Data Flow Diagram with Examples - Supermarket App Example

Written Date : February 16, 2015

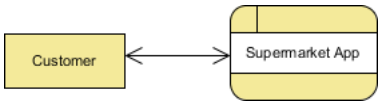
The Supermarket App Example

The data flow diagram is a hierarchy of diagrams that consist of:

1. Context Diagram (conceptually level zero)
2. The Level-1 DFD
3. And possible Level-2 DFD and further levels of functional decomposition, depending on the complexity of your system.

Context DFD

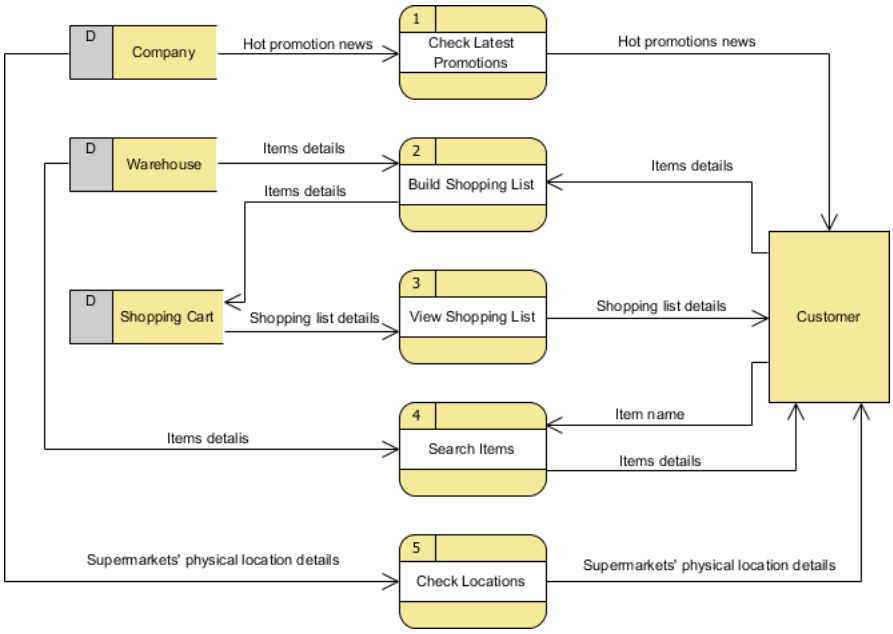
The figure below shows a context Data Flow Diagram that has been drawn for an Android supermarket app. It contains a process (shape) that represents the system to be modeled, in this case, the "*Supermarket App*." It also shows the participants who will interact with the system, called external entities. In this example, there is only one external entity, which is the *Customer*. In between the process and the external entity, there is a bi-directional connector, which indicates the existence of information exchange between the customer and the app, and the information flow is bi-directional.



A Context DFD is the entrance to a data flow model. It contains one and only one process and does not show any data stores, which makes the diagram simple.

Level 1 DFD

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



The Supermarket App Data Flow Diagram example contains five processes, one external entity, and three data stores. Although there is no design guideline that governs 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 a *Customer* can receive *Hot promotion news* from the *Check Latest Promotions* process, and the news is provided by the *Company* database. Note that by common sense, we know that *Check Latest Promotions* is likely to be a feature of the app, but the Data Flow Diagram itself implies no such thing. Theoretically speaking, a process in a Data Flow Diagram may correspond to a feature or a set of features.

A *Customer* can *Build Shopping List* by providing *Items details*, and the details will be stored in the *Shopping Cart* database. The *Warehouse* database will also provide the *Items details* required to complete the process.

A *Customer* can receive *Shopping list details* from the *View Shopping List* process, and such details are provided by the *Shopping Cart* database.

A *Customer* can receive *Items details* by performing the *Search Items* process. They must provide an *Item name* for searching, and the *item details* are returned from the *Warehouse* database after being searched. Although we said that the search result is returned after searching, the Data Flow Diagram, again, implies no such thing. It is our common sense that leads us to interpret the diagram in the way that we naturally understand it. Keep in mind that a Data Flow Diagram only tells you where information exchange takes place. It 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.

Finally, a *Customer* can receive *Supermarkets' physical location details* by performing *Check Locations*, and the details are provided by the *Company* database.

Data Flow Diagram Tips and Cautions

Be Aware of the Level of Detail

In this Data Flow Diagram example, the word "details" is used many times when labeling data. We have "item details," "shopping list details," and "location details." What if we were to write them out explicitly as "item ID," "item name, description, and photo," and "country, city, and address of supermarket"? 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," "information," and "credential" 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

Some designers may feel uncomfortable when they see a connector connecting from a data store to a process without the step of a data request being shown on the diagram. Some of them will try to represent a request by adding a connector between a process and a data store and labeling it "a request" or "request for something," which is surely unnecessary.

Keep in mind that a Data Flow Diagram is designed to represent the exchange of information. Connectors in a Data Flow Diagram 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 either an Activity Diagram or a BPMN Business Process Diagram instead. If you want to model the internal structure of a data store, you may use an Entity Relationship Diagram.

Resources

1. Supermarket-App.vpp

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

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