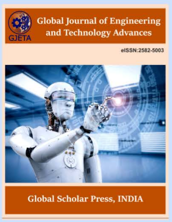
Global Journal of Engineering and Technology Advances
eISSN: 2582-5003
Cross Ref DOI: 10.30574/gjeta
Journal homepage: https://gjeta.com/



(Review Article)

Check for updates

Overview of software testing

Kusum [1,*], Pritika Talwar [2], Amit Puri [2] and Guneet Kumar [2]

[1] *Student of Computer Application, Global Group of Institutes, Amritsar, Punjab, India.*
[2] *Department of Computer Applications, Global Group of Institutes, Amritsar, Punjab, India.*

Global Journal of Engineering and Technology Advances, 2024, 19(01), 104–112

Publication history: Received on 24 February 2024; revised on 03 April 2024; accepted on 06 April 2024

Article DOI: https://doi.org/10.30574/gjeta.2024.19.1.0060

Abstract

Software testing is a task conducted to assess the quality of software and enhance its performance. This process ensures that the software behaves as expected according to user requirements. The software development life cycle (SDLC) outlines various stages such as Analysis, Requirements, Design, Development, Testing, Deployment, and Maintenance to guide the software development process. The primary objective of SDLC is to deliver error-free software that meets user expectations within specified timeframes. Software testing is an integral part of SDLC, serving to identify issues and ensure the completeness, correctness, and quality of the developed software. The ultimate goal is to create efficient software and maintain high-quality assurance throughout the product's lifecycle. This review paper aims to provide a brief discussion on how the software development life cycle contributes to efficient software development and how the software testing life cycle process ensures better quality assurance for the product.

Keywords: Software testing; Software development life cycle (SDLC); Software Testing Life Cycle (STLC); Unit Testing; Integration Testing; System Testing; Acceptance Test; Levels of Testing

1. Introduction

Software testing is a technique employed to evaluate the functionality of a software program. The software's compliance with anticipated requirements is verified, and efforts are made to guarantee that the software is devoid of bugs. The objective of software testing is to detect errors, faults, or omissions in relation to specified requirements. The emphasis is primarily on assessing the specification, functionality, and performance of a software program or application. Systematic software testing is conducted with the aim of discovering defects in a system, and it is essential for system evaluation. With the continuous advancement of technology, we observe a widespread digitization of various aspects. Online banking services, shopping from the convenience of one's home, and numerous other options are made available as technology progresses. The identification and resolution of defects, bugs, or potential issues that could impede the smooth operation of the software is the primary goal of software testing. Through the early detection and addressing of these defects in the development process, a stable and dependable software application is guaranteed by testing. Various phases exist in the software testing life cycle, such as Requirement Analysis, Test Planning, Test Case Development, Environment Setup, Test Execution, and Test Cycle Closure/Reporting. Each of these phases is equipped with specific goals to ensure the enhancement of the quality of any developed software. The testing of software follows various processes across different organizations, but there are common fundamental steps involved. Nowadays, software development typically adheres to a structured approach known as the Software Development Life Cycle (SDLC). Similarly, for conducting the testing process of the software, there is a systematic framework known as the Software Testing Life Cycle (STLC). In addition to these life cycles, various software testing techniques are employed to improve and enhance the quality of the software. These techniques contribute to transforming the software into a higher standard of quality through rigorous testing procedures.

* Corresponding author: Kusum

2. Software development life cycle

A structured process, known as the Software Development Life Cycle (SDLC), is employed for designing, developing, and testing high-quality software. SDLC, a methodology that outlines the entire software development procedure step-by-step, is utilized in this context. The primary goal of the Software Development Life Cycle (SDLC) is to create software of superior quality that not only meets but surpasses customer expectations, all while adhering to predetermined timelines and cost estimates [1].

SDLC, which stands for Software Development Life Cycle, is alternatively referred to as the Software Development Process. It functions as a framework that outlines the tasks carried out at each stage within the software development process.

