Data Flow Diagram

Reference: <https://www.visual-paradigm.com/tutorials/data-flow-diagram-dfd.jsp>

# What is a data flow diagram (DFD)?

A picture is worth a thousand words**. A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system.** A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both.

It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

It is usually beginning with a context diagram as level 0 of the DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower-level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to levels 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that the level of detail for decomposing a particular function depending on the complexity that function.

A context diagram is a data flow diagram that only shows the top level, otherwise known as Level 0. At this level, there is only one visible process node that represents the functions of a complete system in regards to how it interacts with external entities. Some of the benefits of a Context Diagram are:

1. Shows the overview of the boundaries of a system
2. No technical knowledge is required to understand with the simple notation
3. Simple to draw, amend and elaborate as its limited notation

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

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 data stores and external entities on the sides to make it easier to comprehend.

## **DFD Diagram Notations**

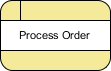
### **External Entity**

An external entity can represent a human, system or subsystem. It is where certain data comes from or goes to. It is external to the system we study, in terms of the business process. For this reason, people used to draw external entities on the edge of a diagram.

notation (external entity)

### **Process**

A process is a business activity or function where the manipulation and transformation of data take place. A process can be decomposed to a finer level of details, for representing how data is being processed within the process.



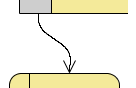
### **Data Store**

A data store represents the storage of persistent data required and/or produced by the process. Here are some examples of data stores: membership forms, database tables, etc.

notation (data store)

