Data Flow Diagram with Examples - Customer Service System

Written Date : February 16, 2015

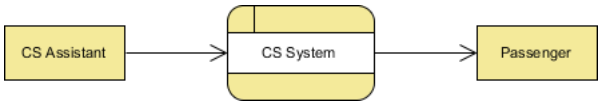
The CS System Example

The data flow diagram is a hierarchy of diagrams that consists 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

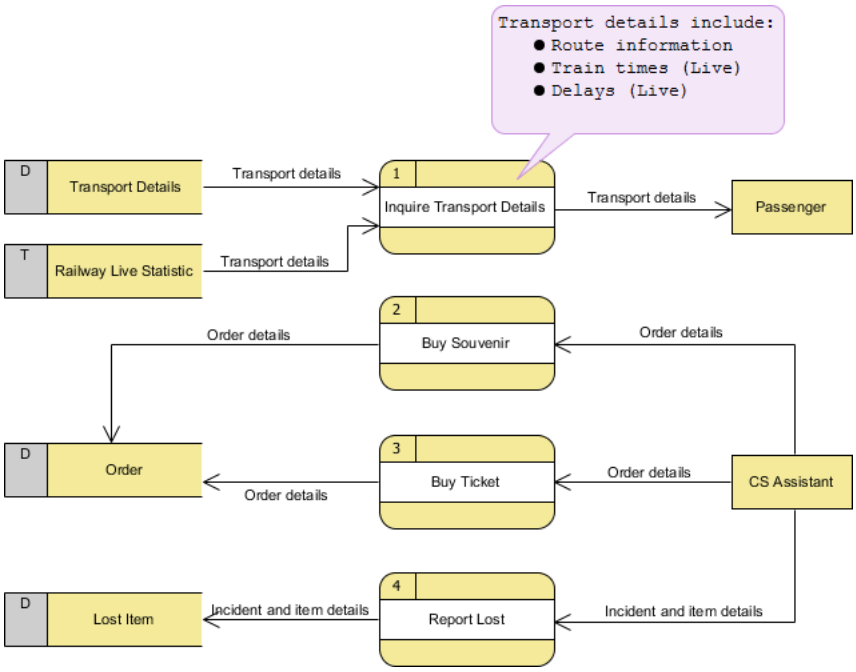
The figure below shows a context Data Flow Diagram that is drawn for a railway company's Customer Service System. It contains a process (shape) that represents the system to be modeled, in this case, the "*CS System*." It also shows the participants who will interact with the system, called the external entities. In this example, *CS Assistant* and *Passenger* are the two 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 CS System 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 CS System Data Flow Diagram example contains four processes, two external entities, and four 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 a *Passenger* can receive *Transport details* from the *Inquiry Transport Details* process, and the details are provided by the data stores *Transport Details* and *Railway Live Statistic*. While the data stored in *Transport Details* is persistent data (indicated by the label "D"), the data stored in *Railway Live Statistic* is transient data that is held for a short time (indicated by the label "T"). A callout shape is used to list the kind of details that can be inquired about by a passenger.

The *CS Assistant* can initiate the *Buy Souvenir* process, which will result in the *Order details* being stored in the *Order* data store. Although the customer is the real person who buys the souvenir, it is the *CS Assistant* who accesses the system to store the order details. Therefore, we make the data flow from the *CS Assistant* to the *Buy Souvenir* process.

The *CS Assistant* can also initiate the *Buy Ticket* process by providing *Order details*, and the details will be stored again in the *Order* data store. A Data Flow Diagram is a high-level diagram that is drawn with a high degree of abstraction. The "Order" data store drawn here does not necessarily imply a real order database or order table in a database. The way order details are stored physically is to be decided later on when implementing the system.

Finally, the *CS Assistant* can initiate the *Report Lost* process by providing the *Incident and item details*, and the information will be stored in the *Lost Item* database.

Data Flow Diagram Tips and Cautions

Stating the Type of Data with D, M, and T

Each data store drawn in a Data Flow Diagram is prefixed by a letter, which is 'D' by default. The letter indicates the kind of data the data store holds. The letter 'D' is used to represent persistent computerized data, which is probably the most common kind of data type in a typical information system. Besides computerized data, data can also be held for a short time in temporary storage. We call this kind of data transient data, and it is represented by the letter 'T'. Sometimes, data is stored without the use of a computer. We call this kind of data manual data, and it is represented by the letter 'M'. Finally, if the data is stored without using a computer and is also held for a short time, this is known as manual transient data and is represented by T(M).

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 "transport details" and "order details." What if we were to write them explicitly as "route information, train times, and delays," "souvenir name, quantity, and amount," and "ticket type and amount"? 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 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

Some designers may feel uncomfortable when they come across a connector connecting from a data store to a process without showing the step of a data request being specified on the diagram. Some designers will attempt to put a request attached to the connector between a process and a data store, labeling it "a request" or "request for something," which is surely unnecessary.

Keep in mind that the Data Flow Diagram was 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 "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. Customer-Service-System.vpp

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

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