Data Flow Diagram with Examples - Vehicle Maintenance Depot

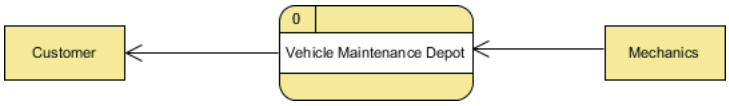
Written Date : February 16, 2015

Data Flow Diagram (DFD) provides a visual representation of the flow of information (i.e. data) within a system. By drawing a Data Flow Diagram, you can tell the information supplied by and delivered to someone who take part in system processes, the information needed in order to complete the processes and the information needed to be stored and accessed. This article describes and explain Data Flow Diagram (DFD) by using a vehicle maintenance depot system as an example.

The Vehicle Maintenance Depot System Example

Context DFD

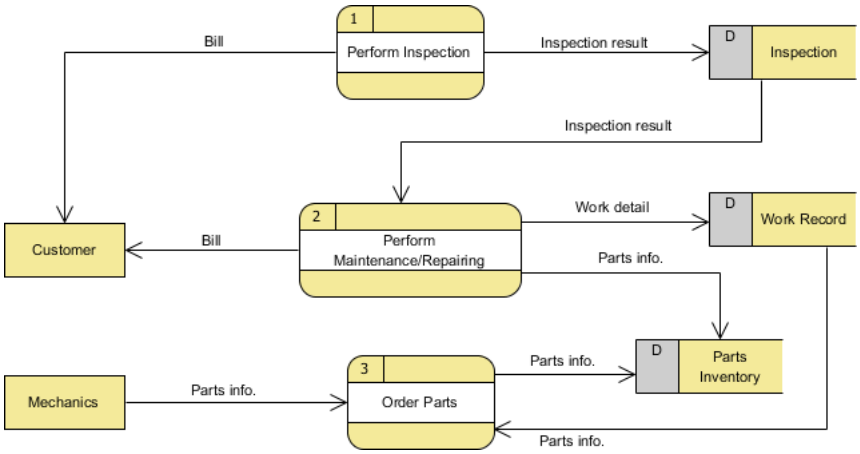
The figure below shows a context Data Flow Diagram drawn for a vehicle maintenance depot system. It contains a process (shape) that represents the system to model, in this case, the "*vehicle maintenance depot system*". It also shows the participants who will interact with the system, called the external entities. In this example, *Customer* and *Mechanics* are the entities who will interact with the system. In between the process and the external entities there are data flow (connectors) that indicate the existence of information exchange between the entities and the system.



Context DFD is the entrance of a data flow model. It contains one and only one process and does not show any data store.

Level 1 DFD

The figure below shows the level 1 DFD, which is the decomposition (i.e. break down) of the system shown in the context DFD. Read through the diagram, and then we will introduce some of the key concepts based on this diagram.



The Data Flow Diagram example contains three processes, two external entities and three data stores. Although there is no design guidelines that governs the positioning of shapes in a Data Flow Diagram, we tend to put the processes in the middle and data stores and external entities on the sides to make it easier to comprehend.

Based on the diagram, we know that the *Perform Inspection* process provides *Bill* to the *Customer* and store the *Inspection result* into the *Inspection* data store.

The *Perform Maintenance/Repairing* process takes *Inspection result* from *Inspection* data store as input, and provides the *Customer* with the *Bill*. Besides, *Work detail* is stored in the *Work Record* data store and *Parts info.* is stored in the *Parts* Inventory data store. Note that Data Flow Diagram does not represent the order of data flow. Strictly speaking, this diagram only tells us the *Perform Maintenance/ Repairing* process receives *Inspection result* as input and produce *Bill*, *Work detail* and *Part info.*, with no order specified. Keep in mind that Data Flow Diagram does not answer in what way and in what order the information is being used throughout a system. If this information is important and worth mentioning, consider to model it with diagrams like BPMN Business Process Diagram or UML Activity Diagram.

A *Mechanics* can *Order Parts* by providing *Parts info.*, and the result is the storage of *Parts info.* in the *Parts Inventory* data store. The process also receives *Parts info.* from *Work Record* data store throughout the process.

Data Flow Diagram Tips and Cautions

Be aware of the level of details

In this Data Flow Diagram example the words "detail" and "info" are used many times when labeling data. We have "work detail", "parts info", etc. What if we write them explicitly as "case id, symptom, problem description, solution" and "part name, quantity, discount"? Is this correct? Well, there is no definite answer to this question but try to ask yourself a question when making a decision. Why are you drawing a DFD?

In most cases, Data Flow Diagram is drawn in the early phase of system development, where many details are yet to be confirmed. The use of general terminologies like "details", "info", "result" certainly leave room for discussion. However, using general terms can be kind of lacking details and make the design lost it usefulness. So it really depends on the purpose of your design.

Don't overdrawn

In a Data Flow Diagram, we focus on the interactions between the system and external parties, rather than the internal communications among interfaces. Therefore, data flows between interfaces and the data stores used are considered to be out of scope and should not be shown in the diagram.

Don't mix up data flow and process flow

Data Flow Diagram was designed for representing the exchange of information. Connectors in a Data Flow Diagram are for representing data, not for representing process flow, step or anything else. When we label a data flow that ends at a data store "a request", this literally means we are passing a request as data into a data store. Although this may be the case in implementation level as some of the DBMS do support the use of functions, which intake some values as parameters and return a result, in Data Flow Diagram we tend to treat data store as a sole data holder that does not possess any processing capability. If you want to model the system flow or process flow, use UML Activity Diagram or BPMN Business Process Diagram instead. If you want to model the internal structure of data store, consider to use Entity Relationship Diagram instead.