2.1. Steps to Software Development Life Cycle

The Software Development Life Cycle (SDLC) process varies somewhat for each team and product**.** However, these are the stages that most SDLC frameworks have in common:

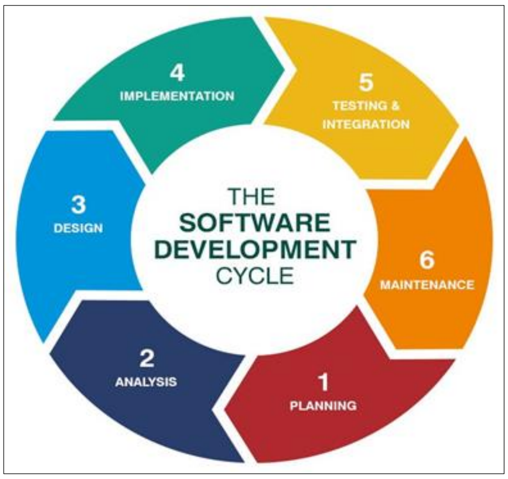


Figure 1 Software Development Life Cycle

The above Figure1 shows the steps to software Development Life Cycle. It is designed by https://datarob.com/essentials-software-development-life-cycle .

- **Planning:** In software development, planning is considered a crucial step, during which requirement analysis is also carried out by the organization's developers. This analysis is derived from customer inputs, as well as surveys conducted by the sales department or market research.[1] At this stage, planning for the quality assurance requirements and identification of the risks associated with the projects are also undertaken [2].
- **Analysis:** Often referred to as Software Requirements Specification (SRS), a complete and comprehensive description of the behavior of the software to be developed is provided. It involves system and business analysts in defining both functional and non-functional requirements [15].
- **Design:** The process of planning and problem-solving for a software solution is undertaken. It involves software developers and designers in defining the plan for a solution, which includes algorithm design, software architecture design, database conceptual schema and logical diagram design, concept design, graphical user interface design, and data structure definition [15].
- **Implementation:** In this phase, the realization of business requirements and design specifications into a concrete executable program, database, website, or software component is referred to. The process involves programming and deployment, where the real code is written and compiled into an operational application, and where the database and text files are created. In other words, it is the process of converting the entire requirements and blueprints into a production environment [15].
- **Testing and Integration**: After the generation of the code, it undergoes testing against the requirements to ensure that the products address and fulfill the needs gathered during the requirements stage. In this stage, unit testing, integration testing, system testing, and acceptance testing are conducted.[2] The primary task in this phase is to identify errors, bugs, and defects in a product after its development. The main objective of this phase in the software development life cycle is to achieve customer satisfaction by improving the product's

quality and fulfilling customer requirements [4]. At the outset of this phase, a Software Requirement Specification (SRS) is created to capture customer requirements and expectations from the product.

- Types of testing to do in this phase:
    - Performance testing
    - Functional testing
    - Security testing
    - Unit testing
    - Usability testing
    - Acceptance testing
- **Maintenance**: Once the developed systems are put into use by the client, real issues arise, and requirements need to be addressed periodically. When a developed product is handed over to the user, the user initiates the utilization of the product, giving rise to various types of problems during its use [2]. These problems must be addressed within a limited timeframe. In this phase, any necessary feature updates, resolution of restrictions, and bug fixes are performed. Essentially, if there are any errors, bugs, or defects, they are resolved in this phase [4].

3. Software testing

Software testing is a method employed to evaluate the functionality of a software program. The assessment involves checking whether the actual software aligns with the expected requirements and ensures that the software is devoid of bugs. The purpose of software testing is to detect errors, faults, or missing requirements in comparison to the actual requirements. The focus is primarily on measuring the specification, functionality, and performance of a software program or application [1]. An independent and objective view of the software, along with assurance of its fitness, is provided by software testing. All components are tested under the required services to confirm whether they satisfy the specified requirements or not. Information about the quality of the software is also conveyed to the client through this process [2]. Test oracles may encompass various factors such as specifications, contracts, comparable products, past versions, intended purpose inferences, user expectations, standards, laws, or other relevant criteria [4].

3.1. Software testing can be divided into two parts:

- **Verification:** Verification involves a series of tasks aimed at confirming that the software accurately executes a particular function. In essence, it addresses the question, "Are we constructing the product correctly?"
- **Validation:** Validation encompasses a distinct set of tasks focused on ensuring that the constructed software aligns with customer requirements. This perspective tackles the query, "Are we constructing the right product?"

