**EOD Report (28/06/2022)**

**Tasks:**

1**.** SDLC

2. DFD

3. ER diagram

4. Use Case

5. Documentation of Project

**SDLC**

**What is Software Development Life Cycle (SDLC)?**

Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product.

**Purpose:**

Purpose of SDLC is to deliver a high-quality product which is as per the customer’s requirement.

**For Example,** A software has to be developed and a team is divided to work on a feature of the product and is allowed to work as they want. One of the developers decides to design first whereas the other decides to code first and the other on the documentation part.

This will lead to project failure because of which it is necessary to have a good knowledge and understanding among the team members to deliver an expected product.

**SDLC Phases**

**Given below are the various phases:**

* Requirement gathering and analysis
* Design
* Implementation or coding
* Testing
* Deployment
* Maintenance

### 1) Requirement Gathering and Analysis

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

**For Example,** A customer wants to have an application which involves money transactions. In this case, the requirement has to be clear like what kind of transactions will be done, how it will be done, in which currency it will be done, etc.

Once the requirement gathering is done, an analysis is done to check the feasibility of the development of a product. In case of any ambiguity, a call is set up for further discussion.

Once the requirement is clearly understood, the SRS (Software Requirement Specification) document is created. This document should be thoroughly understood by the developers and also should be reviewed by the customer for future reference.

### 2) Design

In this phase, the requirement gathered in the SRS document is used as an input and software architecture that is used for implementing system development is derived.

### 3) Implementation or Coding

Implementation/Coding starts once the developer gets the Design document. The Software design is translated into source code. All the components of the software are implemented in this phase.

### 4) Testing

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.

### 5) Deployment

Once the product is tested, it is deployed in the production environment or first [UAT (User Acceptance testing)](https://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/) is done depending on the customer expectation.

In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

### 6) Maintenance

After the deployment of a product on the production environment, maintenance of the product i.e. if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

REFERENCE:

1. <https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/>

2. <https://stackify.com/what-is-sdlc/>

DED

# **Data Flow Diagrams**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system.

**The following observations about DFDs are essential:**

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:



REFERENCE:

1. <https://www.javatpoint.com/software-engineering-data-flow-diagrams>

2. <https://en.wikipedia.org/wiki/Data-flow_diagram>

ER Diagram

## What is an Entity Relationship Diagram (ERD)?

ERD stands for entity relationship diagram. People also call these types of diagrams ER diagrams and Entity Relationship Models. An ERD visulizes the relationships between entities like people, things, or concepts in a database. An ERD will also often visualize the attributes of these entitities.

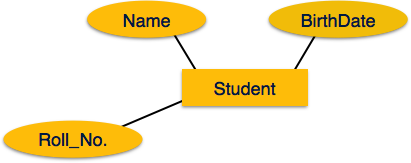
## Entity

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Entities in a school database

## Attributes

Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



## Relationship

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line

**1. One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'.

**2. One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'.

**3**. **Many-to-one** − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'.

**Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship.