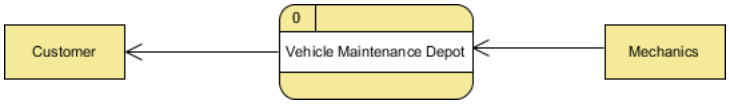
Data Flow Diagram with Examples - Vehicle Maintenance Depot

Written Date : February 16, 2015

The Vehicle Maintenance Depot System Example

Context DFD

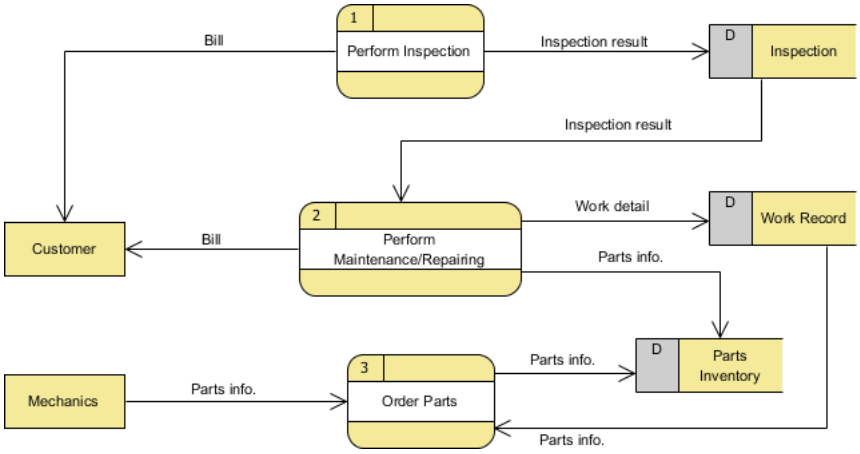
The figure below shows a context Data Flow Diagram drawn for a vehicle maintenance depot system. It contains a process (shape) that represents the system to be modeled, in this case, the "*vehicle maintenance depot system*." It also shows the participants who will interact with the system, called external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



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

Level 1 DFD

The figure below shows the level 1 DFD, which is the decomposition (i.e., breakdown) 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 are no design guidelines that govern the positioning of shapes in a Data Flow Diagram, we tend to put the processes in the middle and the 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 a *Bill* to the *Customer* and stores the *Inspection result* in the *Inspection* data store.

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

A *Mechanic* 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 the *Work Record* data store throughout the process.

Data Flow Diagram Tips and Cautions

Be Aware of the Level of Detail

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 were to write them out 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, a Data Flow Diagram is drawn in the early phase of system development, when many details have yet to be confirmed. The use of general terminologies like "details," "info," and "result" certainly leaves room for discussion. However, using general terms can be lacking in detail and can make the design lose its usefulness. So, it really depends on the purpose of your design.

Don't Overdraw

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 they use are considered to be out of scope and should not be shown in the diagram.

Don't Mix Up Data Flow and Process Flow

A Data Flow Diagram is designed to represent the exchange of information. Connectors in a DFD are for representing data, not for representing a process flow, step, or anything else. When we label a data flow that ends at a data store as "a request," this literally means we are passing a request as data into a data store. Although this may be the case at the implementation level, as some DBMSs do support the use of functions that take in some values as parameters and return a result, in a Data Flow Diagram, we tend to treat a 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, you could use a UML Activity Diagram or a BPMN Business Process Diagram instead. If you want to model the internal structure of a data store, consider using an Entity Relationship Diagram instead.

Resources

1. Vehicle-Maintenance-Depot.vpp

Visual Paradigm home page
(https://www.visual-paradigm.com/)

Visual Paradigm tutorials
(https://www.visual-paradigm.com/tutorials/)