process input & give o/p to the external agent (user) or Interface files or subsystem

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

The data items listed includes all data flows & the contents of all data stores appearing on all the DFDs in a DFD model. It contains system's functional descriptions or external specifications.

Simply, it contains "system descriptions". ie, whatever we gathered as requirements from the customer are placed in it.

Data definition :-

Composite data items can be defined in terms of primitive data items using the following data definition operators.

① + : denotes composition of 2 data items.
Eg: a+b represents data a & b.

② [,,,] : represents selection, i.e any one of the data items listed inside the square bracket can occur. Eg:- [a,b] represents either a or b occurs.

③ (): the content inside the bracket represent optional data which may or may not appear.
Eg:- a+(b) represents either a or a+b occurs.

④ { } : represents iterative data definition.
Eg:- {name}5 → represents 5 name data.

⑤ = : represents equivalence.
Eg:- a = b+c means that a is a composite data item comprising of both b & c.

→ /* */ → Anything appearing within /* and */ 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 is transformed to its corresponding o/p through a hierarchy of DFDs.

→ DFD model of system consists of many of DFDs & a single data dictionary.

→ The top level of DFD is called "level 0" DFD or the "Context diagram". This level is easy to draw & understand.

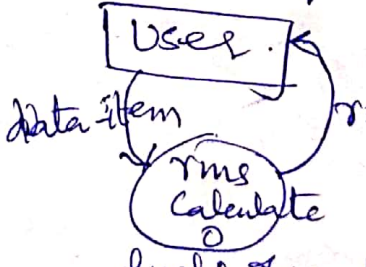
→ At each successive lower level DFDs, more & more details are gradually introduced.

→ To develop a higher-level DFD model, processes are decomposed into their subprocesses & the dataflows among these subprocesses are identified.

→ For any given diag, context diag is drawn using single process diagram.

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

Eg: to calculate ~~relations~~ p. root mean square (rms) the level 0 DFD is shown below.

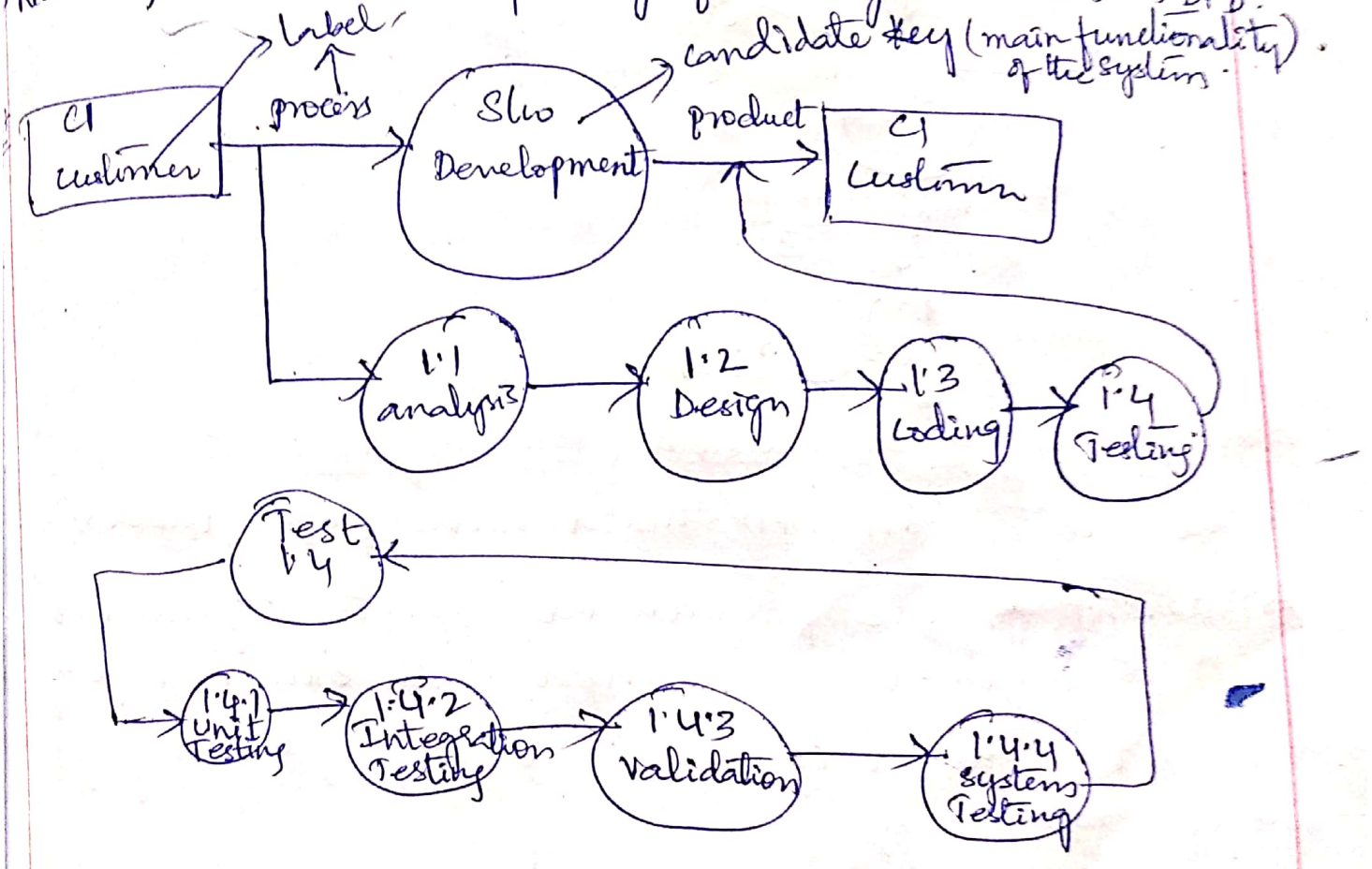


level 0 or context diagram.

identify candidate key & refine them into smaller activities. It should be approximately equal to 1:5.

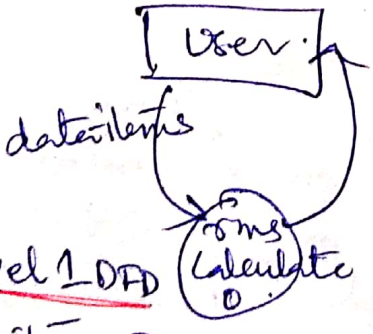
Individual bubble should be refined at a given time.

Names / Labels are compulsory for every notation in DFD.



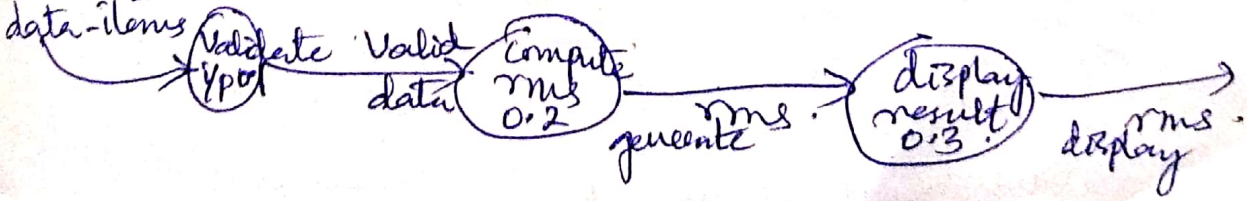
Level in calculating the RMS by the slw systems :-

① LEVEL 0 or context diagram :-



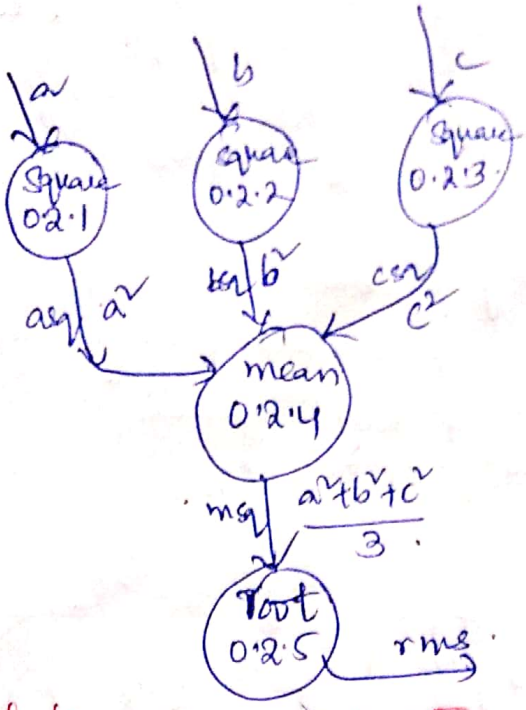
Note : only in level 0 & final level external agents like user, source & sink are represented. Here user is acting as source as well as sink.

