23CS405: Software Engineering

UNIT-I

Topic: Introduction about Software Engineering

Sunitha

Assistant Professor

Computer science and Engineering

Software

Software is a program or set of programs containing instructions that provide the desired functionality.

Engineering

Engineering is the process of designing and building something that serves a particular purpose and finds a cost-effective solution to problems.

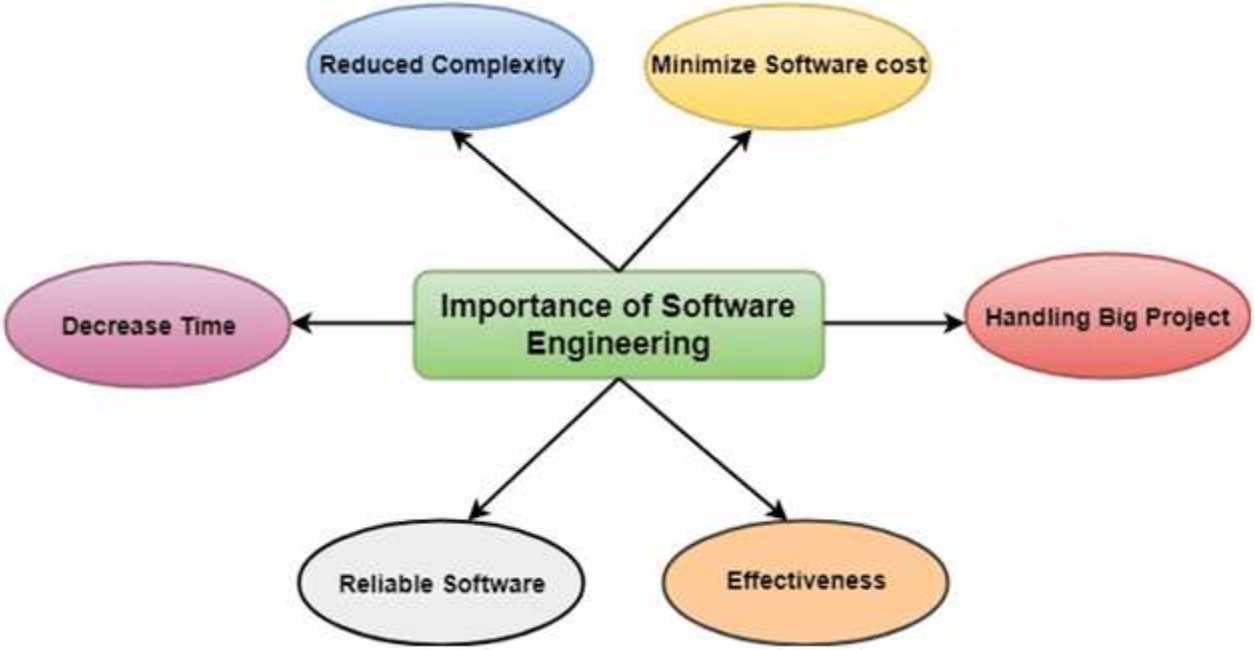
What is Software Engineering

Software Engineering is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software.

Characteristics of Software:

- Software is developed or engineered ;it is not manufactured in the classical sense.
- Software does not “wear out ” (not susceptible to environment effects).
- Reusability of components.

Importance of Software Engineering



Evolving role of Software

There is a dual role of software in the industry. The first one is as a product and the other one is as a vehicle for delivering the product. We will discuss both of them.

1. As a Product It delivers computing potential across networks of Hardware. It enables the Hardware to deliver the expected functionality. It acts as an information transformer because it produces, manages, acquires, modifies, displays, or transmits information.

2. As a Vehicle for Delivering a Product It provides system functionality (e.g., payroll system). It controls other software (e.g., an operating system). It helps build other software (e.g., software tools).

Changing Nature of Software

Nowadays, seven broad categories of computer software present continuing challenges for software engineers

System Software: System software is a collection of programs that are written to service other programs.

Application Software: Application software is defined as programs that solve a specific business need.

Engineering and Scientific Software: This software is used to facilitate the engineering function and task. However modern applications within the engineering and scientific area are moving away from conventional numerical algorithms.

Software Myths

Most, experienced experts have seen myths or superstitions (false beliefs or interpretations) or misleading attitudes (naked users) which creates major problems for management and technical people. The types of software-related myths are listed below. `Types of Software Myths

(i) Management Myths: We have all the standards and procedures available for software development.

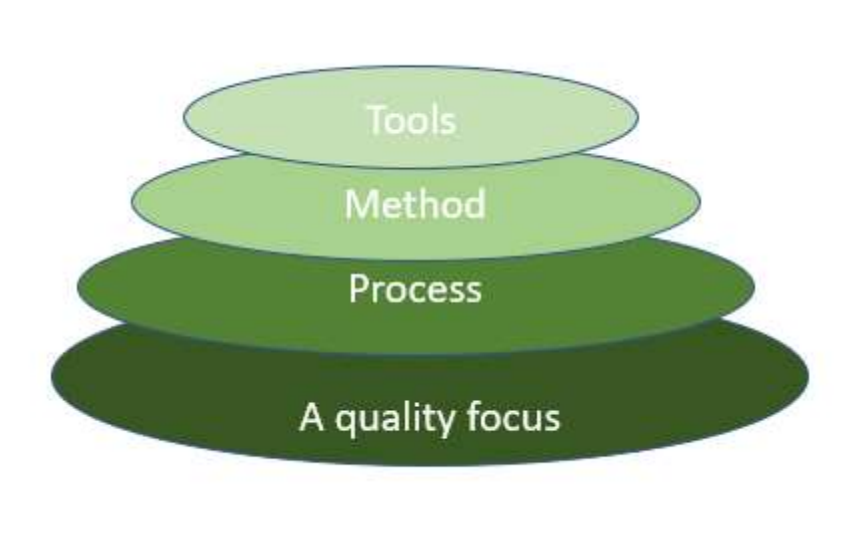
Ii) Customer Myths: The customer can be the direct users of the software, the technical team, marketing / sales department, or other company. Customer has myths leading to false expectations (customer) & that's why you create dissatisfaction with the developer.

Iii) Practitioner's Myths: They believe that their work has been completed with the writing of the plan.

A Generic view of process: Layered Technology in Software Engineering

Software engineering is a fully layered technology, to develop software we need to go from one layer to another. All the layers are connected and each layer demands the fulfillment of the previous layer.

