UNIT 8 FUNCTIONAL MODELING

Structure **Page No.**

8.0 INTRODUCTION

Object Modeling Technique (OMT) is a very popular and established approach for the development of object-oriented systems. OMT model deploys many visualization techniques to explain the working of complex systems in a simple manner. OMT finds application in real-world application domains like the field of medical and health care, telecommunications, transportation and tourism, online shopping etc.

The basis of the OMT model is three main types of models as proposed by James Rambaugh. The first is Object Model, which is based on classes and objects and the features like encapsulation, inheritance, concurrency and abstraction. Next is Dynamic, which depicts states of objects, transition between states and events that trigger the transition from one state to another. The third and final model is the Functional model, which elaborates on concepts like how data flows between various objects called actors, processes, and data stores. This unit focuses on the functional model and all its features explained with the help of data flow diagrams up to level 3 decomposition.

8.1 OBJECTIVES

After going through this unit, you should be able to:

- Explain the functional model,
- Comprehend Data Flow Diagram(DFD) and its purpose in OMT,
- Describe the features of DFD and various symbols used in it,
- Explain the common design flaws in making a DFD,
- Describe the working of a Sample Functional Model (DFD up to 3rd Level) , and
- Describe the relation of functional model to object and dynamic model.

8.2 FUNCTIONAL MODELS

A functional model is a graphical representation of the object-oriented analysis model and focuses on various processes to be performed on input data to obtain the desired output. Here, the system is viewed as a black box that accepts inputs, transforms them by applying various processes/functions and then produces the outputs. A functional model does not show how the transformations occur, when they are performed and why there is a need to do the calculations.

The input to the system can be numbers typed by an operator or control signals produced by a sensor, or some big data files received over a network. The processes that work on the inputs may be performing some arithmetic or logical calculations or applying some algorithm to generate one or more outputs in the form of some numerical value or some control signals that may be used to drive other devices. We use data flow diagrams (DFD) to explain the flow of data from input to output. We can also say that by using the functional model, the Software Requirements Specification (SRS) document is converted to the Data Flow Diagram (DFD) model.

8.3 DATA FLOW DIAGRAM

The DFD is a simple graphical technique that represents an entire software system in terms of inputs fed to it, processing applied on these inputs and the outputs produced by the system, and various data stores used internally to store the data/results. The focus is on the flow of data input to a system by a source, transformed by the application of various processes and then sent to a destination as output. The sources that produce data and the destination that consumes the processed data are known as Actor objects.

It is very easy to understand the functionality of a system with DFD as there are very few symbols used in it, and the basis of DFD is to divide a complex problem into smaller components. Also, it is said that human intellect works on the fundamental of the division of complex concepts into smaller and greater numbers of concepts to make the understanding easier. Hence, DFD uses a hierarchical approach that starts from an abstract model that depicts high-level functions and further introduces details as we go down the hierarchy and create sub-functions. DFD is also called as a bubble chart.

8.4 FEATURES OF A DFD

The features of DFD can be explained with the help of primitive symbols used. There are five different symbols used to draw a DFD, as shown in Figure 8.1.

1. **Circle**: Labelled circles called bubbles are used to show the processing or transformations applied on data. The name of the corresponding function is written inside the bubble.
2. **Arrow:** A labelled arrow is used to represent information and/or data flow occurring between two processes or between an external entity and a process.
3. **Rectangle:** It is used to show an actor who communicates with the software system. The actor may give input (source) to the system or consume the output produced (sink). An actor may be a human being e.g. accountant, or any hardware, maybe a sensor that is feeding signal as input to a process.

4. **Two parallel lines or rounded rectangle**: It is used to represent a Data Store whose name is written between these two parallel lines. The data store is connected to a process that uses it.
5. **A rectangle with the right top corner chopped off** represents the output in the form of a hard copy.

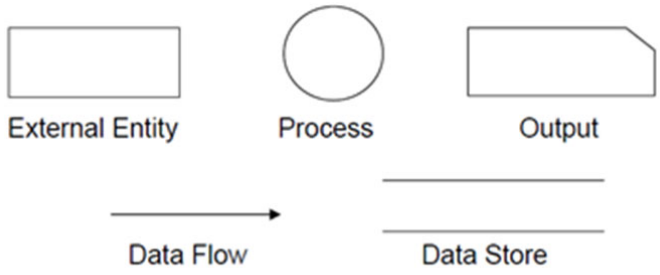

Figure 8.1: Symbols used to draw DFD

Now let us discuss all components of DFD in detail.

8.4.1 Processes

A circle represents a process and the name of function performed by it is written inside the circle. Process denotes a function that is to be performed on one or more inputs to produce one or more outputs. The purpose of the process is to transform data. An example of a process can be Calculate_sum. This process is given three integers as input, and it produces the result as a sum (an integer), as shown in Figure 8.2. Another example is shown in Figure 8.3 where a process Perform_division accepts the dividend and divisor as input and produces quotient and remainder as outputs.

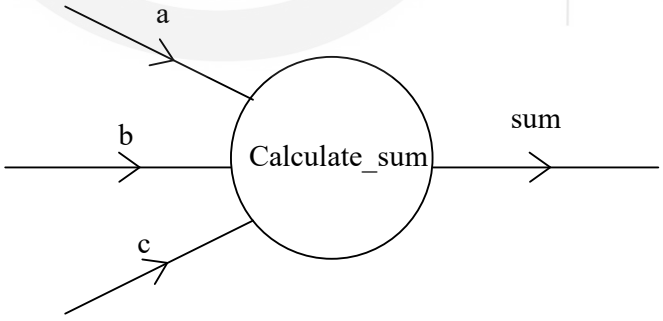

Figure 8.2: Process Calculate_Sum

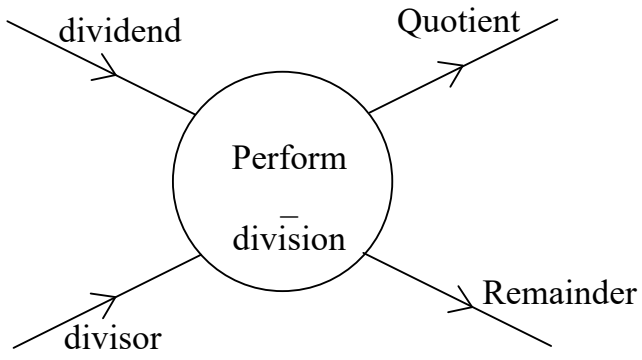

Figure 8.3: Process Perform-division

A process may get input from an external entity called an actor (as shown in Figure 8.4) or from another process (i.e. the output of one process may be input to another process). The number of inputs and outputs for a process is fixed. There may be some non-functional components that provide inputs to a process. For example, a data store may be read to collect input from a file residing on it. Under such circumstances, the process may have side effects because of which the functional model can not provide results of the process. Also, in this situation, the results of the process will depend upon the behaviour of the software system, as explained by the dynamic model.

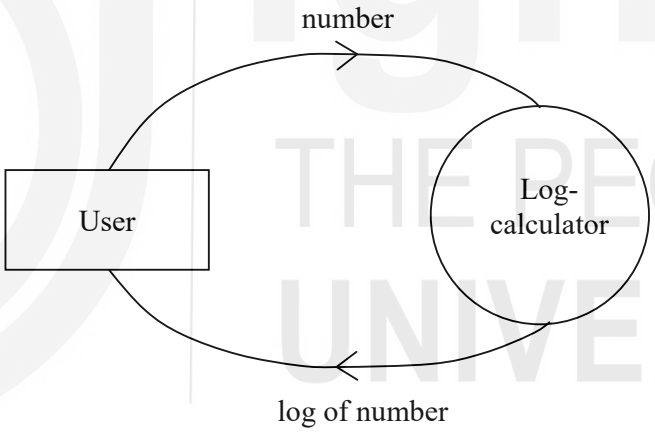

Figure 8.4: Process accepting input from an actor

8.4.2 Data Flows

It denotes the flow of data between any two processes or between a process and an actor, or between a process and a data store. It is represented by an arrow or a directed arc annotated with the name of the data item that it carries at that point in time. Data flow does not modify the content that it carries.

The data item carried by the data flow arrow may be sent to more than one destination. There are two ways in which the data value is forked.

1. A single same value can be sent to more than one place as indicated by a fork with 3 outputs, where each of the arrows would be connected to a different process. As shown in Figure 8.5, the value square_root can be sent to three different places. In this case, there is no need to give a label as the value remains the same as the input.

