ELEMENTS OF COMPUTER SCIENCE AND ENGINEERING
UNIT-2

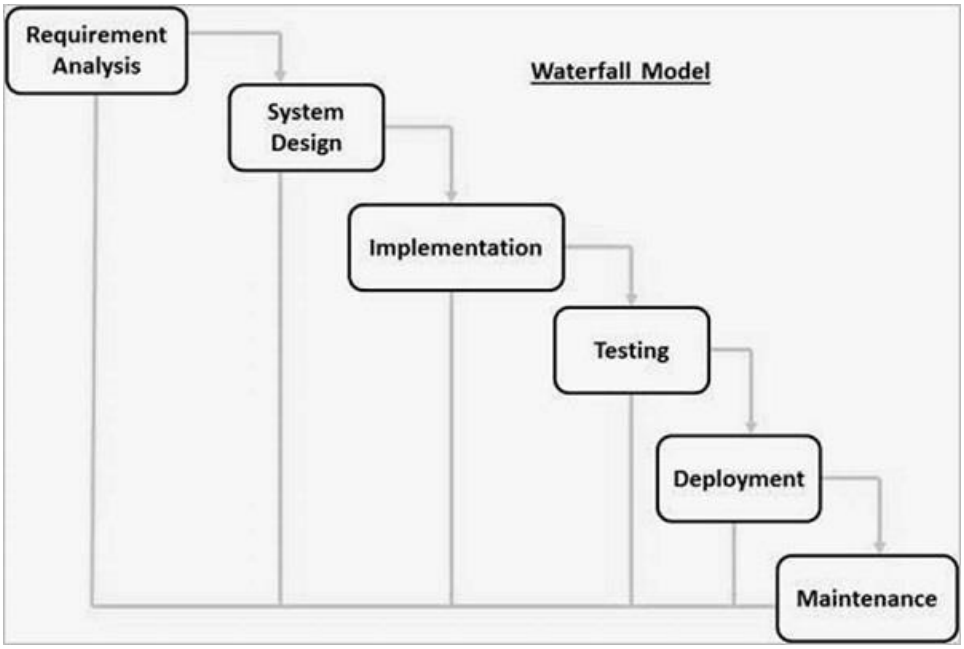
SOFTWARE DEVELOPMENT

Software development is a process by which standalone or individual software is created using a specific programming language. It involves writing a series of interrelated programming code, which provides the functionality of the developed software.

Software development is primarily achieved through computer programming, which is carried out by a software programmer and includes processes such as initial research, data flow design, process flow design, flow charts, technical documentation, software testing, debugging and other software architecture techniques. This is known as the software development life cycle (SDLC).

WATERFALL MODEL

- Waterfall Model is a sequential model that divides software development into pre-defined phases.
- Each phase must be completed before the next phase can begin with no overlap between the phases.
- Each phase is designed for performing specific activity during the SDLC phase.
- It was introduced in 1970 by Winston Royce.



Different Phases of Waterfall Model

Requirement Gathering stage - During this phase, detailed requirements of the software system to be developed are gathered from the client.

Design Stage - Plan the programming language, for Example Java, PHP, .net or database like Oracle, MySQL, etc Or other high-level technical details of the project.

Built Stage - After design stage, it is built stage, that is nothing but coding the software.

Test Stage - In this phase, you test the software to verify that it is built as per the specifications given by the client.

Deployment stage - Deploy the application in the respective environment.

Maintenance stage - Once your system is ready to use, you may later require change the code as per customer request.

Problems in Waterfall Model

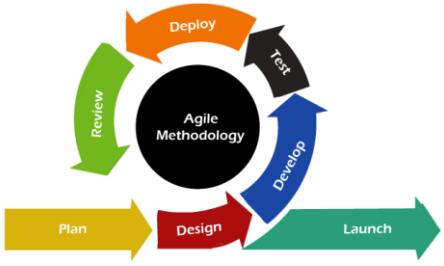
• Real projects rarely follow the sequential flow since they are always iterative.
• The model requires requirements to be explicitly spelled out in the beginning, which is often difficult.
• A working model is not available until late in the project time plan.

AGILE PROCESS

The Processes which are adaptable of changes in requirements, which have incrementality and work on unpredictability. These processes are based on three assumptions which all do refer to the unpredictability in different stages of software process development such unpredictability at time requirements, at analysis and design or at time construction.So these processes are adaptable at all stages on SDLC.