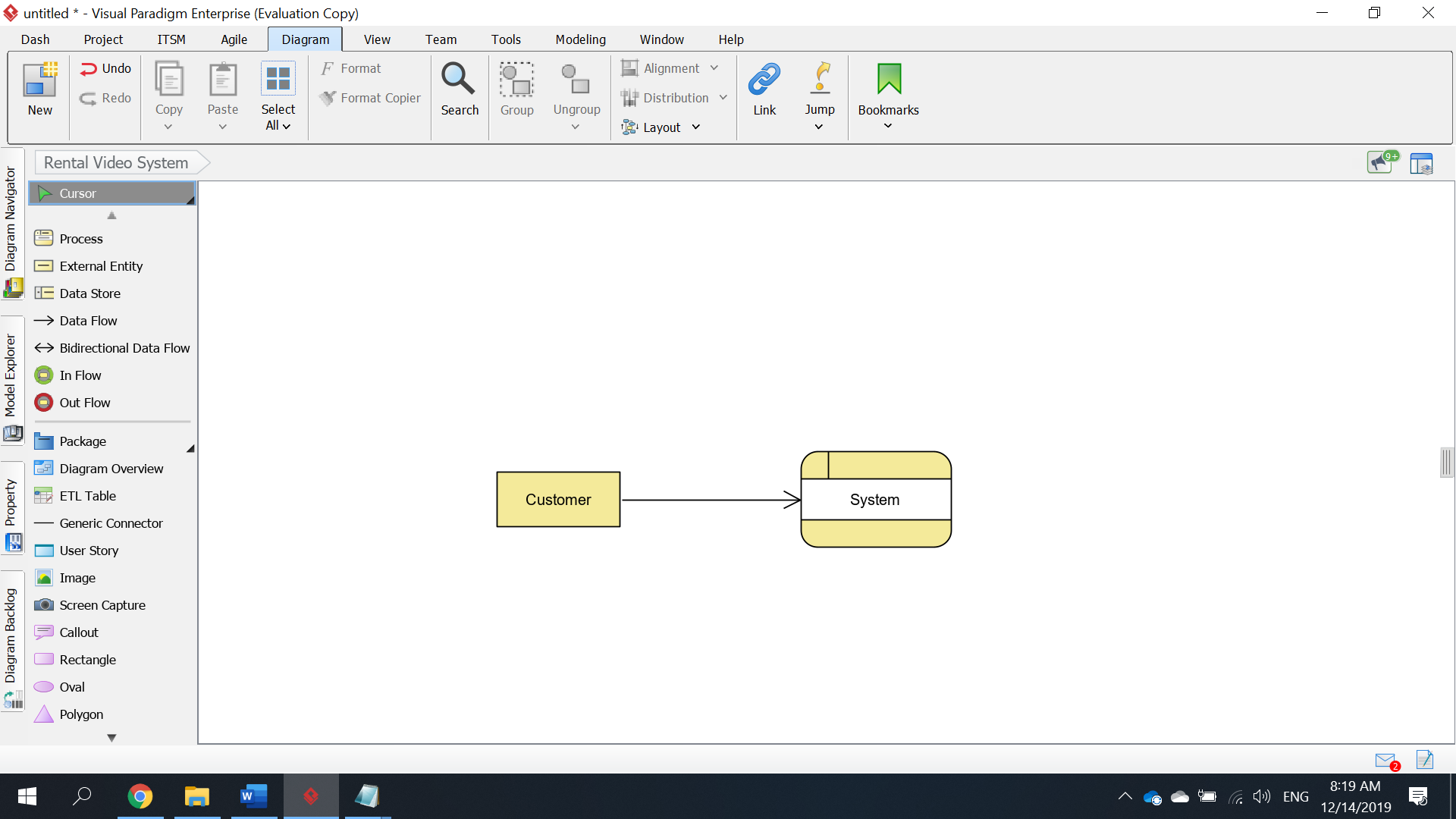
### **Data Flow**

A data flow represents the flow of information, with its direction represented by an arrowhead that shows at the end(s) of flow connector.

