JCSE International Journal of Computer Sciences and Engineering Open Access
Review Paper Vol.-8, Issue-1, Jan 2020 E-ISSN: 2347-2693

Review on Software Analysis & Design Tools

Bindia Tarika

Computer Science & Engineering, Punjab, India

Author: bindiatarika11@gmail.com

DOI: https://doi.org/10.26438/ijcse/v8i1.115119 | Available online at: www.ijcseonline.org

Accepted: 16/Jan/2020, Published: 31/Jan/2020

***Abstract*—**Software analysis and design includes all activities, which help the transformation of requirement specification into implementation. Requirement specifications specify all functional and non-functional expectations from the software. These requirement specifications come in the shape of human readable and understandable documents, to which a computer has nothing to do. Software analysis and design is the intermediate stage, which helps human-readable requirements to be transformed into actual code.

Keywords: *Software Analysis, DFD, HIPO*

I. INTRODUCTION

All the activities that are involved in converting requirements into implementation are considered as Software analysis and design. The expectations from the software are indicated by requirements which are in the normal documents that are in human understandable language. These human readable requirements are converted into the computer code by software analysis and design.
Software design and analysis allows the requirements of an application to be converted to actual code.

TOOLS OF SOFTWARE ANALYSIS AND DESIGN

1. DATA FLOW DIAGRAM (DFD)

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

1.1 Types of DFD

Data Flow Diagrams are either Logical or Physical.
Logical DFD - This type of DFD concentrates on the system process, and flow of data in the system. For example in a Banking software system, how data is moved between different entities.

Physical DFD - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

1.2 DFD Components

DFD can represent Source, destination, storage and flow of data using the following set of components -

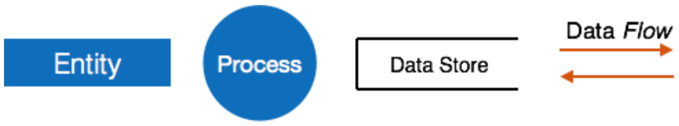


Figure 1. Components of DFD

- **Entities** – The source and destination of the data are considered as entities. DFD depict an entity with a rectangle along with the name of the entity.
- **Process** – Process involves the activities and actions upon the data. DFD depict the process by a circle or by round-edged rectangles.
- **Data Storage** – The storage of the data is depicted by DFD either with an rectangle without one side or with rectangle without both smaller sides.
- **Data Flow** – DFD depicts the flow of data by using pointed arrows. The source of data is represented by the base of the arrow and the destination of data is represented by the head of the arrow, pointing towards the destination.

1.3 Levels of DFD

The different levels of the DFD are as follows:
- **Level 0** – The DFD which represents the complete information system in a single diagram is known as Level 0 DFD and hence is considered as highest abstraction level DFD. They are also called as context level DFDs.

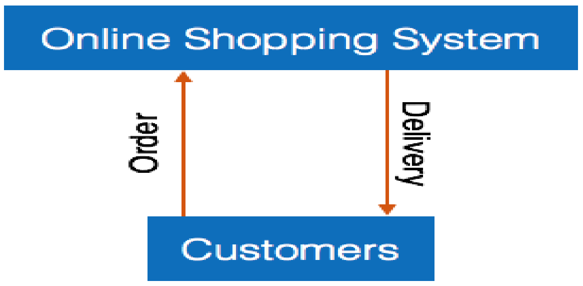


Figure 2. Level0 DFD

Level 1 – In order to put it in a more specific manner Level 0 DFD is split into Level 1 DFD. All the system basic modules and the pattern of data flow within the modules is represented by Level 1 DFD. The details about the information source and the processes involved are also depicted by Level 1 DFD.