Agile Process models

1. Extreme Programming(XP)
2. Adaptive Software development(ASD)
3. Dynamic software Development Method(DSDM)
4. Scrum
5. Crystal
6. Feature Driven development (FDD)
7. Agile Modeling(AM)



Types of Computer Languages

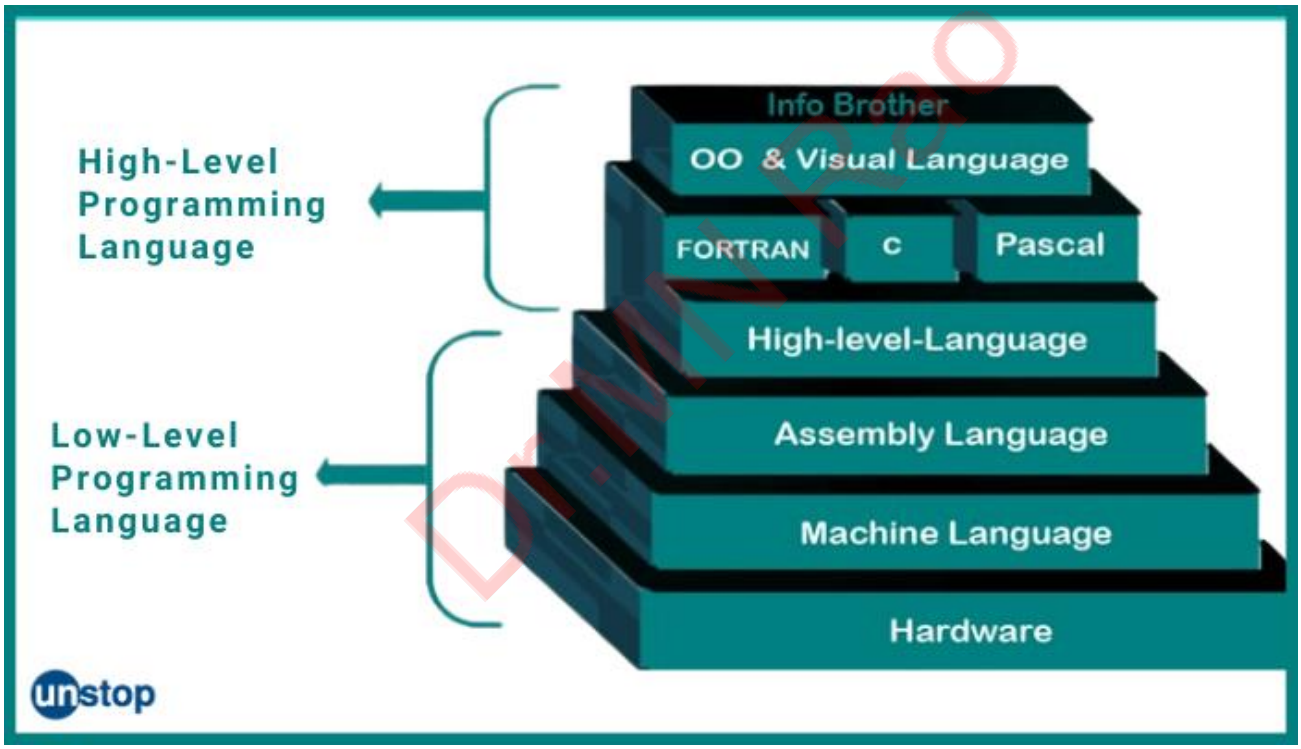
A language is the main medium of communicating between the Computer systems and the most common are the programming languages.

• As we know a Computer only understands binary numbers that is 0 and 1 to perform various operations but the languages are developed for different types of work on a Computer.

• A language consists of all the instructions to make a request to the system for processing a task.

• From the first generation and now fourth generation of the Computers there were several programming languages used to communicate with the Computer.

1,Low-level language
2.High level language



Low Level Language:

• Low level languages are the machine codes in which the instructions are given in machine language in the form of 0 and 1 to a Computer system.

• It is mainly designed to operate and handle all the hardware and instructions set architecture of a Computer.

• The main function of the Low level language is to operate, manage and manipulate the hardware and system components.

