ICT

Data Flow Diagrams

Analytical Tools

Lecturer - Teran Subasinghe
AURORA COMPUTER STUDIES

DATA FLOW DIAGRAMS

Data Flow Diagramming is a means of representing a system at any level of detail with a graphic network of symbols showing data flows, data stores, data processes, and data sources/destinations.

A DFD usually comprises of four components. These four components can be represented by four simple symbols. These symbols can be explained in detail as follows: External entities (source/destination of data) are represented by squares; Processes (input-processing-output) are represented by rectangles with rounded corners; Data Flows (physical or electronic data) are represented by arrows; and finally, Data Stores (physical or electronic like XML files) are represented by open-ended rectangles.

PURPOSE AND OBJECTIVE

The purpose of data flow diagrams is to provide a semantic bridge between users and systems developers. The diagrams are:

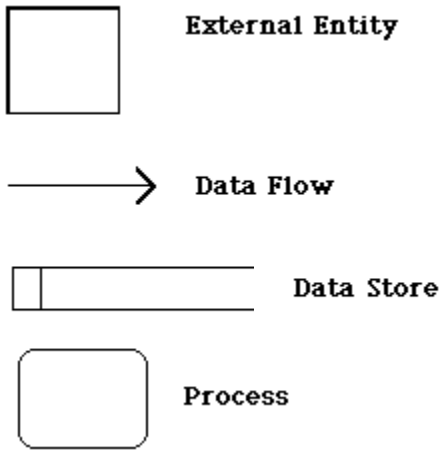
- Graphical, eliminating thousands of words;
- Logical representations, modeling WHAT a system does, rather than physical models showing HOW it does it;
- Hierarchical, showing systems at any level of detail; and
- Jargon less, allowing user understanding and reviewing.

The goal of data flow diagramming is to have a commonly understood model of a system. The diagrams are the basis of structured systems analysis.

Data flow diagrams have the objective of avoiding the cost of:

- User/developer misunderstanding of a system, resulting in a need to redo systems or in not using the system.
- Having to start documentation from scratch when the physical system changes since the logical system, WHAT gets done, often remains the same when technology changes.
- Systems inefficiencies because a system gets "computerized" before it gets "systematized".
- Being unable to evaluate system project boundaries or degree of automation, resulting in a project of inappropriate scope.

SYMBOLS



- The External Entity symbol represents sources of data to the system or destinations of data from the system.
- The Data Flow symbol represents movement of data.
- The Data Store symbol represents data that is not moving (delayed data at rest).
- The Process symbol represents an activity that transforms or manipulates the data (combines, reorders, converts, etc.).
- Any system can be represented at any level of detail by these four symbols.

EXTERNAL ENTITIES:

1. ARE NAMED WITH APPROPRIATE NAME.
2. CAN BE DUPLICATED, ONE OR MORE TIMES, ON THE DIAGRAM TO AVOID LINE CROSSING.
3. DETERMINE THE SYSTEM BOUNDARY. THEY ARE EXTERNAL TO THE SYSTEM BEING STUDIED. THEY ARE OFTEN BEYOND THE AREA OF INFLUENCE OF THE DEVELOPER.
4. CAN REPRESENT ANOTHER SYSTEM OR SUBSYSTEM.
5. GO ON MARGINS/EDGES OF DATA FLOW DIAGRAM.

Data Flows:

1. are represented with a line with an arrowhead on one end. A fork in a data flow means that the same data goes to two separate destinations. The same data coming from several locations can also be joined.
2. should only represent data, not control.
3. are ALWAYS named. Name is not to include the word "data".
4. are referenced by a combination of the identifiers of the constructs that the data flow connects.

Data Stores:

1. are generic for physical files (index cards, desk drawers, magnetic disk, magnetic tape, shirt pocket, human memory, etc.)
2. are named with an appropriate name, not to include the word "file", and numbered with a number preceded with a capital letter D
3. can be duplicated, one or more times, to avoid line crossing.
4. can show two or more systems that share a data store. This is done by adding a solid stripe on the left boundary. This can occur in the case of one system updating the data store, while the other system only accesses the data. For ex ample, the data store could be a freight rate book that one system builds and maintains, but is used by the represented system.
5. are detailed in the data dictionary or with data description diagrams.

Processes:

1. show data transformation or change. Data coming into a process must be "worked on" or transformed in some way. Thus, all processes must have inputs and outputs. In some (rare) cases, data inputs or outputs will only be shown at more detailed levels of the diagrams. Each process in always "running" and ready to accept data.
2. are represented by a rounded corner rectangle
3. are named with one carefully chosen verb and an object of the verb. There is no subject. Name is not to include the word "process". Each process should represent one function or action. If there is an "and" in the name, you likely have more than one function (and process).
4. have physical location shown only for existing physical systems or a physical design is being represented.
5. are numbered within the diagram as convenient. Levels of detail are shown by decimal notation. For example, top level process would be Process 14, next level of detail Processes 14.1-14.4, and next level with Processes 14.3.1-14.3.6.

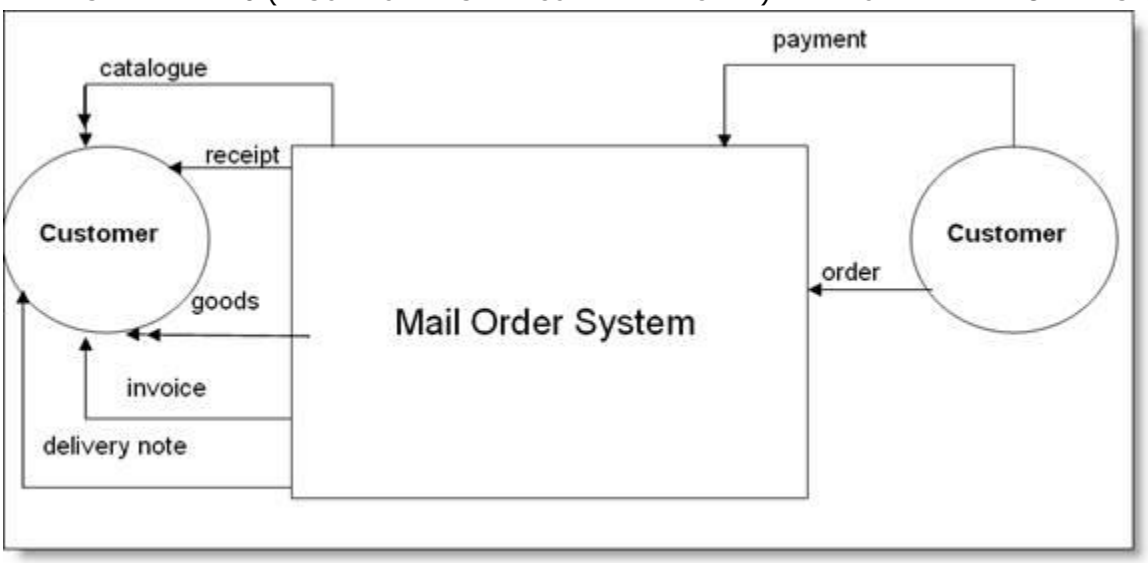
EXAMPLE

SCENARIO

Suppose you are given the details of a small mail order catalogue system that allows people to shop from home. When a customer receives the catalogue and wants to buy something, they can telephone, fax or email their order to the company. The company gets the order and sends the goods and an invoice. When the customer receives the goods with a delivery note, they send payment and receive a receipt for their payment. The first thing we must do is model the main outputs and sources of data in the scenario above. Then we draw the system box and name the system. Next we identify the information that is flowing to the system and from the system.

LEVEL 0 DFD:

HERE IS THE LEVEL 0 (ALSO KNOWN AS THE CONTEXT DIAGRAM) DFD FOR THE MAIL ORDER SYSTEM:



You will see that the Customer (a source of information) has sent in an order.