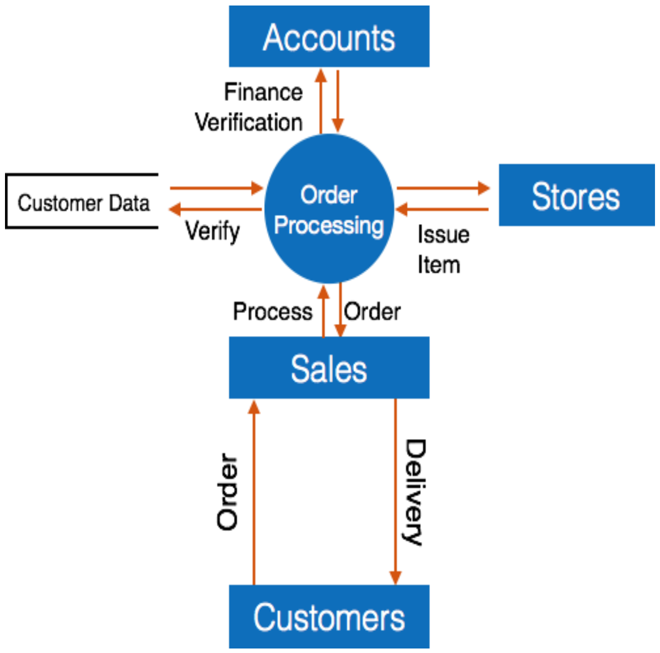


Figure 3. Level1 DFD

Level 2 – The flow of data within the module is represented by Level 2 DFD.
Until and unless the requested specification level is achieved, the higher level DFDs are converted into lower level DFDs.

II. STRUCTURE CHARTS

A chart that can be prepared on the basis on Data flow diagram is known as a structure chart. The system is depicted more clearly and in more detail when compared to that of DFD. The system is split into multiple modules that function independently. For each of such modules, functions and sub-functions are represented by structure charts.

- The module structure is demonstrated by the structure chart in a hierarchy. Tasks are performed at each level of hierarchy.

Structure charts use the following symbols –
2.1 Module – The process of the task is demonstrated by a Module. Modules can be Control Module, Sub-modules and Library module. Control module can be further divided into sub-modules.

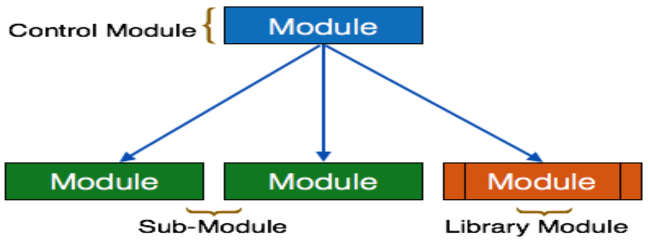


Figure 4.Module

2.2 Condition – When there is a condition on the basis of which the control module need to select the sub-modules, then that condition is represented by a small diamond shape at the base of the control module.

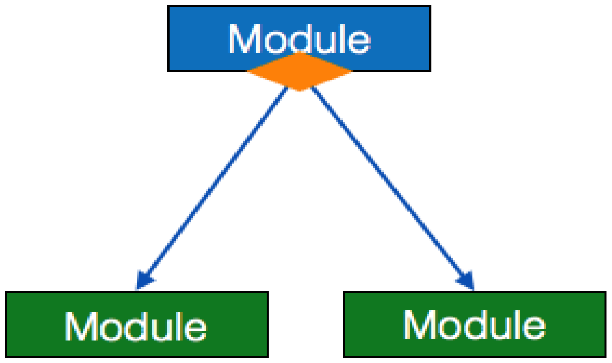


Figure 5. Condition

2.3 Jump – The jumping of the control module inside the middle of the sub-module is represented by an arrow.

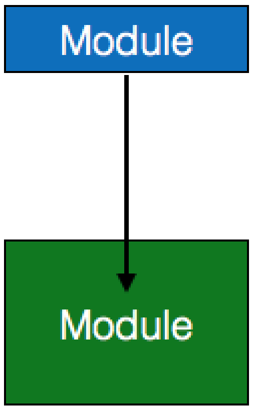


Figure 6. Jump

2.4 Loop – When execution of the sub-modules need to be repeated, then the loop of all the sub-modules is represented by a curved arrow.

