**Unit -II**

**Process and Conceptual Modeling**

**Data Flow Diagram (DFD)**

The DFD (also known as a bubble chart) is a hierarchical graphical model of a system that shows the different processing activities or functions that the system performs and the data interchange among these functions. Each function is considered as a processing station (or process) that consumes some input data and produces some output data. The system is represented in terms of the input data to the system, various processing carried out on these data, and the output data generated by the system. A DFD model uses a very limited number of primitive symbols [as shown in fig.] to represent the functions performed by a system and the data flow among these functions.

Output External Entity Process

Data Flow Data Store

**Importance of DFDs in a good software design**

The main reason why the DFD technique is so popular is probably because of the fact that DFD is a very simple formalism – it is simple to understand and use. Starting with a set of high-level functions that a system performs, a DFD model hierarchically represents various sub-functions. In fact, any hierarchical model is simple to understand. Human mind is such that it can easily understand any hierarchical model of a system – because in a hierarchical model, starting with a very simple and abstract model of a system, different details of the system are slowly introduced through different hierarchies. The data flow diagramming technique also follows a very simple set of intuitive concepts and rules. DFD is an elegant modeling technique that turns out to be useful not only to represent the results of structured analysis of a software problem, but also for several other applications such as showing the flow of documents or items in an organization.

**Example:** Tic-Tac-Toe Computer Game

Tic-tac-toe is a computer game in which a human player and the computer make alternative moves on a 3×3 square. A move consists of marking previously unmarked square. The player who first places three consecutive marks along a straight line on the square (i.e. along a row, column, or diagonal) wins the game. As soon as either the human player or the computer wins, a message congratulating the winner should be displayed. If neither player manages to get three consecutive marks along a straight line, but all the squares on the board are filled up, then the game is drawn. The computer always tries to win a game.

**(a)**

Human Player

Move

Display

Result

Game

Board

**(b)**

**Fig. (a), (b)** Level 0 and Level 1 DFD for Tic-Tac-Toe game described in Example 1

**Balancing a DFD**

The data that flow into or out of a bubble must match the data flow at the next level of DFD. This is known as balancing a DFD. The concept of balancing a DFD has been illustrated in fig.. In the level 1 of the DFD, data items d1 and d3 flow out of the bubble 0.1 and the data item d2 flows into the bubble 0.1. In the next level, bubble 0.1 is decomposed. The decomposition is balanced, as d1and d3 flow out of the level 2 diagram and d2 flows in.

d2

1. Level 1 DFD

d2

d1

d4

d3

1. Level 2 DFD

d1

d3

d22

d23

d21

fig. An example showing balanced decomposition

**Context Diagram**

The context diagram is the most abstract data flow representation of a system. It represents the entire system as a single bubble. This bubble is labeled according to the main function of the system. The various external entities with which the system interacts and the data flow occurring between the system and the external entities are also represented. The data input to the system and the data output from the system are represented as incoming and outgoing arrows. These

data flow arrows should be annotated with the corresponding data names. The name ‘context diagram’ is well justified because it represents the context in which the system is to exist, i.e. the external entities who would interact with the system and the specific data items they would be supplying the system and the data items they would be receiving from the system. The context diagram is also called as the level 0 DFD.

**Example#1:** RMS Calculating Software.

A software system called RMS calculating software would read three integral numbers from the user in the range of -1000 and +1000 and then determine the root mean square (rms) of the three input numbers and display it. In this example, the context diagram (fig.) is simple to draw. The system accepts three integers from the user and returns the result to him.

User

Context Diagram

data-items

RMS

Example 2:

Display

Context Diagram of aTic-Tac-Toe Problem

Move

Human Player

DFD model of a system

A DFD model of a system graphically depicts the transformation of the data input to the system to the final result through a hierarchy of levels. A DFD starts with the most abstract definition of the system (lowest level) and at each higher level DFD, more details are successively introduced. To develop a higher-level DFD model, processes are decomposed into their sub-processes and the data flow among these sub-processes is identified.

To develop the data flow model of a system, first the most abstract representation of the problem is to be worked out. The most abstract representation of the problem is also called the context diagram. After,developing the context diagram, the higher-level DFDs have to be developed.

**Level 1 DFD:-**

To develop the level 1 DFD, examine the high-level functional requirements. If there are between 3 to 7 high-level functional requirements, then these can be directly represented as bubbles in the level 1 DFD. We can then examine the input data to these functions and the data output by these functions and represent them appropriately in the diagram.

If a system has more than 7 high-level functional requirements, then some of the related requirements have to be combined and represented in the form of a bubble in the level 1 DFD. Such a bubble can be split in the lower DFD levels. If a system has less than three high-level functional requirements, then some of them need to be split into their sub-functions so that we have roughly about 5 to7 bubbles on the diagram.

**Decomposition:-**

Each bubble in the DFD represents a function performed by the system. The bubbles are decomposed into sub-functions at the successive levels of the DFD. Decomposition of a bubble is also known as factoring or exploding a bubble.