Fig: The diagram shows the layers of software development



What is a Software Process Framework

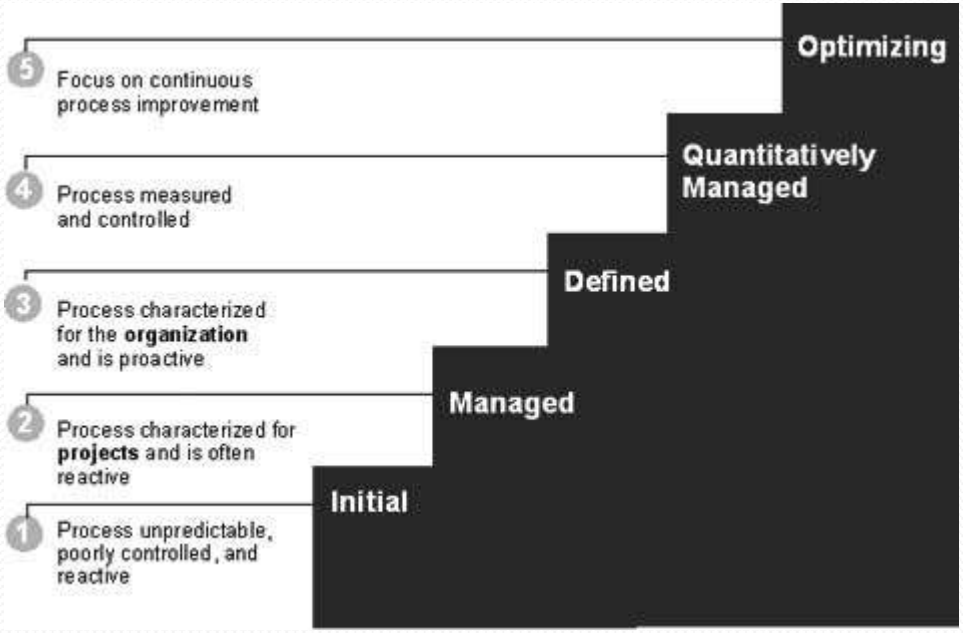
Software Process Framework details the steps and chronological order of a process. Since it serves as a foundation for them, it is utilized in most applications. Task sets, umbrella activities, and process framework activities all define the characteristics of the software development process. Software Process includes :

1. Tasks: They focus on a small, specific objective.
2. Action: It is a set of tasks that produce a major work product.
3. Activities: Activities are groups of related tasks and actions for a major objective.

Capability MaturityModel Integration(CMMI)

A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. Each maturity level provides a layer in the foundation for continuous process improvement.
In CMMI models with a staged representation, there are five maturity levels designated by the numbers 1 through 5
Initial
Managed
Defined
Quantitatively Managed
Optimizing

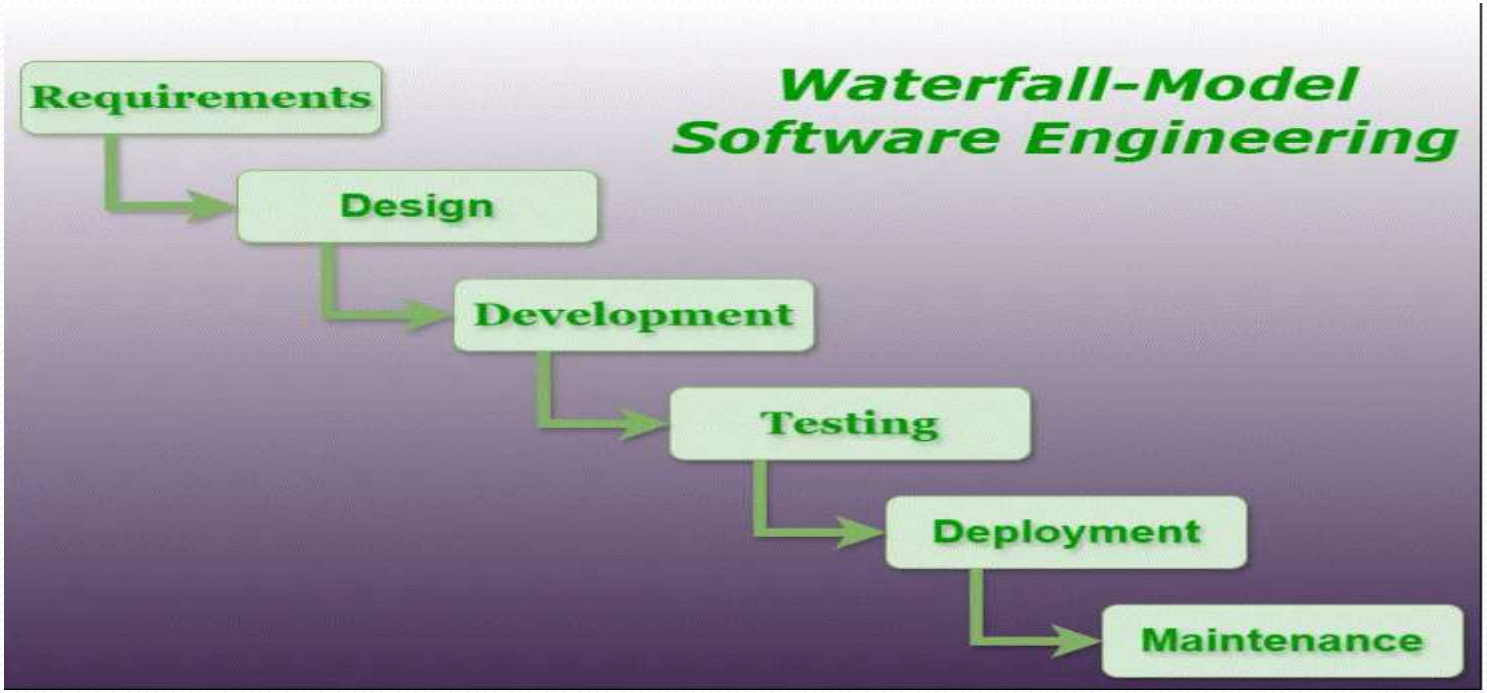
CMMI Staged Representation- Maturity Levels



Waterfall Model

The Waterfall Model is a classical software development methodology. It was first introduced by Winston W. Royce in 1970. It is a linear and sequential approach to software development that consists of several phases. It must be completed in a specific order.

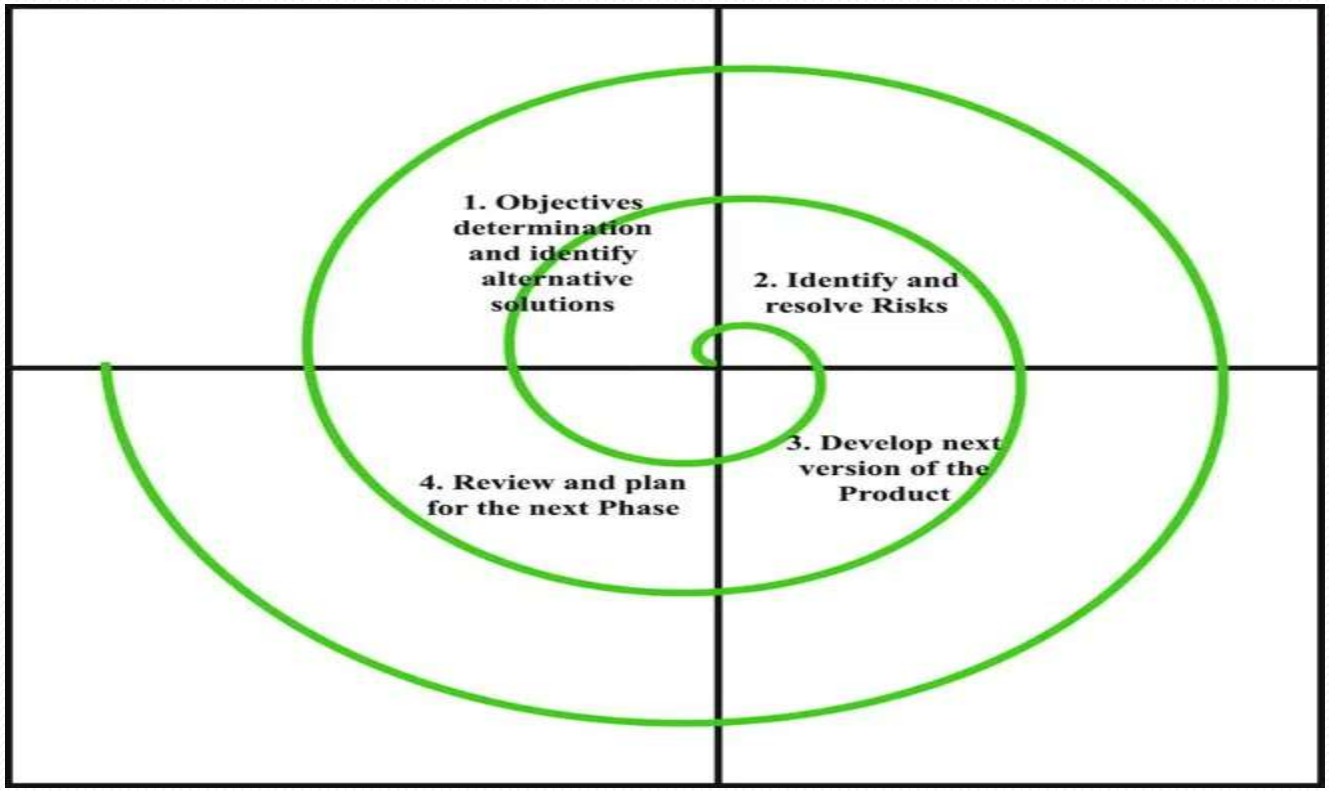
Waterfall Model-Software Engineering



The Spiral Model

is one of the most important Software Development Life Cycle models. The Spiral Model is a combination of the waterfall model and the iterative model. It provides support for **Risk Handling**. The Spiral Model was first proposed by **Barry Boehm**.

Spiral Model



What is Agile Model

The Agile Model was primarily designed to help a project adapt quickly to change requests. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task, agility is required. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project.

Steps in the Agile Model

The agile model is a combination of iterative and incremental process models. The steps involve in agile SDLC models are:

Requirement gathering

Design the Requirements

Construction / Iteration

Testing / Quality Assurance

Dep

Fee

Fig:- Steps in Agile SDLC Model

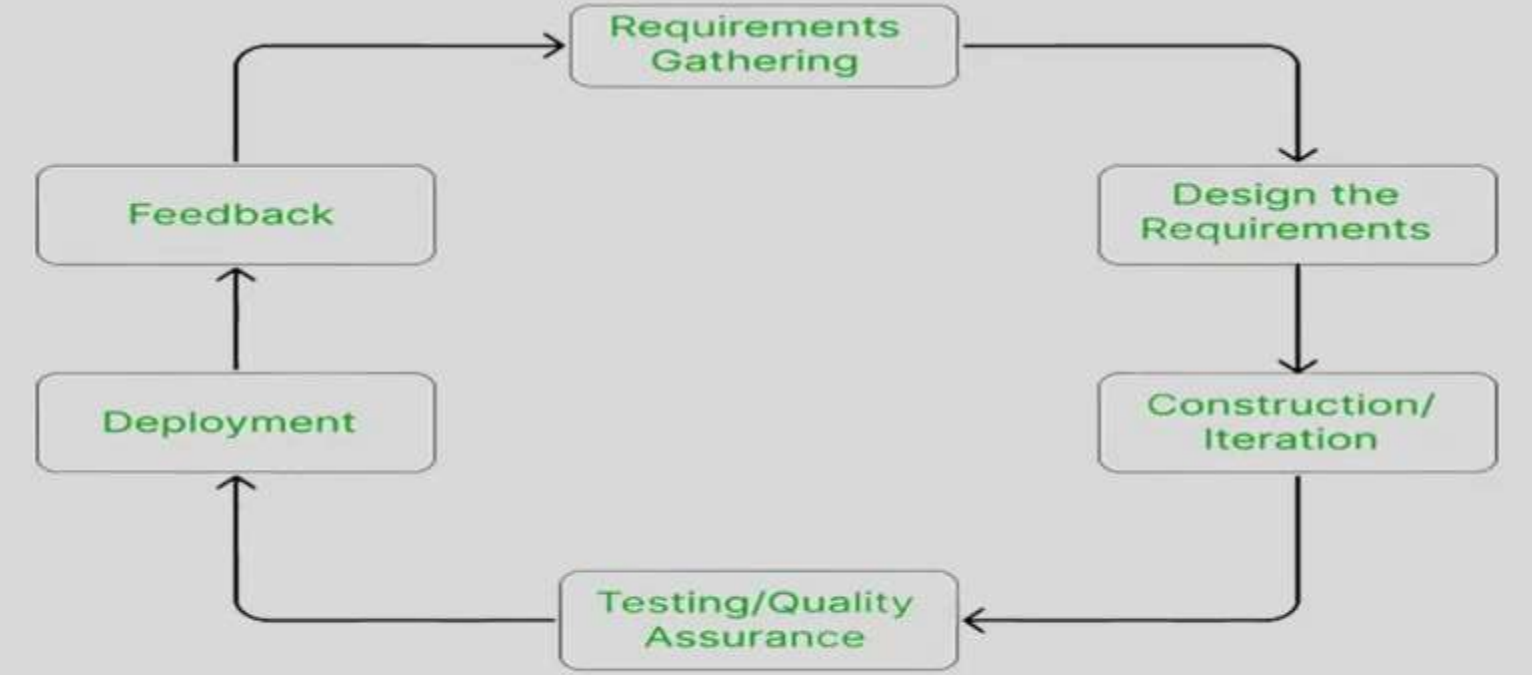


Fig:- Steps in Agile SDLC Model

UNIT-II

Software Requirements

Functional and non-functional requirements

Requirements analysis is a very critical process that enables the success of a system or software projectto be assessed.
Requirements are generally split into two types: Functional and Non-functional requirements.
Understandinganddistinguishingbetweenthesetypesofrequirements isessentialforthesuccessof any project. Our comprehensive System design coursecovers these concepts in detail, providing you with the knowledge and skills to effectively gather, document, and analyze requirements.