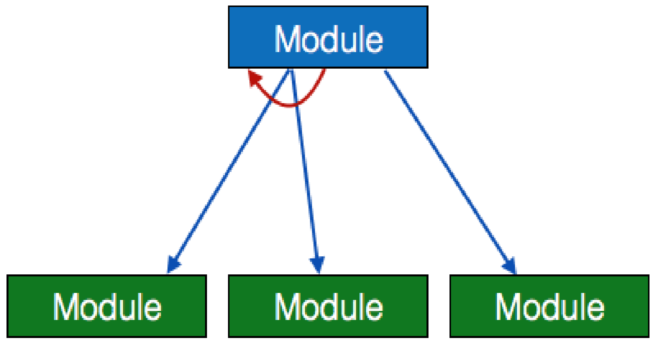


Figure7.Loop

2.5 Data flow – The flow of the data is demonstrated by an arrow with empty circle at the end.

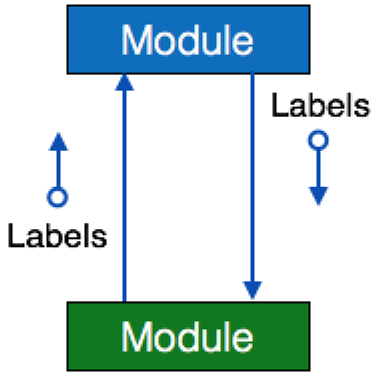


Figure 8. Data Flow

2.6 Control flow – The flow of the control is demonstrated by a directed arrow with filled circle at the end.

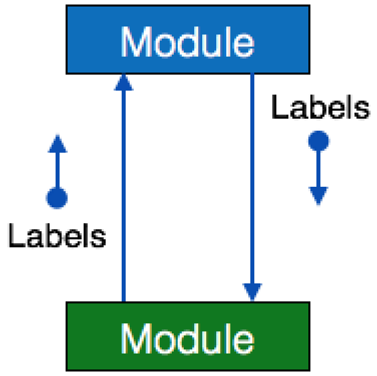


Figure 9. Control Flow

1. HIPO DIAGRAM
The method of analyzing the system combined with the method of facilitating the documentation means, is depicted and is known as Hierarchical Input Process Output, HIPO Diagram.

The modules of the software system are placed in a hierarchy by HIPO diagrams. High-level view for the functions of the system can be achieved by using HIPO diagrams.

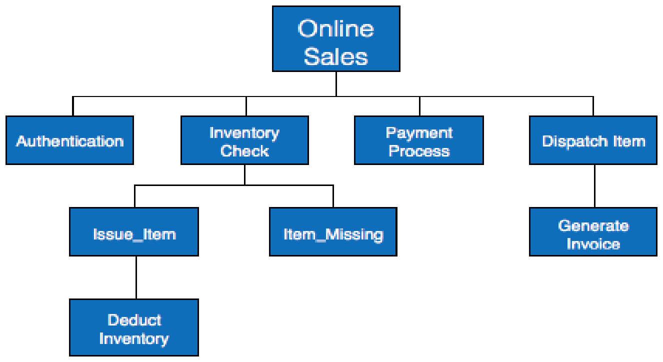
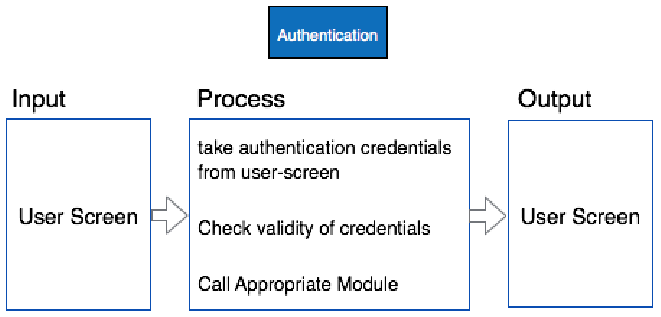


Figure 10. HIPO Diagram

The data flow and control flow information is not provided by HIPO diagram.


Authentication
Input
Process
Output
User Screen
take authentication credentials from user-screen
Check validity of credentials
Call Appropriate Module
User Screen
Figure 11. IPO Diagram

2. STRUCTURED ENGLISH
Structured English enables to reduce the gap of misunderstanding which may occur by using graphs or diagrams.
In structured English, simple words of English are used in the concept of structured programming. It provides detailed information about what is to be coded and how it is to be coded.
Structured programming uses some of the tokens such as:-
- IF-THEN-ELSE
- DO-WHILE-UNTIL

3. PSEUDO-CODE
Unlike structure English, Pseudo-code mostly resembles programming language. All descriptions and comments are included in Pseudo-code.

The constructs of some of the programming languages like C, FORTRAN, and PASCAL are used for writing Pseudo-code.

4. DECISION TABLES
Different conditions and the actions that are taken to address these conditions are represented by Decision Tables.
Decision tables helps in preventing the errors. By using Decision Tables, similar type of information can be combined and grouped into a single table. Decision-making is made more convenient by combining these tables.

