That's it! Teachers are NOT common people and common people are NOT teachers . Please don't choose to become a teacher until you re worth it!

DID YOU KNOW

Teachers in Germany have the highest salary in the country, and when judges, doctors and engineers asked the chancellor of Germany Angela Merkel for the same salary, she told them: "How can I compare you to those who taught you?".

Unit: One (Programming language) 10 Hrs.

1. Introduction to Programming Language
2. Type of Programming Language
3. Language Processor
4. Program Errors
5. Features of Good Program
6. Different Programming Paradigm,
7. Software Development Model
8. Program Development Life Cycle
9. System Design Tools

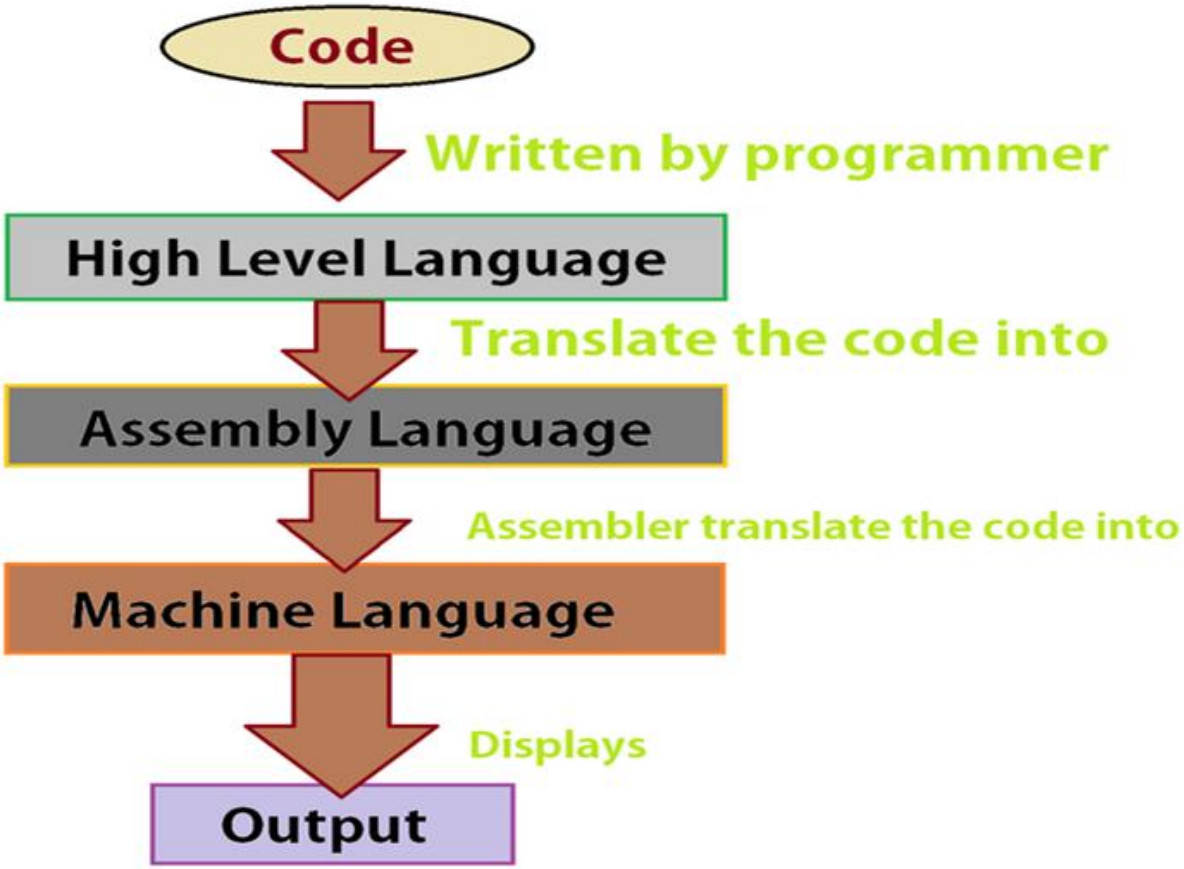
1. Introduction to Programming Language

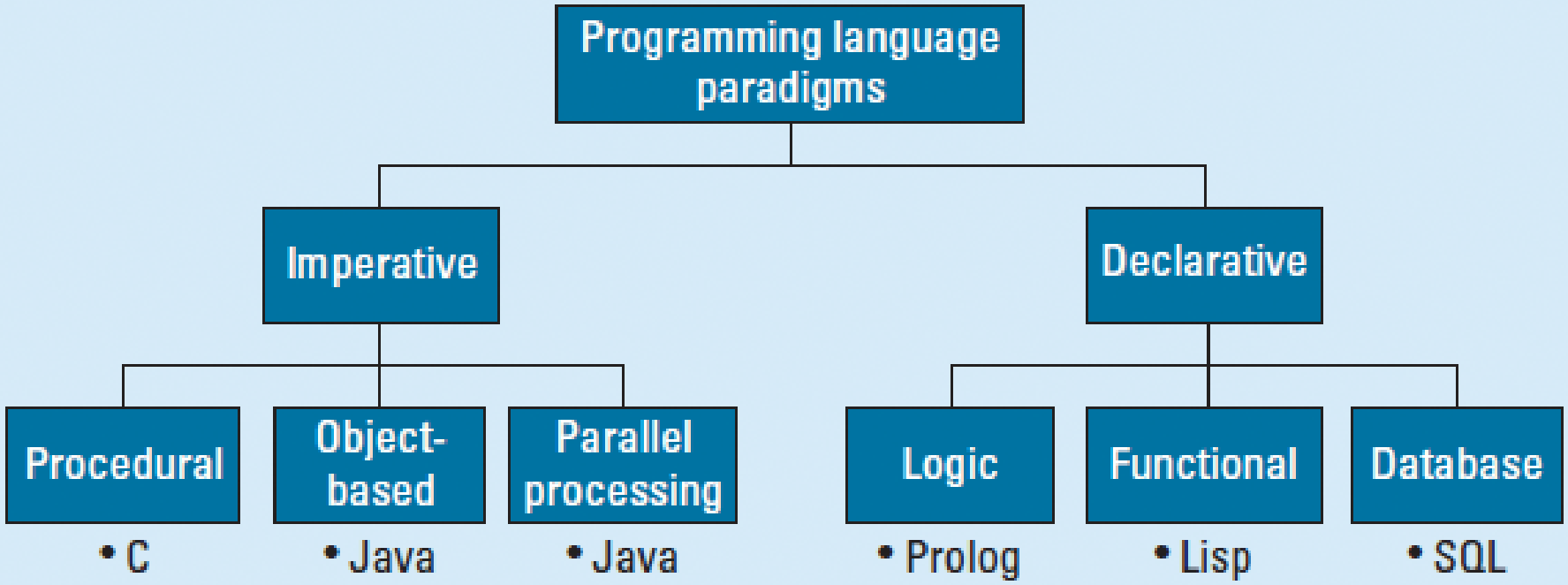
- Computers generally don't understand natural languages like Nepali, English, Japanese etc.
- The languages which are used to instruct the computer to perform certain tasks, are called computer programming languages.
- A programming language is a set of commands, instructions, and other syntax use to create a software program.
- Languages that programmers use to write code are called "high-level languages."
- The codes can be compiled into a "low-level language," which is recognized directly by the computer hardware.
- Programming Language is a set of rules that provides a way of instructing the computer to perform certain operations.

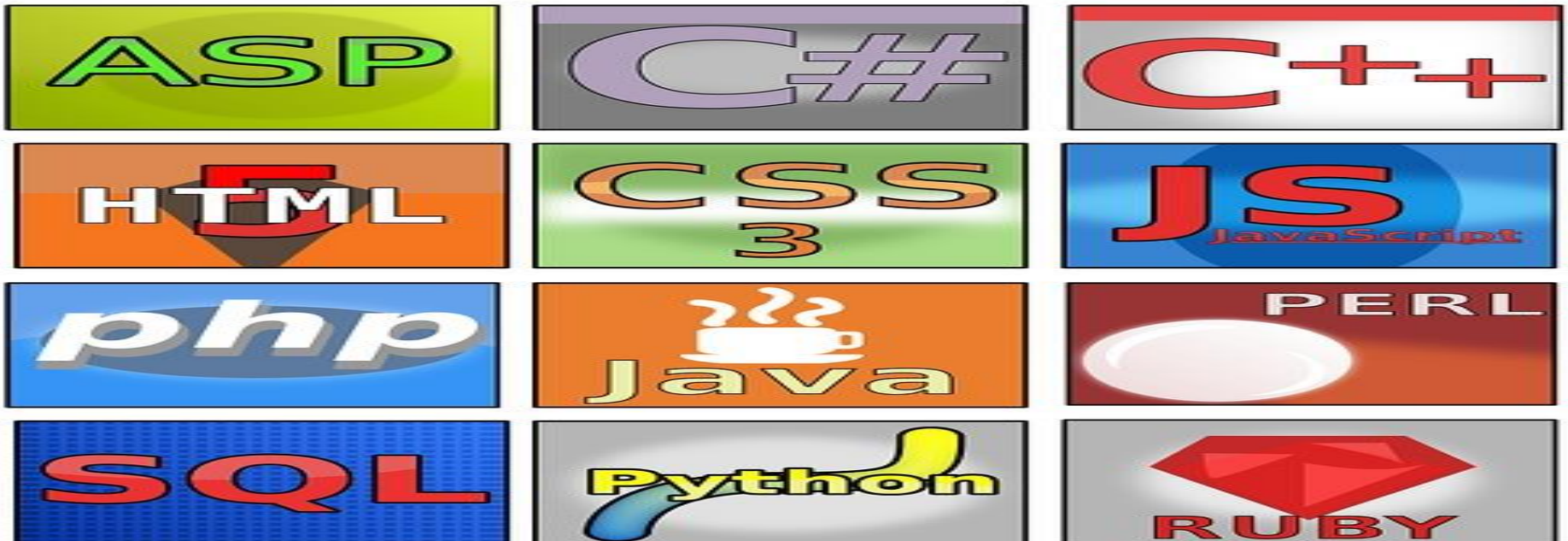
1. Introduction to Programming Language

- High-level languages are designed to be easy to read and understand. This allows programmers to write source code using logical words and symbols.
- For example, reserved words like function, while, if, and else are used in most major programming languages.
- Symbols like <, >, ==, and != are common operators.
- Many high-level languages are similar enough that programmers can easily understand source code written in multiple languages.
- Currently there are many programming languages in the world to write computer programs.
- A computer program is a set of instructions written in a specific programming language that a computer can interpret and execute.