Types of requirement:

User requirements:

Statementsinnaturallanguageplusdiagramsoftheservicesthesystempr ovidesanditsoperational constraints. Written for customers.

System requirements:

Astructureddocumentsettingoutdetaileddescriptionsofthesystem’sfu nctions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Requirements Engineering Process

Feasibility Study
Requirements elicitation
Requirements specification
Requirements for verification and validation
Requirements management

SoftwareRequirementDocument:

Software Requirement Specification (SRS) Formats the name suggests, is a complete specification and description of requirements of the software that need to be fulfilled for the successful development of the software system. These requirements can be functional as well as non-functional depending upon the type of requirement. The interaction between different customers and contractors is done because it is necessary to fully understand the needs of customers.

UNIT-III

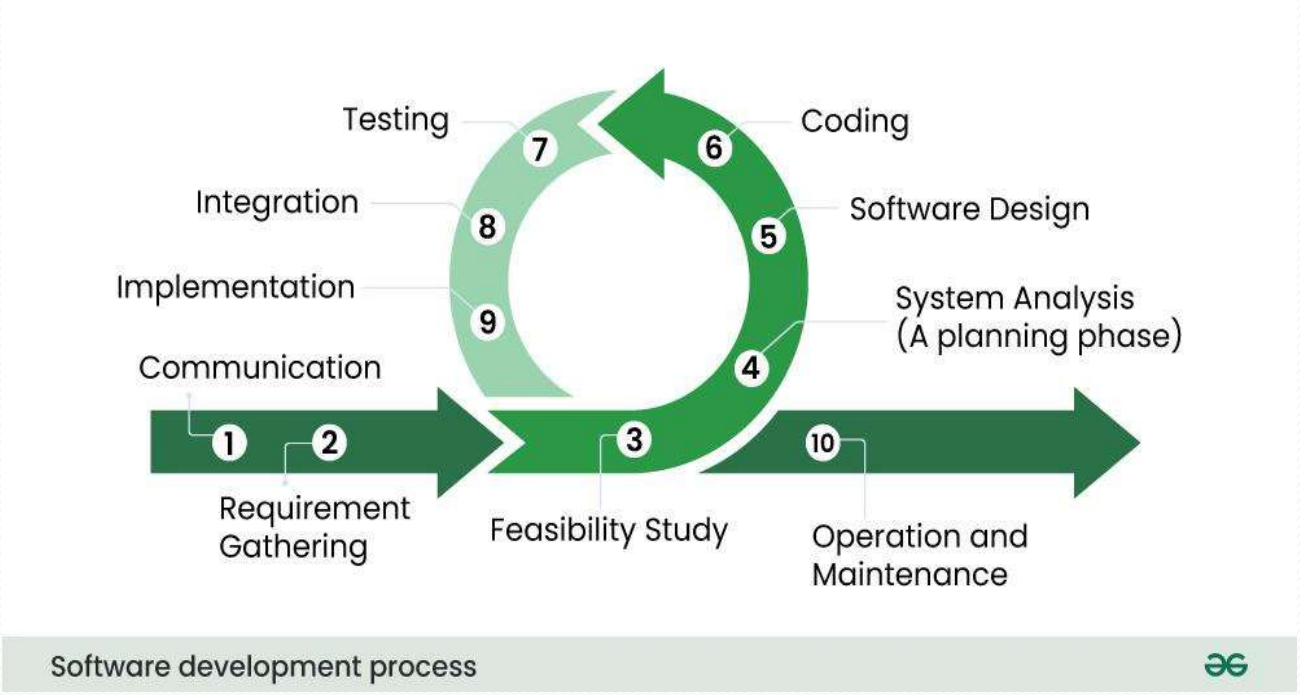
Design Engineering

Software Design Process and quality– Software Engineering

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

- Interface Design
- Architectural Design
- Detailed Design

Software Development Process Process



Software development process

Design Concepts

Concepts are designed and documented during the design phase:

- Different modules are required.
- Control relationships among modules.
- Interface among different modules
- Data structure among the different modules Algorithms are required to be implemented among the individual modules.

Creating an Architectural Design

The software needs an architectural design to represent the design of the software. IEEE defines architectural design as “the process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system.” The software that is built for computer-based systems can exhibit one of these many architectural styles.

System Category: A set of components (eg: a database, computational modules) that will perform a function required by the system. The set of connectors will help in coordination, communication and cooperation between the components.

Architectural Style, Architectural Patterns and Design Patterns

The architectural style shows how do we organize our code, or how the system will look like from 10000 feet helicopter view to show the highest level of abstraction of our system design. Furthermore, when building the architectural style of our system we focus on layers and modules and how they are communicating with each other.

Architectural Style

Structure architectural styles: such as layered, pipes and filters and component-based styles.

Messaging styles: such as Implicit invocation, asynchronous messaging and publish-subscribe style.

Distributed systems: such as service-oriented, peer to peer style, object request broker, and cloud computing styles.

Shared memory styles: such as role-based, blackboard, database-centric styles.

Adaptive system styles: such as microkernel style, reflection, domain-specific language styles.

Architectural Patterns

The architectural pattern shows how a solution can be used to solve a reoccurring problem. In another word, it reflects how a code or components interact with each other. Moreover, the architectural pattern is describing the architectural style of our system and provides solutions for the issues in our architectural style. Personally, I prefer to define architectural patterns as a way to implement our architectural style.

