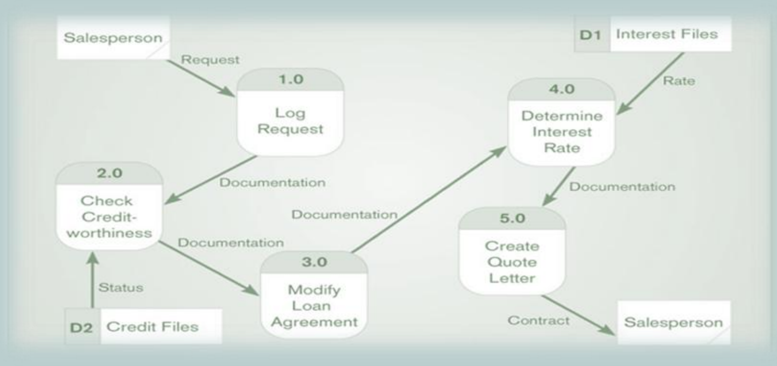
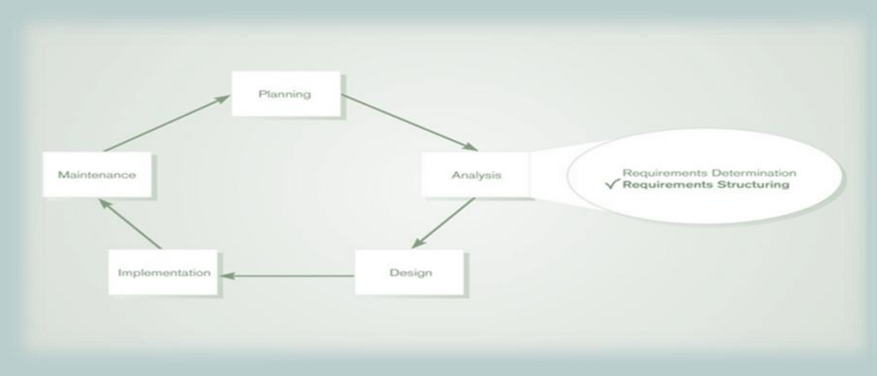
Data Flow Diagram

Structuring System Process Requirements





Overview

- Process Modeling and Data Flow Diagrams (DFDs).
- Draw DFDs of well structured process models.
- Decompose DFDs into lower-level diagrams.
- Balance high-level and low-level DFDs.
- The differences between current physical, current logical, new physical, and new logical DFDs.
- Using DFDs for analyzing information systems.

Structured Analysis vs. Structured Design

The results of structured analysis can be easily understood even by ordinary customers:

does not require computer knowledge

directly represents customer's perception of the problem

uses customer's terminology for naming different functions and data.

The results of structured analysis can be reviewed by customers:

to check whether it captures all their requirements.

Structured Analysis

Based on principles of:

Top-down decomposition approach.

Divide and conquer principle:

each function is considered individually (i.e. isolated from other functions)

decompose functions totally disregarding what happens in other functions.

Graphical representation of results using

data flow diagrams (or bubble charts).

Process Modeling

- A technique for graphically representing the processes that are used to capture, manipulate, store, and distribute data;
  - between a system and its environment,
  - among system components.
- Build a DFD using information gathered during requirements gathering and determination.
- Both processes and data structures are modeled in DFDs.

Process Modeling

Deliverables and Outcomes

- **Context data flow diagram (DFD).**
  - **Shows the scope of a system (i.e., a top-level view).**

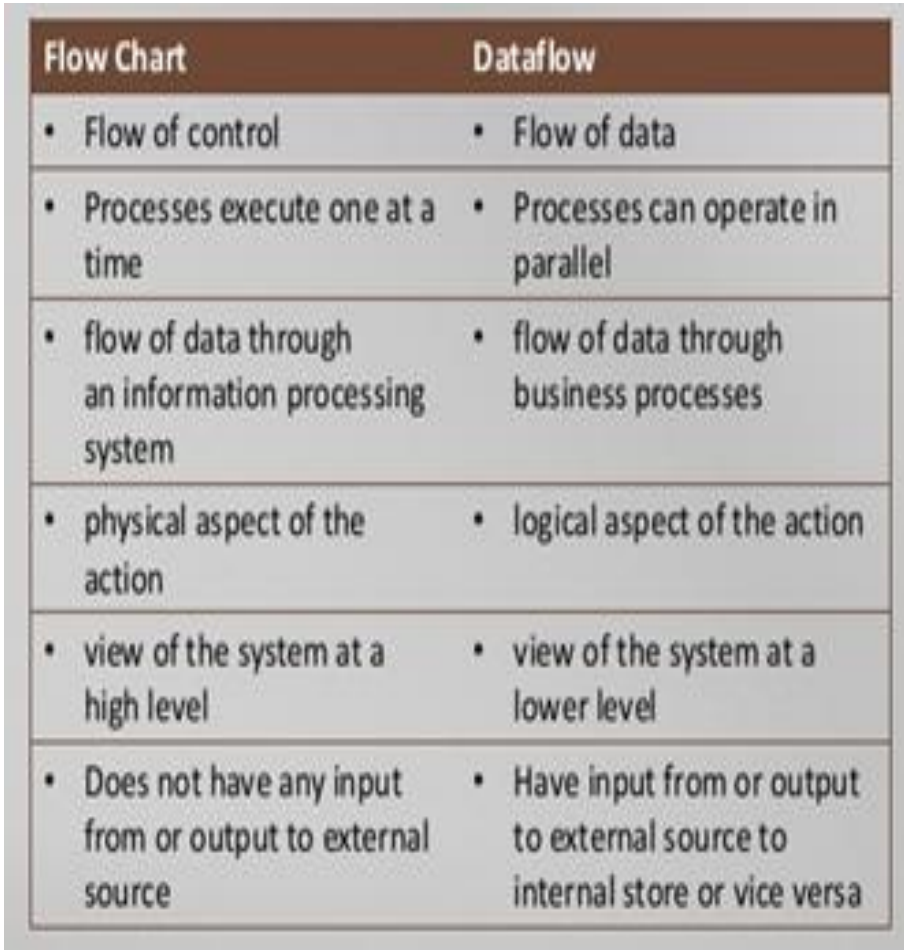
- **Often DFDs are created showing the current physical and logical system.**
  - **It enables analysts to understand how the current system operates.**
  - **The DFD is independent of technology.**
  - **It shows data flows, structure, and functional requirements of the new system.**

- **Includes a thorough description of each DFD component.**

Data Flow Diagram (DFD)

- A picture of the movement of data between **external entities** and the **processes** and **data stores** within a system.

