Subject : Software Engineering

Subject Code : CS3273

Section: Gx

Topic : Different Phases Of Software Development Lifecycle

Enrollment no.	Name
2020CSB007	ARATRIKA PAL
2020CSB009	MAYANK MANAVENDRA KUMAR
2020CSB010	GOURAV KUMAR SHAW
2020CSB011	TAMOGHNA ROY
2020CSB013	ABDUL KHAZMUDDIN
2020CSB014	SIDDARTH JANA

Part I: 1 week

We know that a software lifecycle consists of many different phases like feasibility study, requirement analysis, design, coding, testing and maintenance. Explore online sample

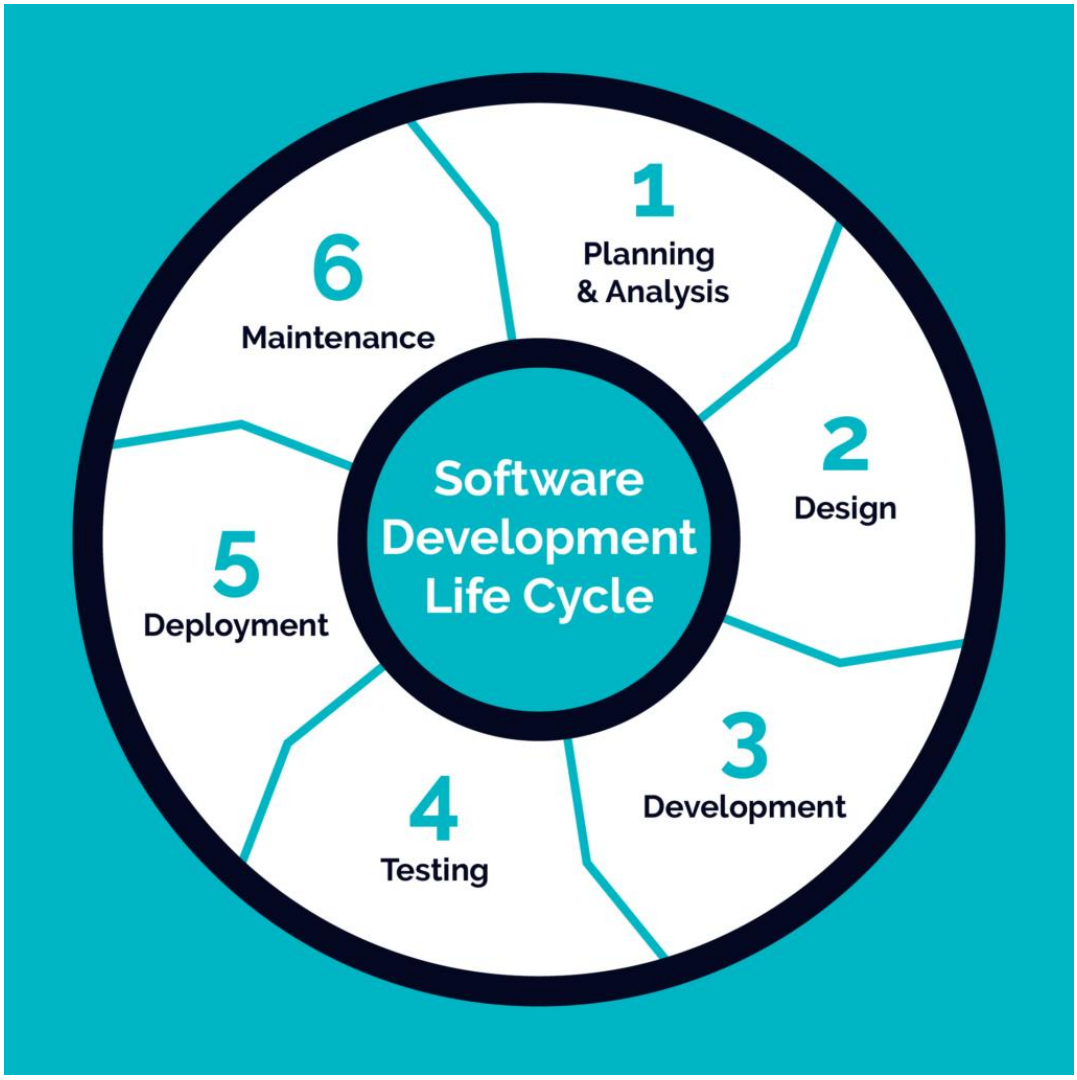
charts/tables/sheets used for formal documentation in each of these phases.

Solution:

Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

- SDLC is the acronym of Software Development Life Cycle.
- It is also called as Software Development Process.
- SDLC is a framework defining tasks performed at each step in the software development process.