② level 1 DFD



level 2 DFD :-

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



Guidelines for constructing / Drawing DFDs :

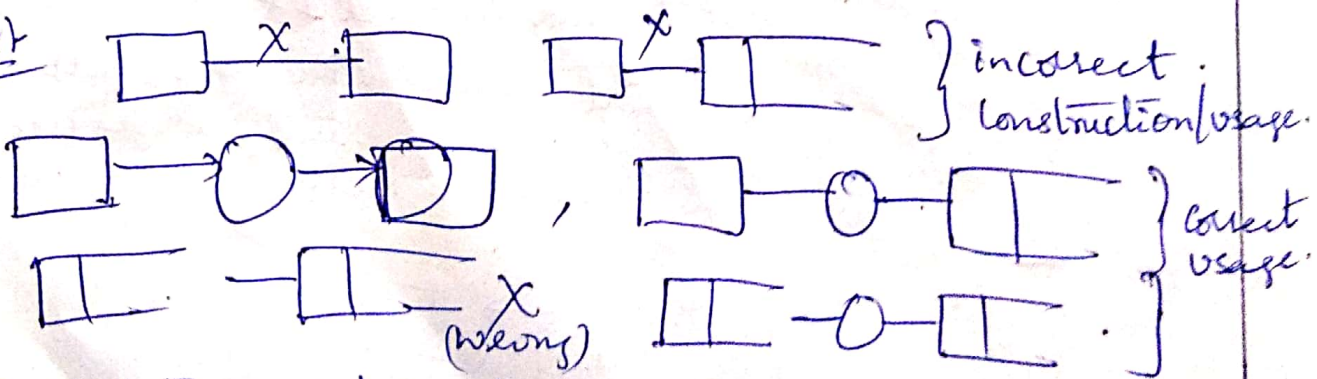
→ All the notations in DFD should have proper labels.

→ DFD doesn't provide procedural aspects i.e, internal speed rather it is concerned with external speed.

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

→ 2 external agents can't be connected directly ~~by using~~ in a DFD. Always there should/will be a process which connects them.

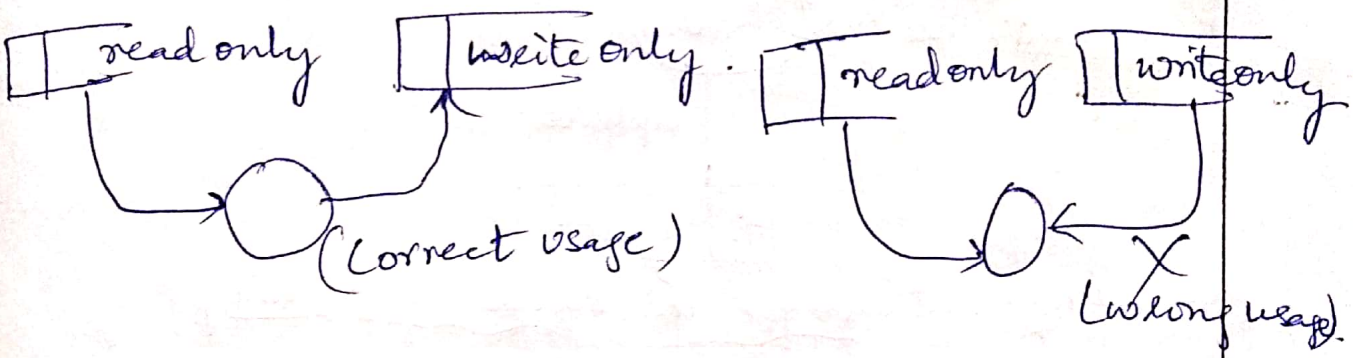
Eg:



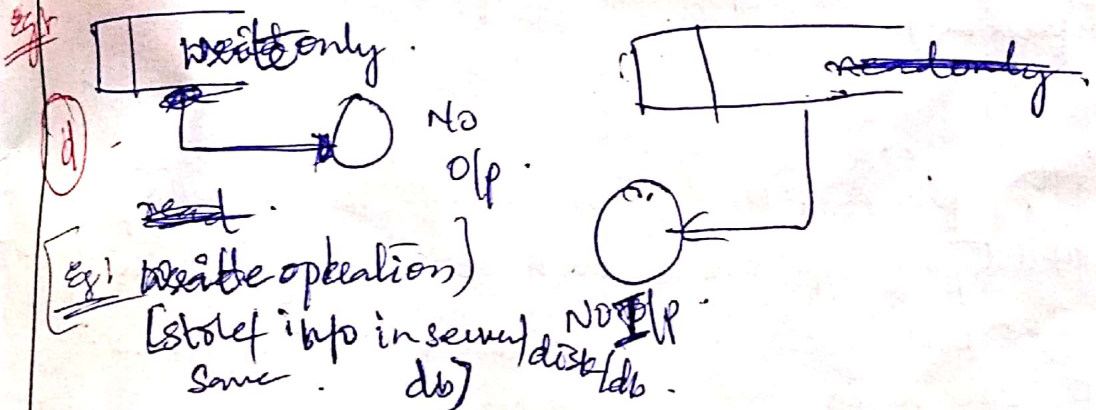
Process is compulsory to connect the external agents.

→ No, notation in the dataflow diag should be isolated.

→ Be aware of read only or write only data stores.



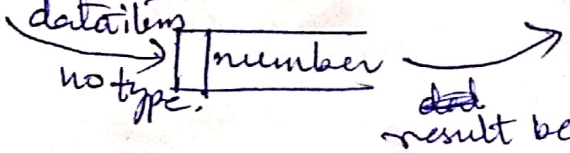
→ Beaware of the process that takes i/p but doesn't produce o/p. & also the process ~~that~~ that doesn't take i/p but produce o/p.



Eg: write operation
[stores info in secondary disk/db. Same db]
NO I/P

→ Ensure that the info leaving out of the data store should be equivalent to info available in it.

Eg:



result be number type only.

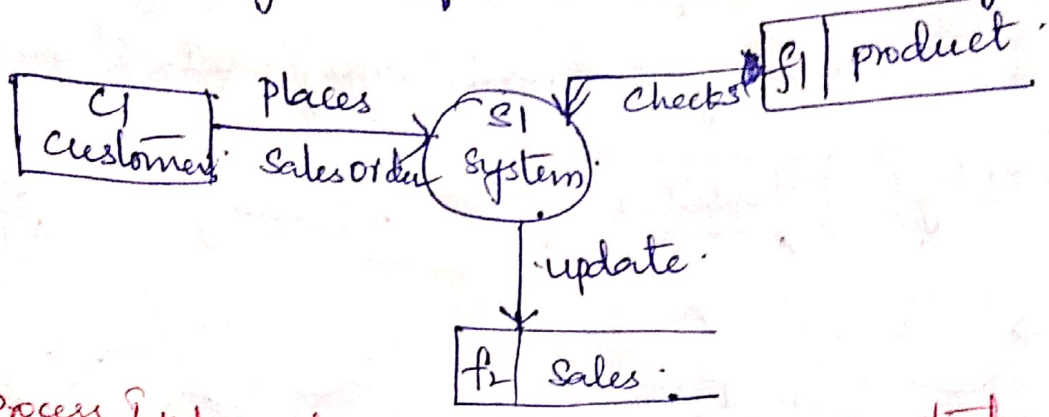
Case study :-

① Customer places product request, the system checks for the product & update the sales info.

Notations :-

Customer → External Agent
system → process
data stores → product files, sales file.

Corresponding DFD for the above case study :