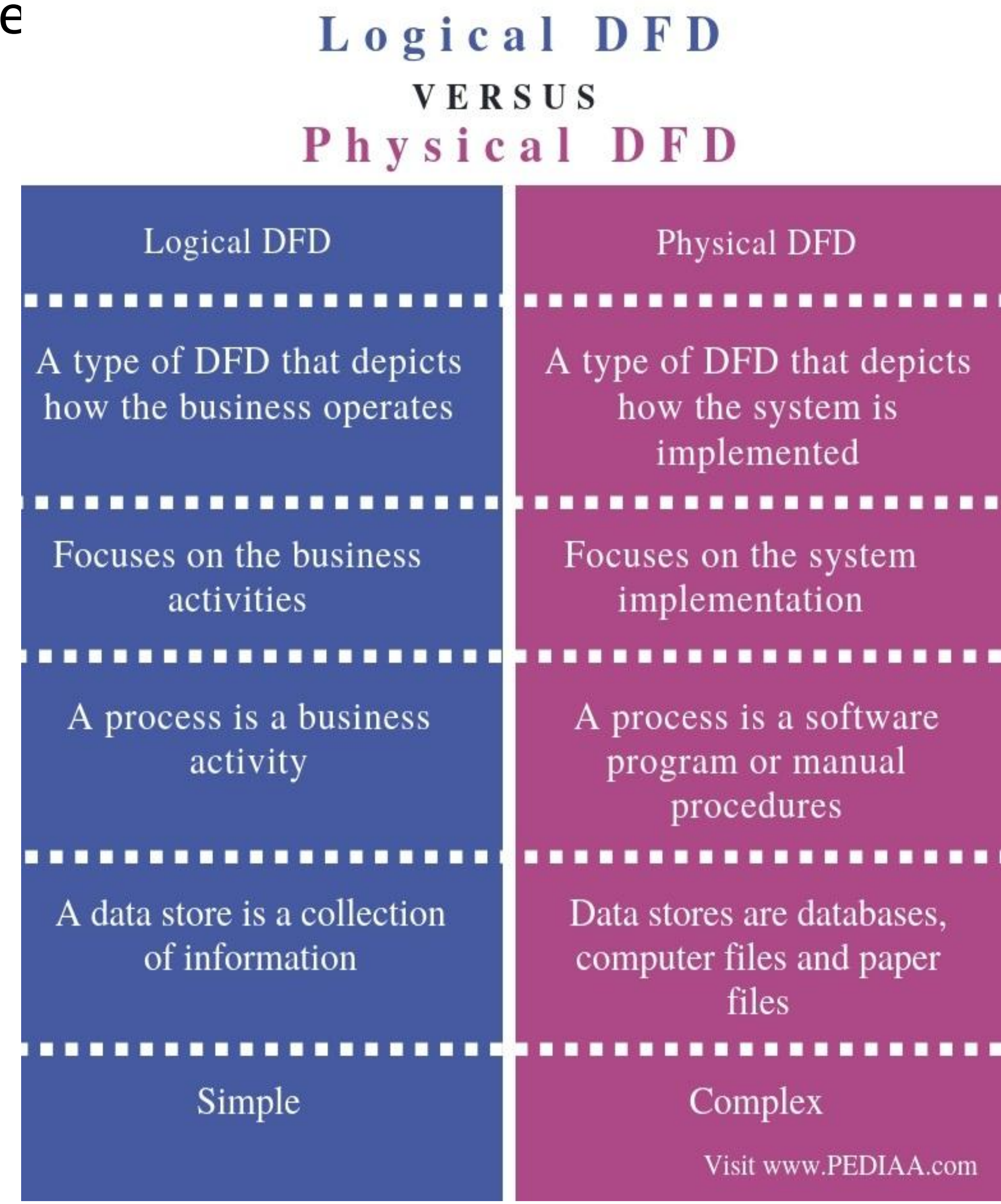
- **How does a DFD differ from a systems flowchart?**
  - DFDs depict **logical data flow** independent of technology.
  - The focus is on **data flows**, not process flows alone.



Flow Chart	Dataflow
• Flow of control	• Flow of data
• Processes execute one at a time	• Processes can operate in parallel
• flow of data through an information processing system	• flow of data through business processes
• physical aspect of the action	• logical aspect of the action
• view of the system at a high level	• view of the system at a lower level
• Does not have any input from or output to external source	• Have input from or output to external source to internal store or vice versa

Types of DFD

Data Flow Diagrams are eithe
Logical or Physical.



Logical DFD
VERSUS
Physical DFD

Logical DFD	Physical DFD
A type of DFD that depicts how the business operates	A type of DFD that depicts how the system is implemented
Focuses on the business activities	Focuses on the system implementation
A process is a business activity	A process is a software program or manual procedures
A data store is a collection of information	Data stores are databases, computer files and paper files
Simple	Complex

Visit www.PEDIAA.com

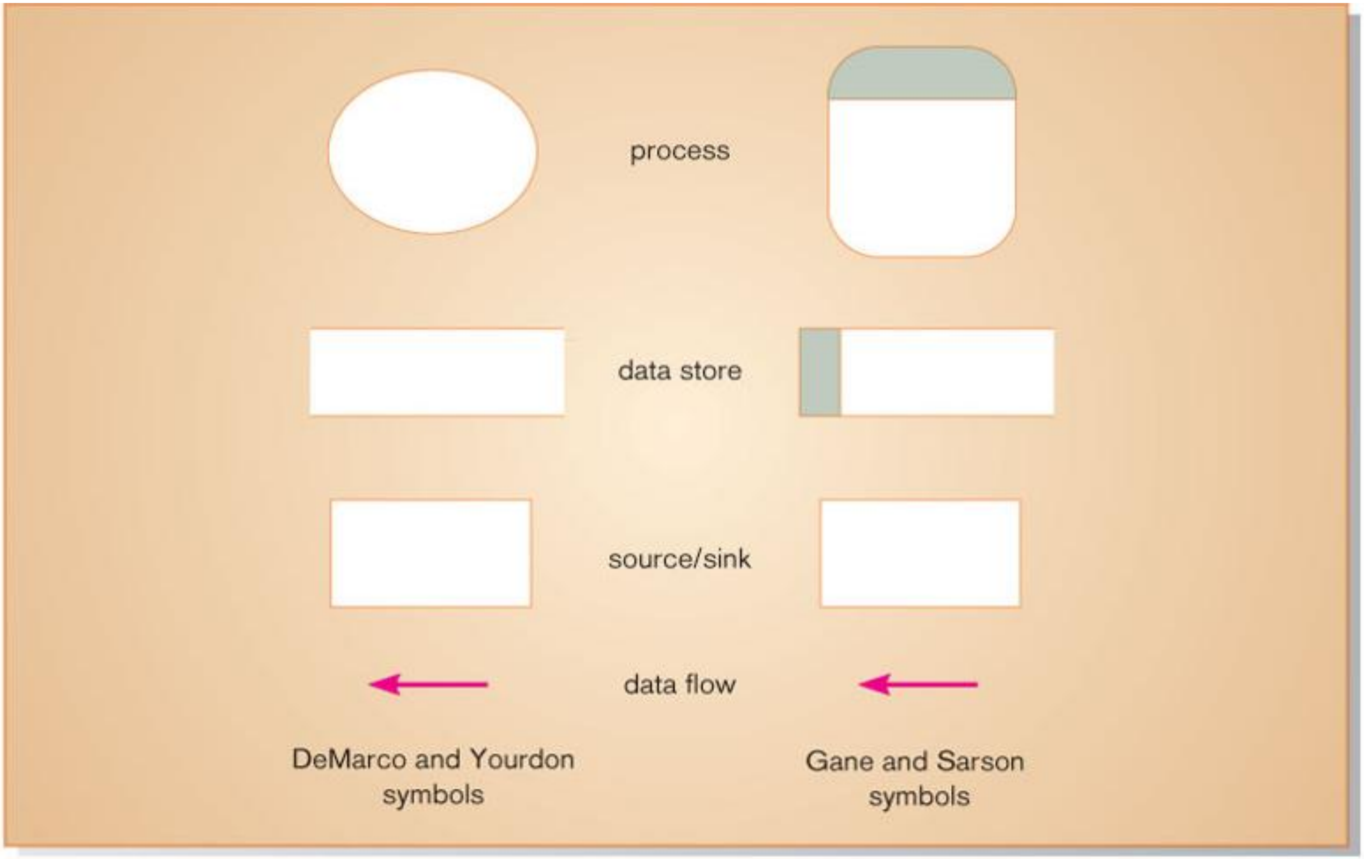
Logical DFD - This type of DFD concentrates on the system process, and flow of data in the system.

For example in a Banking software system, how data is moved between different entities.

Physical DFD - This type of DFD shows how the data flow is actually implemented in the system.

It is more specific and close to the implementation.

Comparison between DFD Symbols Sets

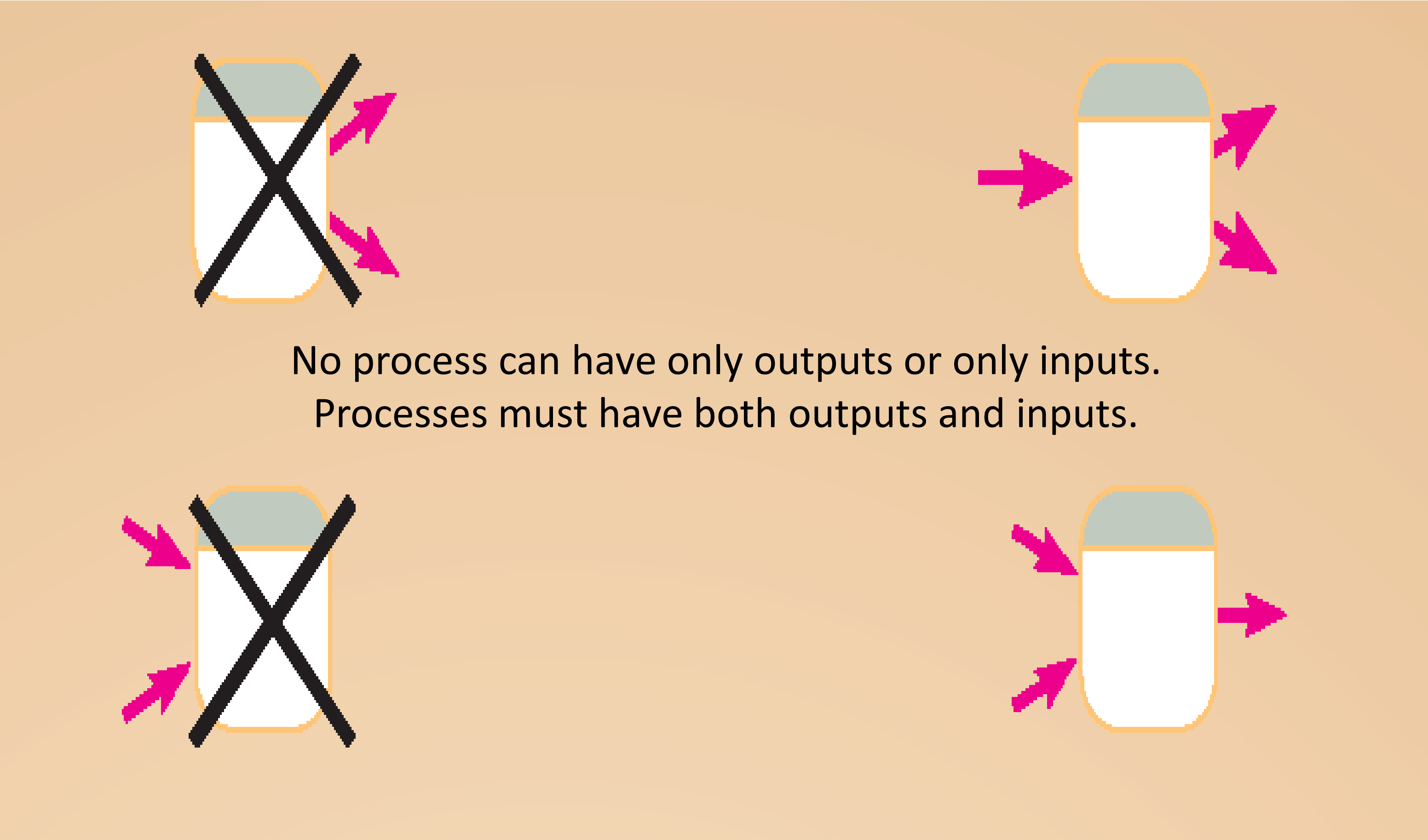


DFD Symbols

- **Process:** work or actions performed on data (inside the system).
- **Data Store:** data at rest (inside the system).
- **Source/Sink:** external entity that is origin or destination of data (outside the system).
- **Data flow:** arrows depicting movement of data.