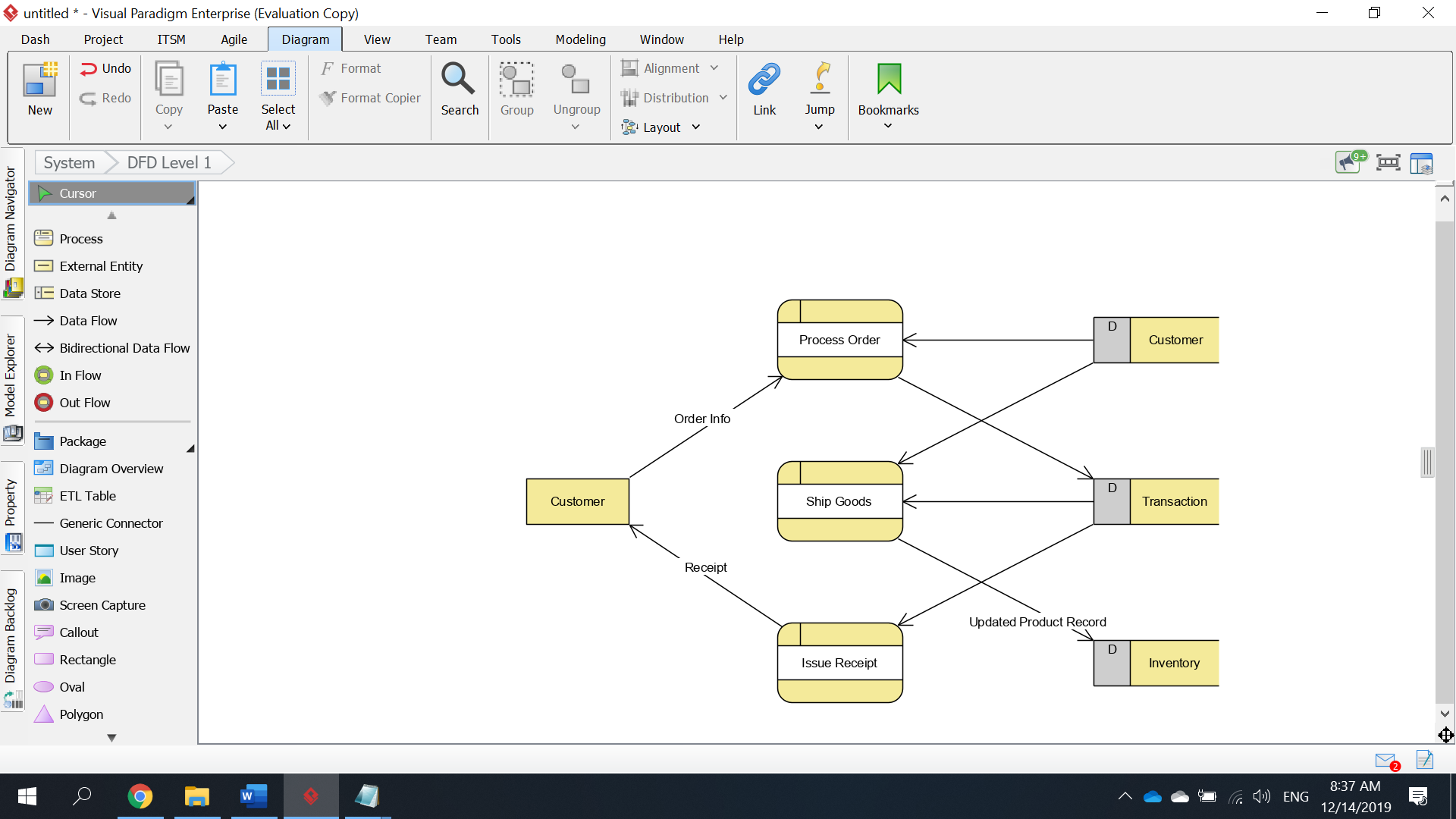


## Data Flow Diagram Example

### **Context DFD**



### **Level 1 DFD**



## **Data Flow Diagram Tips**

1. Process labels should be verb phrases; data stores are represented by nouns
2. A data store must be associated with at least a process
3. An external entity must be associated with at least a process
4. Don't let it get too complex; normally 5 - 7 average people can manage processes
5. DFD is non-deterministic - The numbering does not necessarily indicate sequence, it's useful in identifying the processes when discussing with users
6. Datastores should not be connected to an external entity, otherwise, it would mean that you're giving an external entity direct access to your data files
7. Data flows should not exist between 2 external entities without going through a process
8. A process that has inputs but without outputs is considered to be a black-hole process

## **Data Flow Diagram Cautions**

### **Stating the type of data with D, M and T, T(M)**

Each data store which is drawn in a Data Flow Diagram are 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 a 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. We call this kind of data **transient data** and is represented by letter 'T'. Sometimes, data is stored without the use of a computer. We call this kind of data **manual data** and is represented by letter 'M'. Finally, if the data is stored without using computer and also is held for a short time, this is known as manual transient data and is represented by T(M).

### **Using stereotype for modeling a "special kind of" entity**

Stereotype and tagged values are kind of extensibility mechanisms introduced by [Object Management Group (OMG)](http://www.omg.org/). It allows designers to extend the vocabulary of [UML](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/) in order to create new model elements. As a software design tool, Visual Paradigm extends the support of stereotype to non UML standards like DFD and ERD. Take the securities trading platform as example, we can define a stereotype Third Party for external entity. External entities with the stereotype assigned are said to be "a kind of third party entity".

### **Be aware of the level of details**

In this Data Flow Diagram example the word "details" is used many times when labeling data. We have "customer details", "transaction details", etc. What if we write them explicitly as "customer name, email address, job, address" and "stock number, amount, bid price"? 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", "information", "credential" certainly leave room for discussion. However, using general terms can be kind of lacking details and make the design lost its 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**

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

Keep in mind that 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, use [Entity Relationship Diagram](https://www.visual-paradigm.com/features/database-design-with-erd-tools/#erd).

### **Does not represent the order of data flow**

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 lead us to interpret the diagram in the way that we understand it naturally. Strictly speaking, the diagram only tells us the *Return Video* process receives *Video & Rental info.* and produces *Video info.*, *Rental info.*, and *Return receipt*, with no order specified. Note 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](https://www.visual-paradigm.com/features/bpmn-diagram-and-tools/#business-process-diagram) or [UML Activity Diagram](https://www.visual-paradigm.com/features/uml-tool/#activity-diagram).

Although we said that the attempt to store customer and account details happens after the details are being provided by the *Customer Service Assistant*, the Data Flow Diagram implies no such thing. It is our common sense that lead us to interpret the diagram in the way that we understand it naturally. Strictly speaking, the diagram only tells us the *Open Account* process receives *customer details* and produce customer and account details, with no order specified. Note 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](https://www.visual-paradigm.com/features/bpmn-diagram-and-tools/#business-process-diagram) or [UML Activity Diagram](https://www.visual-paradigm.com/features/uml-tool/#activity-diagram).

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 understand it naturally. Keep in mind that Data Flow Diagram only tells you where information exchange takes place. It 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](https://www.visual-paradigm.com/features/bpmn-diagram-and-tools/#business-process-diagram) or [UML Activity Diagram](https://www.visual-paradigm.com/features/uml-tool/#activity-diagram).

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](https://www.visual-paradigm.com/features/bpmn-diagram-and-tools/#business-process-diagram) or [UML Activity Diagram](https://www.visual-paradigm.com/features/uml-tool/#activity-diagram).

# More DFD Examples

## Video Rental System Example

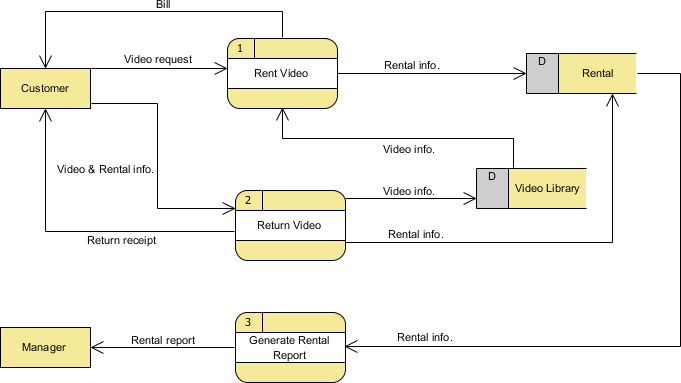
#### **Context DFD**

The figure below shows a context Data Flow Diagram that is drawn for a video rental system. It contains a process (shape) that represents the system to model, in this case, the "*Video Rental Store*". It also shows the participants who will interact with the system, called the 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 indicating the existence of information exchange between customer and the system.



#### **Level 1 DFD**

The figure below shows the level 1 DFD, which is the decomposition (i.e. break down) of the video rental system that is shown in the context DFD.



The Video Rental System Data Flow Diagram example contains three processes, two external entities and two data stores.

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* to the *Customer*, and stores the *Rental info.* into the *Rental* data store.

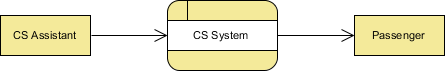
A *Customer* can *Return Video* by providing *Video & Rental info*. The process stores the *Video info.* into the *Video Library* data store and *Rental info.* into the *Rental* data store. As a result, *Return receipt* is delivered to the *Customer*.

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

## Customer Service System Example

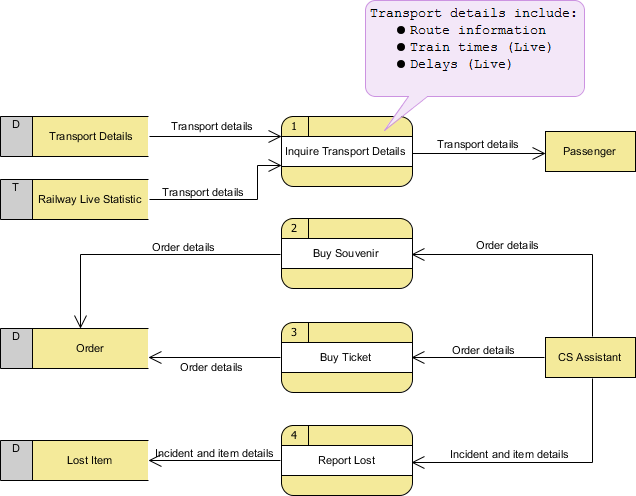
#### **Context DFD**

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 model, 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 flow (connectors) that indicate the existence of information exchange between the entities and the system.



#### **Level 1 DFD**

The figure below shows the level 1 DFD, which is the decomposition (i.e. break down) of the CS System process shown in the context DFD



The CS System Data Flow Diagram example contains four processes, two external entities and four data stores.

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 data stored in *Transport Details* are persistent data (indicated by the label "D"), data stored in *Railway Live Statistic* are transient data that are held for a short time (indicated by the label "T"). A callout shape is used to list out the kind of details that can be inquired by passenger.

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

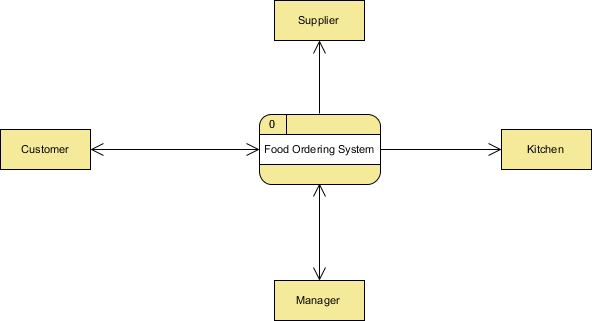
*CS Assistant* can also initiate the *Buy Ticket* process by providing *Order details* and the details will be stored again in the *Order* data store. Data Flow Diagram is a high level diagram that is drawn with a high degree of abstraction. The data store Order which is drawn here does not necessarily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

Finally, *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.

## Food Ordering System Example

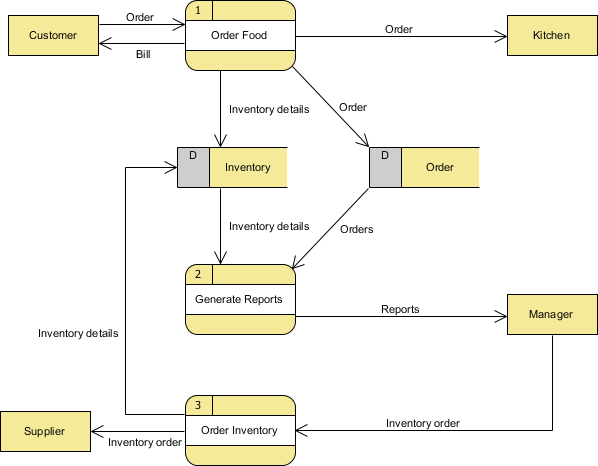
#### **Context DFD**

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



#### **Level 1 DFD**

The figure below shows the level 1 DFD, which is the decomposition (i.e. break down) of the Food Ordering System process shown in the context DFD.



The Food Order System Data Flow Diagram example contains three processes, four external entities, and two data stores.

Based on the diagram, we know that a *Customer* can place an *Order*. The *Order Food* process receives the *Order*, forwards it to the *Kitchen*, store it in the *Order* data store, and store the updated *Inventory details* in the *Inventory* data store. The process also delivers a *Bill* to the *Customer*.

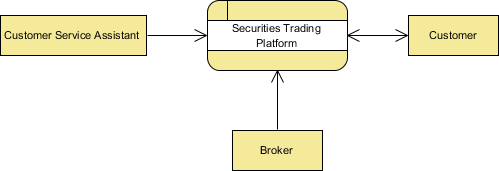
The *Manager* can receive *Reports* through the *Generate Reports* process, which takes *Inventory details* and *Orders* as input from the *Inventory* and *Order* data store respectively.

The *Manager* can also initiate the *Order Inventory* process by providing *Inventory order*. The process forwards the *Inventory order* to the *Supplier* and stores the updated *Inventory details* in the *Inventory* data store.

## Securities Trading Example

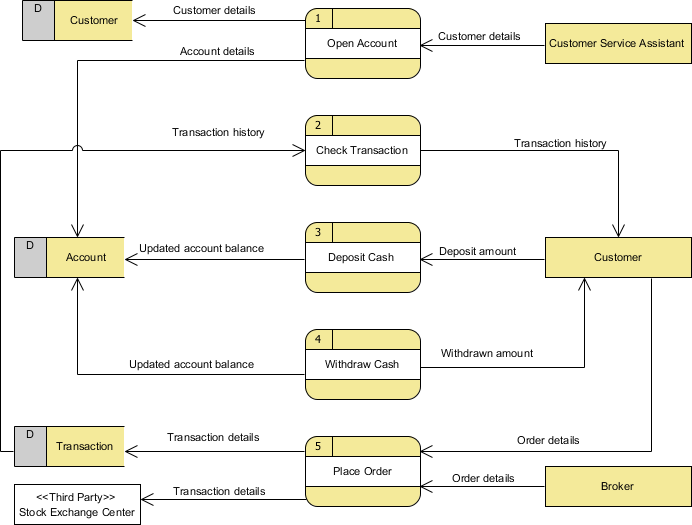
#### **Context DFD**

The figure below shows a context Data Flow Diagram that is drawn for a security trading platform. It contains a process (shape) that represents the system to model, in this case, the "*securities trading platform*". It also shows the participants who will interact with the system, called the external entities. In this example, *CS Assistant*, *Customer* and *Broker* 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.



#### **Level 1 DFD**

The figure below shows the level 1 DFD, which is the decomposition (i.e. break down) of the securities trading platform process shown in the context DFD.



The securities trading platform Data Flow Diagram example contains five processes, three external entities and three data stores.

Based on the diagram, we know that a *Customer Service Assistant* provides customer details to the *Open Account* process. The result is the *Customer details* being stored in *Customer* data store and *Account details* being stored in *Account* data store.

The process *Check Transaction* receives *Transaction details* from the *Transaction* data store and pass it on to *Customer*.