Design Patterns

Design patterns are accumulative best practices and experiences that software professionals used over the years to solve the general problem by – trial and error – they faced during software development. The Gang of Four (GOF, refers to Eric Gamma, Richard Helm, Ralf Johnson, and John Vlissides) wrote a book in 1994 titled with “Design Pattern – Elements of reusable object-oriented software” in which they suggested that design patterns are based on two main principles of object-oriented design:

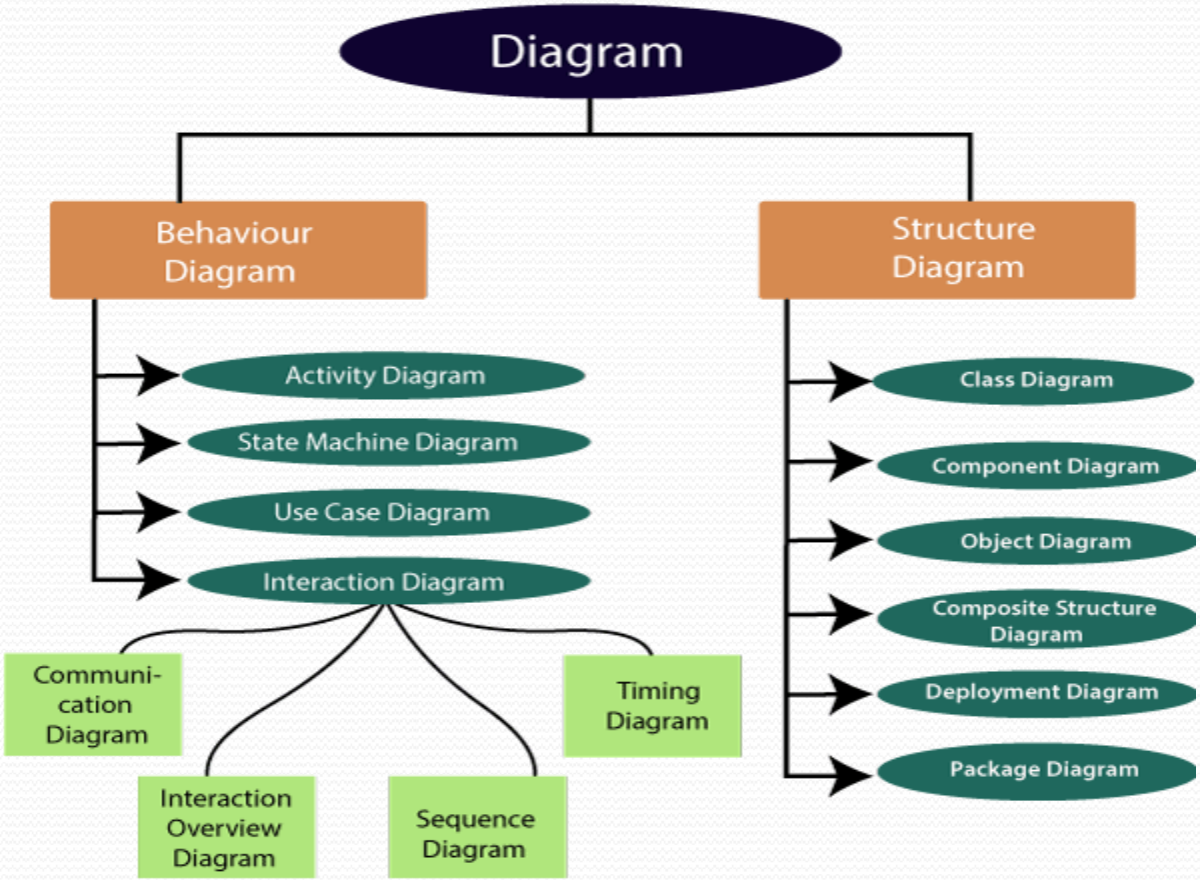
Develop to an interface, not to an implementation.

Favor object composition over inheritance.

Unified Modeling Language (UML)

Unified Modeling Language (UML) is a standardized visual modeling language used in the field of software engineering to provide a general-purpose, developmental, and intuitive way to visualize the design of a system. UML helps in specifying, visualizing, constructing, and documenting the artifacts of software systems.

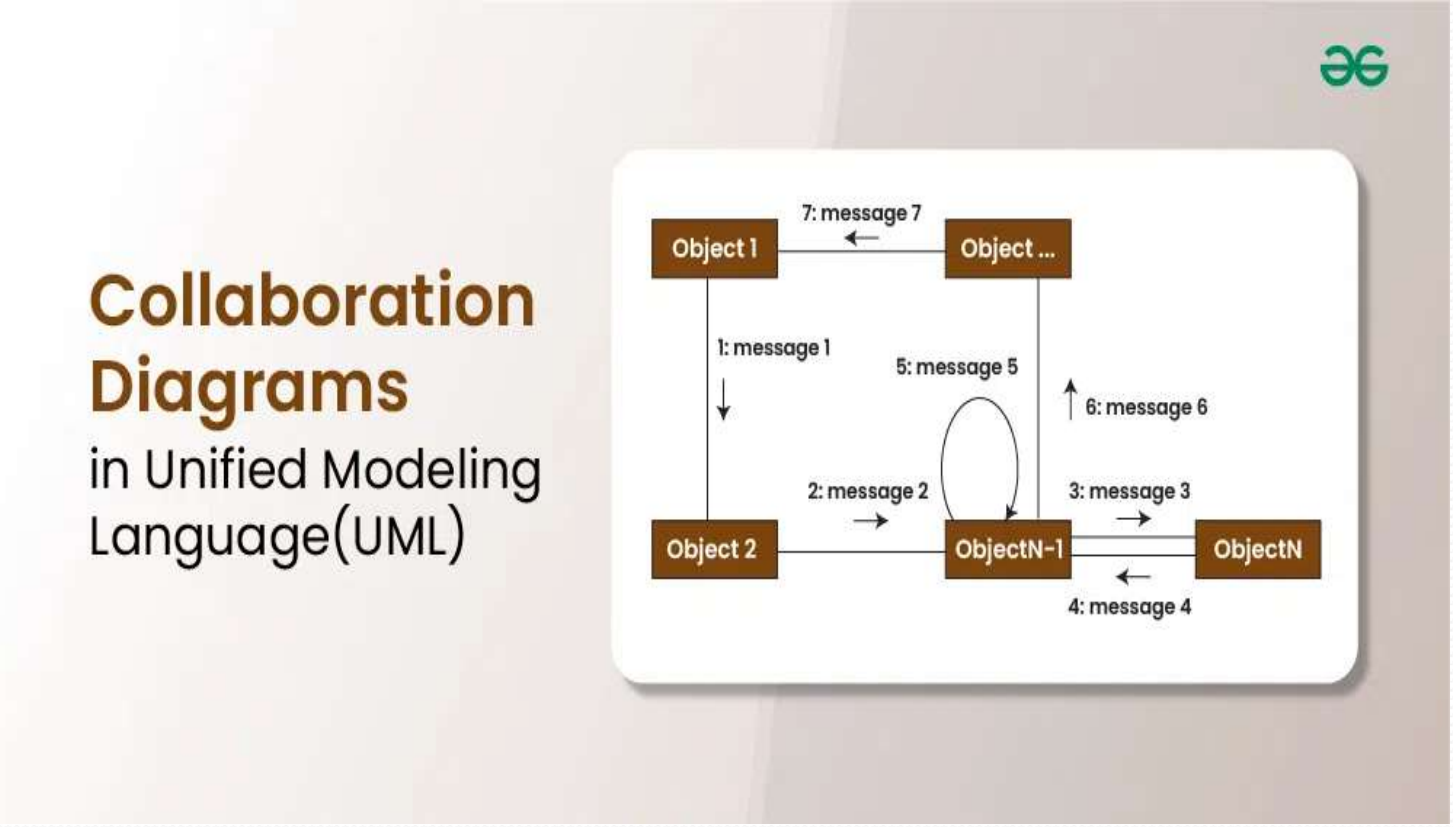
Unified Modelling Language (UML)



Class Diagram: Class diagrams are one of the most widely used diagrams. It is the backbone of all the object-oriented software systems. It depicts the static structure of the system. It displays the system's class, attributes, and methods. It is helpful in recognizing the relation between different objects as well as classes.

