Data Flow Diagram with Examples - Video Rental System Example

Written Date : February 16, 2015

The Video Rental System Example

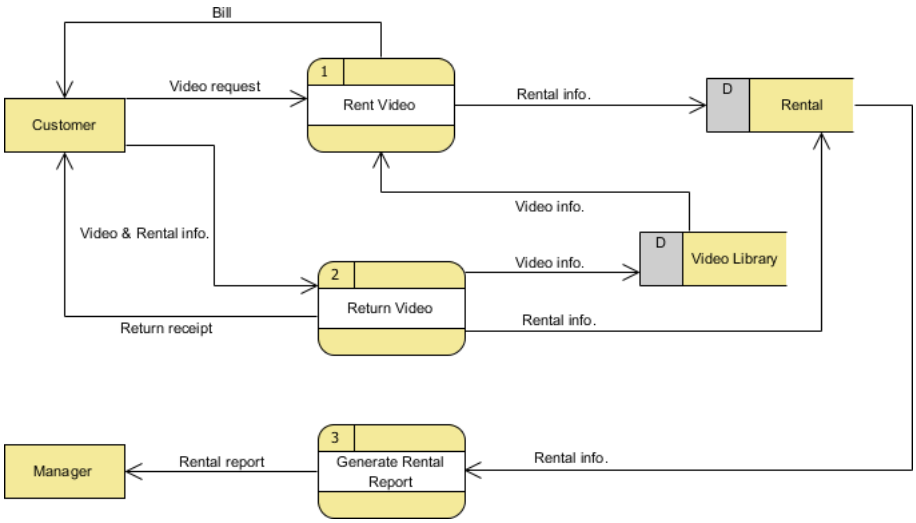
Context DFD

The figure below shows a context Data Flow Diagram that has been drawn for a video rental system. It contains a process (shape) that represents the system to be modeled, in this case, the "*Video Rental Store*." It also shows the participants who will interact with the system, called external entities. In this example, there are two external entities, namely *Customer* and *Manager*. In between the process and the external entities, there are data flow connectors that indicate the existence of information exchange between the customer and the system.

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 video rental system that is shown in the context DFD. Read through the diagram, and then we will introduce some of the key concepts based on this diagram.



The Video Rental System Data Flow Diagram example contains three processes, two external entities, and two 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* makes a *Video request* to the *Rent Video* process. The *Rent Video* process also receives *Video info.* from the *Video Library* data store. As a result, the process produces a *Bill* for the *Customer* and stores the *Rental info.* in the *Rental* data store.

A *Customer* can *Return a Video* by providing *Video & Rental info*. The process stores the *Video info.* in the *Video Library* data store and the *Rental info.* in the *Rental* data store. As a result, a *Return receipt* is delivered to the *Customer*. Although we said that the receipt is delivered as a result of the *Return Video* process, the Data Flow Diagram implies no such thing. It is our common sense that leads us to interpret the diagram in the way that we naturally understand it. Strictly speaking, the diagram only tells us that the *Return Video* process receives *Video & Rental info.* and produces *Video info.*, *Rental info.*, and a *Return receipt*, with no order specified. Note 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.

Finally, a *Manager* can receive a *Rental report* from the *Generate Rental Report* process, and the information involved is provided by the *Rental* data store.

Data Flow Diagram Tips and Cautions

Be Aware of the Level of Detail

In this Data Flow Diagram example, the word "info" is used many times when labeling data. We have "rental info" and "video info." What if we were to write them explicitly as "rental date, video rented, person renting," and "video id, video name, and video status"? 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 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 a UML Activity Diagram or a BPMN Business Process Diagram instead. If you want to model the internal structure of a data store, you can use an Entity Relationship Diagram.

Resources

1. Video-Rental-Store.vpp

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

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