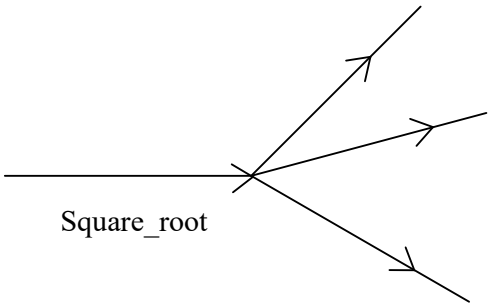

Figure 8.5: Single value sent to three different places

2. An aggregate value can be split into many components. Each of the component is sent to a different process or actor. A fork has depicted this with three arrows coming out of it. In this case, label is written on each arrow, as shown in Figure 8.6

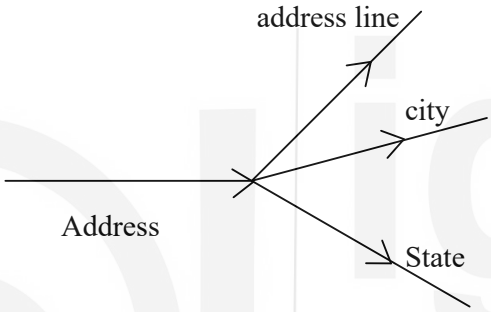

Figure 8.6: Aggregate value Address split in three components

SYNCHRONOUS VS ASYNCHRONOUS DATA FLOW

The DFD in Figure 8.7 shows three data flows—the data item integer flowing into read_number process, the data item named integer flowing from the read_number process to find_modulus process, data item named Modulus_number flowing out from the process find_modulus.

In order to explain this concept, we have shown a part of a software system that reads a number and then finds its modulus. In the case of Synchronous data flow, the output produced by process "read_number" is immediately used by the process "find_modulus", a data flow arrow that directly connects both processes. Hence the speed of operation of both the processes is the same. This is depicted in Figure 8.7.

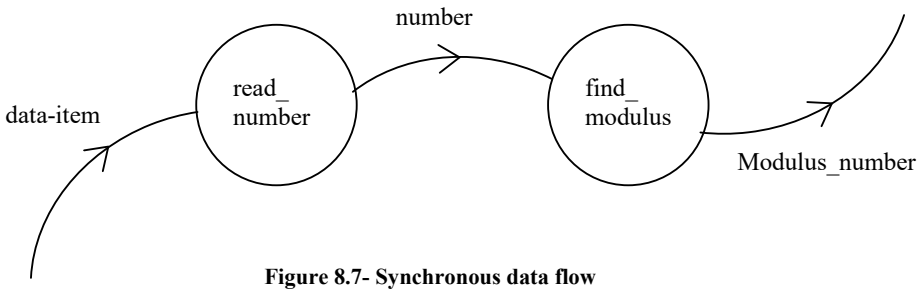

Figure 8.7- Synchronous data flow

In the case of Asynchronous data flow, as shown in Figure 8.8, there sits a data source between these two processes. So, the first process produces the data stored in the data

source. Later on, another process can use this as we can see that the two processes are independent of each other. This type of data flow is called asynchronous.

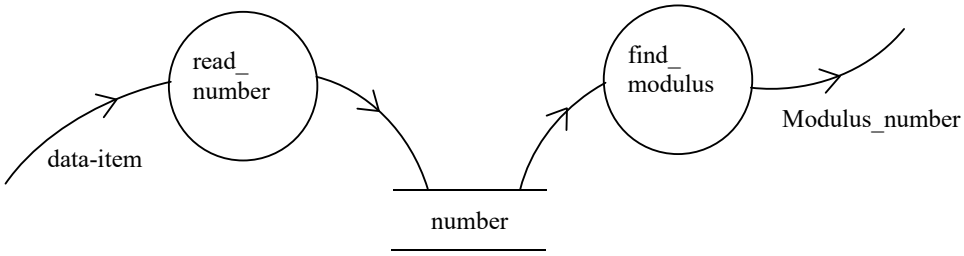

Figure 8.8- Asynchronous data flow

8.4.3 Actors

An actor is represented using a rectangle. An actor is an external entity that produces the data for the system or consumes the information produced by the system. Actors lie at the boundary of the system. They are also called terminators because the data flow is terminated at the input as a source or output as a sink of the system.
We can also say that actors are those that perform some action, for example, a customer of a sales system or a sensor in a temperature management system. The actions performed by actors are explained in the dynamic model in detail, and with reference to DFD, the actor's role is confined only to providing input to system or accepting output produced by the system. The symbol used to represent an actor is a rectangle. An example of two actors, namely Employee and Accountant, interacting with the Payroll_Processing system is shown in Figure 8.9.

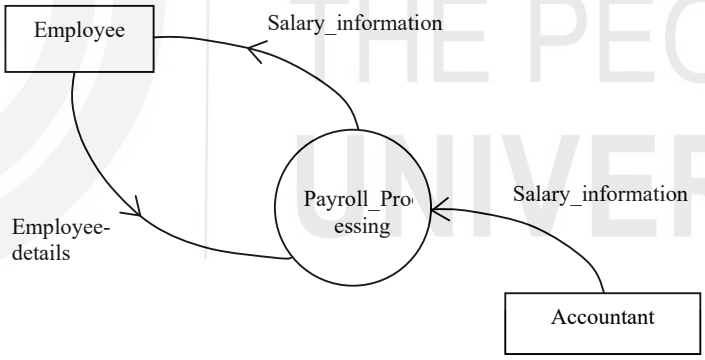

Figure 8.9: Actors interacting with a system

8.4.4 Data Store

A data store is a passive object that represents a repository of data in the software system and is represented using two parallel lines. A data store symbol represents either a data structure or a physical file residing on the disk. We need a data store when the system must store the data for future use by a process. Each data store is connected to one or more processes. Since the data store is a passive object, it does not generate any operation and is used only for reading or writing data into it. Data can be written/modified into the data store using the input arrow, or data may be retrieved from the data store represented using the output arrow. The output arrow is annotated with the name of content retrieved from the data store; if unlabeled, it

denotes full data retrieval. If the arrow is two-way, it represents both storage and retrieval.

Examples of data stores are inventory, library bookstores, payments databases, etc. An example of a Library_Database is shown in Figure 8.10. Another data store named Sales_Database that stores sales details like item name, quantity, cost, date, etc., is being used by the process Find_Average_Sales to retrieve the average value of sales in Figure 8.11.

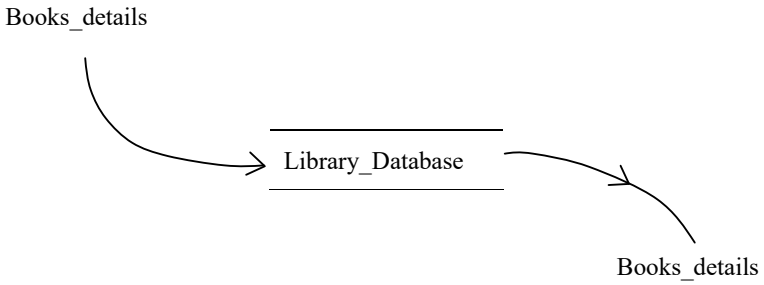

Figure 8.10: Library data store

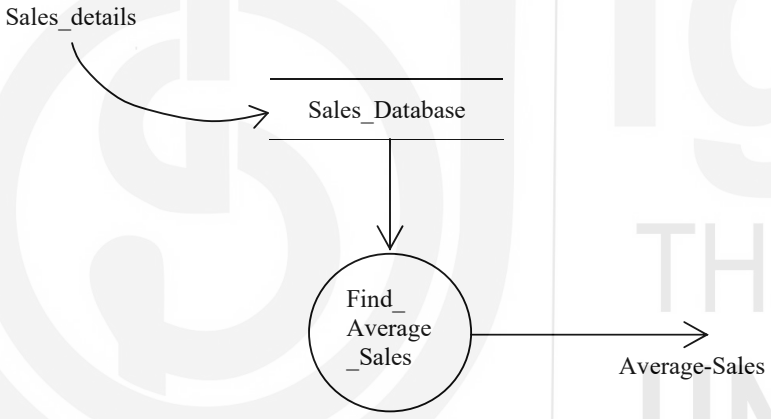

Figure 8.11: Sales-database used by process Find_Average_Sales

8.4.5 Output Symbol

Whenever a hardcopy is generated, we use the output symbol which is a rectangle with the top right corner chopped off.

8.4.6 Constraints

A constraint represents the relationship between two different objects at the same time or between different values of the same object at different times. Constraints mean the conditions or limitations required to be fulfilled over a period of time. One can add new principles or modify the existing ones with the help of constraints.
All models of object-oriented analysis can have constraints in them, as explained below: -.

1. Object Modelling: Here, constraints show the relationship between objects. In this type of modelling, constraints represent the relationship between the different values that an object may take at different times.
2. Dynamic Modelling: Here, the constraints define the relationship between the states and events of different objects.
3. Functional Modelling: In this, the constraints define the limitations on the transformations and computations that can be done on objects.

A constraint between values of an object over time is often called an invariant. For example, in Physics, when a vector is rotated, its length does not change, so the length of a vector is invariant during rotation. The behaviour of operations can be explained with the help of invariants. A constraint is represented in DFD as a string written inside braces.

Figure 8.12 shows a DFD in which, depending on the department of an employee, either he is given an incentive or an increment. As it is visible, if a department is Sales, then the incentive is calculated, and if the department is HR, then an increment is calculated.