1. Feasibility Study

Feasibility Study in Software Engineering is a study to evaluate feasibility of proposed project or system. Feasibility study is one of stage among important four stages of Software Project Management Process. As name suggests feasibility study is the feasibility analysis or it is a measure of the software product in terms of how much beneficial product development will be for the organization in a practical point of view. Feasibility study is carried out based on many purposes to analyze whether software product will be right in terms of development, implantation, contribution of project to the organization etc.

Types of Feasibility Study :
The feasibility study mainly concentrates on below five mentioned areas. Among these Economic Feasibility Study is most important part of the feasibility analysis and Legal Feasibility Study is less considered feasibility analysis.

1. **Technical Feasibility –**
   In Technical Feasibility current resources both hardware software along with required technology are analyzed/assessed to develop project. This technical feasibility study gives report whether there exists correct required resources and technologies which will be used for project development. Along with this, feasibility study also analyzes technical skills and capabilities of technical team, existing technology can be used or not, maintenance and up-gradation is easy or not for chosen technology etc.

2. **Operational Feasibility –**
   In Operational Feasibility degree of providing service to requirements is analyzed along with how much easy product will be to operate and maintenance after deployment. Along with this other operational scopes are determining usability of product, Determining suggested solution by software development team is acceptable or not etc.

3. **Economic Feasibility –**
   In Economic Feasibility study cost and benefit of the project is analyzed. Means under this feasibility study a detail analysis is carried out what will be cost of the project for development which includes all required cost for final development like hardware and software resource required, design and development cost and operational cost and so on. After that it is analyzed whether project will be beneficial in terms of finance for organization or not.

4. **Legal Feasibility –**
   In Legal Feasibility study project is analyzed in legality point of view. This includes analyzing barriers of legal implementation of project, data protection acts or social media laws, project certificate, license, copyright etc. Overall it can be said that Legal Feasibility Study is study to know if proposed project conform legal and ethical requirements.

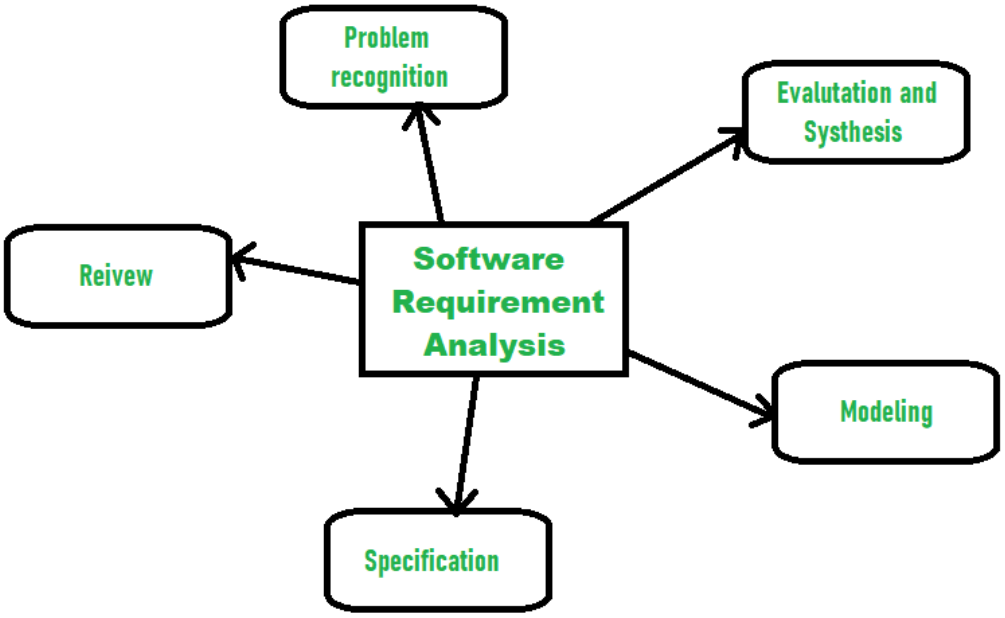
5. **Schedule Feasibility –**
   In Schedule Feasibility Study mainly timelines/deadlines is analyzed for proposed project which includes how many times teams will take to complete final project which has a great impact on the organization as purpose of project may fail if it can't be completed on time.

2. Requirements Analysis

Software requirement means requirement that is needed by software to increase quality of software product. These requirements are generally a type of expectation of user from software product that is important and need to be fulfilled by software. Analysis means to examine something in an organized and specific manner to know complete details about it.

Therefore, **Software requirement analysis** simply means complete study, analyzing, describing software requirements so that requirements that are genuine and needed can be

fulfilled to solve problem. There are several activities involved in analyzing Software requirements. Some of them are given below :

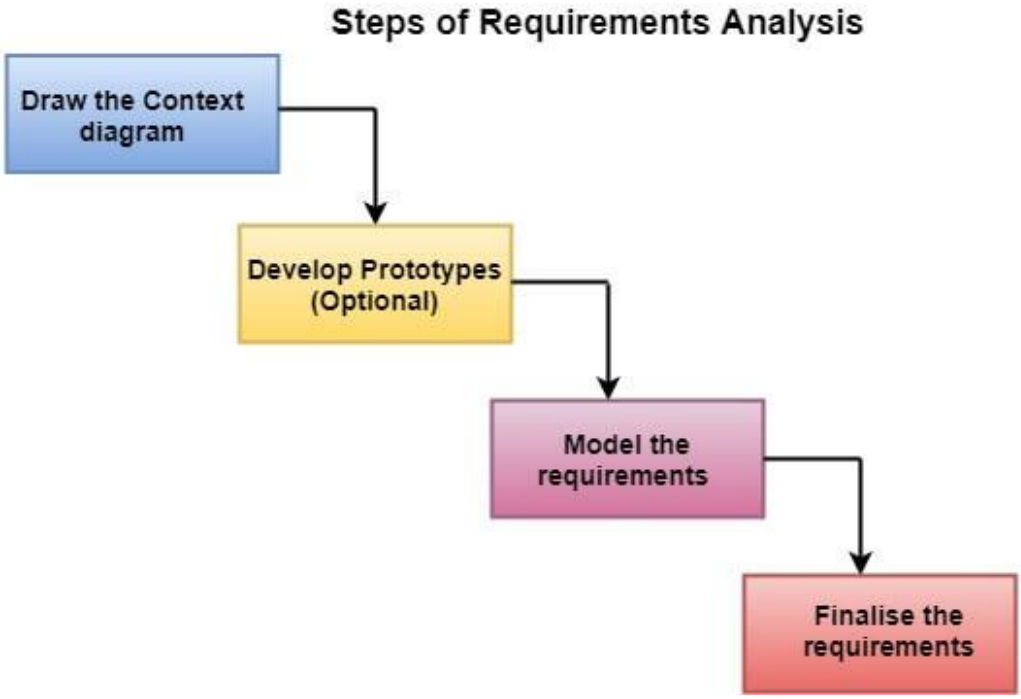


of requirement that includes why it is needed, does it add value to product, will it be beneficial, does it increase quality of the project, does it will have any other effect. All these points are fully recognized in problem recognition so that requirements that are essential can be fulfilled to solve business problems.

2. **Evaluation and Synthesis :**
   Evaluation means judgement about something whether it is worth or not and synthesis means to create or form something. Here are some tasks are given that is important in the evaluation and synthesis of software requirement :
   - To define all functions of software that necessary.
   - To define all data objects that are present externally and are easily observable.
   - To evaluate that flow of data is worth or not.
   - To fully understand overall behavior of system that means overall working of system.
   - To identify and discover constraints that are designed.
   - To define and establish character of system interface to fully understand how system interacts with two or more components or with one another.
3. **Modeling :**
   After complete gathering of information from above tasks, functional and behavioral models are established after checking function and behavior of system using a domain model that also known as the conceptual model.

4. **Specification :**
   The software requirement specification (SRS) which means to specify the requirement whether it is functional or non-functional should be developed.

5. **Review :**
   After developing the SRS, it must be reviewed to check whether it can be improved or not and must be refined to make it better and increase the quality.



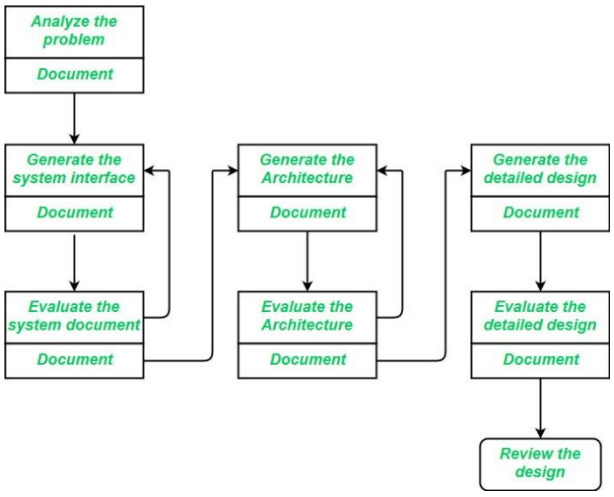
3. Design

The design phase of software development deals with transforming the customer requirements as described in the SRS (software requirement specification) documents into a form implementable using a programming language. The software design process can be divided into the following three levels of phases of design:

1. Interface Design
2. Architectural Design
3. Detailed Design

Elements of a System:

1. **Architecture –** This is the conceptual model that defines the structure, behavior, and views of a system. We can use flowcharts to represent and illustrate the architecture.
2. **Modules –** These are components that handle one specific task in a system. A combination of the modules makes up the system.
3. **Components –** This provides a particular function or group of related functions. They are made up of modules.
4. **Interfaces –** This is the shared boundary across which the components of a system exchange information and relate.
5. **Data –** This is the management of the information and data flow.



Interface Design: *Interface design* is the specification of the interaction between a system and its environment. this phase proceeds at a high level of abstraction with respect to the inner workings of the system

- Precise description of events in the environment, or messages from agents to which the system must respond.
- Precise description of the events or messages that the system must produce.
- Specification on the data, and the formats of the data coming into and going out of the system.
- Specification of the ordering and timing relationships between incoming events or messages, and outgoing events or outputs.

Architectural Design:

Architectural design is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them. In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored. Issues in architectural design includes:

- Gross decomposition of the systems into major components.
- Allocation of functional responsibilities to components.
- Component Interfaces
- Component scaling and performance properties, resource consumption properties, reliability properties, and so forth.
- Communication and interaction between components.

Detailed design may include:

- Decomposition of major system components into program units.
- Allocation of functional responsibilities to units.
- User interfaces
- Unit states and state changes
- Data and control interaction between units
- Data packaging and implementation, including issues of scope and visibility of program elements
- Algorithms and data structures

4. Coding

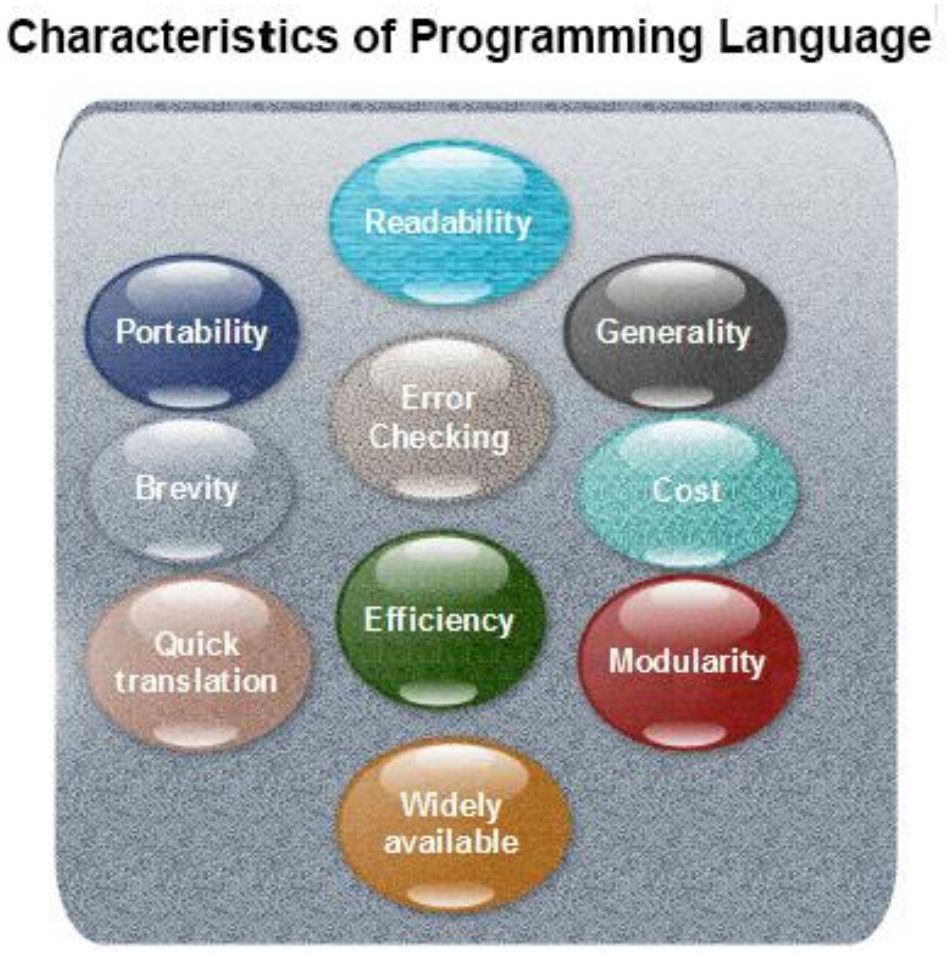
The coding is the process of transforming the design of a system into a computer language format. This coding phase of software development is concerned with software translating design specification into the source code. It is necessary to write source code & internal documentation so that conformance of the code to its specification can be easily verified.

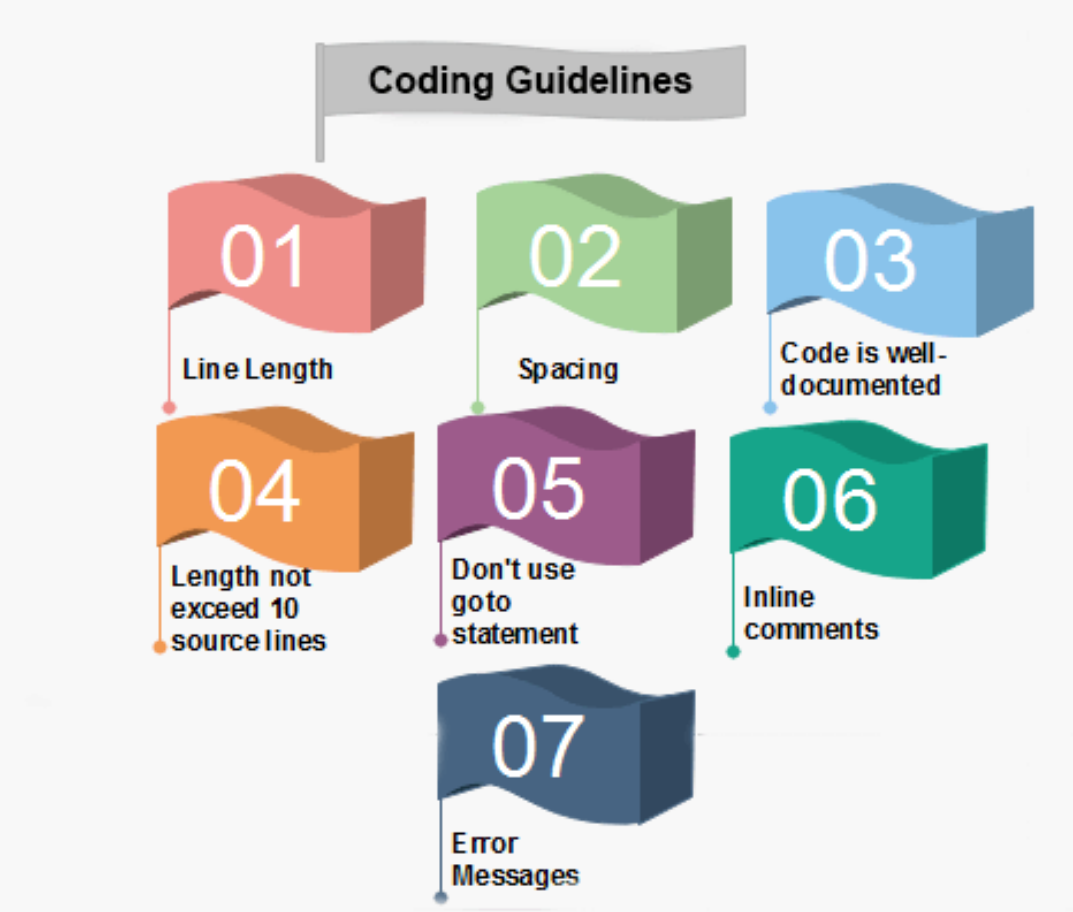
Coding is done by the coder or programmers who are independent people than the designer. The goal is not to reduce the effort and cost of the coding phase, but to cut to the cost of a

later stage. The cost of testing and maintenance can be significantly reduced with efficient coding.

Goals Of Coding

1. **To translate the design of system into a computer language format:** The coding is the process of transforming the design of a system into a computer language format, which can be executed by a computer and that perform tasks as specified by the design of operation during the design phase.
2. **To reduce the cost of later phases:** The cost of testing and maintenance can be significantly reduced with efficient coding.
3. **Making the program more readable:** Program should be easy to read and understand. It increases code understanding having readability and understandability as a clear objective of the coding activity can itself help in producing more maintainable software.





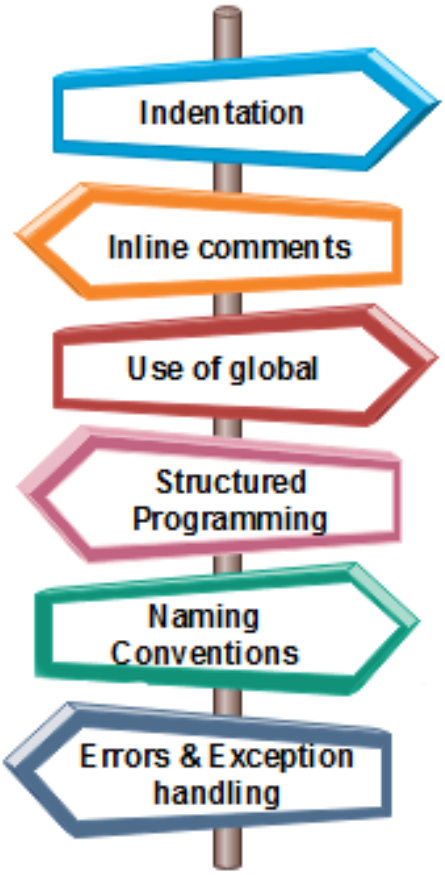
Spacing: The appropriate use of spaces within a line of code can improve readability.

Example:

Bad:
```
cost=price+(price*sales_tax)
fprintf(stdout ,"The total cost is %5.2f\n",cost);
```

Better:
```
cost = price + ( price * sales_tax )
fprintf (stdout,"The total cost is %5.2f\n",cost);
```

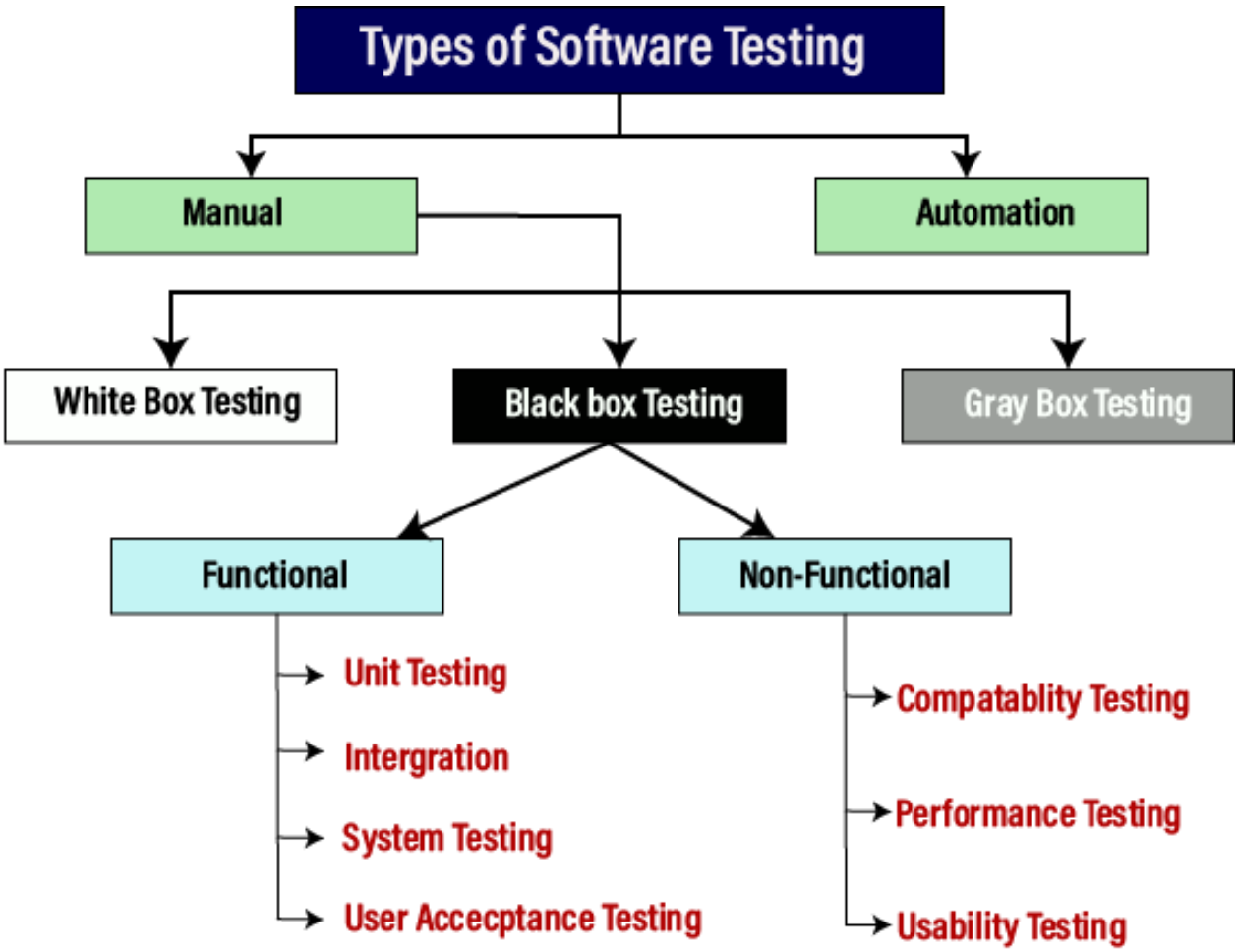
Coding Standards



1. **Indentation:** Proper and consistent indentation is essential in producing easy to read and maintainable programs.
   Indentation should be used to:
   - Emphasize the body of a control structure such as a loop or a select statement.
   - Emphasize the body of a conditional statement
   - Emphasize a new scope block
2. **Inline comments:** Inline comments analyze the functioning of the subroutine, or key aspects of the algorithm shall be frequently used.
3. **Rules for limiting the use of global:** These rules file what types of data can be declared global and what cannot.
4. **Structured Programming:** Structured (or Modular) Programming methods shall be used. "GOTO" statements shall not be used as they lead to "spaghetti" code, which is hard to read and maintain, except as outlined line in the FORTRAN Standards and Guidelines.
5. **Naming conventions for global variables, local variables, and constant identifiers:** A possible naming convention can be that global variable names always begin with a capital letter, local variable names are made of small letters, and constant names are always capital letters.
6. **Error return conventions and exception handling system:** Different functions in a program report the way error conditions are handled should be standard within an

organization. For example, different tasks while encountering an error condition should either return a 0 or 1 consistently.

5 . Software Testing



Testing is the process of executing a program to find errors. To make our software perform well it should be error-free. If testing is done successfully it will remove all the errors from the software.

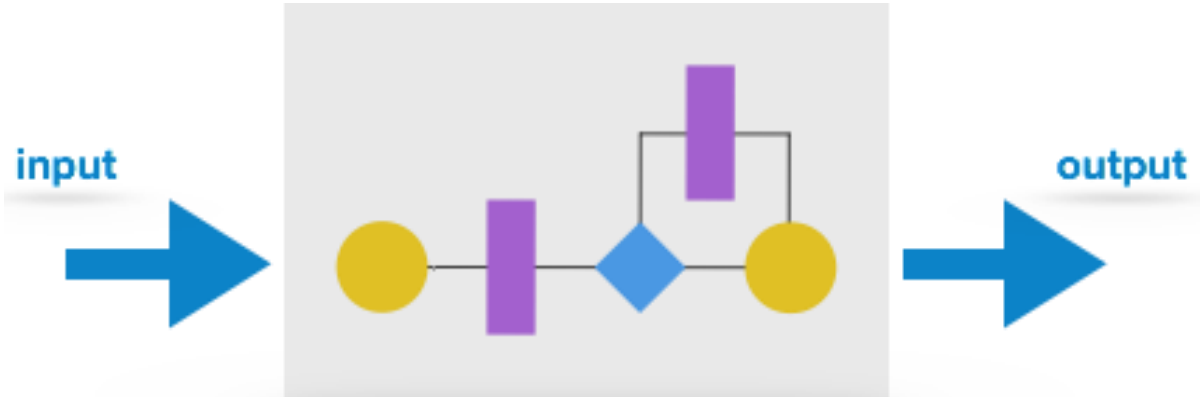
Priciple of software testing:-

(i) All the tests should meet the customer requirements.
(ii) To make our software testing should be performed by a third party.
(iii) Exhaustive testing is not possible. As we need the optimal amount of testing based on the risk assessment of the application.
(iv) All the tests to be conducted should be planned before implementing it

(v) It follows the Pareto rule(80/20 rule) which states that 80% of errors come from 20% of program components.
(vi) Start testing with small parts and extend it to large parts.

White-box Testing

It is conducted to test program and its implementation, in order to improve code efficiency or structure. It is also known as 'Structural' testing.



In this testing method, the design and structure of the code are known to the tester. Programmers of the code conduct this test on the code.

The below are some White-box testing techniques:

- **Control-flow testing** - The purpose of the control-flow testing to set up test cases which covers all statements and branch conditions. The branch conditions are tested for both being true and false, so that all statements can be covered.
- **Data-flow testing** - This testing technique emphasis to cover all the data variables included in the program. It tests where the variables were declared and defined and where they were used or changed.

Black-box Testing

It is carried out to test functionality of the program. It is also called 'Behavioral' testing. The tester in this case, has a set of input values and respective desired results. On providing input, if the output matches with the desired results, the program is tested 'ok', and problematic otherwise.

In this testing method, the design and structure of the code are not known to the tester, and testing engineers and end users conduct this test on the software.

Unit testing

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

Integration testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passing and data updation etc.

System Testing

The software is compiled as product and then it is tested as a whole. This can be accomplished using one or more of the following tests:

- **Functionality testing** - Tests all functionalities of the software against the requirement.
- **Performance testing** - This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance

testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.

- **Security & Portability** - These tests are done when the software is meant to work on various platforms and accessed by number of persons.

Acceptance Testing

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

- **Alpha testing** - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.
- **Beta testing** - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

Regression Testing

Whenever a software product is updated with new code, feature or functionality, it is tested thoroughly to detect if there is any negative impact of the added code. This is known as regression testing.

6. Software Maintenance

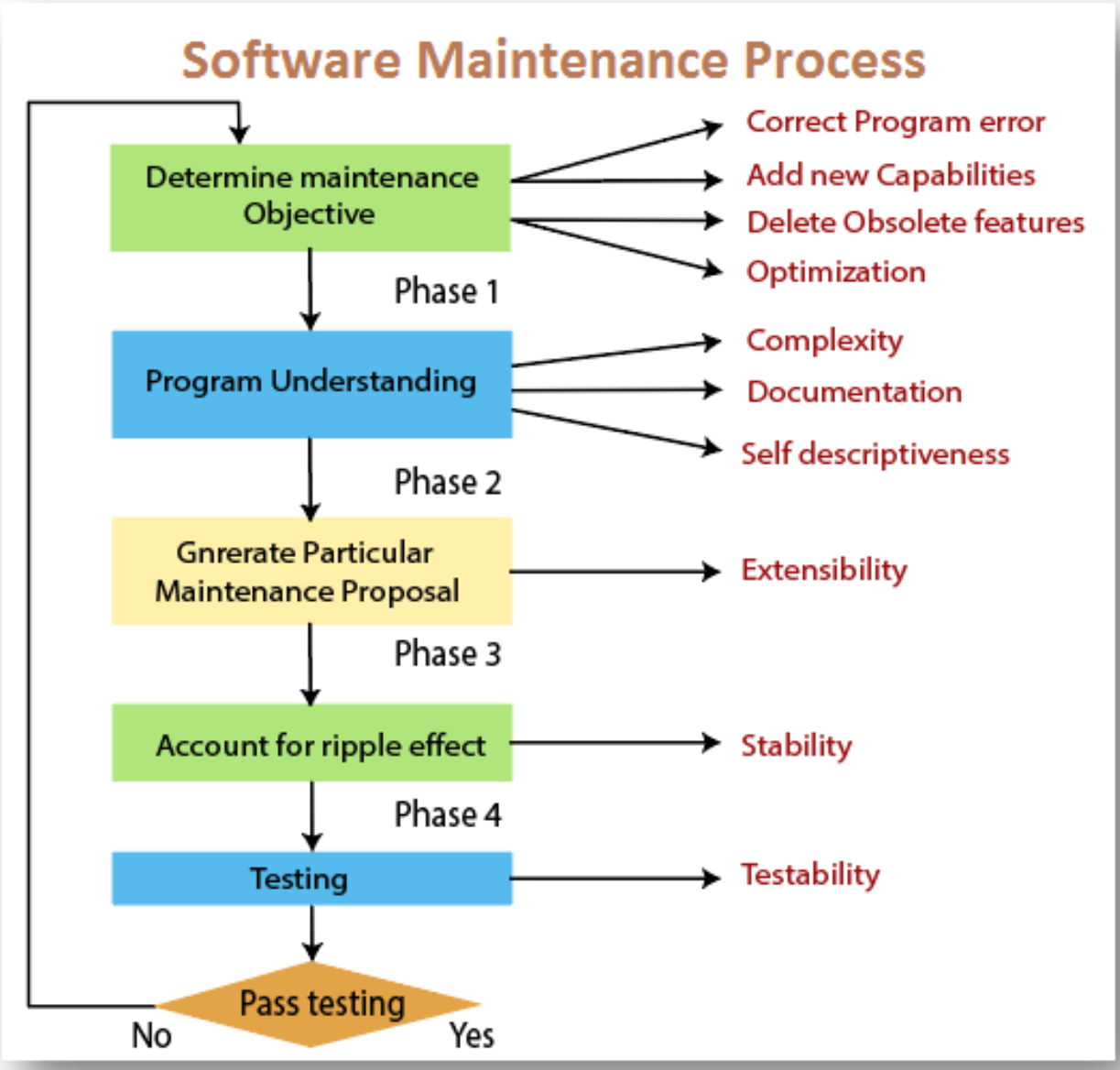
Software Maintenance is the process of modifying a software product after it has been delivered to the customer. The main purpose of software maintenance is to modify and update software applications after delivery to correct faults and to improve performance.

Software maintenance is also an important part of the Software Development Life Cycle (SDLC). To update the software application and do all modifications in software application so as to improve performance is the main focus of software maintenance. Software is a model that run on the basis of real world. so, whenever any change requires in the software that means the need of real world changes wherever possible.

Need for Maintenance –

Software Maintenance must be performed in order to:

- Correct faults.
- Improve the design.
- Implement enhancements.
- Interface with other systems.
- Accommodate programs so that different hardware, software, system features, and telecommunications facilities can be used.
- Migrate legacy software.
- Retire software.
- Requirement of user changes.
- Run the code fast



Categories of Software Maintenance –

Maintenance can be divided into the following:

Corrective maintenance:

Corrective maintenance of a software product may be essential either to rectify some bugs observed while the system is in use, or to enhance the performance of the system.

1. **Adaptive maintenance:**
   This includes modifications and updations when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

2. **Perfective maintenance:**
   A software product needs maintenance to support the new features that the users want or to change different types of functionalities of the system according to the customer demands.

3. **Preventive maintenance:**
   This type of maintenance includes modifications and updations to prevent future problems of the software. It goals to attend problems, which are not significant at this moment but may cause serious issues in future.