- Examples of high-level languages include C++, Java, Perl, and PHP.
- Languages like C++ and Java are called "compiled languages".
- Languages like Perl and PHP are called "interpreted languages“.
- Generally, compiled languages are used to create software applications, while interpreted languages are used for running scripts, such as those used to generate content for dynamic websites.
- Low-level languages include Assembly Level Language (ALL) and Machine Level Language (MLL).
- An ALL contains a list of basic instructions and is much more difficult to read than a high-level language.
- An assembler can be used to translate the assembly code into machine code.
- The machine code, or Machine Level Language (MLL), contains a series of binary codes that are understood directly by a computer's CPU.
- Needless to say, machine language is not designed to be human readable.


Code
Written by programmer
High Level Language
Translate the code into
Assembly Language
Assembler translate the code into
Machine Language
Displays
Output


Programming language paradigms
Imperative
Declarative
Procedural
Object-based
Parallel processing
Logic
Functional
Database
• C
• Java
• Java
• Prolog
• Lisp
• SQL


ASP
C#
C++
HTML
CSS 3
JS
JavaScript
php
Java
PERL
SQL
Python
RUBY

2 . Types of Programming Language:

- Computer Programming Languages of basically two different types.
- They are:
  - LLL(Low Level Language) and HLL (High Level Language).
  - LLL are further divided into two different types. They are:
    - MLL (Machine Level Language) and
    - ALL (Assembly Level Language).
  - HLL are further divided into different categories. They are:
    - Algorithmic Language: FORTRAN, ALGOL, LISP etc.
    - Business Oriented Language: COBOL, SQL etc.
    - Education Oriented Language: BASIC, LOGO, Hypertalk etc.
    - Procedure Oriented Language: C, QBASIC etc.
    - Object Oriented Language: C++, Java, Visual Basic etc.
    - Natural Language: LISP, PROLOG etc.

LLL(Low Level Language)

- It is a computer programming language that is closer to the hardware. Before writing LLL program, one should know detailed internal architecture of computer and its instruction set.
- LLL are further divided into two different types.

They are:

- MLL (Machine Level Language) or 1st Generation Language
- The lowest level language which uses 1's and 0's that is directly understandable by CPU is known as machine language. It is very close to the hardware and also known as 1st Generation Language (1GL).
- Advantages of Machine Level Language are:

- Execute the program very fast.
- No translation of the program is required.

Disadvantages/Limitations of Machine Level Language are:

- Machine dependent due to different hardware architecture.
- Difficult to write program in terms of binary numbers, and it is almost impossible to remember the binary codes.
- Error in writing program because a programmer needs to remember all the machine codes.
- Difficult to error detection and correction.
- Bugs and errors correction is time consuming.
- Decreases the efficiency of the programmers mind remembering the machine language.

ALL (Assembly Level Language)

- It is also called 2nd Generation Language (2GL).
- It uses symbolic notations called mnemonic code to write a program. For example: ADD, SUB, MUL, DIV, MOD etc.
- The assembly program must be translated to machine code to execute by language translator program called assembler.
- Advantages of Assembly Level Language (ALL)

  - Assembly language uses mnemonic code which were easier to understand than binary code.

  - ALL is more user-friendly than machine level language.

  - Easy to understand and use than binary codes.

  - Can run more faster and use less memory space than any high level language.
- User can control on the hardware more easily by Assembly language.

Disadvantages of Assembly Level Language (ALL)

- The program written for one type of computer are not compatible for another type of computers. So, ALL are machine dependent.
- More costly and time consuming to write to write programs for individual computers.
- More specialized and trained person is required to handle the program written in assembly language.
- Difficult to learn and understand. Being, a machine dependent language, every type of computer architecture requires different assembly language, making difficult to understand.
- A single line in the HLL is equivalent to several lines in the assembly language. So, the development time is slower.
- No support for modern software engineering technology.

High Level Language (HLL)

- They are machine independent, transportable, user-friendly programming language and easier to read and understand. High Level Language increases the programmer efficiency.

Advantages of High Level Language

- A programmer/user does not need to be hardware expert in order to work with high level language.
- A program written in high level language can be used in different hardware platform. So, it is machine independent language.
- The language processors like interpreter and compiler are used to debug on occurrence of any errors.
- The development of the program is faster and the cost of the development is also relatively low.
- High level languages are easy to understand and user-friendly too.

Disadvantages of High Level Language

- The user / programmer is unable to control on the hardware system.
- It takes time and more space for execution.
- Slower in execution than Low Level Language.
- Low efficient as far as computation time is concerned.

- Procedure Oriented Language (3 GL)
- These are general purpose languages based on the idea of module. Programs are easy to test, debug, modify, and understand. For example: COBOL, FORTRAN, C etc.
- Problem Oriented Language (4 GL)
- This language is used to solve specific problem using English like syntax and easy to use. Coding is faster and easier. For example: MY SQL, C++, Java, etc.
- Natural Language (5 GL)
- The language used by fifth generation computers to interact like human is called natural language or fifth generation language (5 GL). It uses AI, neural networks and expert system that simulate human behavior by remembering earlier information. For example: LISP, PROLOG etc.

Language Processor

- A software that has the capacity of converting source code into object code. A computer understands instructions written only in object code (i.e. in the form of 0s and 1s).
- A special translator system used to translate the program written in high level language into machine code is called Language Processor.

I. Compiler:

The language processor that reads the complete source program written in high level language as a whole in one pass and translates it into an equivalent machine code is called Compiler.

- Programming languages like C, C++, C#, Java uses compiler.

II. Assembler:

The assembler is used to translate the program written in Assembly level language into machine code. The source program is an input of assembler.

Language Processor

III. Interpreter:

An interpreter is a computer program that is used to directly execute program instructions written using one of the many high-level programming languages. The interpreter transforms the high-level program into an intermediate language that it then executes, or it could parse the high-level source code and then performs the commands directly, which is done line by line or statement by statement. Computer Program called QBASIC (Quick Beginners All-Purpose Symbolic Instruction Code) uses Interpreter or it is also called interpreted version of BASIC.

2 . Types of Programming Language:

- Computer programming language, any of various languages for expressing a set of detailed instructions for a digital computer.
- Such instructions can be executed directly when they are in the computer manufacturer-specific numerical form known as machine language, after a simple substitution process when expressed in a corresponding assembly language, or after translation from some “higher-level” language.
- Although there are many computer languages, relatively few are widely used.
- MLL and ALL are “low-level,” requiring a programmer to manage explicitly all of computer’s features of data storage and operation.
- In contrast, HLL shield a programmer from worrying about such considerations and provide a notation that is more easily written and read by programmers.

Machine and assembly languages

- A machine language consists of the numeric codes for the operations that a particular computer can execute directly. The codes are strings of 0s and 1s, or binary digits (“bits”).
- Machine language instructions typically use some bits to represent operations, such as addition, and some to represent operands, or the location of the next instruction.
- Machine language is difficult to read and write, and its codes vary from computer to computer.

Assembly Level Language (ALL)

- Assembly language is one level above machine language.
- It uses short mnemonic codes for instructions and allows the programmer to introduce names for blocks of memory that hold data.
- Assembly language is designed to be easily translated into machine language.
- Like machine language, assembly language requires detailed knowledge of internal computer architecture.

Algorithmic languages

- Algorithmic languages are designed to express mathematical or symbolic computations. They can express algebraic operations in notation similar to mathematics and allow the use of subprograms that package commonly used operations for reuse. They were the first high-level languages.
- FORTRAN: The first important algorithmic language was FORTRAN(Formula Translation), designed in 1957 by an IBM team led by John Backus.
- It was intended for scientific computations with real numbers and collections of them organized as one- or multidimensional arrays.
- Its control structures included conditional IF statements, repetitive loops (so-called DO loops), and a GOTO statement that allowed non-sequential execution of program code.

ALGOL:

- ALGOL (algorithmic language) was designed by a committee of American and European computer scientists during 1958–60 for publishing algorithms, as well as for doing computations.
- Like LISP, ALGOL had recursive subprograms—procedures that could use themselves to solve a problem by reducing it to a smaller problem of the same kind.
- ALGOL introduced block structure, in which a program is composed of blocks that might contain both data and instructions and have the same structure as an entire program.
- ALGOL was widely used in Europe, and for many years it remained the language in which computer algorithms were published.
- Many important languages, such as Pascal and Ada, are its descendants.

LISP:

- LISP (list processing) was developed about 1960 by John McCarthy at the Massachusetts Institute of Technology (MIT).
- LISP uses a very simple notation in which operations and their operands are given in a parenthesized list.
- For example, (+ a (* b c)) stands for a + b*c.
- LISP also uses the list structure to represent data, it is easy for a LISP program to operate on other programs as data.
- LISP became a common language for Artificial Intelligence (AI) programming.

Business-oriented languages

- COBOL: COBOL (common business oriented language) has been heavily used by businesses since its inception in 1959.
- A committee of computer manufacturers and users and U.S. government organizations established CODASYL (Committee on Data Systems and Languages) to develop and oversee the language standard in order to ensure its portability across diverse systems.
- COBOL uses an English-like notation.
- COBOL introduced the record data structure for record clusters heterogeneous data such as a name, ID number, age, and address into a single unit.