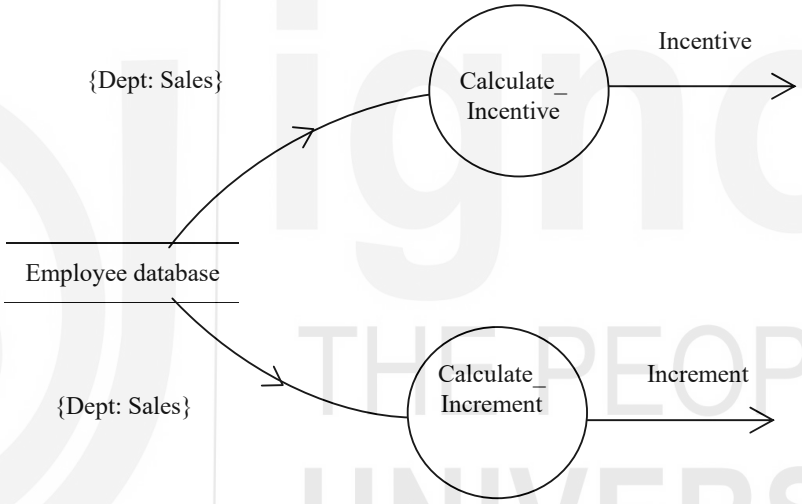

Figure 8.12: Constraints in DFD

8.4.7 Control Flows

A control flow signal is generated by a process so as to inform other processes to start or stop its processing. This is typically a Boolean value passed to another process on the basis of which the other process decides its course of action. It is denoted by a dotted line.

An example of control flow is shown in Figure 8.13. The customer enters the password in a net banking website, which is verified. Only after that, he can transfer money to somebody else's account.

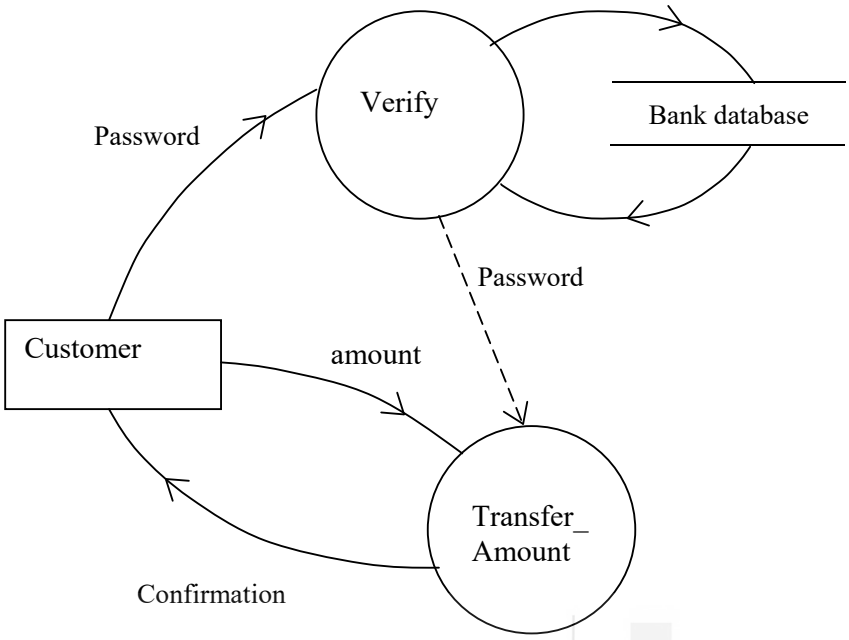

Figure 8.13: Control How to inform Transfer_amount process to start processing

8.5 DESIGN FLAWS IN DFD

There are certain limitations of the DFD, which may lead to a few design flaws. The most disturbing limitation is that with DFD, we cannot visualize the complete picture of the software system and hence some of the essential physical entities are left out. Also, at times, it can be tough to conceptualize an object-oriented structure in a hierarchical format.

There are different categories of design flaws:

1. Flaws committed by beginners:
    - The context diagram must have just one bubble that represents the complete software system at an abstract level and has arrows of input and output interactions corresponding to data flows with the world. Many designers draw more than one bubble in the context diagram, which must be avoided.
    - External entities are drawn only in the context diagram. However, many designers represent the external entities at all levels of DFD, which is not correct.
    - Each process/ bubble should be decomposed into between 3 to 7 bubbles, but sometimes designers do not follow this principle. Different levels of DFD must be balanced. Balancing of DFD means that the data flow into or out of a process symbol must match the flow at the next higher Level of the DFD.
2. Representing control information in a DFD:

    One must clearly understand that DFD is not a flow chart. DFDs cannot capture the following important features (When, Why and How): -
    - When: As the Time-dependent behaviour of a software system cannot be shown in a DFD, therefore the transformations are done by processes that cannot be shown in a DFD.
    - Why: DFDs cannot specify why the value of an object has changed.

- How: DFDs cannot mention the frequency of transformations done on an object.
- If a bubble X invokes either the bubble Y or the bubble Z depending upon some conditions, we need only to represent the data that flows between bubbles X and Y or bubbles X and Z and not the conditions depending on which the two modules are invoked.

3. Mistakes arising out of poor Diagramming approach: Black holes, miracles and grey holes.
   Such design flaws occur when the outputs from a process do not match its inputs. A few examples of this design flaw are listed below:
   - **Black hole:** If a processing step has just input flows, but no output flows.
   - **Miracle:** If a processing step has just output flows, but no input flows.
   - **Grey hole:** As the name suggests that it is a confusing situation. Such a situation may arise when a processing step has outputs that are greater than the sum of its inputs - e.g., its inputs could not produce the output shown.
4. **Illegal data flows with data store:** It is very important to realize that a store can only be connected to bubbles using data arrows. A data store can never be connected to another data store or to an external entity.

☞ Check Your progress-1

1) What is the Functional model? Why is it required in Object-oriented analysis and design?

.....

.....

.....

.....

2) What do the data stores represent in a DFD?

.....

.....

.....

.....

3) What do you understand by Data Flow Diagram? Explain with a diagram.

.....

.....

.....

.....

4) Differentiate between Synchronous and Asynchronous data flow.

.....

.....

.....

.....

5) What do you understand by "balancing a DFD"? Explain.

.....

.....

.....

.....

6) Explain the design flaws arising out of a poor Diagramming approach.

.....

.....

.....

.....

8.6 A SAMPLE FUNCTIONAL MODEL (EXAMPLE DFD UPTO 3rd LEVEL)

Before you learn to draw a DFD for a complex system, a few important guidelines for drawing a functional diagram are mentioned below:

1. Every process must have a meaningful name which is unique and a unique number as its identifier.
2. The Level-0 DFD, also called a Context Diagram, must have just one process.
3. Every data flow arrow must be named.
4. A bubble should be decomposed into a minimum of three and a maximum of seven bubbles in further level. Hence, each DFD can have no more than seven processes.
5. A process can be connected only with another process, external entity and data store.
6. No two external entities can be connected directly with each other.
7. No loop is allowed in a DFD.
8. There should be logical consistency in all levels of DFD.
9. The data flow of DFD should be easy to understand.
10. We should try to use arrow's direction to depict the order of events. For explaining the order of occurrence of the event, a flow chart should be used.

11. The numbering of the bubbles is to be done very carefully. Each bubble is given a unique identification number. The bubble in the context diagram that is Level- 0 is given "0" number. Bubbles in level-1 are numbered like 0.1, 0.2, 0.3 and so on. Bubbles in level-2 that are formed after factoring/ decomposing each of the bubbles of level 1. Say 0.1 bubble (of level 1) is factored, its children are numbered as 0.1.1, 0.1.2, 0.1.3, 0.1.4 and so on. Hence, we can say if a bubble having the number "x" is exploded, its children (that is, bubbles at the next Level of DFD) are numbered as x.1, x.2, x.3 and so on.
    So, suppose a bubble has number 0.4.5, it means it belongs to Level- 2 (Level is equal to the number of dots), and it is 5^{th} bubble after decomposing 0.4^{th} bubble (which was at level- 1). That means every bubble number will start with "0".

A complete working example of DFD up to level-3 is has been explained here. In order to represent the DFD of a system, first of all level-0 DFD, also called as Context diagram is drawn. It represents the complete software system at an abstract level by showing just one bubble having arrows of input and output interactions, corresponding to data flows, with the world.

The bubble is labelled as per the main context of the system, which is usually written using a noun. The data flow arrows are annotated corresponding to various data items that would be either supplied to system or generated from the system. This bubble is numbered as "0".

Functional Model for Online Shopping System

We have taken an example of DFDs up to level-3 for a website for Online Shopping System (OSS) for buying products of various categories viz. apparel,food and beverages, electronic items etc. There are two types of users, also called actors, who interact with this OSS. These are the Administrator and the customer. The Administrator is the sole incharge of the website and is responsible for all features and facilities offered by the site.

Level- 0 DFD

Only the Login request and response for customer and Administrator are depicted in this. The name of the process is "Online Shopping system" which is a noun. The context diagram is shown in Figure 8.14.

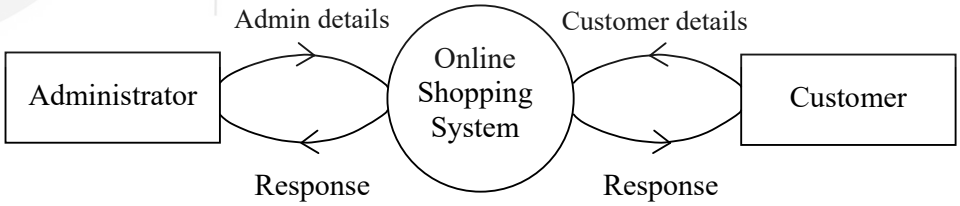

Figure 8.14: Context diagram for Online Shopping System

Level- 1 DFD

The context diagram is further decomposed in a hierarchical way to explain the system's entire working. This is done by converting the Level-0 bubble into three to seven bubbles which generally have labels denoting the functionality using verbs. This produces level-1 DFD.

Since there are many functions pertaining to the customer side and administrator side, the Level- 1 DFD for both is shown separately.

Level- 1 DFD : Customer side

The customer can interact with the OSS in one or more of the following ways.

1. Registration
2. Customer Login
3. Manage account
4. Purchase items
5. Make payment
6. Return item

The Level- 1 DFD is shown in Figure 8.15.

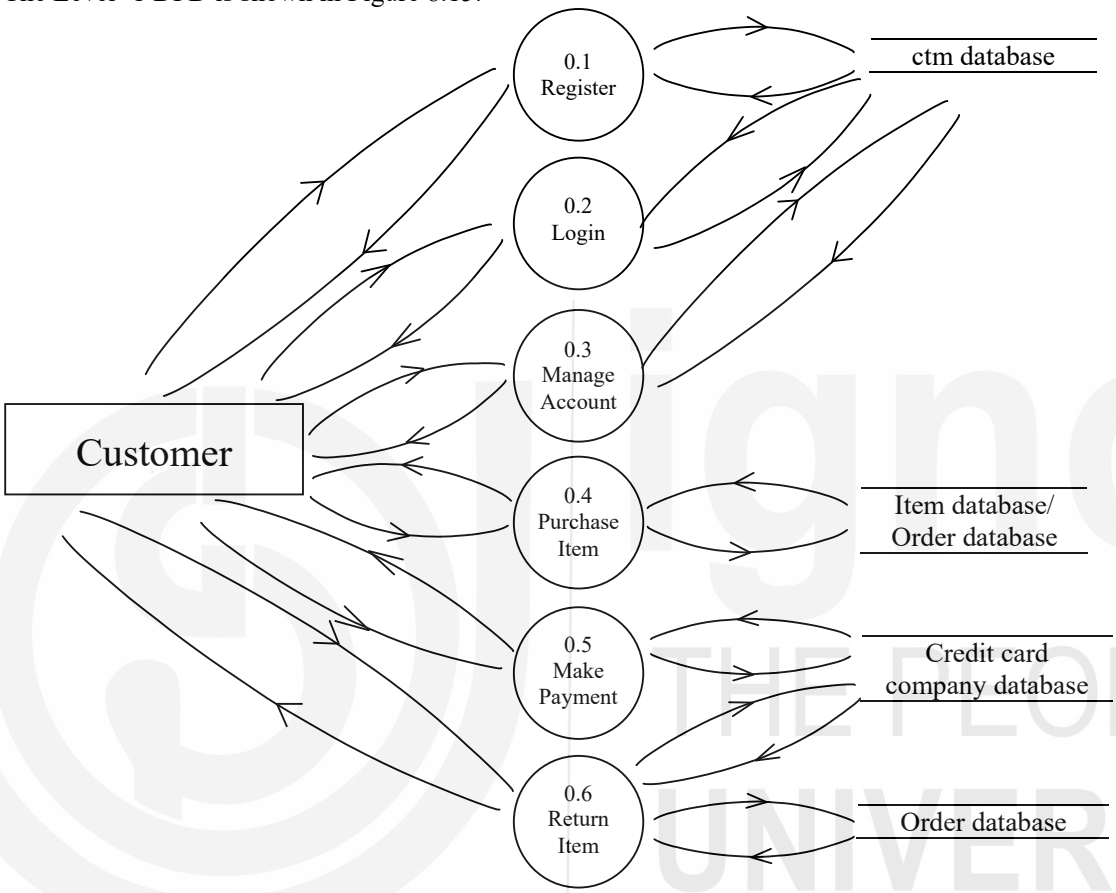

Figure 8.15: Level 1 DFD for costumer side

In Level- 1 DFD, we have shown all the databases that are used by various processes. As we can see, the process module's numbering is of type 0.x where 0 denotes that this process's parent is level 0. The DFD that is 0.x means this process module belongs to Level- 1, x denotes the unique serial number in a DFD.

The process modules except Register, Login and Return items are complex in nature, so they will be further decomposed into sub-processes in the next higher Level DFDs. Hence, they are not explained here.

A new customer uses the Register module to register himself/herself by giving the email id, username, password and mobile number for verification with OTP (one-time-password).

These are stored in the customer database. Only after successful registration the customer can login. While logging in, the credentials (username and password) are validated by checking the customer database. Once it is validated, the customer can Manage accounts, Purchase items, Make Payments and Return items if required.

As we know, using DFD, the "when" part or the control flow cannot be explained, so there is not any timing information.

Level- 1 DFD: Administrator side

The Administrator is the only person managing the addition of new items, managing orders received by the company, taking care of deliveries, checking whether payments are received or not, and generating various reports. The functions that the Administrator can perform are listed here.

1. Admin Login
2. Manage category
3. Manage item
4. Manage order
5. Manage report

Various databases that are used by each process module are shown in Figure 8.16.

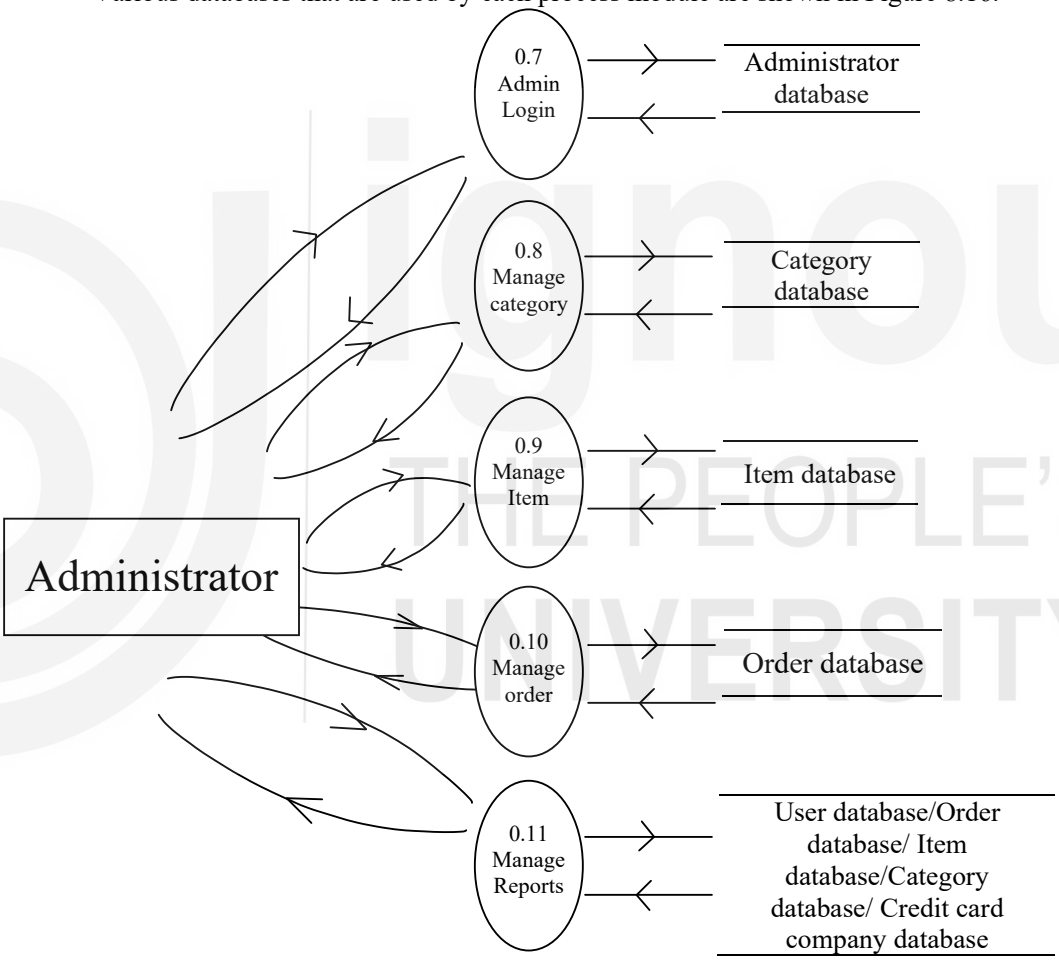

Figure 8.16: Level 1 DFD for administrator side

Now, each of the processes of Level- 1 DFD may or may not be exploded into more process modules depending upon the complexity of functionality performed by it. Here, Login module is not exploded into submodules because the purpose of this is quite simple, just the Administrator is going to Login to the OSS by using his username and password, and this process module accesses the Admin database for the same. The Administrator can perform the rest of the functionalities.

Level- 2 DFD

Again, each of the bubbles of level-1 DFD maybe decomposed into three to seven more bubbles so that the functions can be explained at an atomic level. This expansion generates level 2 DFD. Hence each process is decomposed into many sub-processes so that all functions can be explained in a simpler manner. In most situations, decomposition up to Level- 2 is sufficient to explain the functionality of the entire software system. Level-2 DFDs for the Customer side and Administrator side are shown separately.

Level -2 DFD: Customer side

Now, we have explained the customer side processes from Level- 1 DFD, one by one, starting from DFD number 0.3 to 0.5.

Level- 2 DFD: Customer side, for "Manage Account" Module 0.3 of Level 1 DFD

The process "Manage Account" has been decomposed into three sub-processes View Account, Change Password and Edit Profile. All of these three sub-processes interact with the customer database, as shown in Figure 8.17.

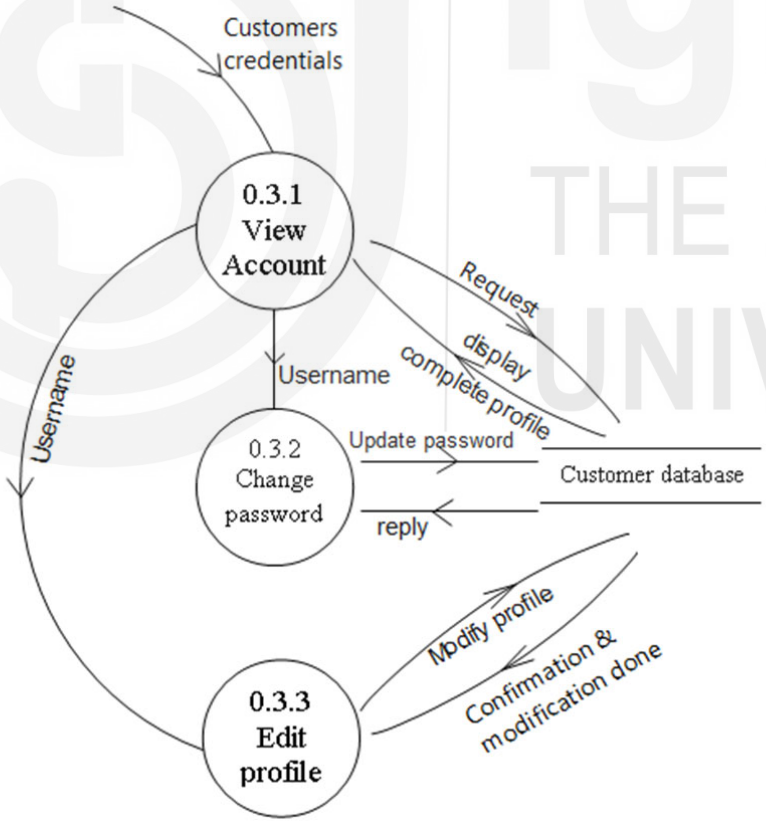

Figure 8.17: Level 2 DFD of "Manage Account" process (0.3) of customer side

Level- 2 DFD: Customer side, for "Purchase Items" Module 0.4 of Level 1 DFD

While purchasing items, the customer can search for an item (apply filters for color selection and price range selection), search for available offers, and manage his shopping cart. Different data stores that are accessed by these processes are shown in Figure 8.18. The process “Manage cart” is further exploded to show all sub-processes involved, in Figure 8.24 (Level 3DFD).

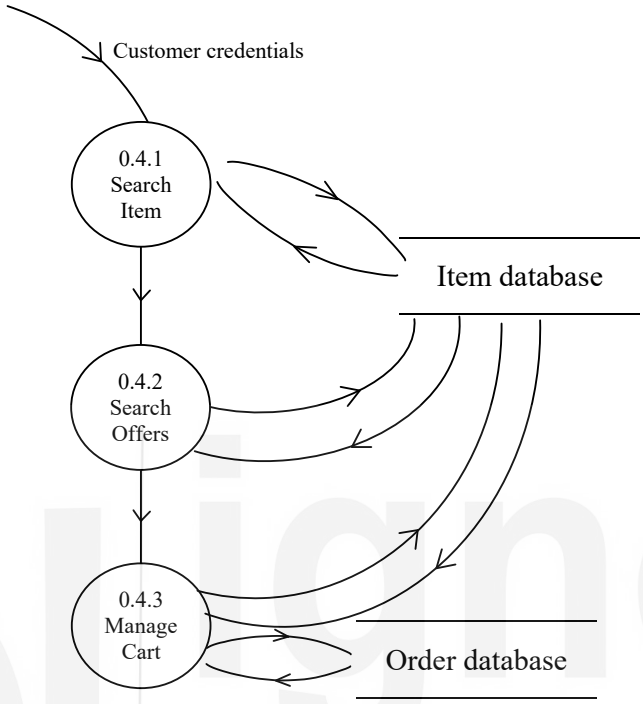

Figure 8.18: Level 2 DFD of “Purchase item” (0.4) of customer side

Level- 2 DFD : Customer side, for “Make Payment” Module 0.5 of Level 1 DFD

To make a payment, the customer types in his / her credit card details that are verified by the payment gateway by contacting the credit card database. After the payment is made, the customer can cancel the order if he desires to. If not cancelled in the stipulated time, the item details of the placed order are added to the company's order database. This DFD is shown in Figure 8.19.

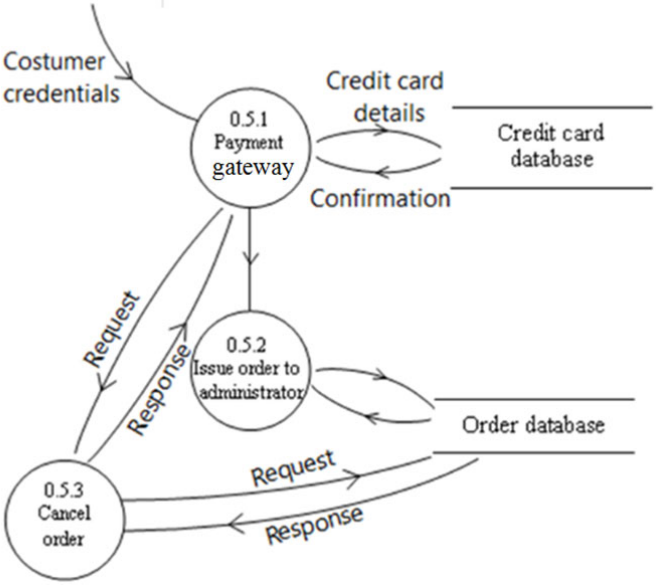

Figure 8.19: Level 2 DFD for “Make payment” (0.5) of customer side

Level -2 DFD: Administrator side

Now, we have explained the administrator side processes from Level 1 DFD, one by one, starting from DFD number 0.8 to 0.11.

Level- 2 DFD: Administrator side, for "Manage Category" Module 0.8 of Level 1 DFD

The process "Manage Category" is meant to add a new category of items on the website. In this new category, new items are added. An existing category may be updated by changing its name or shifting the items here and there among existing categories; maybe an item belongs to more than one category like Milk falls in the Dairy products category as well as the Daily needs category. If the OSS company thinks of not selling a few categories completely, then that particular category can be deleted altogether. The categories list can also be viewed. The DFD, with all the sub-processes explained here, is shown in Figure 8.20.