Sequence Diagram: It shows the interactions between the objects in terms of messages exchanged over time. It delineates in what order and how the object functions are in a system.

Collaboration Diagrams



A collaboration diagram is a behavioral UML diagram which is also referred to as a communication diagram**.** It illustrates how objects or components interact with each other to achieve specific tasks or scenarios within a system.

Use Case Diagram: It represents the functionality of a system by utilizing actors and use cases. It encapsulates the functional requirement of a system and its association with actors. It portrays the use case view of a system.

Component Diagram: It portrays the organization of the physical components within the system. It is used for modeling execution details. It determines whether the desired functional requirements have been considered by the planned development or not, as it depicts the structural relationships between the elements of a software system.

UNIT-IV

Testing Strategies

Testing Strategies

A Strategic Approach to Software Testing

Softwaretestingistheprocessofevaluatingasoftwareapplicationtoidentifyifitmeetsspecified requirements and to identify any defects. The following are common testing strategies:

1. Black box testing– Teststhefunctionalityofthesoftwarewithoutlookingattheinternalcode structure.
2. White box testing–Tests the internal code structure and logic of the software.
3. Unit testing– Tests individual units or components of the software to ensure they are functioning as intended.

Test Strategies for Conventional Software:

Conventional testing is defined as traditional testing where the main aim is to check whether all the requirements stated by the user are achieved.

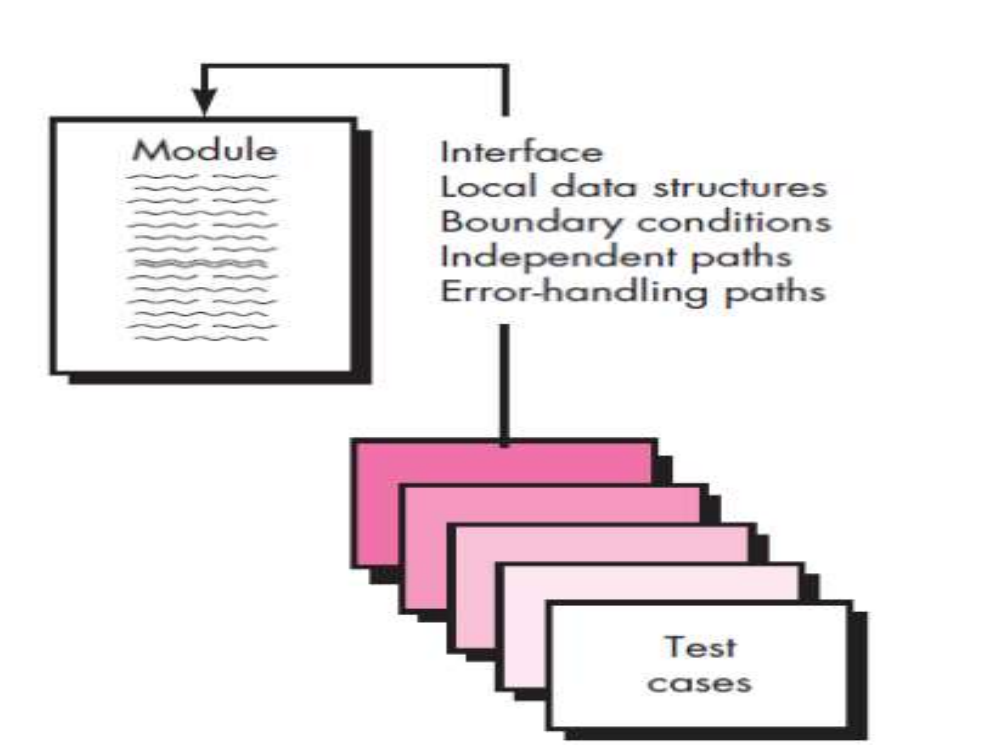
The difference between conventional testing and other testing approach is that it concentrates on checking all the requirements given by the user rather than following a software development life cycle.

Conventional testing mainly focuses on functional testing.

TEST STRATEGIES FOR CONVENTIONAL SOFTWARE

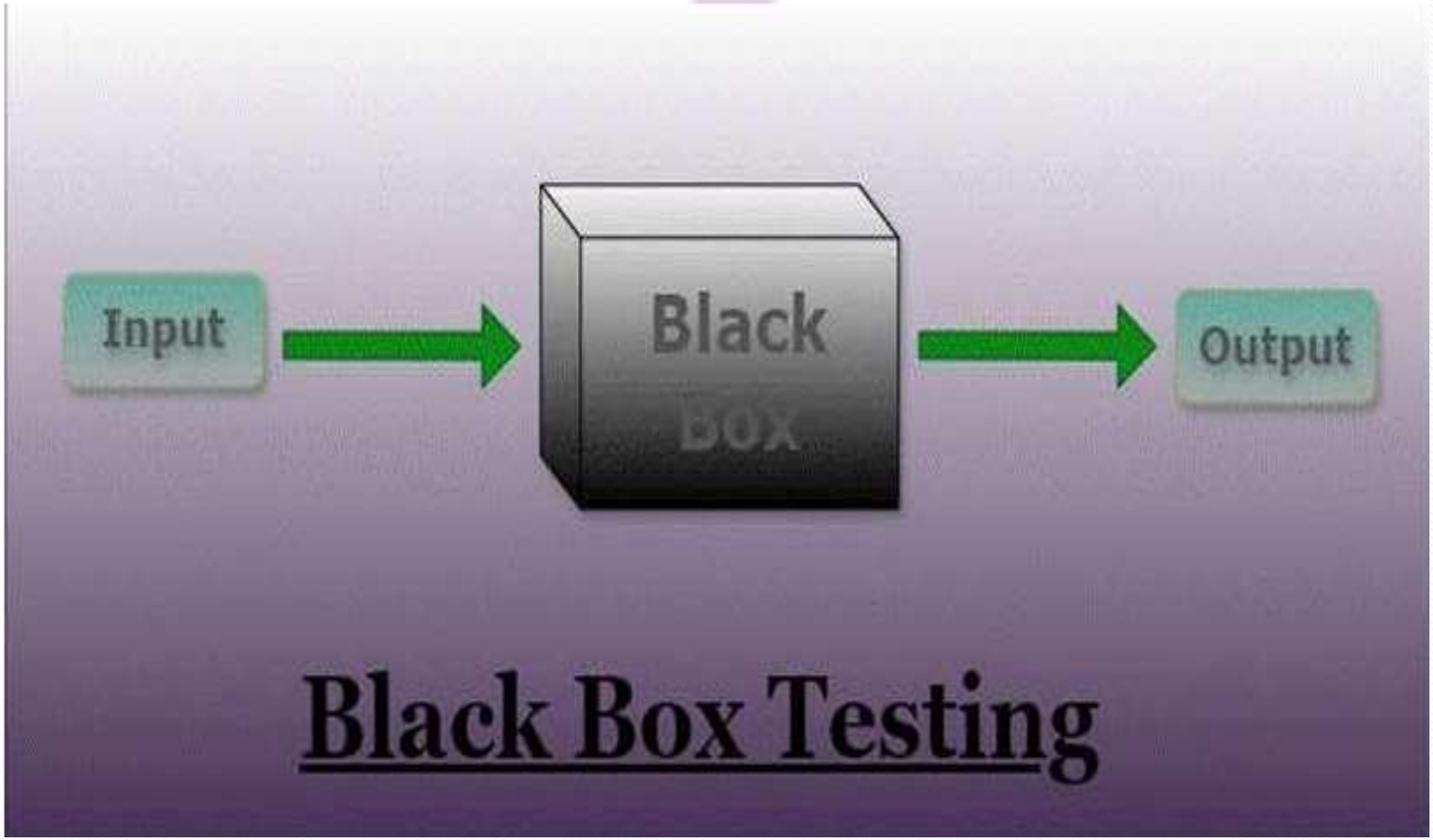
Unit Testing The unit test focuses on the internal processing logic and data structures within the boundaries of a component. This type of testing can be conducted in parallel for multiple components. Unit-test considerations:- 1. The module interface is tested to ensure proper information flows (into and out). 2. Local data structures are examined to ensure temporary data store during execution.

Test Strategies for Conventional Software



Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.


Input
Black
BOX
Output
Black Box Testing

White Box Testing

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

Types Of White Box Testing

White box testing can be done for different purposes. The three main types are:

1. Unit Testing
2. Integration Testing
3. Regression Testing

Unit Testing Checks if each part or function of the application works correctly. Ensures the application meets design requirements during development.

Integration Testing Examine show different parts of the application work together.

Regression Testing Verifies that changes or updates don't break existing functionality. Ensures the application still passes all existing tests after updates.

Validation

Validation is the process of checking whether the software product is up to the mark or in other words product has high-level requirements. It is the process of checking the validation of the product i.e. it checks what we are developing is the right product. It is validation of the actual and expected products. Validation is dynamic testing.

System Testing

System Testing is a type of software testing that is performed on a completely integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated.

Debugging

Debugging in Software Engineering is the process of identifying and resolving errors or bugs in a software system. It's a critical aspect of software development, ensuring quality, performance, and user satisfaction. Despite being time-consuming, effective debugging is essential for reliable and competitive software products.

Metrics for Process and products

Software Metrics A metric is a measurement of the level at which any impute belongs to a system product or process. Software metrics are a quantifiable or countable assessment of the attributes of a software product. There are 4 functions related to software metrics:

1. Planning
2. Organizing
3. Controlling
4. Improving

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Metrics for Software Quality

In Software Engineering, Software Measurement is done based on some Software Metrics where these software metrics are referred to as the measure of various characteristics of a Software. In Software engineering Software Quality Assurance (SAQ)assures the quality of the software. A set of activities in SAQ is continuously applied throughout the software process. Software Quality is measured based on some software quality metrics.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

UNIT-V

Risk management & Quality management

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

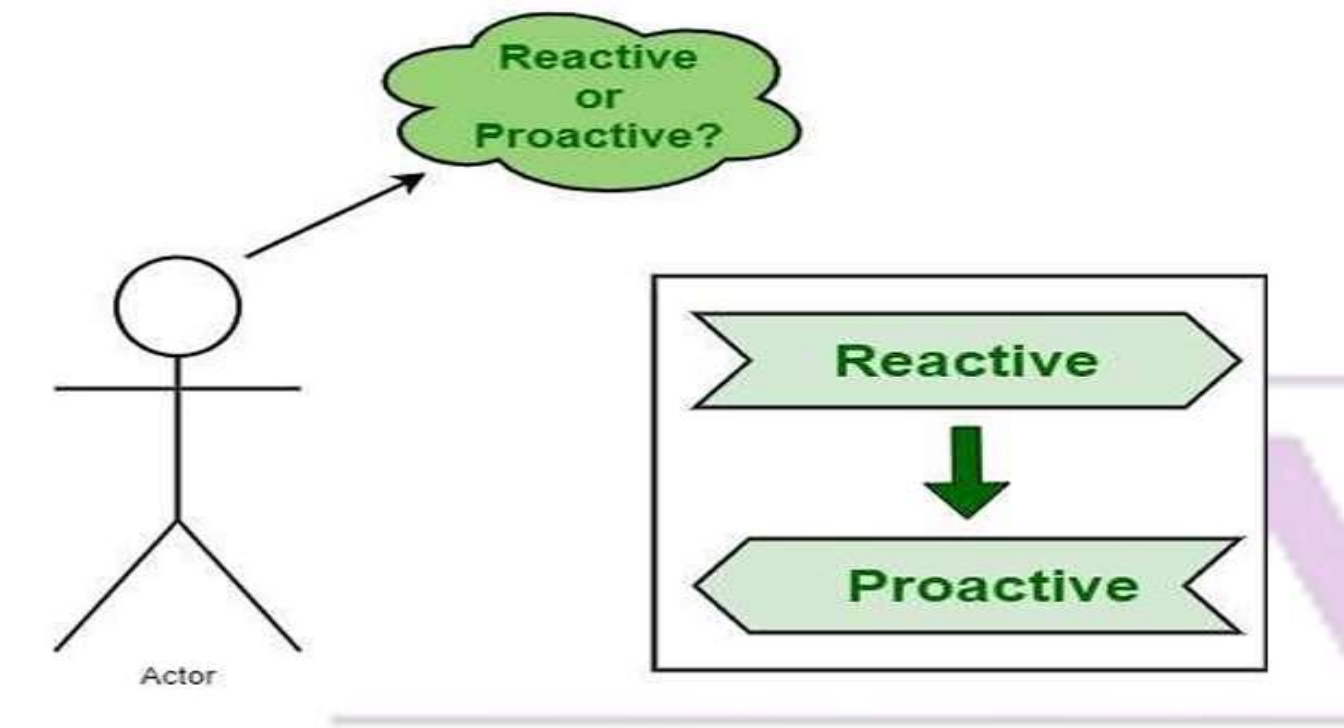
Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the software's internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements.