SQL

- SQL (structured query language) is a language for specifying the organization of databases (collections of records).
- Databases organized with SQL are called relational because SQL provides the ability to query a database for information that falls in a given relation.
- For example, a query might be “find all records with both last_name Smith and city New York.”
- Commercial database programs commonly use a SQL-like language for their queries.

Education-oriented languages

- BASIC: BASIC (beginner's all-purpose symbolic instruction code) was designed at Dartmouth College in the mid-1960s by John Kemeny and Thomas Kurtz.
- It was intended to be easy to learn, particularly non-computer science majors, and to run well on a time-sharing computer with many users.
- It had simple data structures and notation and it was interpreted: a BASIC program was translated line-by-line, which made it easy to locate programming errors.
- Its small size and simplicity also made BASIC a popular language for early personal computers.
- Its recent forms have adopted many of the data and control structures of other contemporary languages, which makes it more powerful but less convenient for beginners.

Pascal:

- About1970, Niklaus Wirth of Switzerland designed Pascal to teach structured programming, which emphasized the orderly use of conditional and loop control structures without GOTO statements.
- Although Pascal resembled ALGOL in notation, it provided the ability to define data types with which to organize complex information, a feature beyond the capabilities of ALGOL as well as FORTRAN and COBOL.
- User-defined data types allowed the programmer to introduce names for complex data, which the language translator could then check for correct usage before running a program.
- During the late 1970s and ’80s, Pascal was one of the most widely used languages for programming instruction.

Logo

- Logo originated in the late 1960s.
- It had a more conventional syntax.
- The name came from an early project to program a turtle like robot.
- Turtle graphics used body-centered instructions, in which an object was moved around a screen by commands, such as “left 90” and “forward,” that specified actions relative to the current position and orientation of the object rather than in terms of a fixed framework.
- Together with recursive routines, this technique made it easy to program intricate and attractive patterns.

Hypertalk:

- Hypertalk was designed as “programming for the rest of us” by Bill Atkinson for Apple’s Macintosh.
- Using a simple English-like syntax, Hypertalk enabled anyone to combine text, graphics, and audio quickly into “linked stacks” that could be navigated by clicking with a mouse on standard buttons supplied by the program.
- Hypertalk was particularly popular among educators in the 1980s and early ’90s for classroom multimedia presentations.

Object-oriented languages:

- Object-oriented languages help to manage complexity in large programs.
- Objects package data and the operations on them so that only the operations are publicly accessible and internal details of the data structures are hidden.
- This information hiding made large-scale programming easier by allowing a programmer to think about each part of the program in isolation.
- Object-oriented programming began with the Simula language (1967), which added information hiding to ALGOL.
- Another influential object-oriented language was Smalltalk (1980), in which a program was a set of objects that interacted by sending messages to one another.

C++

- The C++ language, developed by Bjarne Stroustrup at AT&T in the mid-1980s, extended C by adding objects to it while preserving the efficiency of C programs.
- It has been one of the most important languages for both education and industrial programming.
- Large parts of many operating systems, such as the Microsoft Corporation's Windows 98, were written in C++.

Ada

- Ada was named for Augusta Ada King, countess of Lovelace, who was an assistant to the 19th-century English inventor Charles Babbage, and is sometimes called the first computer programmer.
- Ada, the language, was developed in the early 1980s for the U.S. Department of Defense for large-scale programming.
- While no longer mandated for use in work for the Department of Defense, Ada remains an effective language for engineering large programs.

Java

- In the early 1990s, Java was designed by Sun Microsystems, Inc., as a programming language for the World Wide Web (WWW).
- Although it resembled C++ in appearance, it was fully object-oriented.
- In order to be portable, Java programs are translated by a Java Virtual Machine specific to each computer platform, which then executes the Java program.
- In addition to adding interactive capabilities to the Internet through Web “applets,” Java has been widely used for programming small and portable devices, such as mobile telephones.

Visual Basic

- Visual Basic was developed by Microsoft to extend the capabilities of BASIC by adding objects and “event-driven” programming: buttons, menus, and other elements of graphical user interfaces (GUIs).
- Visual Basic can also be used within other Microsoft software to program small routines.

Declarative Languages

- Declarative languages, also called nonprocedural or very high level, are programming languages in which a program specifies what is to be done rather than how to do it.
- In such languages there is less difference between the specification of a program and its implementation than in the procedural languages described so far.
- The two common kinds of declarative languages are logic and functional languages.

Logic programming languages

- PROLOG (programming with logic) is the best known, state a program as a set of logical relations (e.g., a grandparent is the parent of a parent of someone).
- Such languages are similar to the SQL database language.
- PROLOG has been used extensively in natural language processing and other AI programs.

Functional languages

- Functional languages have a mathematical style.
- A functional program is constructed by applying functions to arguments.
- Functional languages, such as LISP, ML, and Haskell, are used as research tools in language development, in automated mathematical theorem provers, and in some commercial projects.

Scripting languages

- Scripting languages are sometimes called little languages. They are intended to solve relatively small programming problems that do not require the overhead of data declarations and other features to make large programs manageable.
- Scripting languages are used for writing operating system utilities, for special-purpose file-manipulation programs, and, because they are easy to learn, sometimes for considerably larger programs.
- PERL (practical extraction and report language) was developed in the late 1980s. It was intended to have all the capabilities of earlier scripting languages. In the 1990s it became popular as a system-programming tool, both for small utility programs and for prototypes of larger ones.

TeX

- TeX was developed during 1977–86 as a text formatting language by Donald Knuth, a Stanford University professor, to improve the quality of mathematical notation in his books.
- Text formatting systems, unlike WYSIWYG ("What You See Is What You Get") word processors, embed plain text formatting commands in a document, which are then interpreted by the language processor to produce a formatted document for display or printing. TeX marks italic text, for example, as {\it this is italicized}, which is then displayed as this is italicized.
- TeX largely replaced earlier text formatting languages. Its powerful and flexible abilities gave an expert precise control over such things as the choice of fonts, layout of tables, mathematical notation, and the inclusion of graphics within a document. TeX contains numerous standard "style sheets" for different types of documents, and these may be further adapted by each user.

PostScript

- PostScript is a page-description language developed in the early 1980s by Adobe Systems Incorporated on the basis of work at Xerox PARC (Palo Alto Research Center). Such languages describe documents in terms that can be interpreted by a personal computer to display the document on its screen or by a microprocessor in a printer or a typesetting device.
- PostScript commands can, for example, precisely position text, in various fonts and sizes, draw images that are mathematically described, and specify color or shading. PostScript uses postfix, also called reverse Polish notation, in which an operation name follows its arguments. Although PostScript can be read and written by a programmer, it is normally produced by text formatting programs, word processors, or graphic display tools.

SGML

- SGML (standard generalized markup language) is an international standard for the definition of markup languages; that is, it is a meta language. Markup consists of notations called tags that specify the function of a piece of text or how it is to be displayed. SGML emphasizes descriptive markup, in which a tag might be "<emphasis>." Such a markup denotes the document function, and it could be interpreted as reverse video on a computer screen, underlining by a typewriter, or italics in typeset text.
- SGML is used to specify DTDs (document type definitions). A DTD defines a kind of document, such as a report, by specifying what elements must appear in the document—e.g., <Title>—and giving rules for the use of document elements, such as that a paragraph may appear within a table entry but a table may not appear within a paragraph. A marked-up text may be analyzed by a parsing program to determine if it conforms to a DTD. Another program may read the markups to translate the document into PostScript for printing, generate audio for readers with visual or hearing disabilities.

World Wide Web, Display Languages, HTML

- The World Wide Web is a system for displaying text, graphics, and audio retrieved over the Internet on a computer monitor. Each retrieval unit is known as a Web page, and such pages frequently contain “links” that allow related pages to be retrieved. HTML (hypertext markup language) is the markup language for encoding Web pages. It was designed by Tim Berners-Lee at the CERN nuclear physics laboratory in Switzerland during the 1980s and is defined by an SGML DTD. HTML markup tags specify document elements such as headings, paragraphs, and tables. They mark up a document for display by a computer program known as a Web browser. The browser interprets the tags, displaying the headings, paragraphs, and tables in a layout that is adapted to the screen size and fonts available to it.

HTML

- HTML documents also contain anchors, which are tags that specify links to other Web pages. An anchor has the form <A HREF= “http://www.britannica.com”> Encyclopædia Britannica</A>, where the quoted string is the URL (universal resource locator) to which the link points (the Web “address”) and the text following it is what appears in a Web browser, underlined to show that it is a link to another page. What is displayed as a single page may also be formed from multiple URLs, some containing text and others graphics.

XML

- HTML does not allow one to define new text elements; that is, it is not extensible. XML (extensible markup language) is a simplified form of SGML intended for documents that are published on the Web. Like SGML, XML uses DTDs to define document types and the meanings of tags used in them. XML adopts conventions that make it easy to parse, such as that document entities are marked by both a beginning and an ending tag, such as <BEGIN>…</BEGIN>. XML provides more kinds of hypertext links than HTML, such as bidirectional links and links relative to a document subsection.
- Because an author may define new tags, an XML DTD must also contain rules that instruct a Web browser how to interpret them—how an entity is to be displayed or how it is to generate an action such as preparing an e-mail message.

Web scripting

- Web pages marked up with HTML or XML are largely static documents. Web scripting can add information to a page as a reader uses it or let the reader enter information that may, for example, be passed on to the order department of an online business. CGI (common gateway interface) provides one mechanism; it transmits requests and responses between the reader's Web browser and the Web server that provides the page. The CGI component on the server contains small programs called scripts that take information from the browser system or provide it for display. A simple script might ask the reader's name, determine the Internet address of the system that the reader uses, and print a greeting. Scripts may be written in any programming language, but, because they are generally simple text-processing routines, scripting languages like PERL are particularly appropriate.