How to create a Decision Table?
The steps for creating a decision table are as follows -
- All the conditions that are likely to be addressed are identified.
- For the conditions thus identified, the relevant actions are determined.
- To the maximum possible extent, rules are created.
- For each of the specific rule, action needs to be defined.

5. ENTITY-RELATIONSHIP MODEL
On the basis of the concept of real-world entities and their relationship, a database model is developed, which is known as Entity-Relationship model. ER database model enables to link with a real world scenario. The entities along with the attributes and constraints are created by an ER model.

The database can be designed on the basis of concept by using ER Model. ER model can be depicted as follows –

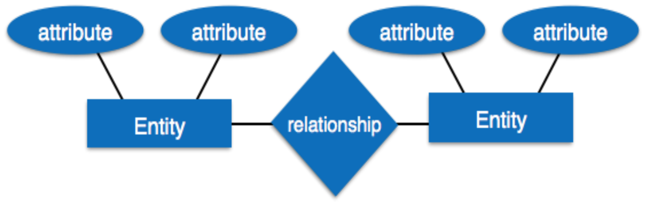


Figure 12. ER Model

- **Entity** – The real world being associated with properties is known as an entity. The properties are known as attributes. The respective set of values defines an attribute. The set of values is known as domain.For instance, in a database of a school, student is considered as an entity. The attributes of the entity are name, id, age and class etc.

- **Relationship** – The entities when linked with each other in a logical manner is known as relationship. Different combinations of mapping can be done for developing relationships. Between two entities, number of associations is defined by the cardinalities of mapping.

The different mapping cardinalities are –
- one to one
- one to many
- many to one
- many to many

6. DATA DICTIONARY
The complete information about the data is collected and stored centrally is Data dictionary. The information about the data that is stored in data dictionary is information related to origin of the data, relations with other data, format of data usage. The users and software designers are facilitated with all the definitions pertaining to any data.

7. CONTENT OF DATA DICTIONARY
Data Flow: Data Flow is described by means of DFDs.

=	**Composed of**
{}	Repetition
()	Optional
+	And
[/]	Or

Data Elements: Data elements consist of Name and descriptions of Data and Control Items, Internal or External data stores etc. with the following details:

○ Primary Name
○ Secondary Name (Alias)
○ Use-case (How and where to use)
○ Content Description (Notation etc.)
○ Supplementary Information (preset values, constraints etc.)

Data Store: It stores the information from where the data enters into the system and exists out of the system.

Data Processing: There are two types of Data Processing: Logical, Physical.

III. CONCLUSION

Creating these models and diagrams mentioned is a pre-cursor to choosing a design pattern or implementing any code. Creating a usable understanding of how a system will work is a crucial first step in any design. When it comes time to design an application one will definitely make sure to the usage these tools before you think about the actual design pattern or architecture you want to use.

REFERENCES

[1] Cameron, J.R., An overview of JSD, *IEEE Transactions on Software Engineering*, **Vol. SE-12**, No.2, February, pp 222–240. **1986.**
[2] Richard Fairley ,Software Engineeering Concepts ,Tata Mcgraw Hill.
[3] Pankaj Jalote , An Integrated Approach to Software engineering, *Narosa Publication*.

[4] Diethelm, I., L. Geiger, and A. Zundorf "Teaching Modeling with Objects First," WCCE 2005, *8th World Conference on Computers in Education, Cape Town*, South Africa, **2005.**

[5] Whitgift, David, "*Methods and Tools for Software Configuration Management*", J. Wiley, **1991.**

[6] Coad Peter and Edward Yourdan, *Object-Oriented Design*, 1991.

[7] *Software Management Guide*, Vol. I, Software Technology Support Center, p. **23**, **October 1993.**

[8] Dyer, Mike, "*he Cleanroom Approach to Quality Software Development*", **1993.**

[9] Blum Bruce I., "*Software Engineering: A holistic View*", **1992**.

[10] Booch, Grady, *Software Engineering with Ada*, p**. 25, 1994.**

AUTHOR PROFILE

Bindia Tarika did her B.Tech. and M.Tech. in Computer Science & Engineering from Guru Nanak Dev Engineering College, Ludhiana, Punjab Technical University, Kapurthala , Punjab, India . Her research interests are in the fields of Medical Image Processing, Detection of Cancer, Software Engineering. Her research papers have been published in various international journals.