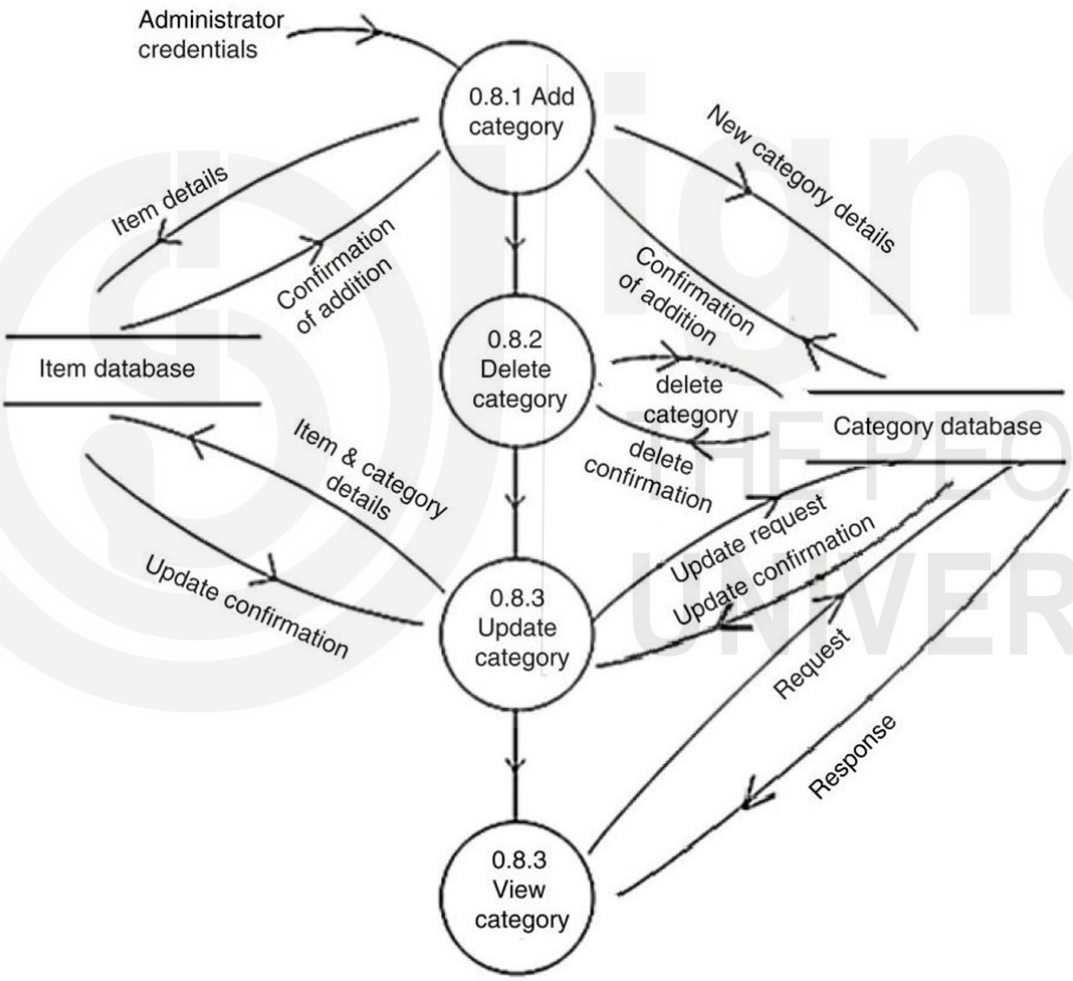

Figure 8.20 : Level 2 DFD "Manage category" (0.8) of administrator side

Level - 2 DFD : Administrator side, for "Manage Items" Module 0.9 of Level 1 DFD

Within the "Manage Items" process, the Administrator can add an item, delete item, view an item, and update an item. Update item is considered here at an abstract level, and its sub-processes are explained in Level 3 DFD, Figure 8.21.

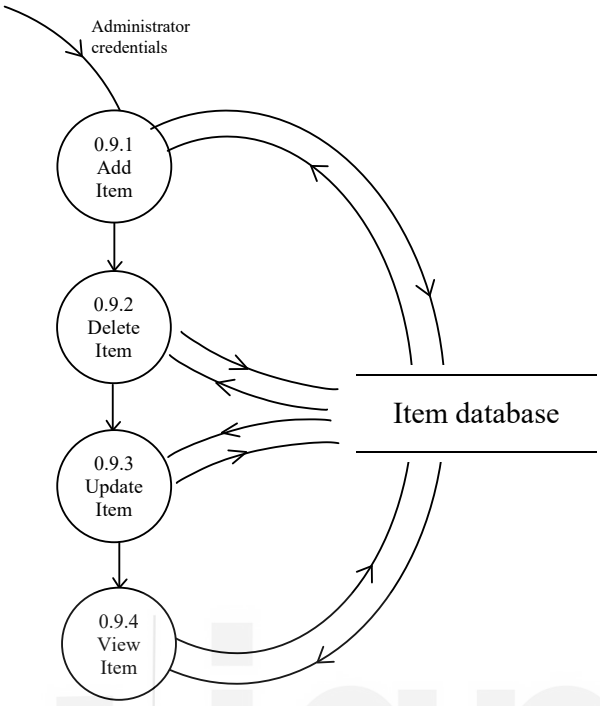

Figure 8.21: Level 2 DFD for "Manage item" (0.9) of administrator side

Level- 2 DFD : Administrator side, for "Manage Order" Module 0.10 of Level 1 DFD

Corresponding to the "Manage orders" process bubble, the Administrator can perform the followings:

- Acknowledge the order,
- View order details (cost of each item with item description and total bill), cancel the order (in case the item is not available), and
- Dispatch the order (this includes sub-processes related to supplier, shipment, returned item pick up; hence it is taken up in Level 3 DFD.).

For all these, the Order database has been accessed. This is shown in Figure 8.22.

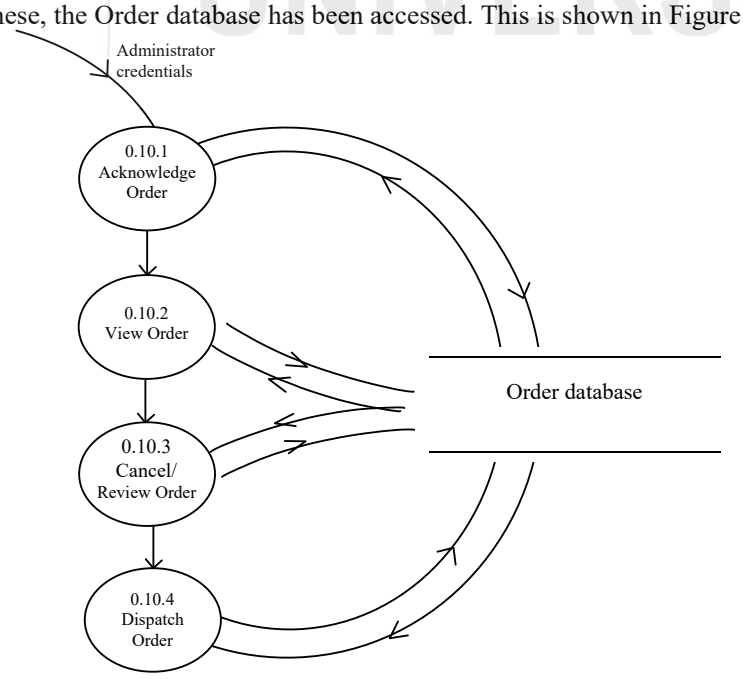

Figure 8.22: Level 2 DFD for "Manage order" (0.10) of Administrator side

Level 2 DFD – Administrator side, for "Manage Report" Module 0.11 of Level 1 DFD

Corresponding to the process "Manage Report", the Administrator can apply various types of filters on different databases to generate various reports as required. For example, given below a list of few reports that can be generated.

1. The report of users details along with their orders in a particular time duration can be generated.
2. The report on items that are delivered can be generated.
3. The report on payments received by the company can be generated.
4. The report on various suppliers can also be generated by this process.

Various databases namely user, item, category, credit card company database are accessed according to type of reports required. This DFD is shown in Figure 8.23.

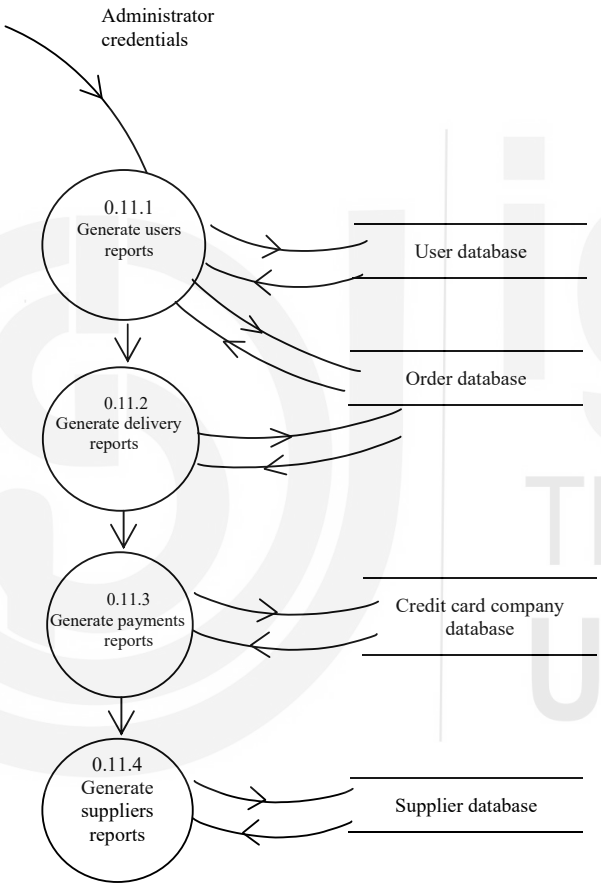

Figure 8.23: Level 2 DFD for "Manage Reports" (0.11) of Administrator side

Level- 3 DFD : Customer side, for "Manage Cart" Module 0.4.3 of Level 2 DFD

Further, we can see the decomposition of the "Manage cart" process of Level 2 DFD on the customer side. Here, all sub-processes involved are Add item, View Cart, Edit Cart, and check out. The corresponding databases that are accessed by each of the processes are shown in Figure 8.24.