- Another approach is to use a language designed for Web scripts to be executed by the browser. JavaScript is one such language, designed by the Netscape Communications Corp., which may be used with both Netscape's and Microsoft's browsers. JavaScript is a simple language, quite different from Java. A JavaScript program may be embedded in a Web page with the HTML tag <script language="JavaScript">.
- JavaScript instructions following that tag will be executed by the browser when the page is selected. In order to speed up display of dynamic (interactive) pages, JavaScript is often combined with XML or some other language for exchanging information between the server and the client's browser. In particular, the XML Http Request command enables asynchronous data requests from the server without requiring the server to resend the entire Web page. This approach, or "philosophy," of programming is called Ajax (asynchronous JavaScript and XML).

- VB Script is a subset of Visual Basic. Originally developed for Microsoft's Office suite of programs, it was later used for Web scripting as well. Its capabilities are similar to those of JavaScript, and it may be embedded in HTML in the same fashion.
- Behind the use of such scripting languages for Web programming lies the idea of component programming, in which programs are constructed by combining independent previously written components without any further language processing. JavaScript and VB Script programs were designed as components that may be attached to Web browsers to control how they display information.

Interpreter vs Compiler

SN	Interpreter	SN	Compiler
1.	It takes source code of HLL to convert into MLL line by line.	1.	It takes entire source code of HLL to covert into MLL at once.
2.	It takes less amount of time to analyze the source code.	2.	It takes large amount of time to analyze the entire source code.
3.	The execution time of program by interpreter is slower.	3.	The execution time of program using compiler is comparatively faster.
4.	Its debugging is easier as it translates line by line.	4.	Its debugging is comparatively hard as the error can be present anywhere within a program.
5.	No immediate object code is generated.	5.	Generates immediate object code.
6.	For examples: QBASIC, Python, Perl etc.	6.	For examples: C, C++, Java etc.

1.3 Features of Good Program

- Every computer requires proper and correct instructions to obtain desired output. A program should be written in such a way that it is easier to understand the underlying logic.
- A good program should have following characteristics:

  1. Portability: The ability of an application to run on different operating systems with or without minimal charges is portability. If a program is developed for a platform, the life span of the program is severely affected.

  2. Readability: If a program is written structurally, it helps users or programmers to understand their own program in better way without much effort increasing computational efficiency and to create a user-friendly environment.

3. Efficiency: A good program is efficient when it utilizes the least amount of memory and processing time for instructions and data.

4. Structural: If a program is developed structurally, it becomes more readable, and the testing and documentation process also gets easier.

5. Flexibility: A program should be flexible to handle most of the changes without having to rewrite the entire program without having to reconstruct the entire application.

6. Generality: A program developed for a particular task should also be used for all similar tasks of the same domain. If a program is developed for a particular college, it should suit all the other similar colleges.

7. Documentation: Documentation is one of the most important component of application development. A well-documented applications is useful for other programmers as well as end users because even in the absence of the author, they can understand it.

1.4 Programming Paradigm

Programming Paradigm or Programming Method is an approach to solve problems using some programming language or tools and techniques that are available to us following some approach. There are various programming languages as well as paradigms to fulfill each and every demand. Some of the programming paradigms are discussed below:

1.4.1. Imperative Programming Paradigm:

- It is the oldest programming paradigm.
- It describes a sequence of steps and tells the computer "how" to accomplish the task.
- Very simple to implement.
- It contains loops, variables etc.
- Complex programs cannot be solved.
- Parallel programming is not possible.

1.4 Programming Paradigm

1.4.1. Imperative Programming Paradigm:

Imperative programming paradigm is further divided into two types: PPP and OOP.

A) Procedural Programming Paradigm (PPP):

- It uses top-down approach.
- It has the ability to reuse the code.

B) Object Oriented Paradigm (OOP):

- They uses classes and objects (basic entities) for communication.
- More emphasis is on data rather than procedure.
- It can handle all kinds of real life problems of today providing features of data security, inheritance, code reusability, flexibility, data abstraction etc.

1.4 Programming Paradigm

1.4.2. Declarative Programming Paradigm:

It is divided into Logic, Functional and Database. The only difference between imperative (how to do) and declarative (what to do) programming paradigm.

A) Logical Programming Paradigm (LPP):

- It would solve logical problems like puzzles, series etc.
- The execution of the program is very much like proof of mathematical statements. For eg: PROLOG.

B) Functional Programming Paradigm (FPP):

- It is language independent and based on mathematics.
- It is used for the execution of series of mathematical functions.
- Functions can be replaced with their values without changing the meaning.
- Languages like perl and javascript mostly uses this paradigm.

1.4 Programming Paradigm

1.4.2. Declarative Programming Paradigm:

C) Database/Data Driven Programming Paradigm (DPP):

- It is based on data and its movement.
- It is a heart of business information system.
- It provides file creation, data entry, update, query and reporting functions.
- Languages like SQL mostly use this paradigm for database application.

1.5 Program Development Life Cycle

The program development life cycle is a set of steps that are used to develop a program in any programming language.

- It is a systematic way of developing quality software.
- It provides an organized plan for breaking down the task before moving on to the next phase.

Steps used in programming development are:

1. Problem Analysis
2. Algorithm Development
3. Drawing Flowchart
4. Writing Source Code (Coding)
5. Compilation and Execution
6. Debugging and Testing
7. Support and Maintenance
8. Documentation

1.5 Program Development Life Cycle

Steps used in programming development are:

1. Problem Analysis: During this stage, the system analysts and programmers communicate with users in order to fully understand what the software should do. The program's inputs, outputs, different ways of solving problems, software and hardware requirements and available time period should be known in advance.

- It consists at least following tasks:
- Define objectives of the program.
- Determine desired outputs.
- Determine input requirements.
- Determine processing requirements.
- Evaluate feasibility of the program.
- Document the analysis.

1.5 Program Development Life Cycle

Steps used in programming development are:

2. Algorithm Development: An algorithm is finite set of instructions to perform any particular task in finite number of steps obtaining values as input, correct set of output, simple steps, precise rules, finite time and steps to get output.

Write an algorithm for finding the sum of any two numbers.

Step 1: Start

Step 2: Display “Enter two numbers”

Step 3: Read A and B

Step 4: Add A and B, store the sum in C

Step 5: Display “C as sum of two numbers”

Step 6: Stop

1.5 Program Development Life Cycle

Write an algorithm for calculating the simple interest.

Step 1: Start

Step 2: Display “Enter values of P, T and R”

Step 3: Read the first value as P.

Step 4: Read the second value as T.

Step 5: Read the third value as R.

Step 6: Calculate simple interest using formula, P, T and R are multiplied. The product of P, T and R are divided by 100 and the result is stored in SI. [SI = (P*T*R)/100]

Step 7: Display SI as simple interest

Step 8: Stop

1.5 Program Development Life Cycle

Write an algorithm to test whether the number is odd or even?

Step 1: Start

Step 2: Display “Enter a number for even or odd”

Step 3: Read NUM for user

Step 4: Calculate remainder REM using modular division of NUM by 2

Step 5: If REM = 0, THEN

Print “The number is even”

ELSE

Print “The number is odd”

Step 6: Stop

1.5 Program Development Life Cycle

3. Drawing Flowchart: Flowchart is graphical representation of an algorithm using special-purpose symbols connected by arrows.

Symbols used in flowcharts.

A flowchart uses different symbols to denote different types of instructions. The symbols are listed below.

Name of Symbol	Use in flowchart
Oval or Rounded Rectangle	Denote the beginning / start or end of a program
Flow line	Denotes the direction of logic flow in a program
Parallelogram	Denotes input or an output operation
Rectangle	Denotes a process to be carried out
Diamond	Denotes a decision to be made or branch that should continue along one of two routes (e.g. IF / THEN /ELSE)
Connector	Connects parts of a flowchart

1.5 Program Development Life Cycle

4. Writing Source Code (Coding)

The coding is the process of using the logic into a computer language format. This stage translates the program design into computer instructions using some programming languages like: C, C++, Java, Python etc. Coding must eliminate all syntax and logical errors.

5. Compilation and Execution

The code of the program written in HLL should be translated into MLL. The translation process is called compilation; which is done by special software called compiler. Object code is the executable file that is compatible with particular platform (Windows, Linux etc.)

1.5 Program Development Life Cycle

6. Debugging and Testing

Whenever error appears, debugging is necessary for the correction of programming errors. Different debugging tools are simulators, logic analyzers, breakpoints, etc. Testing ensures that program performs the required task correctly.

7. Support and Maintenance:

In this phase, the solution is used by the end user. If the user encounters any problem or wants any enhancement, then all the phases will be repeated from the starting, so that the encountered problem is solved or enhancement is added.

1.5 Program Development Life Cycle

8. Documentation

It is the description of the program and its logic written to support understanding the program. There are two types of documentation. They are:

- Programmer's documentation (Technical documentation)

  It is prepared for future references to the programmers who maintain, redesign and upgrade the system.

- User Documentation (User Manual)

  It provides support to the end user of the program.

  The requirements of the user are always changing.

The above steps in software development are repeated for n-cycles to fulfill the demand of user. So, it is called Life Cycle.

1.6 System Design Tools

1. DFD (Data Flow Diagram)

- It is a pictorial representation of the flow of data through a process / sub-process.
- It consists of data flows, processes, sources, destination and store, each of which are described by easily understandable symbols.
- It shows external entities, internal entities, their relationship and their flow direction.

The key elements of DFD are as follows:

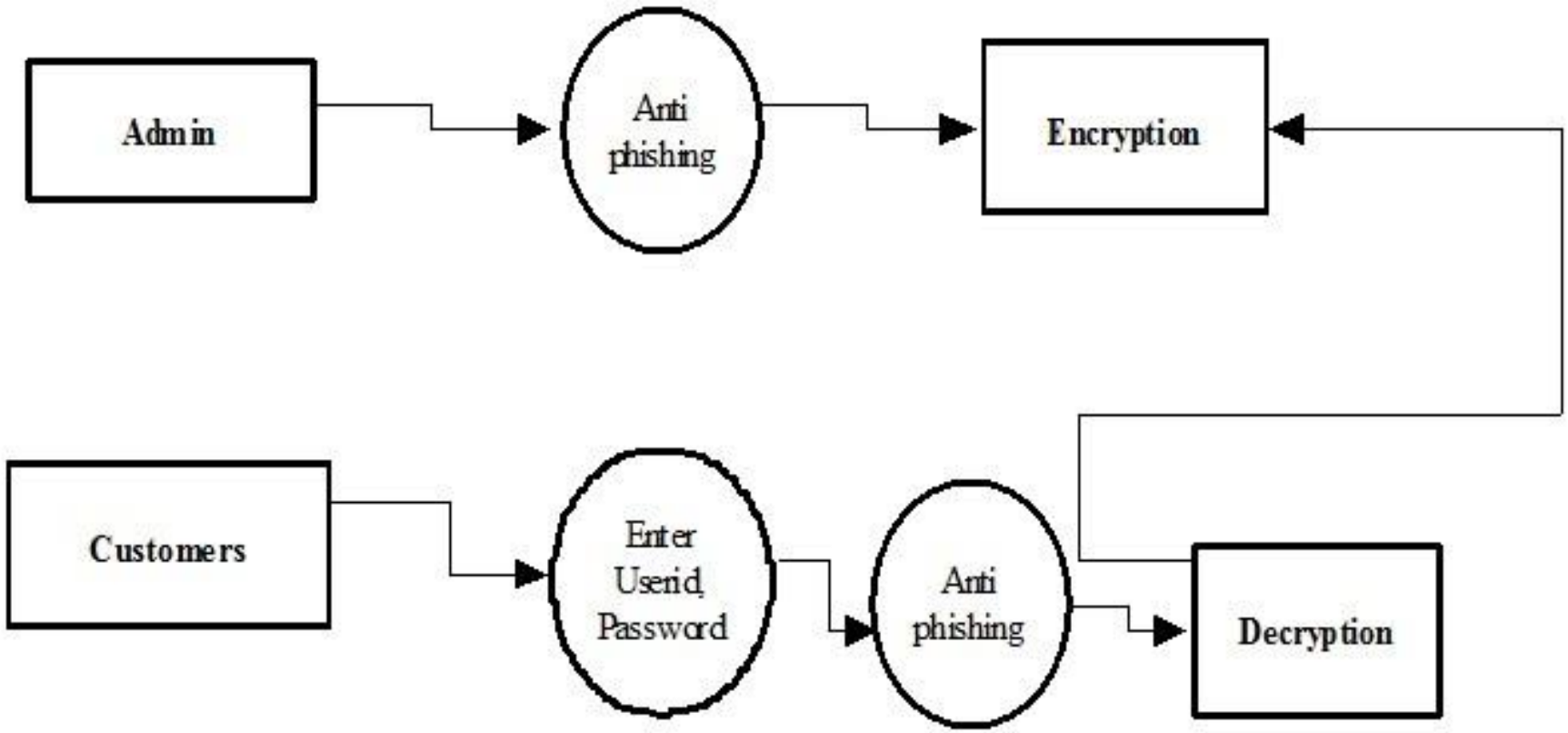
A) Process: A process is represented by a circle, and it transforms an input data flow into an output data flow.

B) Data Flow: It is indicated by a line with an arrow which shows the input or output of data or information.

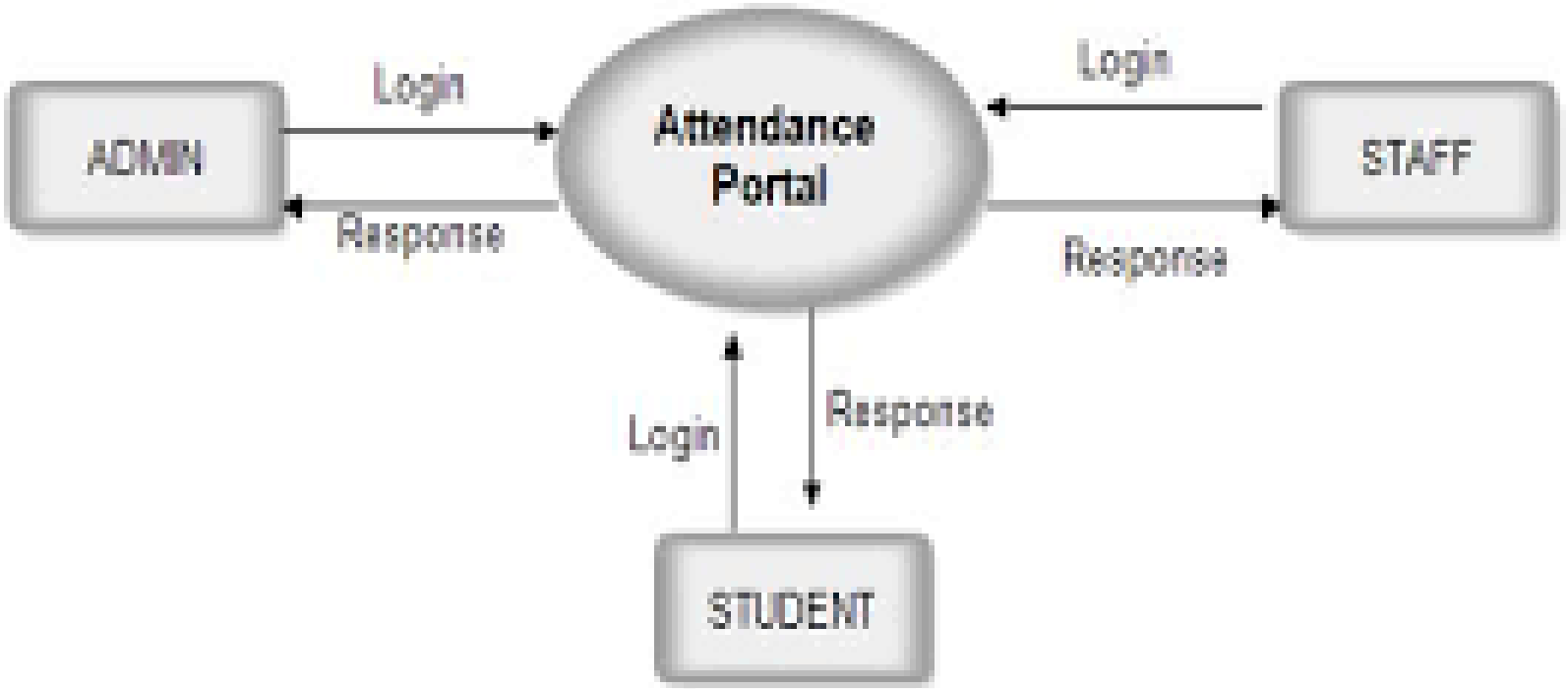
C) Data Store: A data stores is shown as open ended rectangle or slant line.

D) External Entity: They are represented through the rectangle used for the presentation of source and destination. For example: A person, organization or a system etc.