Each bubble at any level of DFD is usually decomposed to anything between 3 to 7 bubbles. Too few bubbles at any level make that level superfluous. For example, if a bubble is decomposed to just one bubble or two bubbles, then this decomposition becomes redundant. Also, too many bubbles, i.e. more than 7 bubbles at any level of a DFD makes the DFD model hard to understand.

Decomposition of a bubble should be carried on until a level is reached at which the function of the bubble can be described using a simple algorithm.

**Numbering of Bubbles**

It is necessary to number the different bubbles occurring in the DFD. These numbers help in uniquely identifying any bubble in the DFD by its bubble number.

The bubble at the context level is usually assigned the number 0 to indicate that it is the 0 level DFD. Bubbles at level 1 are numbered, 0.1, 0.2, 0.3, etc, etc.

When a bubble numbered x is decomposed, its children bubble are numbered x.1, x.2, x.3, etc. In this numbering scheme, by looking at the number of a bubble we can unambiguously determine its level, its ancestors, and its successors.

**Example:-**

A supermarket needs to develop the following software to encourage regular customers. For this, the customer needs to supply his/her residence address, telephone number, and the driving license number. Each customer who registers for this scheme is assigned a unique customer number (CN) by the computer. A customer can present his CN to the check out staff when he makes any purchase. In this case, the value of his purchase is credited against his CN. At

the end of each year, the supermarket intends to award surprise gifts to 10 customers who make the highest total purchase over the year. Also, it intends to award a 22 caret gold coin to every customer whose purchase exceeded Rs.10,000. The entries against the CN are the reset on the day of every year after the prize winners’ lists are generated. draw context level, level 1 and Level 2 diagrams for the above problem.

Gen. - Winner

Command

Customer Details

CN

Customer

Manager

Sales Details

Winner - List

Sales- Clerk

Fig. Context diagram for supermarket problem

Sales -

Details

CN

Customer -

Details

Fig. Level 1 Diagram for Supermarket Problem

Generate -

Winner -Command

Winner -

List

Sales -

info

Customer -

data

Generate - winner - commannd

Surprise- Gift- winner - list

Sales - Info

Fig. Level 2 diagram for supermarket problem

Sales - Info

Total - Sales

**Commonly made errors while constructing a DFD model**

Although DFDs are simple to understand and draw, students and practitioners alike encounter similar types of problems while modelling software problems using DFDs. While learning from experience is powerful thing, it is an expensive pedagogical technique in the business world. It is therefore helpful to understand the different types of mistakes that users usually make while constructing the DFD model of systems.

* Many beginners commit the mistake of drawing more than one bubble in the context diagram. A context diagram should depict the system as a single bubble.
* Many beginners have external entities appearing at all levels of DFDs. All external entities interacting with the system should be represented only in the context diagram. The external entities should not appear at other levels of the DFD.
* It is a common oversight to have either too less or too many bubbles in a DFD. Only 3 to 7 bubbles per diagram should be allowed, i.e. each bubble should be decomposed to between 3 and 7 bubbles.
* Many beginners leave different levels of DFD unbalanced.
* A common mistake committed by many beginners while developing a DFD model is attempting to represent control information in a DFD.
* A data store should be connected only to bubbles through data arrows. A data store cannot be connected to another data store or to an external entity.
* All the functionalities of the system must be captured by the DFD model. No function of the system specified in its SRS document should be overlooked.
* Only those functions of the system specified in the SRS document should be represented, i.e. the designer should not assume functionality of the system not specified by the SRS document and then try to represent them in the DFD.
* Improper or unsatisfactory data dictionary.
* The data and function names must be intuitive. Some students and even practicing engineers use symbolic data names such a, b, c, etc. Such names hinder understanding the DFD model.

**Shortcomings of a DFD model**

DFD models suffer from several shortcomings. The important shortcomings of the DFD models are the following:

* DFDs leave ample scope to be imprecise. In the DFD model, the function performed by a bubble is judged from its label. However, a short label may not capture the entire functionality of a bubble. For example, a bubble named find-book-position has only intuitive meaning and does not specify several things, e.g. what happens when some input information are missing or are incorrect. Further, the find-bookposition bubble may not convey anything regarding what happens when the required book is missing.
* Control aspects are not defined by a DFD. For instance, the order in which inputs are consumed and outputs are produced by a bubble is not specified. A DFD model does not specify the order in which the different bubbles are executed. Representation of such aspects is very important for modeling real-time systems.
* The method of carrying out decomposition to arrive at the successive levels and the ultimate level to which decomposition is carried out are highly subjective and depend on the choice and judgment of the analyst. Due to this reason, even for the same problem, several alternative DFD representations are possible. Further, many times it is not possible to say which DFD representation is superior or preferable to another one.
* The data flow diagramming technique does not provide any specific guidance as to how exactly to decompose a given function into its subfunctions and we have to use subjective judgment to carry out decomposition.