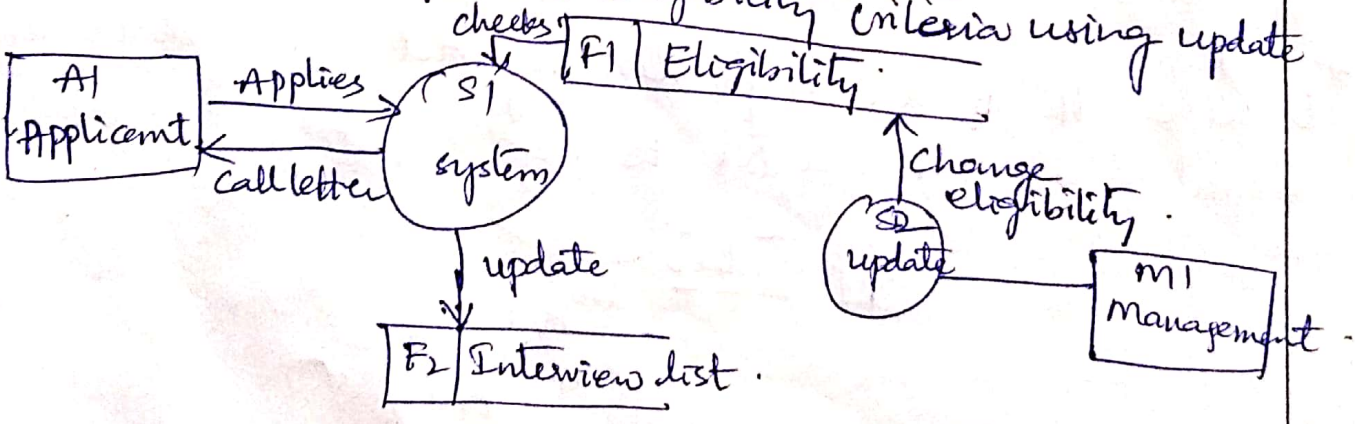


Process Information for the above case study :

No of i/p — 1
No of o/p — 1
" " Inquiries — 0
" " Files — 1
" " Interfaces — 1

User — 1
Internal file — 1
external file — 1
external interface — 1.

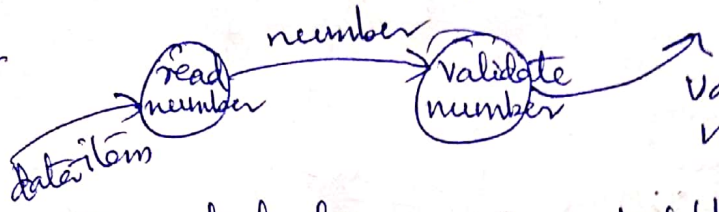
(2) Applicants applies for a job, system checks for eligibility based on eligibility, applicants will be called for interview. System maintains list of candidates called for interview. Management changes the eligibility criteria using update process.



Synchronous & asynchronous operations :-

Synchronous :- If two bubbles are directly connected by a data flow arrow, then they are synchronous. This means that they operate at the same speed.

eg :-

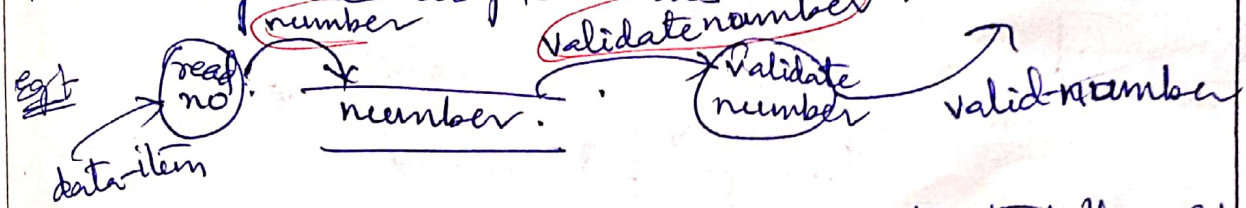


Here, the validate-number bubble can start processing only after the read-number bubble has supplied data to it. & read-number bubble has to wait until the validate-number bubble has consumed its data.

Asynchronous :-

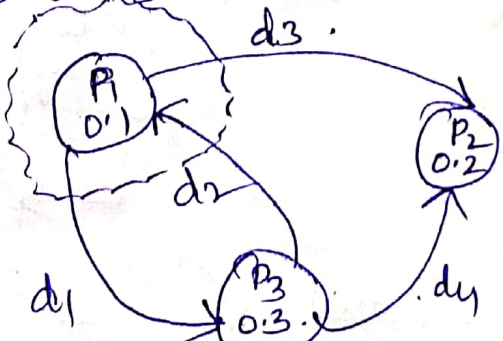
If two bubbles are connected through a data store, then the speed of operation of the bubbles are independent then they are asynchronous.

labels are compulsory.