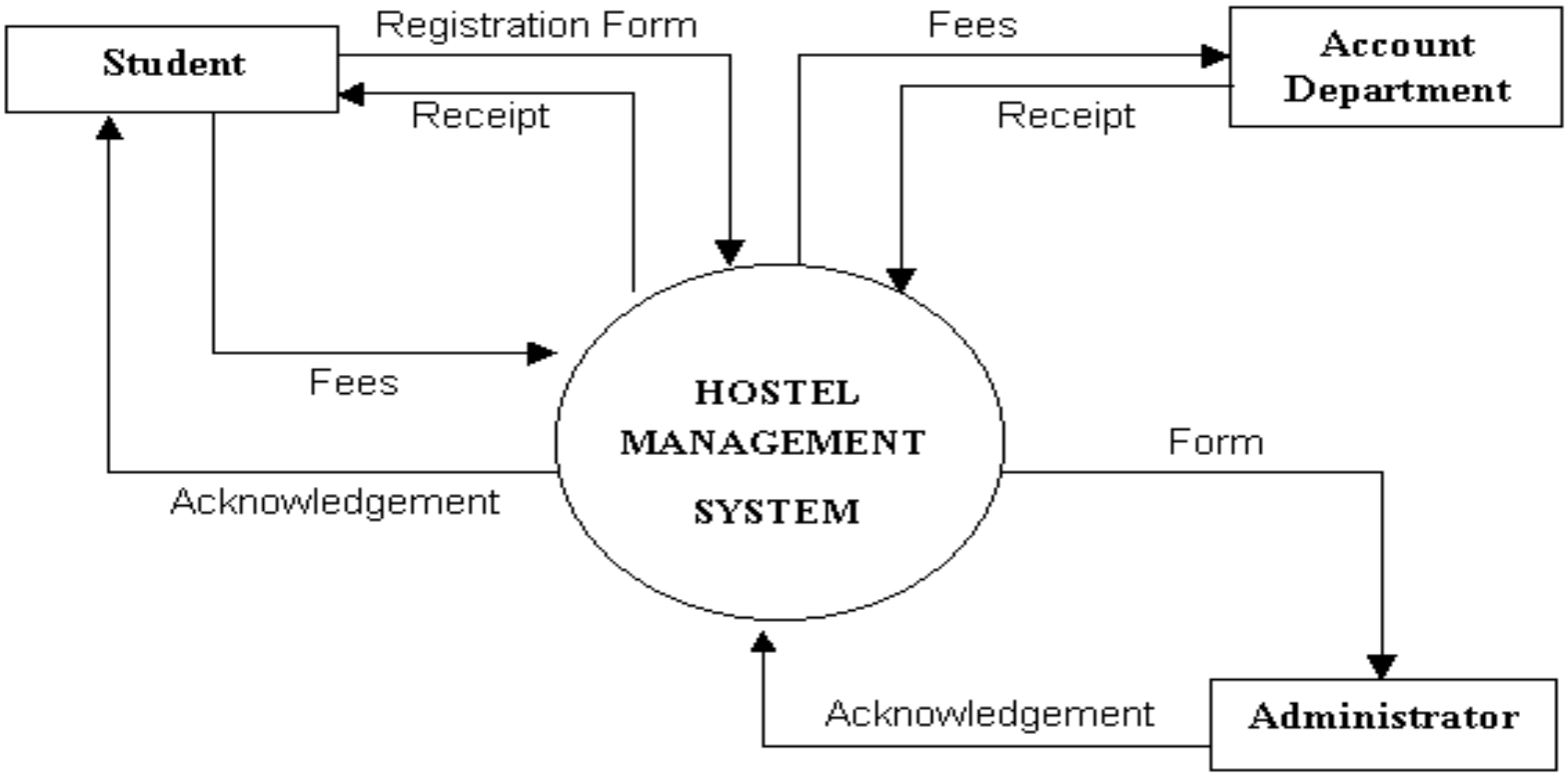
DFD OF BANK MANAGEMENT SYSTEM



0 - Level DFD : Context Level



Context Level Diagram for Hostel Management System



Context Diagram

- This diagram gives the system overview.
- It is top level DFD (level 0) that treats the whole system as a single process with all its inputs, outputs, links and sources without any sub-process in context diagram.
- After the context diagram is broken down into many sub-process into many sub-systems, then the output is DFD.

Decision Table

It is a table that defines a logical procedure by means of a set of conditions and related actions. It consists of two parts. They are:

- Conditions
- Actions
- Decision table is preferred when one of the large number of actions is to be selected
- The action selected depends upon a large number of conditions

Decision table for diagnosing printer problem is given below:

		Rules				
Conditions	Printer doesn't print	Y	Y	-	-	Y
	A red light is flashing	N	-	-	Y	-
	Printer is unrecognized	Y	Y	Y	-	N
Actions	Check the power cable	X				
	Check the printer-computer cable		X			
	Ensure printer software installed			X		
	Replace ink				X	
	Check for paper jam					X

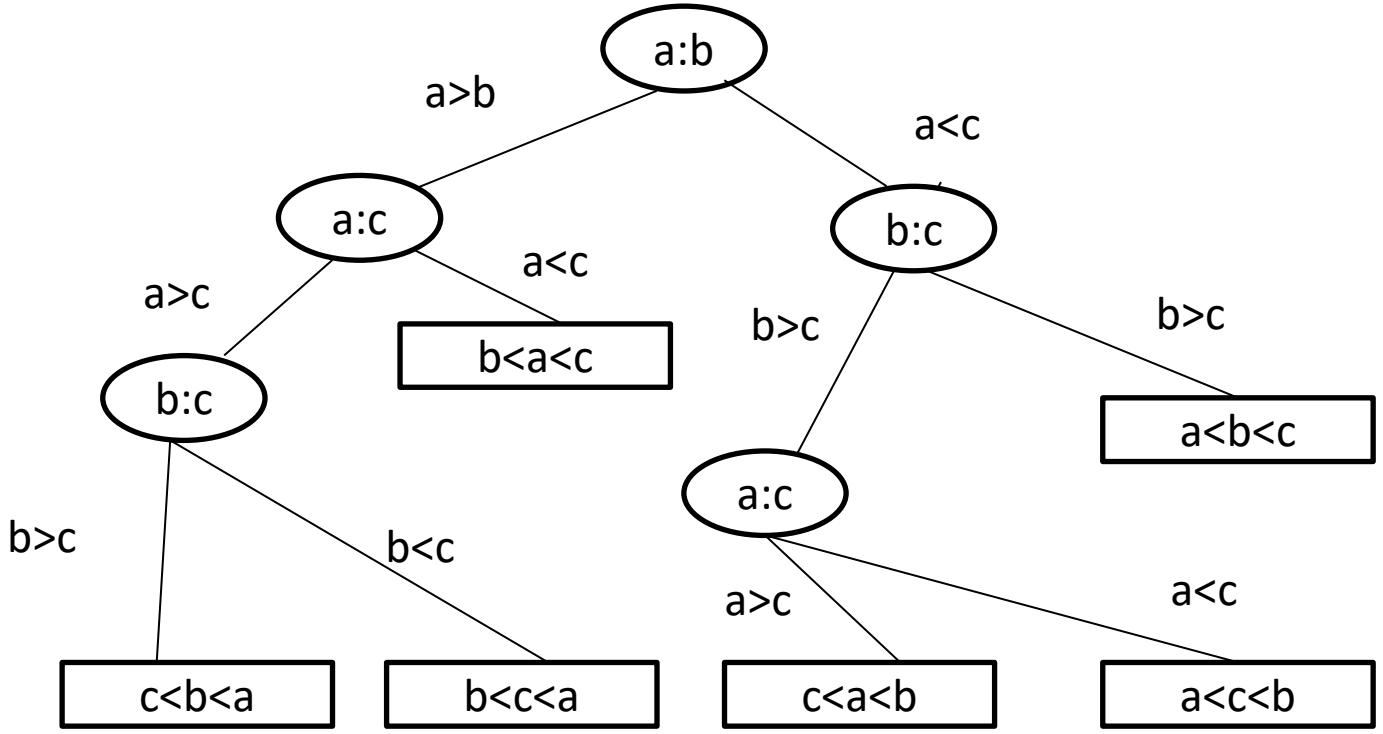
Decision table for diagnosing printer problem is given below:

Requirement Number					
Condition	1	2	3	4	5
User is authorized	F	T	T	T	T
Chemical is available	—	F	T	T	T
Chemical is hazardous	—	—	F	T	T
Requester is trained	—	—	—	F	T
Action					
Accept request			X		X
Reject request	X	X		X	

Decision Tree

- It is a decision support tool that uses a tree like graph or model of decisions.
- It is used to identify the path of actions based on conditions, except that it follows the tree structure and each node of tree denotes conditions and leaf node denote action.
- It is user friendly because it provides a graphical hierarchical diagram view of the conditions and actions.

- A decision tree for sorting three distinct elements, in which internal node correspondents to decision and sub tree of these vertices are possible outcome of the decision.
- Decision tree for identifying smallest among three numbers.



Software Process Model

- Software Processes is a coherent set of activities for specifying, designing, implementing and testing software systems
- A software process model is an abstract representation of a process that presents a description of a process from some particular perspective

There are many different software processes but all involve:

- Specification – defining what the system should do;
- Design and implementation – defining the organization of the system and implementing the system;
- Validation – checking that it does what the customer wants;
- Evolution – changing the system in response to changing customer needs.

Waterfall Model

- It is the simplest and linear order software development model
- This model takes the fundamental process activities such as requirement specification, design, implementation, testing and so on

The phases in waterfall model are shown as follows:

- Requirements analysis and definition
- System and software design
- Implementation and unit testing
- Integration and system testing
- Operation and maintenance

Waterfall Model

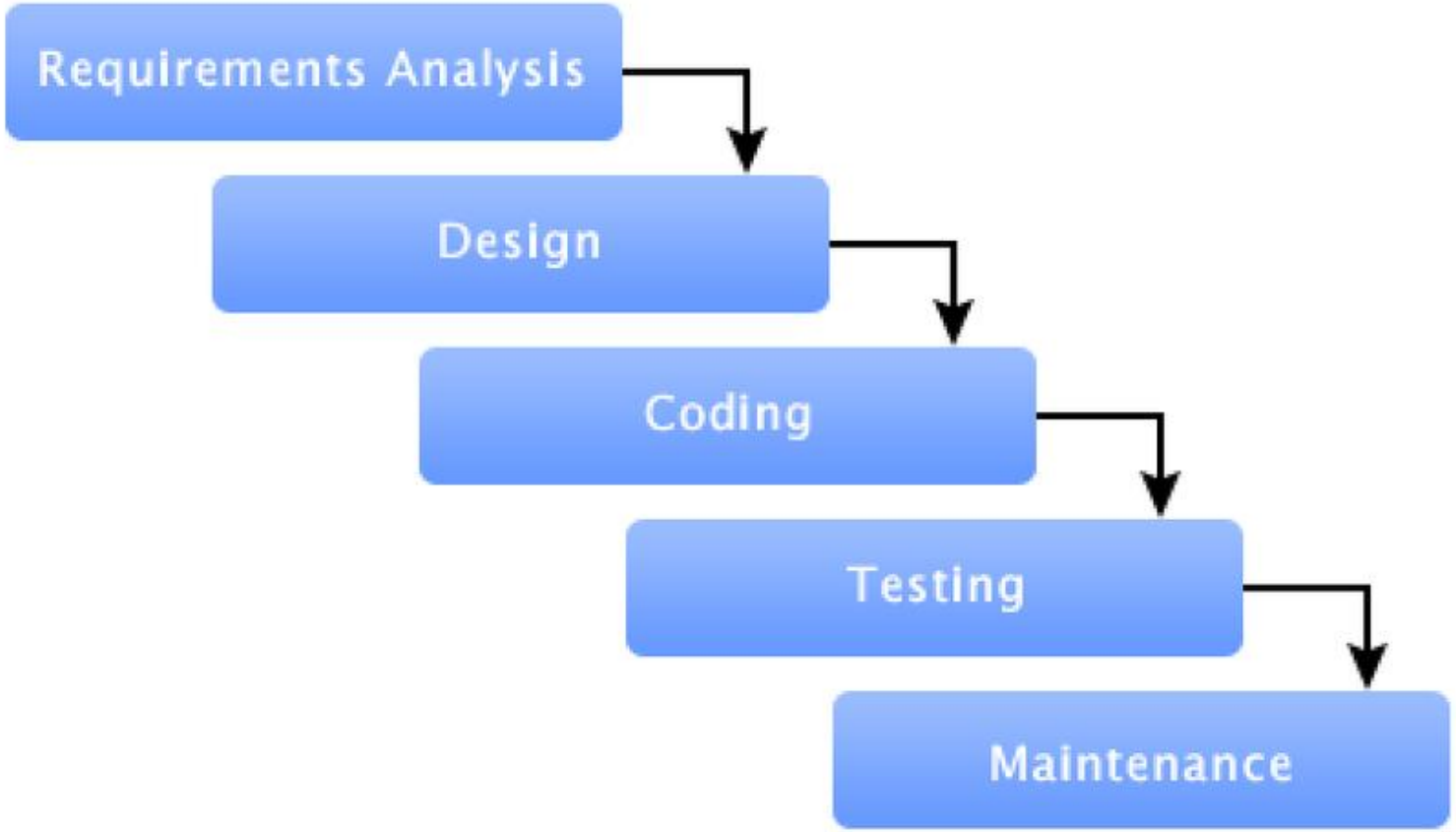


Figure: Waterfall Model

Waterfall Model

The phases in waterfall model are shown as follows:

- Requirements analysis and definition: They are often in detail and serve as a system specification.
- System and software design: It established overall system architecture and their relationship.
- Implementation and unit testing: It includes writing source code and verify that each unit meets its specification.
- Integration and system testing: The programs are integrated and tested before delivering to the customer.
- Operation and maintenance: The system after testing is installed and put into practical use.

Advantages of Waterfall Model

- The advantages of waterfall model are as follows:
  - Easy to understand.
  - Each phase has well defined inputs and outputs.
  - Helps the project manager in proper planning of the project.

Disadvantages of Waterfall Model

- The disadvantages of waterfall model are as follows:
  - Difficult to change after the process is underway because of sequential nature.
  - Inflexible of the project into distinct stage.
  - Difficult to response to changing customer requirements.

2. Prototyping Model

An alternative method to traditional method has been developed, called prototyping model is based on the assumption that it is difficult know all the requirement of user in advance.

- The prototyping model consists of following steps:
  - Determine the Requirements.
  - Quick Design.
  - Build/Create Prototype.
  - Use and Evaluate Prototyping.
  - Refine the Prototype.
  - Design.
  - Implementation.
  - Testing.
  - Operation and Maintenance.

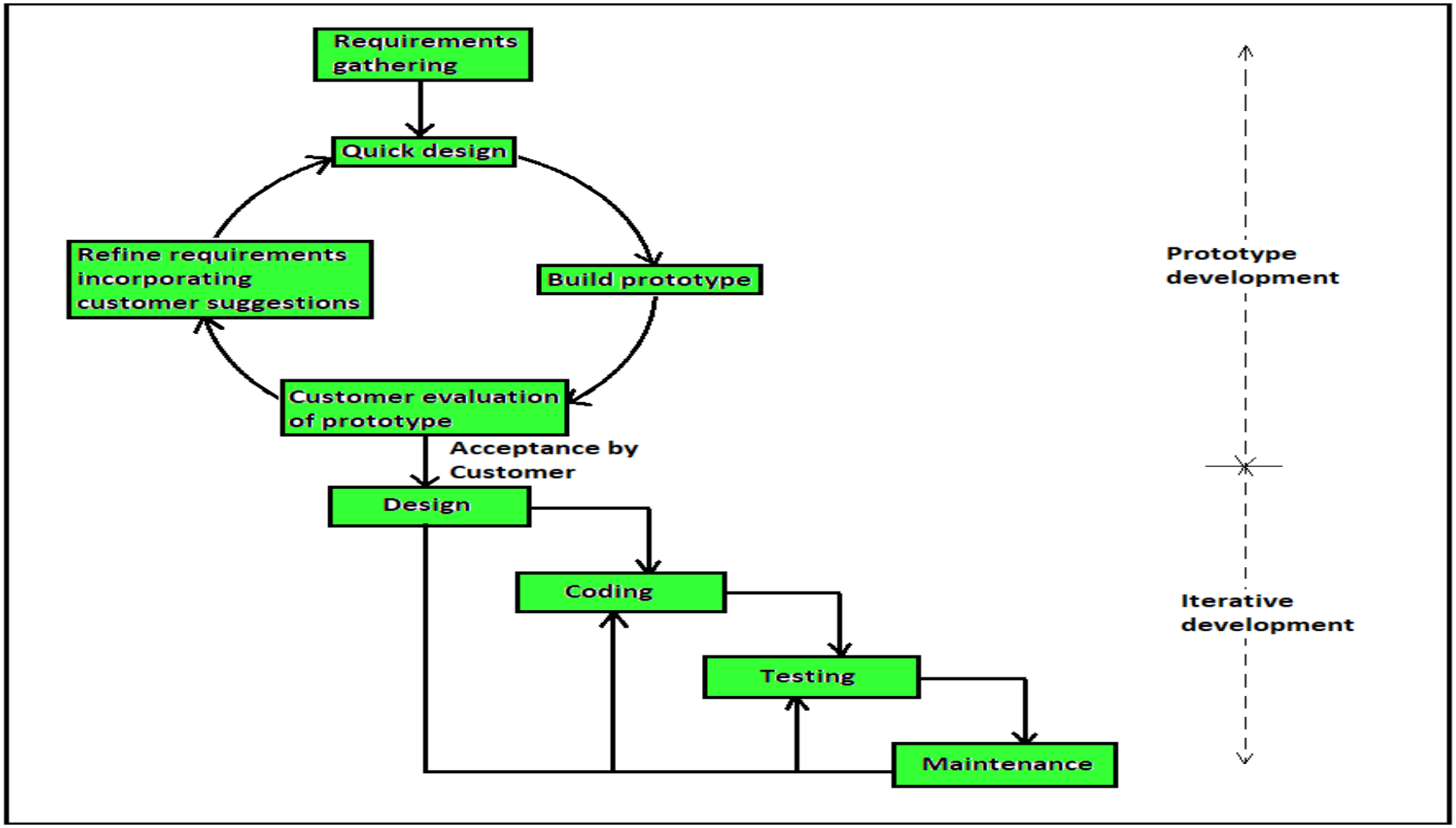


Figure: Prototype Model

2. Prototyping Model

- The prototyping model consists of following steps:

  - **Determine the Requirements:** This is the first phase of prototyping model. In this phase, the system's services, constraints and goals are established by consultation with system users.

  - **Quick Design:** On the basis of known requirement, rough design of system is drawn for discussion and decision.

  - **Build/Create Prototype**: In this phase, the information from the design is rolled into prototype which is the blueprint of the system.

  - **Use and Evaluate Prototyping:** In this phase, user comments and suggestions are called and areas of refinements are determined.

  - Refine the Prototype: Based on the suggestions of user, the system developer revises to make more effective and efficient to meet customer requirements.

2. Prototyping Model

Design: Once the prototype is refined and accepted by the users, by the users, actual system is designed to satisfy customer.

Implementation: During this stage, the software design is realized as a set of programs and program units.

Testing: Once the program modules are ready, each of the program modules is tested independently as per the specifications of the user and debugged.

Operation and Maintenance: In this stage, the system is installed and put into practical use.

Advantages of Prototyping Model

The advantages of waterfall model are as follows:

- The user gets a proper clarity and can suggest changes and modifications.
- Feedbacks from customer are received periodically and changes don't come at last minute.
- Iteration between developer and user provides a good environment during project.
- Time can be saved having better idea to the developer.

Disadvantages of Prototyping Model

The disadvantages of waterfall model are as follows:

- Once the user requirement is taken, it may be of no use. So, this model is as "Throw-away" prototype.
- Too much involvement of client is not always preferred by the developer.
- To many changes can disturb the rhythm of the development team.

3. Spiral Model

- The spiral model, also known as spiral lifecycle model or Boehm's model was introduced by Barry Boehm in 1988.
- This model combines the features of prototyping model and waterfall model in order to eliminate almost every possible known risk factors.
- This model follows a spiral process as the development takes place

- The spiral model has following major phases:
  - Customer Communication
  - Planning
  - Risk Analysis
  - Engineering
  - Construction and Release
  - Customer Evaluation