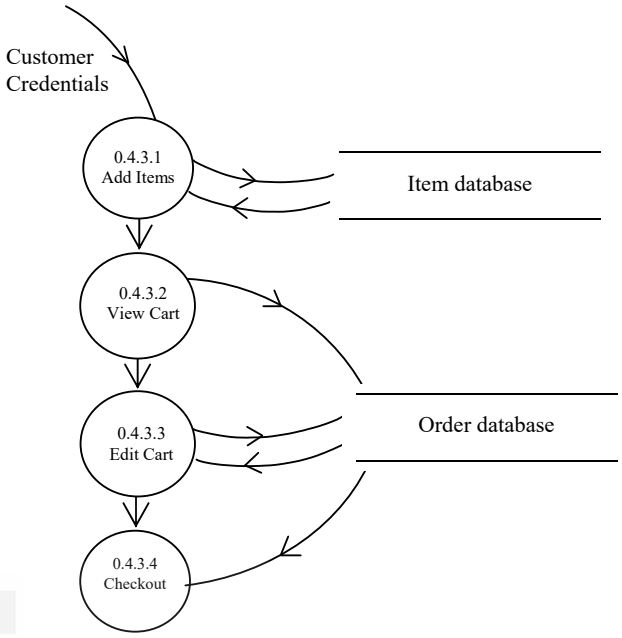

Figure 8.24: Level 3 DFD corresponding to decomposition of "Manage cart" (0.4.3) of level 2 of customer side

Level- 3 : DFD – Administrator side, for "Dispatch order" Module 0.10.4 of Level 2 DFD

The sub-process "Dispatch order" comprises many sub-functions Hence, it is exploded into three sub-processes: Contact Supplier, Contact Delivery Agent, Manage Returns, and the corresponding databases, as shown in Figure 8.25.

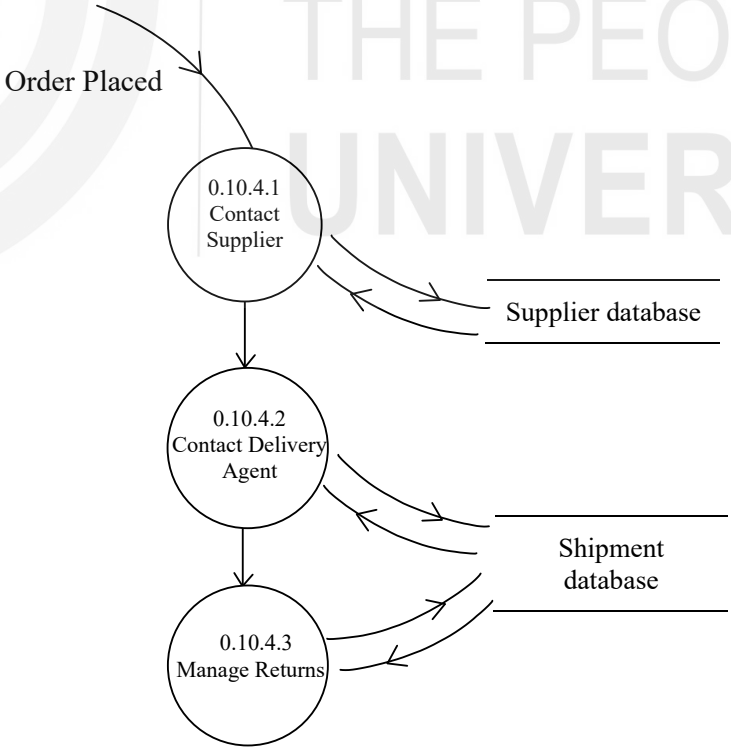

Figure 8.25: Level 3 DFD after decomposing "dispatch order" (0.10.4) bubble of Level 2 of administrator side

8.7 FUNCTIONAL Vs. OBJECT Vs. DYNAMIC MODEL

The object-modeling technique (OMT) consists of three models that explain the different aspects of the software system. These three models are as given below: -

1. **Object Model**: The purpose of this model is to show the static aspect of the software system, including depicting the attributes and operations that an entity can perform, i.e. Here, the focus is on how an object can be implemented and what operations an object is capable of performing.
   Diagrams: In an Object model, we use object diagram and class diagram.

2. **Dynamic Model**: The purpose is to show the control aspect, as well as the temporal and behavioural nature of the software system. The sequencing of operations is shown in the dynamic model. The focus is on when the operations are done in response to an external stimuli.
   Diagrams: We model the states, transitions, events and actions with the help of interaction diagrams (like sequence and collaborative diagrams) and state transition diagrams.

3. **Functional Model**: This model is used to depict the functional aspects by showing various processes that include many sub-processes within them, along with various transformations applied to data as it moves in the system. This model focuses on what the operations do in terms of transformations on objects.
   Diagrams: We use data flow diagrams to show collaboration between different objects to achieve the system's desired behaviour.

In order to explain any software system in totality, we need to understand the relationship between all these three models. These three models work hand in hand for the implementation of methods. The functional model acts as a guide to the methods. The processes in the functional model correspond to operations in the object model. Most of the time, there is a direct relationship between each Level.

A top-level process resembles an operation on a complex object. In contrast, lower-level processes correspond to operations on more basic objects that are part of the complex object or that implement it. Sometimes, one process corresponds to several operations, and one operation corresponds to several processes.
In a functional model, various processes show objects that are related by function. Most of the time, one of the inputs to a process can be identified as the target object, and the remaining inputs are the parameters to the operations. The target object is a client of the other objects because the target object uses other objects to perform the operations. The target has complete knowledge of the clients, but the clients may not know the target. The target object class is dependent on the argument classes for its operations. Client-supplier relationships create the implementation dependencies among classes. The clients' implementation is done by using supplier classes; hence, client class depends upon the supplier classes.
A process is implemented as a method. If the same class of object is an input as well as an output, then the object is usually the target, and the other inputs are arguments. Data store is always considered as a target, irrespective of the situation, whether it is input or output to a data store.

Whenever there is a process that interacts with a data store (provides input to data store or gets output from the data store), it uses two methods, of which one is an implicit selection or update of the data store. Further, if an input to or output from a data store is provided or obtained by an actor, then the actor itself is the target.

In the object model, if an input is an object and an output is a part of the object or a neighbor of the object, then the object is considered as the target. If an output object is created by making use of input parts, then the process represents a class method. If none of these situations is true, then the target is often implicit and is not one of the inputs or outputs. In most situations, the target of a process is the target of the whole sub-diagram. In the object model, actors are explicit objects. Whenever there is any interaction between the data store and actor, it is considered as operations with the actor (which is nothing but object). The data flow values are the arguments or results of the operations. Because actors are self-motivated objects, the functional model is not sufficient to indicate when they act. The dynamic model for an actor object specifies when it acts.

Data stores are also objects in the object model or attributes of objects. Each flow into a data store is called an update operation. Each flow out of a data store is a query operation with no side effects on the data store object. Data stores are passive objects that respond to queries and updates, so the dynamic model of the data store is irrelevant to its behaviour. A dynamic model of the actors in a diagram is necessary to determine the order of operations. Data flows are values in the object model. Many data flows are simply pure values, such as numbers, strings, or lists of pure values. Pure values can be modeled as classes and implemented as objects in most languages.

Relation of Functional Model to Object Model

In the functional model we use four components that have one to one correspondence with components of the object model. These are explained here.

Processes: Processes are the functions/ methods that are implemented in the objects in Object Model.

Actors: Correspond to the object of Object Model.

Data Stores: These are nothing but the objects of the Object Model or the attributes of the Objects.

Data Flow: Data flow that occurs as a result of operations performed on an object, is shown in Object Model. We depict data flow to or from the data store corresponding to a query or update operation in an Object Model.

Relation of Functional Model to Dynamic Model

In the functional model, we can specify how the operations are performed and which arguments were required for those operations, whereas the dynamic model depicts the timing of an operation. The dynamic model specifies when it acts on an active object only, i.e. on an actor of functional model. The dynamic model does not mention anything related to data stores as it is a passive object. The data store responds only to the updates and queries that too in the functional model only.

☞ Check Your Progress - 2

1) State True / False for the followings:
   a) External entities may appear at all levels of DFDs
   b) A DFD captures the order in which the processes (bubbles) operate
   c) A DFD model of a system represents the functions performed by the system and the data flow taking place among these functions.

.....

.....

.....

.....

2) What type of information can be represented in a dynamic model?

.....

.....

.....

.....

3) Why do we need many different levels in a DFD? What is the purpose of each of them?

.....

.....

.....

.....

4) Draw a DFD up to level-1 to show the registration system to register a student for subjects in a semester.

.....

.....

.....

.....

8.8 SUMMARY

In OMT there are three models namely Object Model, Dynamic Model and Functional Model that are required to be generated in order to explain the detailed working of any Software System in totality. In this unit, we have seen the design and working of the functional model in order to address the question "What" the software system is doing, and we have used Data Flow Diagrams to explain this. We have discussed the features of a DFD in terms of different symbols used along with their purpose. The symbols are used to denote process, data flow, actor, data store, constraints and control flows. Next, the design flaws that can occur while making a DFD have also been explained in detail in this unit. Further for the practical understanding of the topic, generation of DFDs up to Level-3 has been done for an Online Shopping System by decomposition of processes as we go from Level-0 to Level-3. Finally, an attempt has been made to throw some light on the relation between the functional model and the relation of the functional to the dynamic model.

8.9 SOLUTIONS / ANSWERS TO CHECK YOUR PROGRESS

☞ Check your progress -1

1) **Functional Model:** A functional model provides a framework to demonstrate the working of a software system in a graphical manner. It explains the functions of internal processes with the help of DFD (Data Flow Diagram).

   Importance of Functional model in OOAD: The Functional Model plays a key role in the OOAD process as it describes the functions and processes, assists with the discovery of information needs, helps in identifying opportunities, and establishes a basis for determining product and service costs. This model is similar to a real system, which helps the analyst understand and predict the effect of changes on the system. In a simple term, modelling is creating a model which represents a system, including all of its properties.

2) **Data stores:** They show data collection without mentioning how the data is stored. It represents the databases or any other medium of storage of data.

3) **Data flow diagram:** A data flow diagram (DFD) charts out the flow of information for any software system. It makes use of symbols to denote the flow of data from input to output. DFD can be a simple hand-drawn one or maybe a detailed multi-level one that shows the complete in-depth handling of the data. As we know, "A picture is worth a thousand words.", so a DFD can explain so many things that are difficult to explain in words and also, it can be understood by top management easily who is non-technical in nature.
   Below is a DFD that explains that input being fetched from the database passes through the system and produces the desired output for the customer.

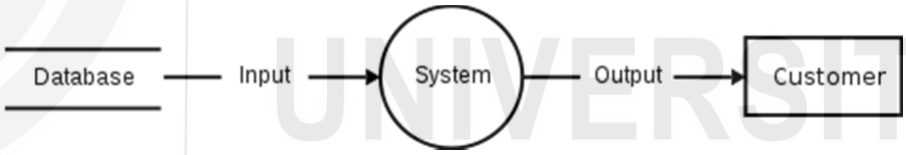

Figure 8.26: A data flow diagram

4) **Synchronous vs Asynchronous data flow**: Synchronous data flow is a situation when a data flow directly connects the two processes. Both processes must have the same speed of operation as there is no storage/buffer between the two processes to temporarily hold the data to adjust for the speed mismatch of the two processes. Hence, both processes are dependent on each other directly. In case there sits a data source between the two processes, the first process produces the data, which is stored in a data source. Later on, this can be used by another process. This type of data flow is called asynchronous.

5) **Balancing a DFD** means that the data that flow into or out of a bubble must match the data flow at the next Level of DFD.

6) **Design flaws arising out of poor Diagramming approach:** A poor diagrammatic approach poses many challenges and issues in understanding a software system. Many design flaws crop in because of this. The design flaws

occur when the outputs from a process does not match its inputs. A few examples of this design flaw are listed below: -

- Black hole: If a processing step has just input flows but no output flows.
- Miracle: If a processing step has just output flows, but no input flows.
- Grey hole: As the name suggests that it is a confusing situation. Such a situation may arise when a processing step has outputs greater than the sum of its inputs - e.g., its inputs could not produce the output shown.

☞ Check your progress -2

1) State True / False
   a) False. (External entities may appear only at level 0 of DFD.)
   b) False.
   c) True

2) **Dynamic Model** depicts the time and sequencing of the operations. We use it to specify and implement the control aspect of the system. State diagrams are used to show all such information.

3) Data flow diagrams can be made in several nested layers. Many different levels in a DFD are required in order to explain the functions of the software system in a deep and detailed manner. Level-0 DFD is a top-level data flow diagram (also known as "A context diagram"). It only contains one process node ("Process 0") that generalizes the function of the entire system in relationship to external entities.
   In level-1 data flow diagram, the single process node from the context diagram is broken down into subprocesses. As these processes are added, the diagram will need additional data flows and data stores to link them together.
   In a similar way, Level-1 DFD is exploded into further detailed sub-sub-processes to depict basic modules in the system and the flow of data among various modules and produces Level-3 DFD.

4) **DFD upto level- 1 to show the registration system to register a student for subjects in a semester.**

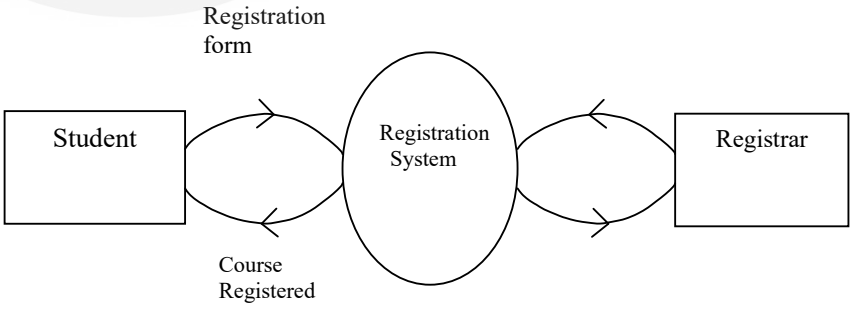

Figure 8.27: Context Diagram (Level- 0) for Student Registration System

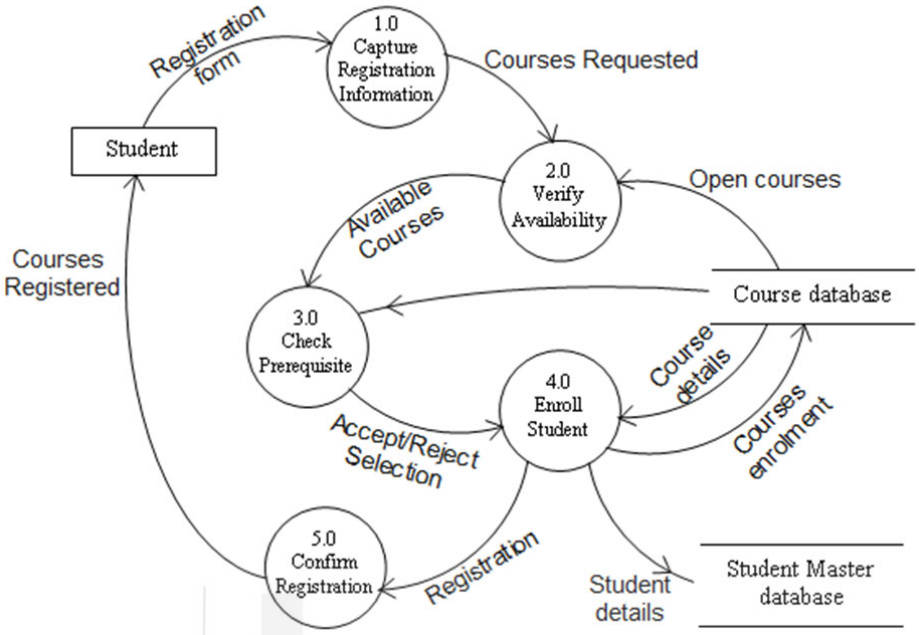

Figure 8.28: Level 1 DFD for Registration of Students for Subjects in a semester

8.10 REFERENCES / FURTHER READING

- Grady Booch, James Rumbaugh and Ivar Jacobson,“ The Unified Modeling Language User Guide”, 2nd Edition, Addison-Wesley Object Technology Series, 2005.
- Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, Kelli A. Houston, “Object-Oriented Analysis and Design with Applications,” 3rd Edition, Addison-Wesley, 2007.
- James Rumbaugh, Ivar Jacobson, and Grady Booch, “Unified Modeling Language Reference Manual,” 2nd Edition, Addison-Wesley Professional, 2004.
- Rajib Mall ,“Fundamentals of Software Engineering”, PHI, 2018.
- Bruce R. Maxim, and Roger S. Pressman, “Software Engineering: A Practitioner’s Approach”, McGraw Hill Education, 8th edition,2019.
- https://www.tutorialspoint.com/object_oriented_analysis_design/ooad_functional_modeling.htm
- https://www.geeksforgeeks.org/functional-modelling-in-object-oriented-analysis-and-design/
- https://en.wikipedia.org/wiki/Function_model
- https://www.geeksforgeeks.org/what-is-dfddata-flow-diagram/
- https://www.javatpoint.com/software-engineering-data-flow-diagrams
- https://blog.hubspot.com/marketing/data-flow-diagram