DFD Diagramming Rules

Process

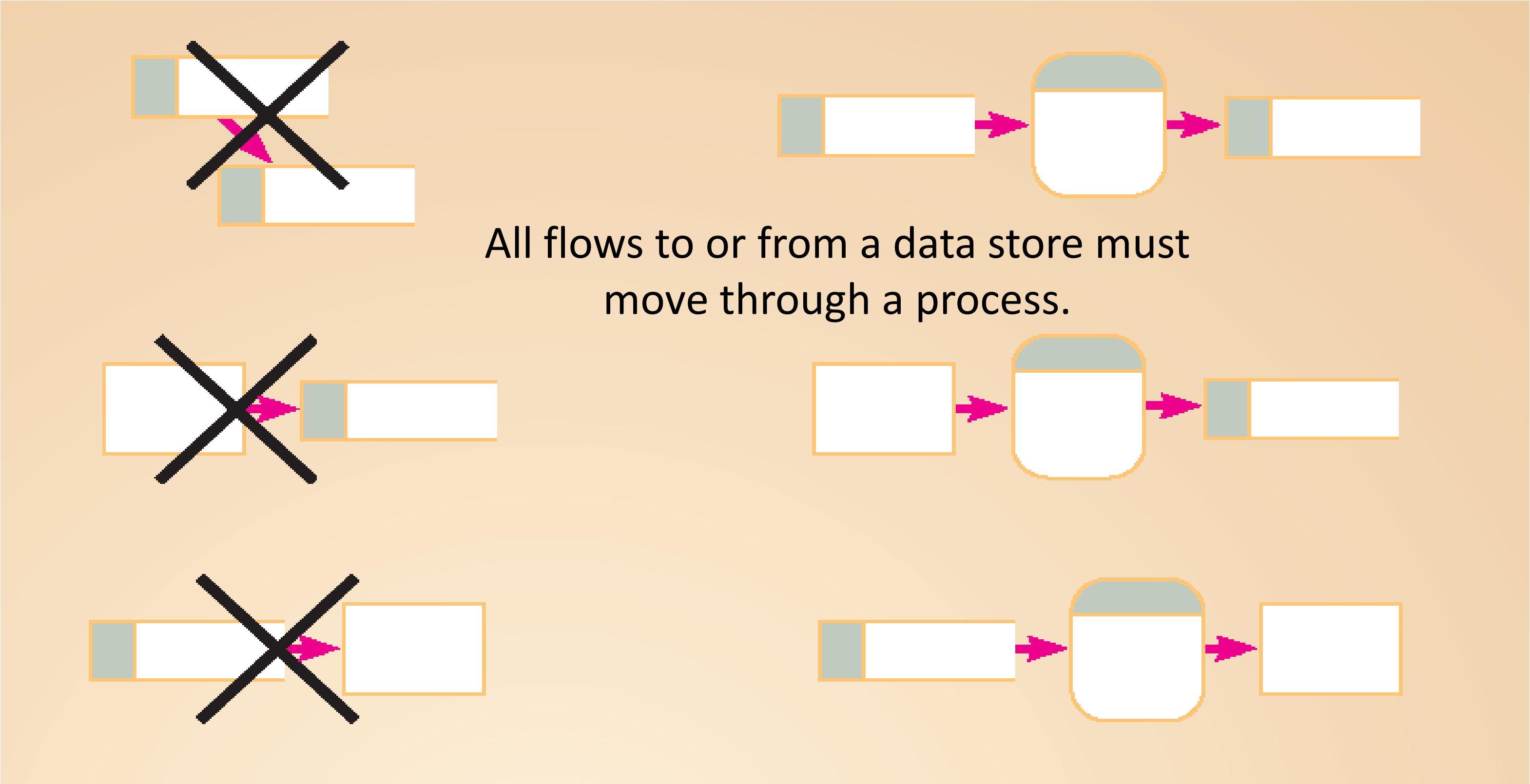


No process can have only outputs or only inputs.
Processes must have both outputs and inputs.

Process labels should be verb phrases.

DFD Diagramming Rules

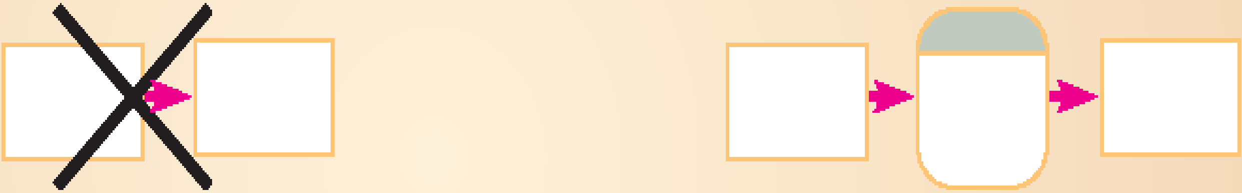
Data Store



All flows to or from a data store must move through a process.

Data Store labels should be noun phrases.

DFD Diagramming Rules
Source/Sink



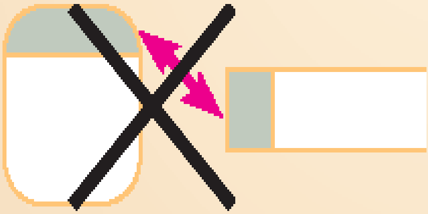
No data moves directly between external entities without going through a process.

Interactions between external entities without intervening processes are outside the system and therefore not represented in the DFD.

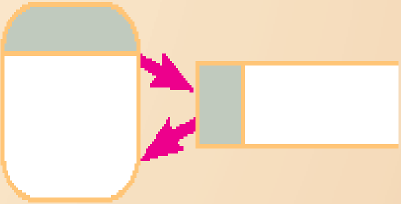
Source and Sink labels should be noun phrases.

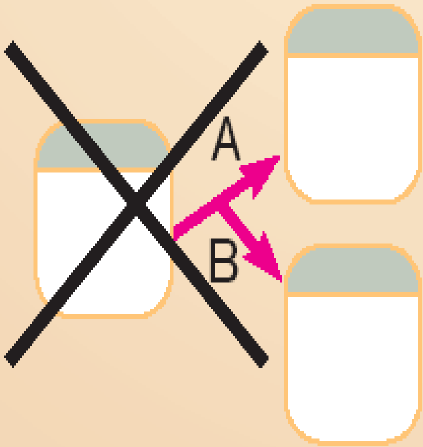
DFD Diagramming Rules

Data Flow

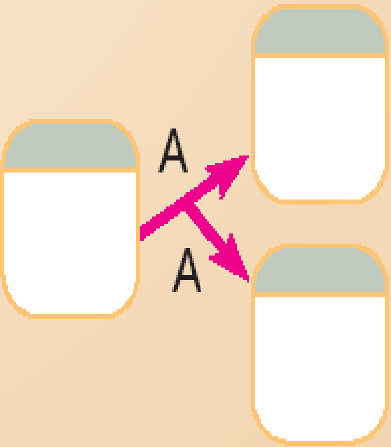


Bidirectional flow between process and data store is represented by two separate arrows.

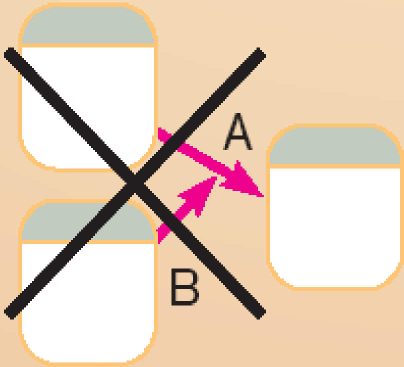




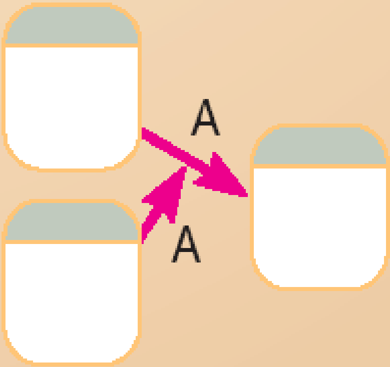
Forked data flow must refer to exact same data item (not different data items) from a common location to multiple destinations.

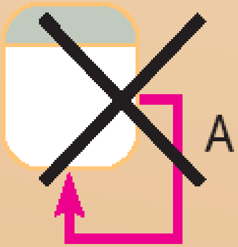


DFD Diagramming Rules
Data Flow

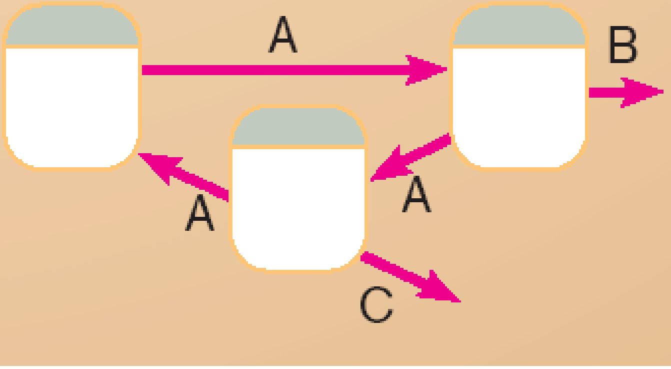


Joined data flow must refer to exact same data item (not different data items) from multiple sources to a common location.





Data flow cannot go directly from a process to itself, must go through intervening processes.



DFD Diagramming Rules
Data Flow

- **Data flow from a process to a data store means update (insert, delete or change).**
- **Data flow from a data store to a process means retrieve or use.**
- **Data flow labels should be noun phrases.**

Functional Decomposition

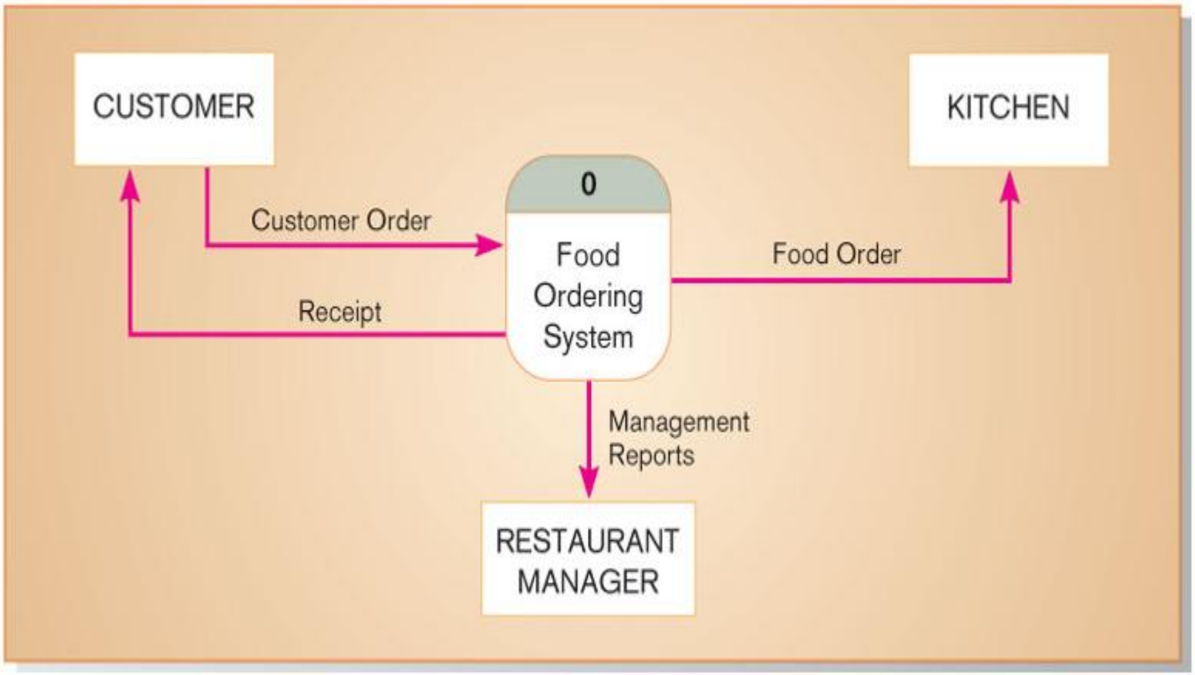
- An iterative process of breaking a system description down into finer and finer detail.
- High-level processes described in terms of lower-level sub-processes.
- DFD charts created for each level of detail.

DFD Levels

- **Context DFD**
  - Overview of the organizational system.
- **Level-0 DFD**
  - Representation of system’s major processes at high level of abstraction.
- **Level-1 DFD**
  - Results from decomposition of Level 0 diagram.
- **Level-n DFD**
  - Results from decomposition of Level n-1 diagram.

Context Diagram

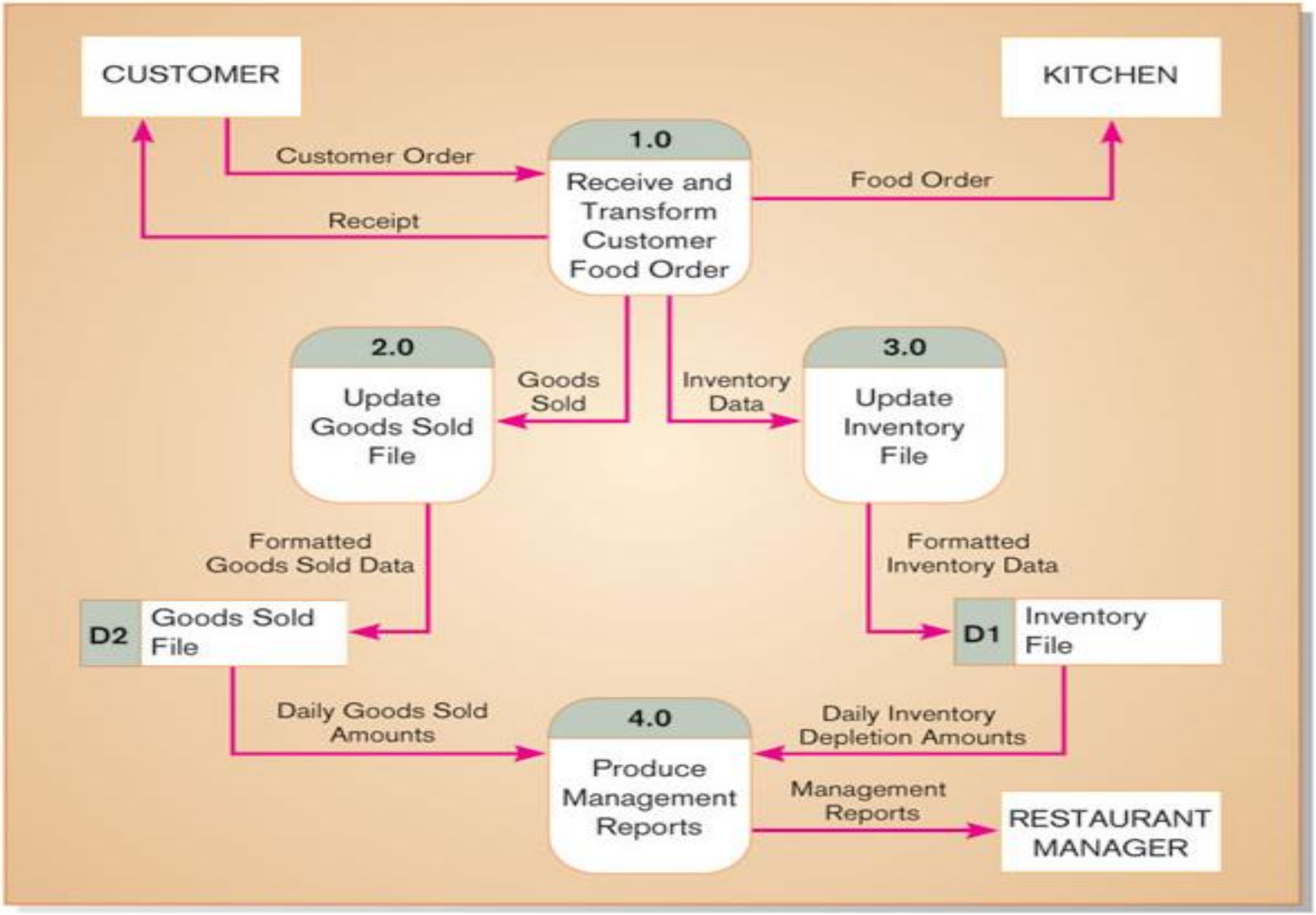
of Hoosier Burger’s food ordering system



Context diagram shows the **system boundaries, external entities** that interact with the system, and **major information flows** between entities and the system.

NOTE: only one process symbol, and no data stores shown.

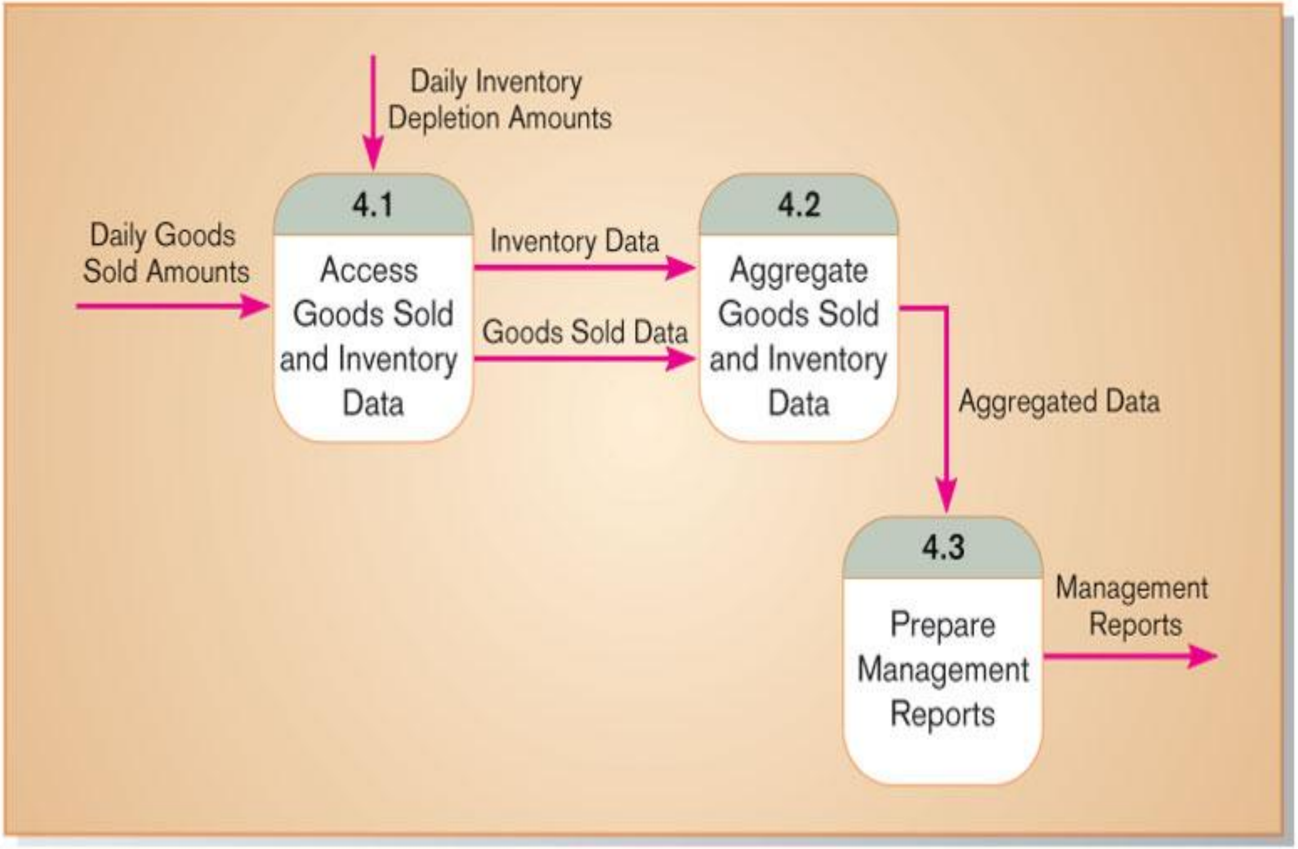
Level-0 DFD



Level-0 DFD shows the system's major processes, data flows, and data stores at a high level of abstraction.

Processes are labeled 1.0, 2.0, etc. These will be decomposed into more primitive (lower-level) DFDs.

Level-1 DFD

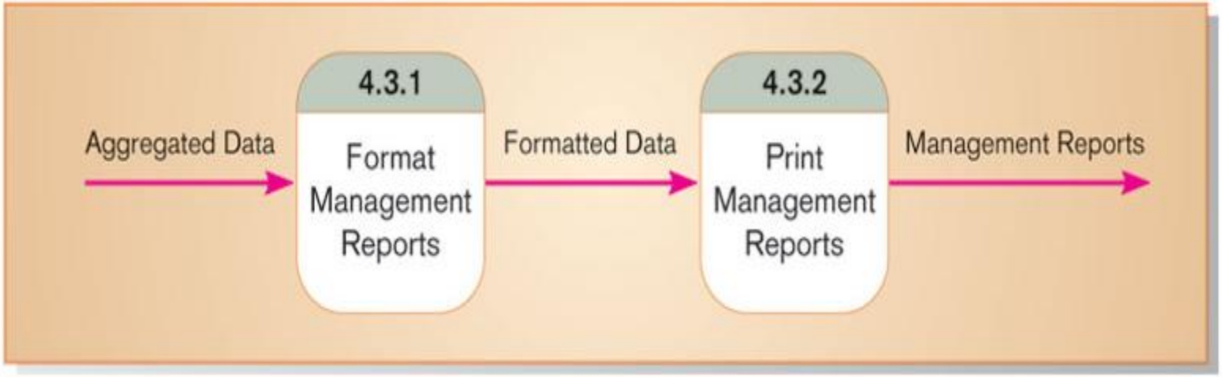

Daily Inventory Depletion Amounts
Daily Goods Sold Amounts
4.1
Access Goods Sold and Inventory Data
Inventory Data
Goods Sold Data
4.2
Aggregate Goods Sold and Inventory Data
Aggregated Data
4.3
Prepare Management Reports
Management Reports

Level-1 DFD shows the sub-processes of one of the processes in the Level-0 DFD.

This is a Level-1 DFD for Process 4.0.

Processes are labeled 4.1, 4.2, etc. These can be further decomposed in more primitive (lower-level) DFDs if necessary.

Level-n DFD



Level-*n* DFD shows the sub-processes of one of the processes in the Level *n-1* DFD.

This is a Level-2 DFD for Process 4.3.

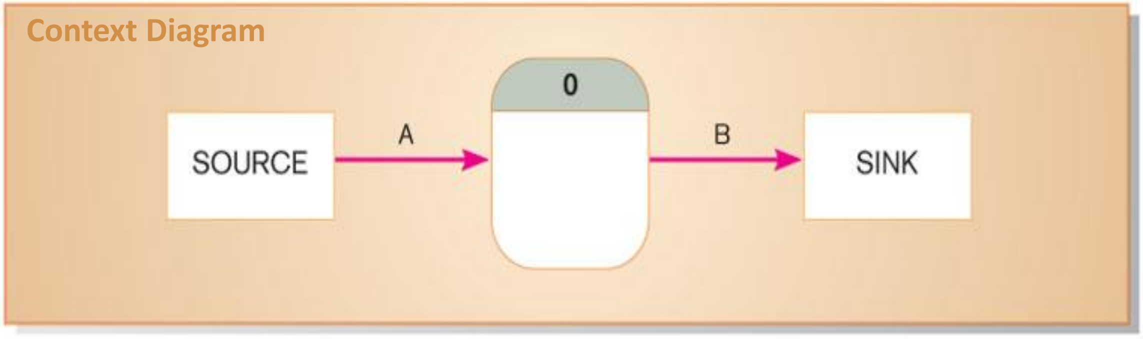
Processes are labeled 4.3.1, 4.3.2, etc. If this is the lowest level of the hierarchy, it is called a *primitive DFD.*

DFD Balancing

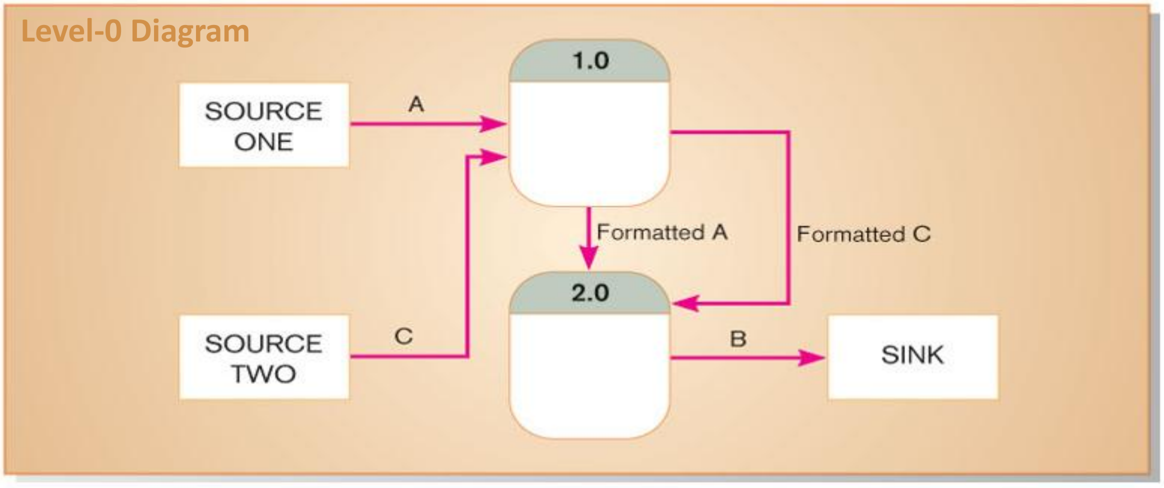
- The conservation of inputs and outputs to a data flow process when that process is decomposed to a lower level.

- **Balanced means:**
  - Number of inputs to lower level DFD equals number of inputs to associated process of higher-level DFD.
  - Number of outputs to lower level DFD equals number of outputs to associated process of higher-level DFD.

Unbalanced DFD



Context Diagram



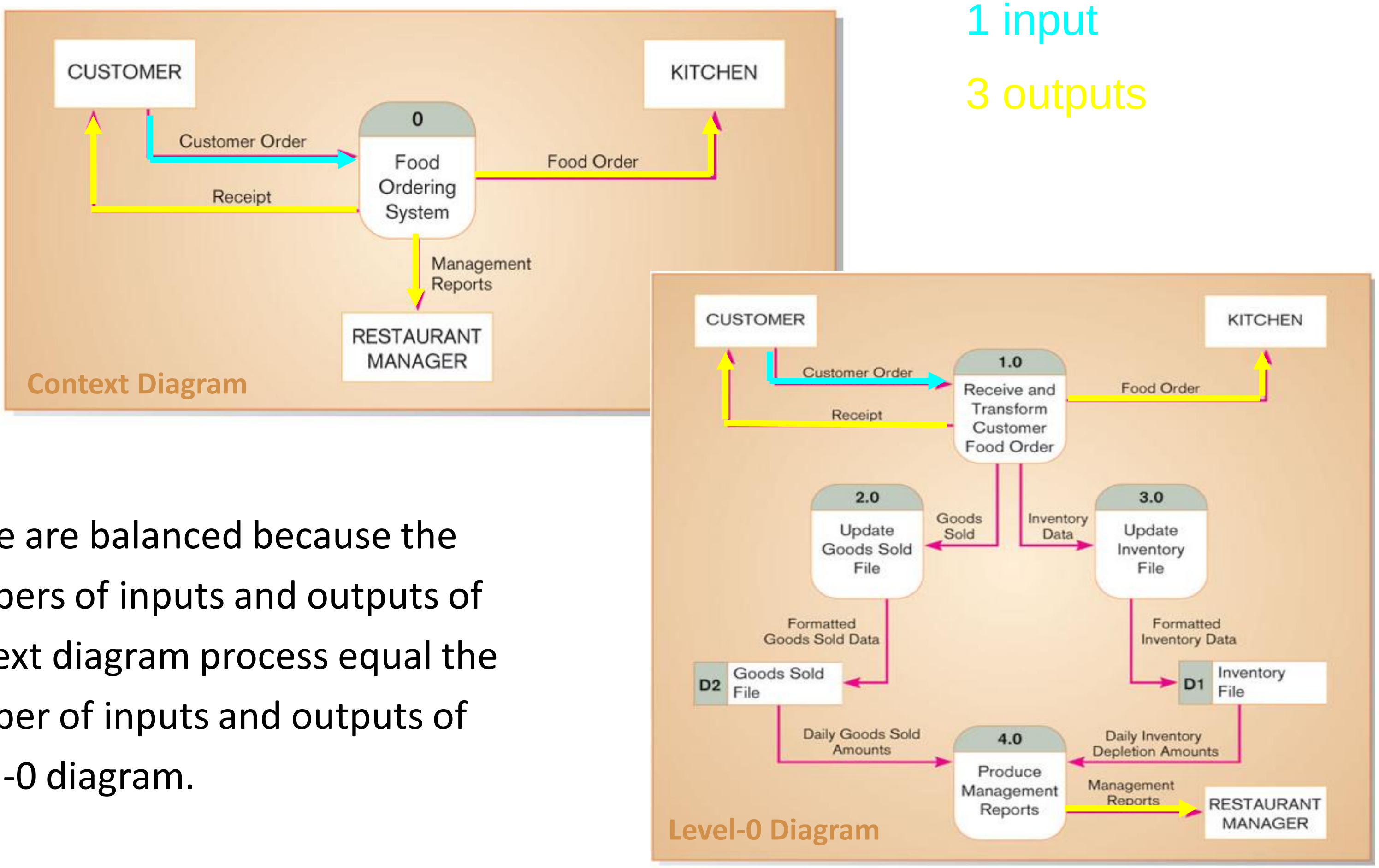
Level-0 Diagram

This is unbalanced because the process of the context diagram has only one input but the Level-0 diagram has two inputs.