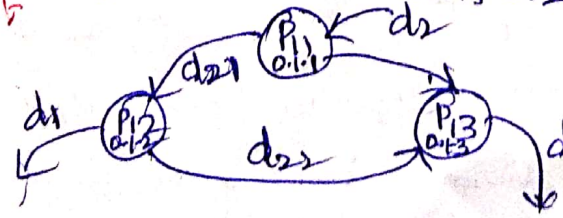
Balancing DFD's :- In the DFD model, the data that flow into or out of a bubble must match the data flow at the next level of DFD. This is known as balancing a DFD.

level 1 DFD :-



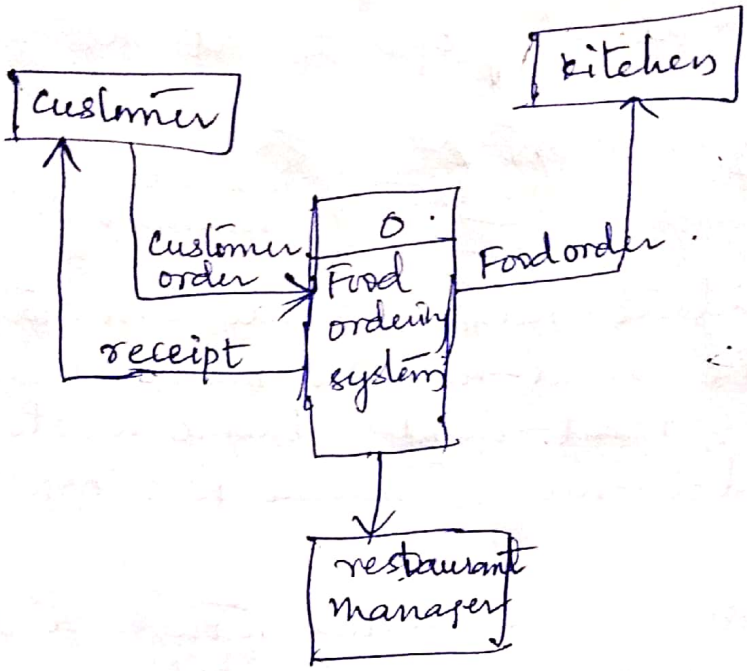
In the level 1 DFD, data items d_1 & d_3 flow out of the bubble 0.1 & the data item d_2 flows into the bubble 0.1.