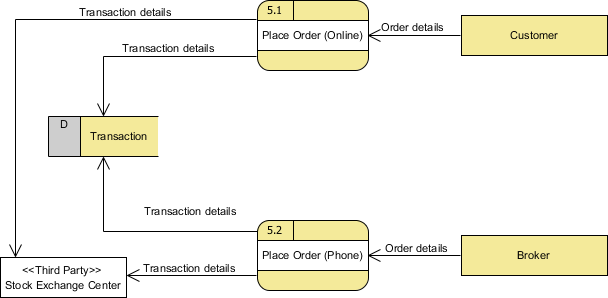
A *Customer* can *Deposit Cash* by providing the *Deposit amount* and the result is the *Updated account balance* being stored in the *Account* data store.

Similarly, a *Customer* can *Withdraw Cash*. The result is that he will receive the *Withdrawn amount* and the *Updated account balance* will be stored in the *Account* data store.

Finally, both the *Customer* and *Broker* can initiate the *Place Order* process, which results in the *Transaction details* being stored in the *Transaction* data store. The *Place Order* process also passes the *Transaction details* to the *Stock Exchange Center*, which is an entity out of the system scope. In the next section, we will introduce a way to represent this kind of entity.

#### **Level 2 DFD**

Just like the process in context DFD, processes in level 1 DFD can also be decomposed into a deeper level or even levels of process details.



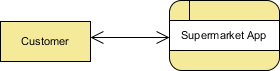
The external entities and data stores in this DFD correspond with those shown in the upper level (i.e. the diagram above). What makes it different is the breakdown of *Place Order* process into *Place Order (Online)* process and *Place Order (Offline)* process.

Based on this diagram, we know that a *Customer* can perform *Place Order (Online)* by supplying *Order details* while a *Broker* can perform *Place Order (Phone)* also by supplying *Order details*; in either case causing *Transaction* details to be stored in the *Transaction* data store and passed to the *Stock Exchange Center*.

## Supermarket App Example

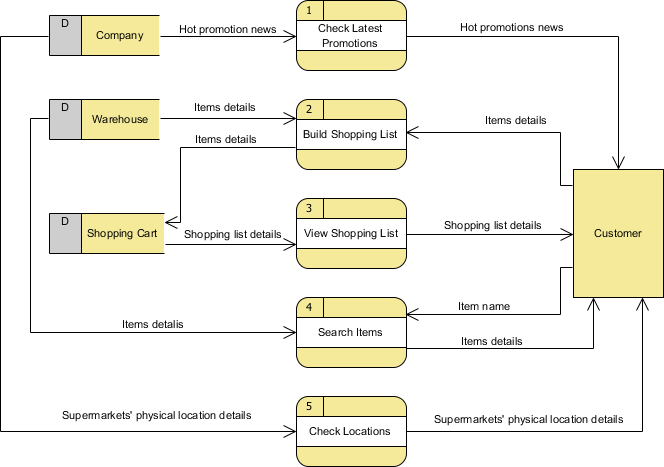
#### **Context DFD**

The figure below shows a context Data Flow Diagram that is drawn for an Android supermarket app. It contains a process (shape) that represents the system to model, in this case, the "*Supermarket App*". It also shows the participants who will interact with the system, called the 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 customer and the app, and the information flow is bi-directional.



#### **Level 1 DFD**

The figure below shows the level 1 DFD, which is the decomposition (i.e. break down) of the Supermarket App process that is shown in the context DFD.



The Supermarket App Data Flow Diagram example contains five processes, one external entity and three data stores.

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 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 is provided by the *Shopping Cart* database.

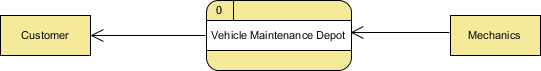
A *Customer* can receive *Items details* by performing the *Search Items* process. He/she must provide an *Item name* for searching and the *item details* are returned from the *Warehouse* database after searched.

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

## Vehicle Maintenance Depot Example

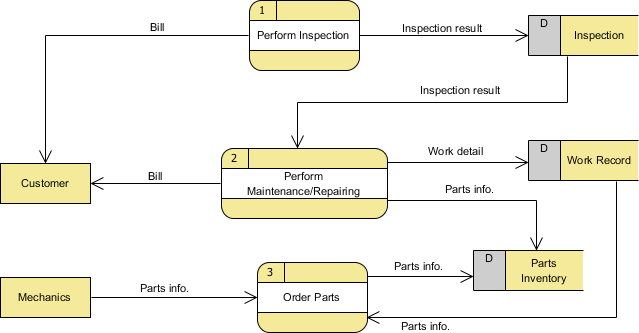
#### **Context DFD**

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.



#### **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.



The Data Flow Diagram example contains three processes, two external entities and three data stores.

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.

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.