Reactive and Proactive Risk Strategies:



one of the best methods to identify main cause or root cause of problems or events in very systematic way or process. RCA is based on the idea that for effective management, we need to find out way to prevent arising or occurring problems.

The main question that arises is whether RCA is reactive or proactive Some people think that RCA is only required to solve problems or failures that have already occurred. But, it's not true. One should know that RCA can be both i.e. reactive and proactive as given below –

Advantages:

Helps one to prioritize tasks according to its severity and then resolve it. Increases team work and their knowledge.

Disadvantages: Sometimes, resolving equipment after failure can be more costly than preventing failure from an occurrence. Failed equipment can cause greater damage to system and interrupts production activities.

SoftwareRisks:

Software risk analysis in software development is a system atic process that involves identifying and evaluating any problem that might happen during the creation, implementation and maintaining of software systems. It can guarantee that projects are finished on schedule, within budget, and with the appropriate quality. It is a crucial component of software development.

Software risk analysis in Software Development

Software risk analysis in Software Development involves identifying which application risks should be tested first. Risk is the possible loss or harm that an organization might face. Risk can include issues like project management, technical challenges, resource constraints, changes in requirements, and more Finding every possible risk and estimating are the two goals of risk analysis.

Why perform software risk analysis

Using different technologies, software developers add new features in Software Development. Software system vulnerabilities grow in combination with technology. Software goods are therefore more vulnerable to malfunctioning or performing poorly. Many factors, including timetable delays, inaccurate cost projections, a lack of resources, and security hazards, contribute to the risks associated with software in Software Development.

Risks Identification:

Identifying risk is one of most important or essential and initial steps in risk management process. By chance, if failure occurs in identifying any specific or particular risk, then all other steps that are involved in risk management will not be implemented for that particular risk. For identifying risk, project team should review scope of program, estimate cost, schedule, technical maturity, parameters of key performance, etc. To manage risk, project team or organization are needed to know about what risks it faces, and then to evaluate them. Generally, identification of risk is an iterative process

Risk Mitigation, Monitoring and Management(RMMM)

A risk management technique is usually seen in the software Project plan. This can be divided into Risk Mitigation, Monitoring, and Management Plan (RMMM). In this plan, all works are done as part of risk analysis. As part of the overall project plan project manager generally uses this RMMM plan. In some software teams, risk is documented with the help of a Risk Information Sheet (RIS). This RIS is controlled by using a database system for easier management of information i.e creation, priority ordering, searching, and other analysis.

QUALITY CONCEPTS:

(1) A quality management approach,
(2) Effective software engineering technology(methods and tools),
(3) Formal technical reviews that are applied throughout the software process,
(4) a multitier testing strategy,
(5) Control of software documentation and the changes made to it,
(6) A procedure to ensure compliance with software development standards(when applicable),
(7) measurement and reporting mechanisms.

Software Quality Assurance

Software Quality Assurance (SQA)is simply a way to assure quality in the software. It is the set of activities that ensure processes, procedures as well as standards are suitable for the project and implemented correctly. Software Quality Assurance is a process that works parallel to Software Development. It focuses on improving the process of development of software so that problems can be prevented before they become major issues. Software Quality Assurance is a kind of Umbrella activity that is applied throughout the software process.

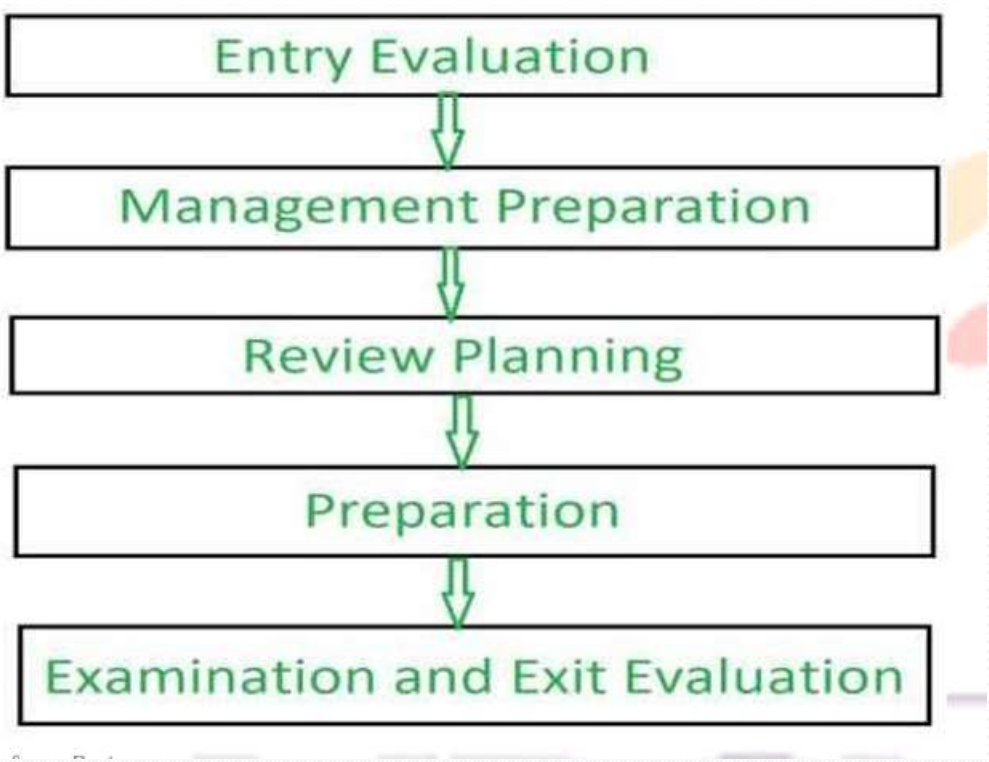
Major Software Quality Assurance (SQA) Activities

SQA Management Plan : Make a plan for how you will carry out the SQA throughout the project. Think about which set of software engineering activities are the best for project. check level of SQA team skills.

2. Set The Check Points : SQA team should set checkpoints. Evaluate the performance of the project on the basis of collected data on different check points.

3. Measure Change Impact : The changes for making the correction of an error sometimes re introduces more errors keep the measure of impact of change on project. Reset the new change to check the compatibility of this fix with whole project.

Process of Software Review



FormalTechnicalReviews:

FormalReview generallytakesplaceinpiecemealapproachthatconsistsofsix different steps that are essential. Formal review generally obeys formal process. It is also one of the most important and essential techniques required in static testing.

Phases of Formal Reviews

ISO9000Certification

TheInternationalorganizationforStandardizationisaworldwidefederationofnational standard bodies. The International standards organization (ISO) is a standard which serves as a for contract between independent parties. It specifies guidelines for development of quality system.

ISO9000Certification