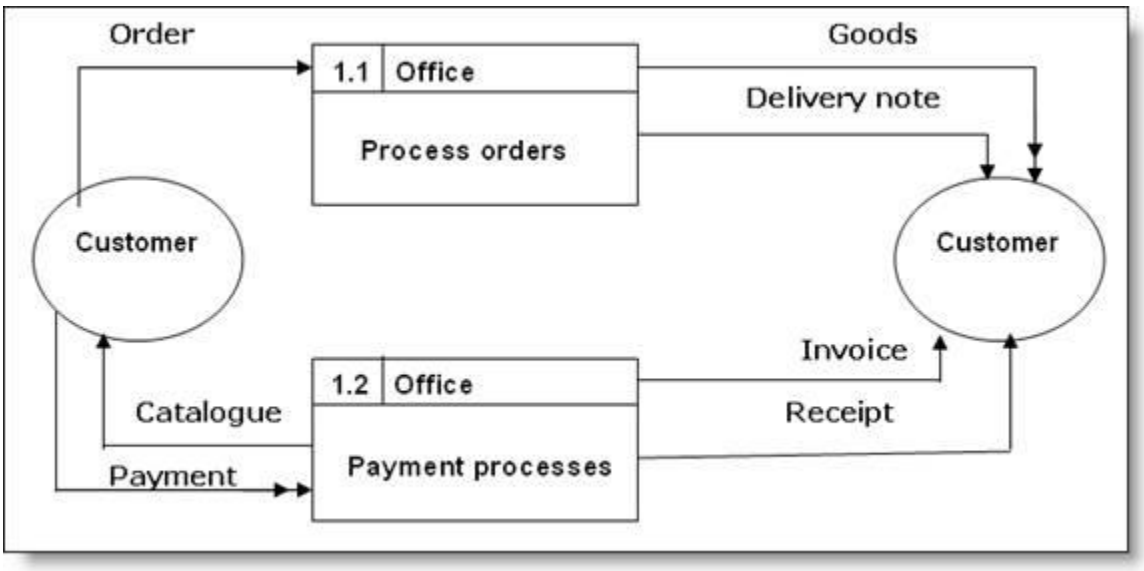
- The system has sent out an invoice data flow, a delivery note data flow and goods ('goods' is not information; it is shown as a double-headed arrow because it is a physical or resource data flow).
- Payment comes in - i.e. a payment data flow is received, and a receipt data flow goes out to the customer.
- Catalogue goes out to the customer - this is also a physical data flow.

We now have a top-level view of the information flow in and out of the system.

LEVEL 1 DFD:

THE NEXT STAGE IS TO CREATE THE LEVEL 1 DATA FLOW DIAGRAM. THIS HIGHLIGHTS THE MAIN FUNCTIONS CARRIED OUT BY THE SYSTEM. AS A RULE, WE TRY TO DESCRIBE THE SYSTEM USING BETWEEN TWO AND SEVEN FUNCTIONS - TWO BEING A SIMPLE SYSTEM AND SEVEN BEING A COMPLICATED SYSTEM.

Some functions could be described in sentences like: Gets orders from customers and receives payment when they are delivered.



However, we are not quite finished with this diagram yet. There are certain rules we must learn in order to ensure that we create a valid DFD. Here they are below - it is recommended that you learn these well!

Rule 1 - Does each function have input and output?

Rule 2 - Does each function have all the information it needs in order to produce its output?

Rule 3 - If not, then what information does it need and where will it get that information from?

Applying these simple rules ensures a valid Data Flow Model. So let us apply these rules to our Level 1 DFD.

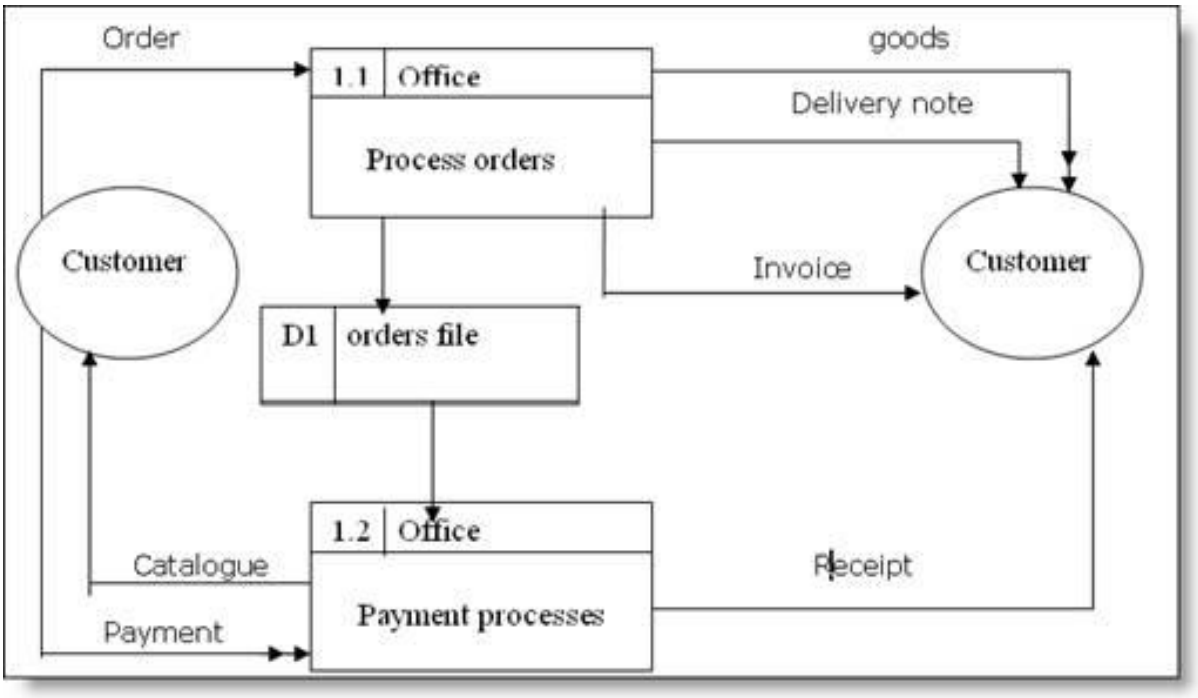
Rule 1: Answer YES.