• There are various programs and applications written in low level languages that are directly executable without any interpretation or translation.
• The most famous and the base of all programming languages "C" and "C++" are mostly used Low level languages till today.
• Low level language is also divided into two parts are

- Machine language
- Assembly language

Machine Language

The first generation language developed for communicating with a Computer.
• It is written in machine code which represents 0 and 1 binary digits inside the Computer string which makes it easy to understand and perform the operations.
• As we know a Computer system can recognize electric signals so here 0 stands for turning off electric pulse and 1 stands for turning on electric pulse.
• It is very easy to understand by the Computer and also increases the processing speed.
Advantage : – no need of a translator or interpreter to translate the code, as the Computer can directly understand.
Disadvantage : – Remember the operation codes, memory address every time you write a program and also hard to find errors in a written program.

Assembly Language

The second generation programming language that has almost similar structure and set of commands as Machine language.
• Here we use words or names in English forms and also symbols.
• The programs that have been written using words, names and symbols in assembly language are converted to machine language using an Assembler.
• Because a Computer only understands machine code languages that's why we need an Assembler that can convert the Assembly level language to Machine language so the Computer gets the instruction and responds quickly.
Disadvantage : – it is written only for a single type of CPU and does not run on any other CPU. • But its speed makes it the most used low level language till today which is used by many programmers.

High Level Languages (Third generation languages/3GLs)

• The assembly language was easier to use compared with machine language as it relieved the programmer from a burden of remembering the operation – codes and addresses of memory location.
• Even though the assembly languages proved to be great help to the programmer, a search was continued for still better languages nearer to the conventional English language.
• The languages developed which were nearer to the English language, for the use of writing the programmer in 1960 were known as High Level languages.
• The different high level languages which can be used by the common user are FORTRAN, COBOL, BASIC, PASCAL, PL-1 and many others.

• Each high level language was developed to fulfil some basic requirements for particular type of problems.
• But further developments are made in each language to widen its utility for different purposes.

Fourth Generation Languages(4GLs)

• The 3GLs are procedural in nature i.e., HOW the problem get coded i.e., the procedures require the knowledge of how the problem will be solved.
• Contrary to them, 4GLs are non procedural.
• That is only WHAT of the problem is coded i.e., only 'What is required' is to be specified and rest gets done on its own. • Thus a big program of a 3GLs may get replaced by a single statement of a 4GLs.
• The main aim of 4GLs is to cut down on development and maintenance time and making it easier for users.

Program Development

When we want to develop a program using any programming language, we follow a sequence of steps. These steps are called phases in program development. The program development life cycle is a set of steps that are used to develop a program in any language. Generally, the program development life cycle contains 6 phases, they are as follows….

- Problem Definition
- Problem Analysis
- Algorithm Development
- Coding & Documentation
- Testing & Debugging
- Maintenance

1. Problem Definition

In this phase, we define the problem statement and we decide the boundaries of the problem. In this phase we need to understand the problem statement, what is our requirement, what should be the output of the problem solution. These are defined in this first phase of the program development life cycle.

2. Problem Analysis

In phase 2, we determine the requirements like variables, functions, etc. to solve the problem. That means we gather the required resources to solve the problem defined in the problem definition phase. We also determine the bounds of the solution.

3. Algorithm Development

During this phase, we develop a step by step procedure to solve the problem using the specification given in the previous phase. This phase is very important for program development. That means we write the solution in step by step statements.

4. Coding & Documentation

This phase uses a programming language to write or implement the actual programming instructions for the steps defined in the previous phase. In this phase, we construct the actual

program. That means we write the program to solve the given problem using programming languages like C, C++, Java, etc.,

5. Testing & Debugging

During this phase, we check whether the code written in the previous step is solving the specified problem or not. That means we test the program whether it is solving the problem for various input data values or not. We also test whether it is providing the desired output or not.

6. Maintenance

During this phase, the program is actively used by the users. If any enhancements found in this phase, all the phases are to be repeated to make the enhancements. That means in this phase, the solution (program) is used by the end-user. If the user encounters any problem or wants any enhancement, then we need to repeat all the phases from the starting, so that the encountered problem is solved or enhancement is added.