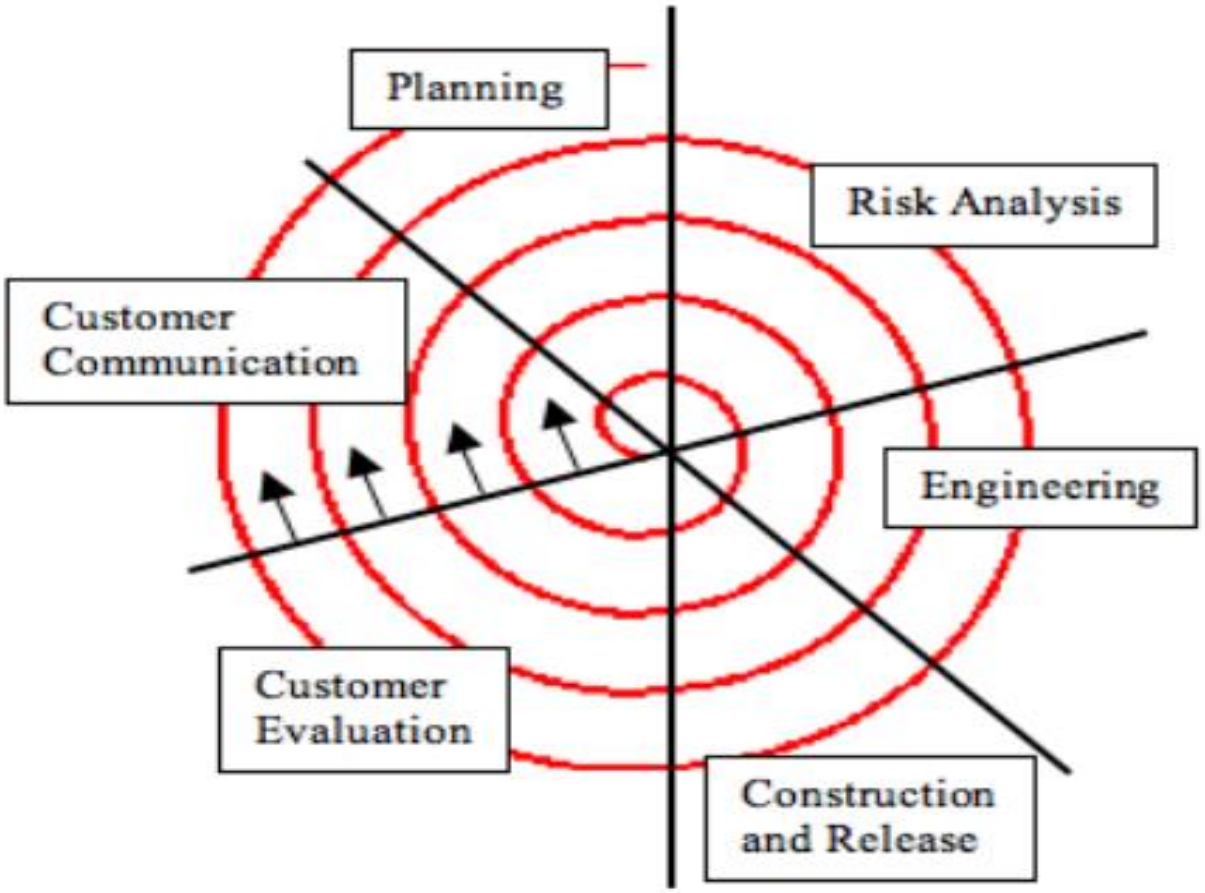


Figure: Spiral Model

3. Spiral Model

The spiral model has following major phases:

Customer Communication: Tasks required establishing effective communication between developer and customer.

Planning: In this phase, objectives and specifications are fixed in order to decide which strategies to follow during project life cycle.

Risk Analysis: In this phase all possible alternatives, which can help in developing a cost effective project are analyzed and strategies are decided to use them.

Engineering: In this phase, the output is passed through all the phases iteratively in order to obtain improvements in same.

Construction and Release: Tasks required to construct, test, install, and provide user support (e.g., documentation and training).

Customer Evaluation: In this phase, developed products are passed onto the customer in order to receive customer comments and suggestions which can help in identifying and resolving potential in the developed software.

Advantages of Spiral Model

The advantages of spiral model are as follows:

- Avoid risk via risk analysis.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Software is produced early in the software life cycle.

Disadvantages of Spiral Model

The disadvantages of spiral model are as follows:

- It can be costly to use.
- Risk analysis requires highly specific expertise.
- Not suitable for smaller projects.
- Project's success is highly dependent on the risk analysis.

4. Incremental Model

- Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle
- Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance
- Every subsequent release of the module adds function to the previous release
- The process continues until the complete system achieved

4. Incremental Model

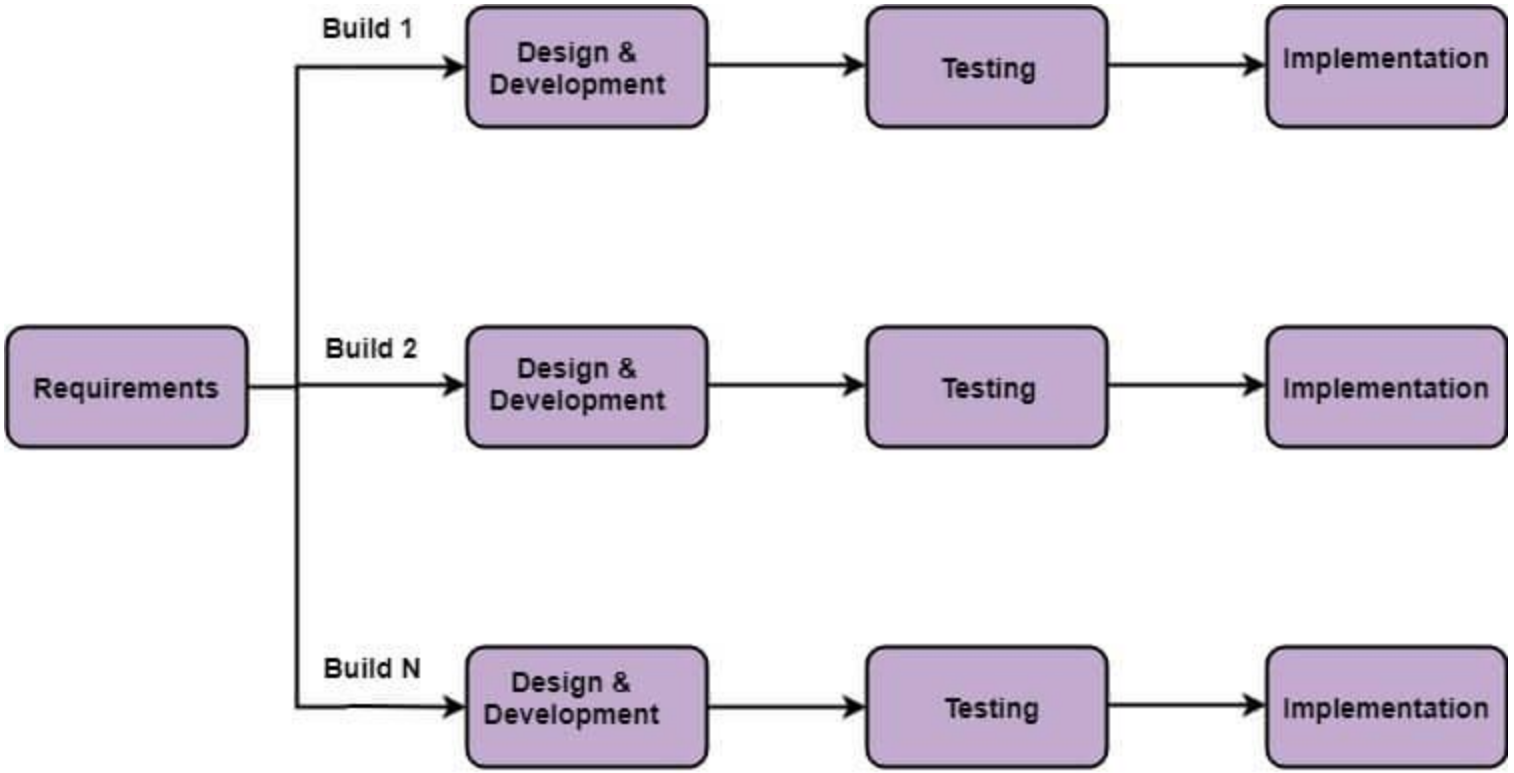


Fig: Incremental Model

4. Incremental Model

The various phases of incremental model are as follows:

- **Requirement analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

- **Design & Development:** In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

- **Testing:** In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

- **Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

4. Incremental Model

Advantage of Incremental Model

- Errors are easy to be recognized
- Easier to test and debug
- More flexible
- Simple to manage risk because it handled during its iteration
- The Client gets important functionality early

Disadvantage of Incremental Model

- Need for good planning
- Total Cost is high
- Well defined module interfaces are needed

Multiple Choice Questions:

1. Language in which single statement can be written to accomplish substantial tasks is termed as:

a. MLL b. ALL c. HLL d. MediumLL

2. The mnemonics are used in

a. ALL b. MLL c. HLL d. OOPL

3. Translator which is used to convert codes of ALL into MLL is termed as:

a. Assembler b. Interpreter c. Compiler d. Debugger

4. Which of the following error can't be detected by compiler?

a. Syntax Error b. Run Time Error

c. Logical Error d. Both b and c

5. Which of the following is OOP language?

a. BASIC b. C c. Java d. FORTRAN

Multiple Choice Questions:

6. …… error is occurred due to insufficient knowledge of programming language.

a. Syntax b. Logical c. Run Time d. Exception

7. In program development life cycle, design phase is followed by

a. Coding b. Testing c. Maintenance d. All of them

8. The diamond shaped symbol is used in the flowchart to show the

a. Decision b. Process c. Input d. Stop

9. What is the operation that is used in algorithm to repeat the particular block of statement?

a. Selection b. Sequence c. Iteration d. Reverse

10. Risk analysis is key feature in …….. Software process model.

a. Waterfall b. Spiral c. Prototype d. Both a and b

Short Questions:

1. Define programming language. Explain LLL and HLL with examples.
2. What do you mean by language processor? Differentiate between compiler and interpreter.
3. What is programming paradigm? Compare different types of paradigm.
4. What is algorithm? Explain the characteristics.
5. What is flowchart? Explain symbols used in flowchart with their semantics.
6. Define program compilation and execution? What is its importance?
7. What do you mean by program design tools? Explain context diagram, DFD and Decision Table in brief.
8. Write an algorithm and draw flowchart to determine whether the number is positive, negative or zero.
9. Write an algorithm and draw flowchart to find smallest among three numbers.
10. Write an algorithm and draw flowchart to find the real roots of quadratic equation: ax2 + bx + c = 0
11. Draw flowchart to determine whether entered number is prime or not.

Long Questions:

1. Explain MLL and ALL with their merits and demerits.
2. Explain various types of programming errors. Compare them on the basis of causes, sources, complexity and impact on output of the program.
3. What is program development life cycle? Explain the phases of program development life cycle.
4. Define software process model. Explain waterfall and spiral model with advantages and disadvantages.


THANK YOU
FOR YOUR ATTENTION