Rule 2: Function 1: YES - it does have all the information it needs. It only needs order details to produce its output - invoice.

Function 2: NO - it does not have all the information it needs to produce its output (receipt). So we must apply Rule 3 to Function 2.

Rule 3: Function 2 has payment details but it also needs order details so that it can match up payment with order. So where will it get information from? We can see that Function 1 receives order details from the customer, so it must receive order details from Function 1.

How do we show this in our DFD? Well, we can assume that when orders enter the system they are stored in the system for further use. We can assume that Function 1 stores order details. This is shown on the below:



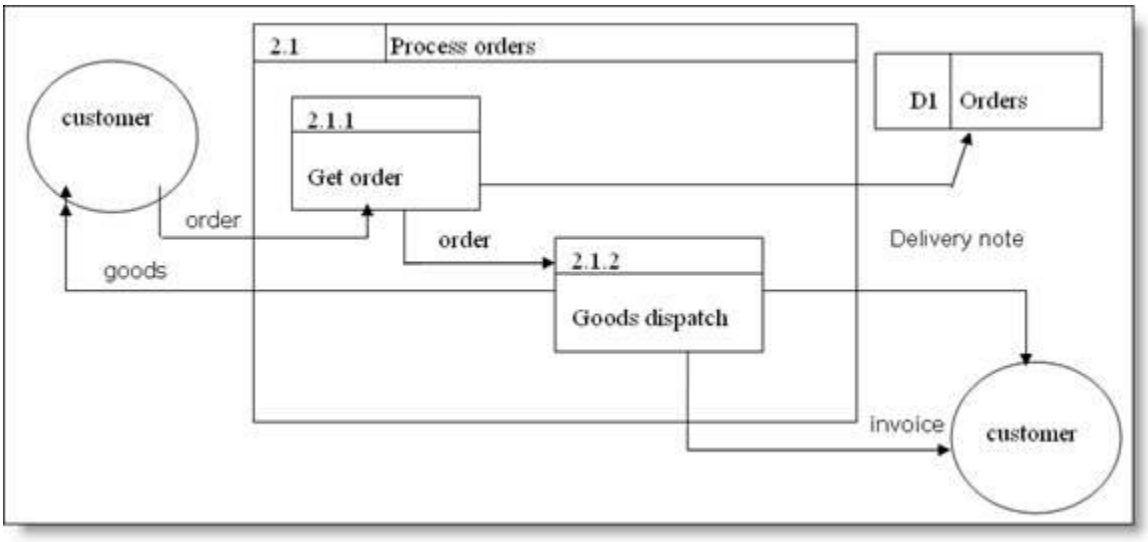
We have used the data store symbol to show the stored orders. Note that Function 1 *creates* the order data store and Function 2 *reads* the data store. Note also how important it is to apply all the rules - if we only applied the main rule (i.e. all functions to input and output), the diagram would have been incomplete.

LEVEL 2 DFD:

WE NOW CREATE THE LEVEL 2 DATA FLOW DIAGRAMS. A POSSIBLE SOLUTION IS AS SHOWN BELOW. WE DESCRIBE EACH FUNCTION USING TWO TO SEVEN SENTENCES. THESE SENTENCES THEN BECOME THE FUNCTION'S PROCESSES. SINCE THIS IS A SIMPLE SYSTEM, WE WILL USE TWO PROCESSES FOR EACH FUNCTION:

Function 1 - Process Order	1. Receive order 2. Issue invoice and goods.
Function 2 - Process Payments	1. Receive payment 2. Issue receipt and catalogue

PROCESS ORDERS UPDATED DIAGRAM



PROCESS PAYMENTS UPDATED DIAGRAM WITH A DATA STORE