**Conceptual Modelling:**

It is a set of [concepts](http://medical-dictionary.thefreedictionary.com/concept), with propositions that describe them, express the relationship between them, or set forth the basic assumptions of the model. Both the concepts and the propositions of a conceptual model are abstract and very general and are often expressed in a vocabulary that is distinctive for each model.

Note: Preposition is the act of offering or suggesting something to be considered, accepted, adopted, or done.

**Entity Relationship (ER) Diagram**

* An *Entity-relationship model*(ERM) is an abstract and conceptual representation of data.
* ERmodelling is a DBmodelling method, used to produce a type of *conceptual schema*of a system.
* Diagrams created by this process are called *ER diagrams*.
* ER Model used to interpret, specify & document requirements for DBs irrespective of DBMS being used.

**ER Definitions:**

***Entity (Instance)*:**

– An instance of a physical object in the real world.

Student

– *Entity Class*: Group of objects of the same type.

– E.g. Entity Class “Student”, Entities “John”, “Trish” etc

***Attributes*:**

– Properties of Entities that describe their characteristics.

**– Types:**

• *Simple*: Attribute that is not divisible, e.g. age.

• *Composite*: Attribute composed of several simple attributes, e.g. address (house number, street, district)

• *Multiple* : Attribute with a set of possible values for the same entity, e.g. Phone (home, mobile etc.) or email

* ***Key*:** Uniquely Ids the Entity e.g. PPSN, Chassis No.

***Relationships*:**

1. Are bi-directional (ie can be put 2 ways)

*2. Degree*:

– *binary* (i.e. involve only two entities),

Lectures

Student

Lecturer

– *ternary* (i.e. involve three participating entities).

Course

Recommends

Lecturer

Textbook

3. Cardinality: Entity types may be linked in more than one way.

(a) One to One

1

1

1:1

Marries

Woman

Man

(b) One to Many

1

m

1:m

Lecturer

Teaches

Student

(c) Many to Many

n:m

Course

Student

Enrolls

m

n

4. May have properties (attribs).

Marries

5. Can be Recursive.

Person

Keys/Key Attributes

•Some definitions:

–*Super Key –*Set of attributes uniquely identifying a row

–*Candidate Key –*(Irreducible)combination of attributes which is a unique identifier within a–*Primary Key -*One of the candidate keys.

–*Alternate Key -*The candidate key(s) (if any) not chosen as the primary key.

–*Foreign Key -*A (combination of) attribute(s) in one relation whose value(s) are required to equal in the primary key of another relation.

**ER- Example1: An Entity with Attributes**

A student has a student number (identifying), a name, an address (with street number, street and district) and several phone numbers.

Student

**ER-Example 2: A Hospital Case**

Patients are treated in a single ward by the doctors assigned to them. Usually each patient will be assigned a single doctor, but in rare cases they will have two. Heathcare assistants also attend to the patients, a number of these are associated with each ward.

n

m

1

m

n

m

Seen by

Seen by

Seen by

Patient

Ward

Assistant

Doctor

**ER- Example 3: A university Database**

A lecturer, identified by his or her number, name and room number, is responsible for organising a number of course modules. Each module has a unique code and also a name and each module can involve a number of lecturers who deliver part of it. A module is composed of a series of lectures and because of economic constraints and common sense, sometimes lectureson a given topic can be part of more than one module. A lecture has a time, roomand dateand is delivered by a lecturer and a lecturer may delivermore than one lecture. Students, identified by numberand name, can attendlectures and a student must be registered for a number of modules. We also store the date on which the student first registered for that module. Finally, a lecturer acts as a tutor for a number of students and each student has only one tutor.

**Entities** and their Attributes (key)

–**Lecturer**(Number, Name, Office)**, Student(**Number, Name)

–**Module**(Code, Name),**Lecture**(Room, Date, Time)

•**Entities**and their Relationships**(Cardinality) *italics***

–“A lecturer is responsible for organising a number of course modules”

–**Lecturer-Module**(1:N) *is responsible for*

–“Each module can involve a number of lecturers who deliver part of it.”

–**Lecturer-Module**(N:M) *lectures*

–“A Module is composed of a series of Lectures and Lectures on a given topic can be part of more than one Module.”

–**Module-Lecture** (N:M)*is part of*

–“A Lecture is delivered by a Lecturer and a lecturer may deliver more than one lecture.”

–**Lecturer-Lecture**(1:N)*delivers*

–“Students, can attend Lectures”

–**Student-Lecture**(N:M) *attend*

–“and a Student must be registered for a number of Modules”

–**Student-Module**(N:M) *registers*(Attribute: Date)

–“Lecturer acts as a tutor for a number of Students and each Student has only one tutor”

–**Lecturer-Student**(1:N) *tutors*

n

1

Plays for

Student

Lecturer

n

m

m

1

n

n

m

n

1

n

m

n

Lecture

Parts of

Delivers

Registers

Attends

Repsp. for

Lectures

Module