ALGORITHM:

The word "algorithm" relates to the name of the mathematician Al-khowarizmi, which means a procedure or a technique. Software Engineer commonly uses an algorithm for planning and solving the problems. An algorithm is a sequence of steps to solve a particular problem or algorithm is an ordered set of unambiguous steps that produces a result and terminates in a finite time

Algorithm has the following characteristics

- **Input**: An algorithm may or may not require input.
- **Output**: Each algorithm is expected to produce at least one result.
- **Definiteness**: Each instruction must be clear and unambiguous.
- **Finiteness**: If the instructions of an algorithm are executed, the algorithm should terminate after finite number of steps.

The algorithm and flowchart include following three types of control structures.

1. Sequence: In the sequence structure, statements are placed one after the other and the execution takes place starting from up to down.

2. Branching (Selection): In branch control, there is a condition and according to a condition, a decision of either TRUE or FALSE is achieved. In the case of TRUE, one of the two branches is explored; but in the case of FALSE condition, the other alternative is taken. Generally, the 'IF-THEN' is used to represent branch control.

3. Loop (Repetition): The Loop or Repetition allows a statement(s) to be executed repeatedly based on certain loop condition e.g. WHILE, FOR loops.

Advantages of algorithm

• It is a step-wise representation of a solution to a given problem, which makes it easy to understand.
• An algorithm uses a definite procedure.
• It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
• Every step in an algorithm has its own logical sequence so it is easy to debug.

HOW TO WRITE ALGORITHMS

Step 1: Define your algorithms input: Many algorithms take in data to be processed, e.g. to calculate the area of rectangle input may be the rectangle height and rectangle width.
Step 2: Define the variables: Algorithm's variables allow you to use it for more than one place. We can define two variables for rectangle height and rectangle width as HEIGHT and WIDTH (or H & W). We should use meaningful variable name e.g. instead of using H & W use HEIGHT and WIDTH as variable name.
Step 3: Outline the algorithm's operations: Use input variable for computation purpose, e.g. to find area of rectangle multiply the HEIGHT and WIDTH variable and store the value in new variable (say) AREA. An algorithm's operations can take the form of multiple steps and even branch, depending on the value of the input variables.
Step 4: Output the results of your algorithm's operations: In case of area of rectangle output will be the value stored in variable AREA. if the input variables described a rectangle with a HEIGHT of 2 and a WIDTH of 3, the algorithm would output the value of 6.

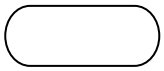
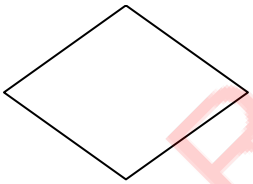
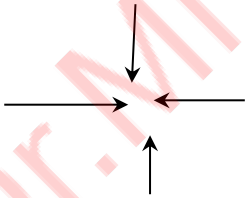
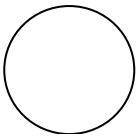
FLOWCHART

The first design of flowchart goes back to 1945 which was designed by John Von Neumann. Unlike an algorithm, Flowchart uses different symbols to design a solution to a problem. It is another commonly used programming tool. By looking at a Flowchart one can understand the operations and sequence of operations performed in a system. Flowchart is often considered as a blueprint of a design used for solving a specific problem.

Advantages of flowchart

• Flowchart is an excellent way of communicating the logic of a program.
• Easy and efficient to analyze problems using flowchart.
• During the program development cycle, the flowchart plays the role of a blueprint, which makes the program development process easier.
•After successful development of a program, it needs continuous timely maintenance during the course of its operation. The flowchart makes program or system maintenance easier.
• It is easy to convert the flowchart into any programming language code.

Flowchart is a diagrammatic /Graphical representation of the sequence of steps to solve a problem. To draw a flowchart following standard symbols are use

Symbol Name	Symbol	function
Oval		Used to represent start and end of flowchart
Parallelogram		Used for input and output operation
Rectangle		Processing: Used for arithmetic operations and Data manipulations
Diamond		Decision making. Used to represent the operation in which there are two/three alternatives, true and false etc
Arrows		Flow line Used to indicate the flow of logic by connecting symbols
Circle		Page Connector

DATA STRUCTURES

Data may be organised in many different ways logical or mathematical model of a program particularly organisation of data. This organised data is called “Data Structure”

Or

The organised collection of data is called a ‘Data Structure’.

Data Structure involves two complementary goals. The first goal is to identify and develop useful mathematical entities and operations and to determine what class of problems can be

solved by using these entities and operations. The second goal is to determine representation for those abstract entities to implement abstract operations on this concrete representation.