Assignment-Module – 01 SDLC

**1.What is software? What is software engineering?**

Software is a set of instructions, data or programs used to operate computers and execute specific tasks. It is the opposite of hardware, which describes the physical aspects of a computer.

Software engineering is defined as a process of analyzing user requirements and then designing, building, and testing software application which will satisfy those requirements.

**2.Explain types of software**

In a computer system, the software is basically a set of instructions or commands that tells a computer what to do. Or in other words, the software is a computer program that provides a set of instructions to execute a user’s commands and tell the computer what to do.

Generally, there are two main classifications of software, which are namely, System Software along with the Application Software.

1. System Software

2. Application Software

1. System Software

In case of a system software, it helps the user as well as the hardware to function and even interact with each other easily. Essentially, it is a software which is used to manage the behaviour of the computer hardware in order to offer basic functionalities which are needed by the user. In simpler word, it can be said that system software is essentially an intermediator or even a middle layer between the user as well as the hardware.

These software sanction an environment or platform for the other software to easily work in. Hence, it is the reason why the system software is quite important in the management of the entire computer system. Whenever you turn on the computer first, it is this system software which gets initialized and then gets loaded in the system’s memory. A system software essentially runs in the background, and it isn’t actually utilized by the end-users. Due to this reason, the system software is also known popularly as “low-level software”. Companies usually hire the best software development company to build a system software.

Few of the common system software examples are:

a. Operating System

Being a prominent example for system software, it is essentially a collection of software which handles resources as well as offers general services for various other application which actually run over them. There are different types of operating systems like embedded, real-time, distributed, single-user, multi-user, mobile, internet and much more. Full stack web development services develop apps to operate on a mobile operating system like Android and iOS. Some of the key examples of operating systems are as follows:

MS Windows

macOS

Linux

iOS

Android

CentOS

Ubuntu

Unix

Printer Drivers

USB Drivers

Sound Card Driver

VGA Drivers

2. Application Software

They are also popularly known as end-user programs or even productivity programs which assist the user in completing various tasks like conducting online research, making notes, designing graphics, maintaining accounts, carrying out calculations or even playing computer games. They essentially lie above the system software. They are actually used by the end-user as well as have specific functionality or tasks which they are designed to perform. These software are often developed through custom software development, based on the requirements of the users. There is a variety of application software. Some of them are:

a. Word Processors

Such applications are meant for documentation. It also assists in storing as well as formatting and even printing of the documents. Key examples of such software are:

MS Word

Apple iWork-Pages

Corel WordPerfect

Google Docs

b. Database Software

It is used to create as well as manage a database and also known as Database Management System or in short, DBMS. Such software assists in the data organization. Some of the examples of DBMS are:

MS Access

FileMaker

dBase

Clipper

MySQL

FoxPro

Windows Movie Maker

**3.What is SDLC? Explain each phase of SDLC**

The process of planning, creating, testing, and deploying software is called Software Development Life Cycle or SDLC. Different tasks to be performed in each step of the software development process are explained well in SDLC. Different phases of SDLC are planning, requirements, design, development, testing, deployment, and maintenance. Various SDLC models are the waterfall model, spiral model, V-shaped model, iterative model, big bang model, and agile model. Each part of software development is evaluated easily and helps programmers to work concurrently using SDLC phases. This is a process to be followed and not a technique.

**Types of Phases in SDLC**

There are several phases in the lifecycle of software which are given below:

Requirement Phase

Analysis Phase

Design Phase

Development Phase

Testing Phase

Deployment & Maintenance Phase

1. Requirement Phase

This is the first and fundamental step in the Life Cycle of Software Development. It starts with gathering the requirements from customers or clients. In most organizations, this role is taken care of by Business Analysts. A Business Analysts interacts with the customer/clients, set up daily meetings, document the requirements in Business Requirement Specifications (or Simple Business Specification), and hand over the final documented requirement to the development team. It is the responsibility of Business Analysts that every detail is captured and documented and also to make sure that everyone clearly understands the client requirements.

2. Analysis Phase

Once the Requirement Gathering phase is completed, the next task is to analyze the requirements and get them approved by the customer/clients. This is achieved through Software Requirement Specification (SRS), which consists of all the requirements gathered and developed during the Requirements Gathering phase. This phase is mainly done by Project Managers, Business Analysts, and Consultants.

3. Design Phase

Once the Analysis Phase is over, next comes the need to come up with the most accurate, robust, efficient and cost-effective architecture of the product that needs to be developed. Usually, more than one design is proposed in this phase, and the best one is selected based on different parameters such as robustness, durability, timeline, cost-effectiveness, and many more! The different design architecture is generally documented in Design Document Specification or DDS.

This phase consists of 2 design approaches:

Low-Level Design: This task is performed by the Senior Developers where they specify the function of each module of the product architecture that has to be developed.

High-Level Design: This task is performed by Architects/Senior Architects where they design different possible architectures of the product that has to be developed.

4. Development Phase

This phase is where the actual implementation of programming languages and different frameworks are being utilized for the development of the product. In this phase, all developers are involved. Developers are expected to follow certain predefined coding standards and guidelines; they are expected to complete the project modules within the defined deadline for the project. This phase is also the longest and one of the most critical phases in the Software Development Life Cycle. This phase is documented as a Source Code Document (SCD).

5. Testing Phase

Once the Development phase is completed, the next step is to test the developed software. The developed software is sent to the testing team, where they conduct different types of testing thoroughly on the software and look for defects. If any defect is found, the testing team records and document which is again sent back to the development team for error removal. This role is taken care of by Software Testers and Quality Analysts of the company. The testing team has to make sure that each component of the software is error-free and it works as expected.

6. Deployment and Maintenance Phase

After the testing phase is over, the first version of the software is deployed and delivered to the customer for their use. Once the customer starts using the developed software, there is the scope of bug fixing that was not detected during the testing phase as when a large group of end-users starts using the software; there could be some probability that few boundary cases might have been missed. There is also scope for upgrading the software with newer versions and the latest security patches and technologies. And finally, there is also scope for enhancement of the software by adding more features into the existing software.

**4.What is DFD? Create a DFD diagram on Flipkart**

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

Components of DFD

The Data Flow Diagram has 4 components:

Process

Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence

Data Flow

Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.

Warehouse

The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updation.

Terminator

The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

Rules for creating DFD

The name of the entity should be easy and understandable without any extra assistance(like comments).

The processes should be numbered or put in ordered list to be referred easily.

The DFD should maintain consistency across all the DFD levels.

A single DFD can have maximum processes upto 9 and minimum 3 processes.

Levels of DFD

DFD uses hierarchy to maintain transparency thus multilevel DFD’s can be created. Levels of DFD are as follows:

0-level DFD

1-level DFD:

2-level DFD:

Advantages of DFD

It helps us to understand the functioning and the limits of a system.

It is a graphical representation which is very easy to understand as it helps visualize contents.

Data Flow Diagram represent detailed and well explained diagram of system components.

It is used as the part of system documentation file.

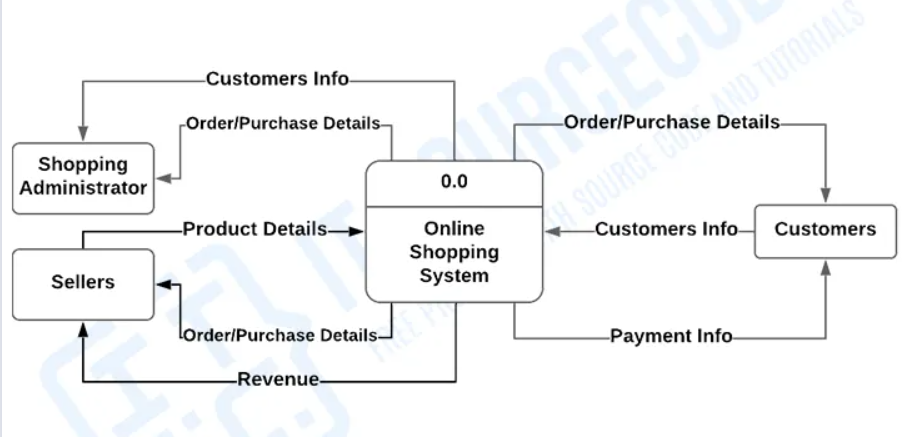
Data Flow Diagrams can be understood by both technical or nontechnical person because they are very easy to understand.

Disadvantages of DFD

At times DFD can confuse the programmers regarding the system.

Data Flow Diagram takes long time to be generated, and many times due to this reasons analysts are denied permission to work on it.

DFD diagram on Flipkart



**5.What is Flow chart? Create a flowchart to make addition of two numbers**

A flowchart is a diagrammatical representation of data that helps in depicting a particular process or flow of instructions of an algorithm that is basically a step-wise approach in solving a bigger task. A flowchart diagram usually represents the particular steps as boxes and the way of communication or the connection between boxes is done using arrows. With the help of this diagrammatic approach, a flowchart shows the overall model of a particular problem and finds the solution using the right approach. It is used in analyzing and designing a program or a process in many sectors.

A flowchart is a visual representation in terms of representing data using symbols. It is better approached by designing it in a diagrammatic approach for its easy behavior and understanding

**6.What is Use case Diagram? Create a use-case on bill payment on paytm**

A use case diagram is a visual summarization of interactions and relationships within a system. These diagrams show a very broad view of a system. They may show systems in computer software, businesses or customer experiences. A use case diagram shows a model scenario in which individuals interact with a system using a series of specialized symbols and connectors.

What is involved in a use case diagram?

A case use diagram includes a few key components:

Actors

Actors are individuals who have a relationship within the system. They interact with each step of the process. This may include users, customers, clients or employees Actors may interact with the system internally or externally.

Use cases

Use cases, often represented by an oval or circle encompassing text, describe a system function that is an actor or person can start or use. This function may be automatic or manual and may link to an actor or function independently.

Communication links

A solid, straight line can represent an actor's interaction with a use case or use cases' interactions. Actors may connect to use cases and other people through a variety of communication links representing different relationships such as:

Associations: Association links represent interactions between actors and use cases.

Generalizations: Generalization links represent relationships between actors that depend on each other's functions to complete their role within a system.

Extensions: Extension links represent optional functions within a system. These links may connect actors to use cases or multiple related use cases without depending on one another for functionality.

Inclusions: Inclusion links represent additional functions necessary to complete a system and may form connections between two use cases or use cases and actors.

System boundary

The system boundary is a visual limitation of the entire system module. For complex systems, there may be many system boundaries that represent smaller, more specific modules. For example, a business use case diagram may have system boundaries for smaller components, such as marketing, sales and accounting.

Purpose of a use case diagram

Use case diagrams offer a clear visualization of a system's functional requirements. They can help an organization make important design choices. They may also identify external and internal factors that could affect the system's interactions.

Use-case diagram