level 2 :-

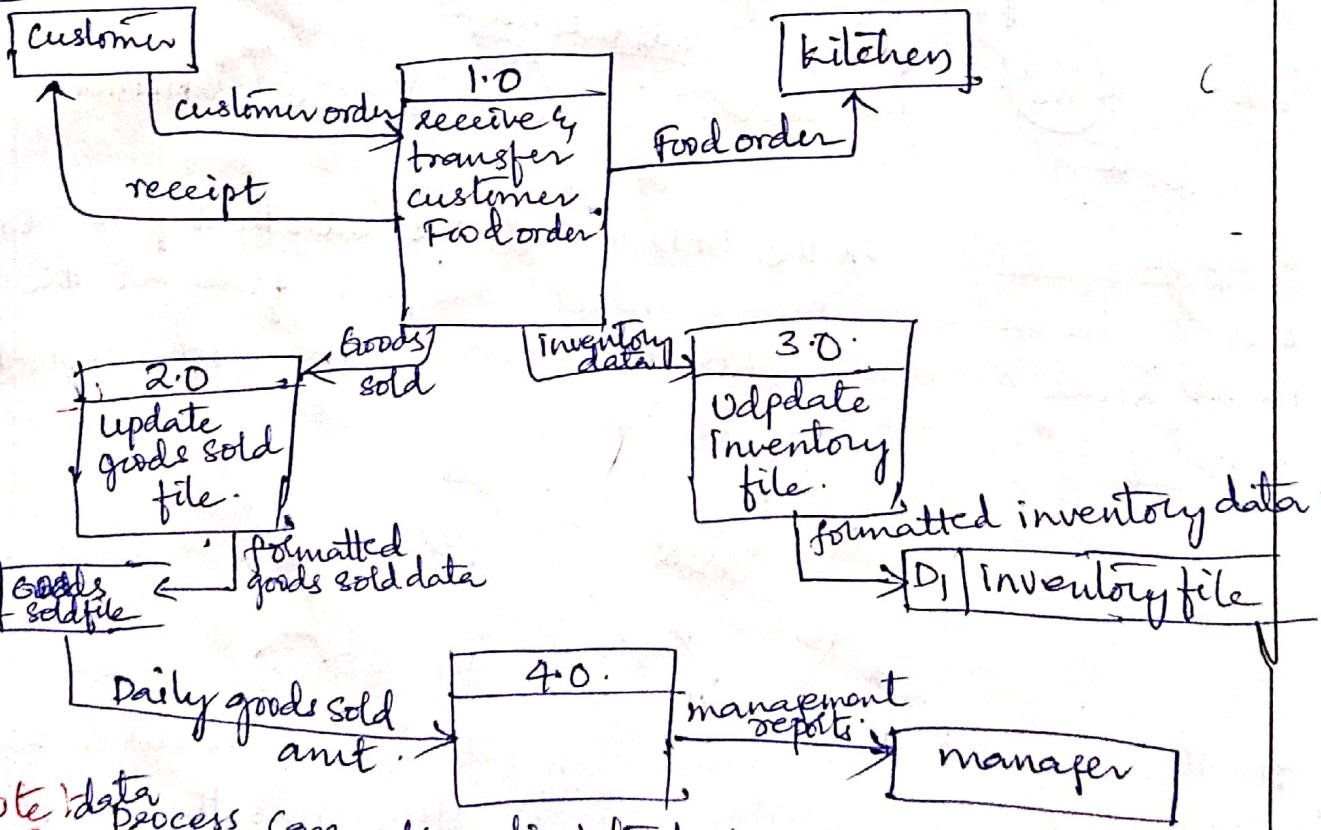


→ In this level, bubble 0.1 is decomposed into 3 DFDs. d_1 & d_3 flow out of the level 2 diagram & d_2 flows in.
∴ The decomposition is balanced.

Ex: level 0 :



level 1 :



Note: data process can also depicted by a circle or rectangle.

→ level 0 DFD of food ordering system has 1 i/p & 3 o/ps from system.
→ level 1, the system is divided into 4 subsystems before

reaching to manager. I/p & O/p number is equal, so it is a balanced DFD.

Content Diagram (level 0 DFD)

The content diagram is the most abstract level of data flow representation of a system. It represents the entire system as a single bubble. This bubble is labelled according to the main functions of the system.

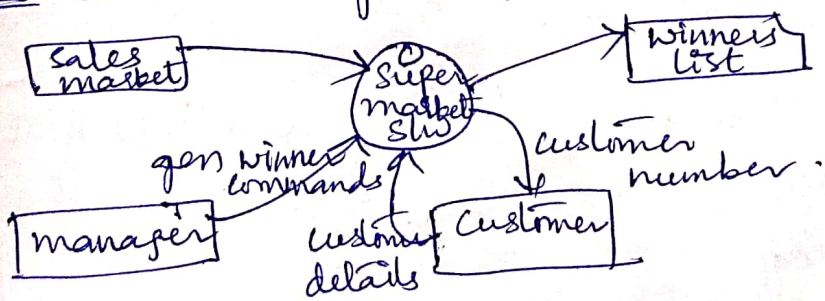
The various external entities with which the system interacts & the data flow occurring between the system & the external entities are also represented.

Content diagram is also called "level 0 DFD".

To develop the content diagram of the system, it is required to analyze the SRS document to identify the different types of users who would be using the system & the kinds of data they would ~~be~~ give as input to the system.

The bubble in the content diagram is content diagram is annotated with the name of the s/w system being developed (usually a noun). This is in contrast with the bubbles in all other levels which are annotated with the verbs.

Ex: content diagram for super market problem (Case study).



Bootstrap loader:

Bootloader is the first sw prog that runs when a computer starts. ~~are used to boot~~

→ It is responsible for loading & transferring control to the OS kernal sw (such as Hurd or Linux). The kernel in turn, initializes the rest of the OS.

→ GNU GRUB is a Multi boot boot loader.

GRUB stands for GRand Unified Bootloader.

Difference bw bootloader & a bootstrap loader

→ Bootloader is a piece of code that runs before any OS is running. Bootloaders are used to boot other OSs, usually each OS has a set of bootloaders specific for it.

Bootstrap loader is a prog that resides in the comp EPROM, ROM or other non-volatile memory that automatically executed by the processor when turning on the comp.

→ The bootstrap loader reads the hard drives boot boot sector to continue the process of loading the computer's operating system.

commonly made errors while constructing a DFD model.

Many beginners commit the mistake of drawing more than one bubble in the context diagram.

A context diagram should depict system as a single bubble.

All external entities interacting with the system should be represented only in the context diagram. The external entities shouldn't appear at other levels of DFD.

Too less or too many bubbles should not be there in a DFD. only 3-7 bubbles per diagram should be allowed.

Many beginners leave different levels of DFD unbalanced.

Representation of control information on a DFD is incorrect.