Balanced DFD

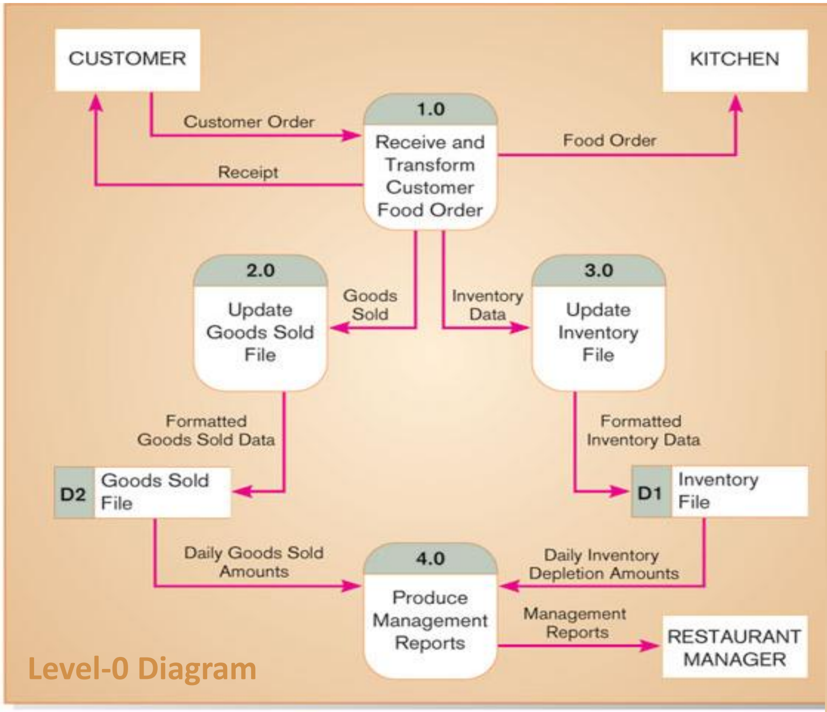
1 input

3 outputs



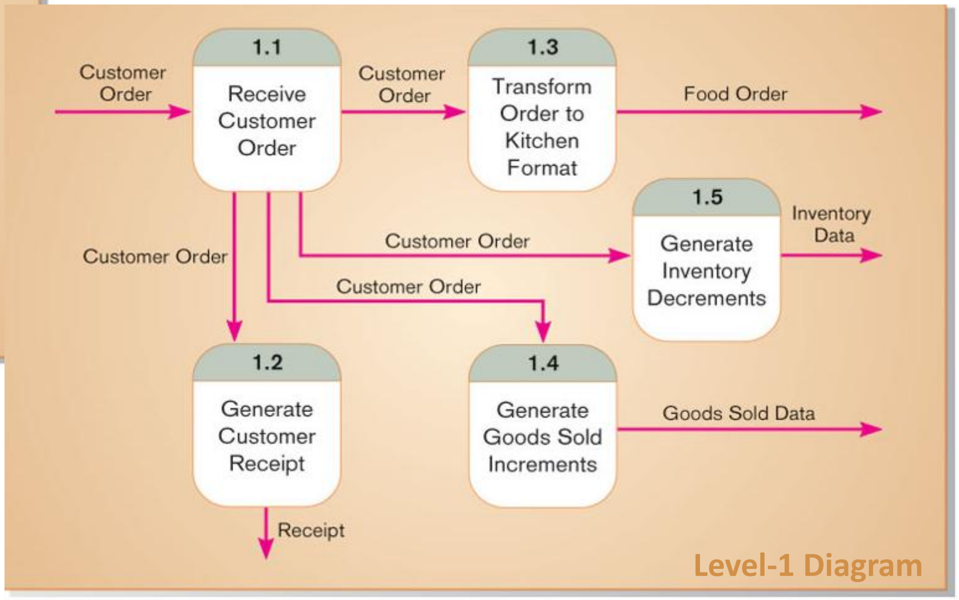
These are balanced because the numbers of inputs and outputs of context diagram process equal the number of inputs and outputs of Level-0 diagram.

Balanced DFD



Level-0 Diagram

These are balanced because the numbers of inputs and outputs to **Process 1.0** of the Level-0 diagram equals the number of inputs and outputs to the Level-1 diagram.

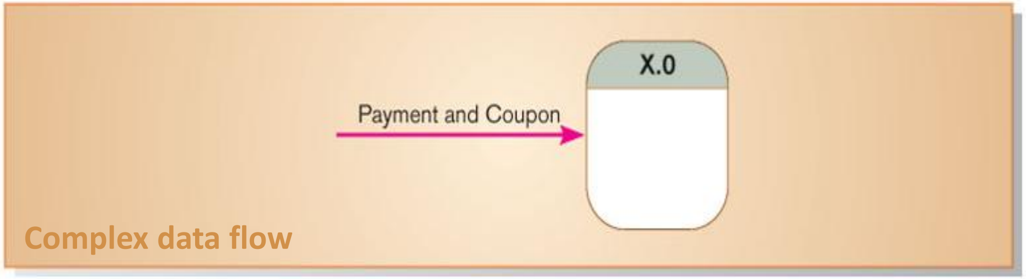


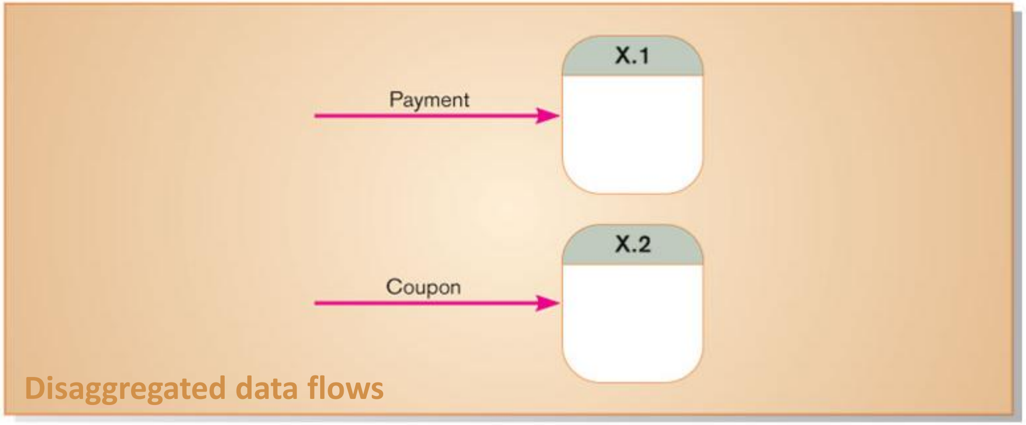
Level-1 Diagram

1 input

4 outputs

Data Flow Splitting





A composite data flow at a higher level may be split if different parts go to different processes in the lower level DFD.

This **remains balanced** because the same data is involved, but split into two parts.

Example: Data Dictionary

response: [bill + material-issue-slip, reject-message]

query: period /* query from manager regarding sales statistics*/

period: [date+date,month,year,day]

date: year + month + day

year: integer

month: integer

day: integer

order: customer-id + {items + quantity}*

accepted-order: order /* ordered items available in inventory */

reject-message: order + message /* rejection message */

pending-orders: customer-id + {items+quantity}*

customer-address: name+house #+street #+city+pin

Data dictionary

A data dictionary lists all data items appearing in the DFD model of a system.

The data items listed include all data flows and the contents of all data stores appearing on the DFDs in the DFD model of a system.

A data dictionary lists the purpose of all data items and the definition of all composite data items in terms of their component data items.

For example, a data dictionary entry may represent that the data grossPay consists of the components regularPay and overtimePay.

grossPay = regularPay + overtimePay

For the smallest units of data items, the data dictionary lists their name and their type.

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

+ : denotes composition of two data items, e.g. a+b represents data a and b.

[,,] : represents selection, i.e. any one of the data items listed in the brackets can occur. For example, [a,b] represents either a occurs or b occurs.

() : the contents inside the bracket represent optional data which may or may not appear. e.g. a+(b) represents either a occurs or a+b occurs.

{} : represents iterative data definition, e.g. {name} 5 represents five name data. {name}* represents zero or more instances of name data.

= : represents equivalence, e.g. a=b+c means that a represents b and c.

/* */ : Anything appearing within /* and */ is considered as a comment.

Observation

From the examples,

observe that DFDs help create:

data model

function model

As a DFD is refined into greater levels of detail:

the analyst performs an implicit functional decomposition.

At the same time, refinements of data takes place.

Guidelines For Constructing DFDs

Context diagram should represent the system as a single bubble:

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

All external entities should be represented in the context diagram and level 0:

external entities should not appear at any other level of DFD.

Only 3 to 7 bubbles per diagram should be allowed:

each bubble should be decomposed to between 3 and 7 bubbles.

Guidelines For Constructing DFDs

A common mistake committed by many beginners:
attempting to represent control information in a DFD.
e.g. trying to represent the order in which different functions are executed.

A DFD does not represent control information:

when or in what order different functions (processes) are invoked

the conditions under which different functions are invoked are not represented.

For example, a function might invoke one function or another depending on some condition.