Verification and validation together form crucial aspects of software development, ensuring both accuracy in implementation and alignment with customer expectations [1].

4. Software quality life cycle

The Software Testing Life Cycle (STLC) is a systematic method employed for testing a software application, aiming to confirm its compliance with requirements and absence of defects. This process is characterized by a sequence of steps or phases, each having well-defined objectives and deliverables. The STLC is utilized to guarantee that the software attains a high level of quality, reliability, and fulfills the requirements of end-users [1].

The Software Testing Lifecycle (STLC) is a method for testers to adhere to a sequence of steps aimed at delivering an application free of bugs and aligned with user requirements. In testing, STLC represents a systematic approach with the goal of fulfilling software quality requirements. Despite its apparent similarity to the Software Development Lifecycle (SDLC), it significantly diverges from that concept [5]. The Software Testing Life Cycle (STLC) involves the execution of seven fundamental steps for testing purposes. The various steps in the software testing life cycle include Requirement Analysis, Test Planning, Test Case Development, Environment Setup, Test Execution, and finally Reporting and Closure activities upon completion of the evaluation [4].

The below **figure2** shows the Software Quality Life Cycle. It is designed by https://www.openxcell.com/blog/software-testing-all-you-need-to-know/ .

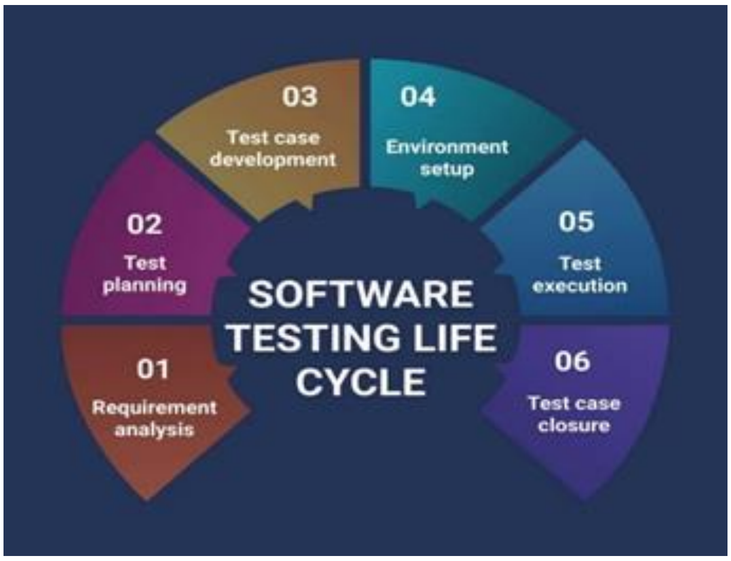


Figure 2 Software Testing Life Cycle

- **Requirement Analysis:** The first step of the Software Testing Life Cycle (STLC) is Requirement Analysis. In this phase, the quality assurance team comprehends the requirements, determining what is to be tested. If anything is missing or unclear, the quality assurance team collaborates with stakeholders to gain a more detailed understanding of the requirements [1]. The requirements may fall into two categories: functional or non-functional. Additionally, during this stage, an assessment of the feasibility of automation for the testing project is also conducted [5].
  Activities
  - Create a Requirement Traceability Matrix (RTM)[5].
  - The details of the test environment where testing will occur [5].
  - Conduct an analysis of automation feasibility if necessary [5].
- **Test Planning:** The most efficient phase of the software testing life cycle, where all testing plans are defined, is known as Test Planning. In this phase, the estimated effort and cost for the testing work are calculated by the manager of the testing team. After the Requirement Analysis Phase has been successfully completed, this phase comes into play[1]. It yields testing strategy and effort estimation documents. Test case execution can commence once Test Planning is successfully accomplished [2].
  Activities
  - Defining the goals and boundaries of the testing process Selection of test tools [2].
  - Effort estimation for testing, along with resource planning and the determination of roles and responsibilities, is carried out [2].
  - Training requirements are identified [1].
- **Test case development**: The test case development phase commences after the completion of the test planning phase. In this phase, detailed test cases are documented by the testing team, and the necessary test data is also prepared. Subsequently, the quality assurance team reviews the prepared test cases [2]. If the test environment is provided by the development team, the involvement of the test team may not be necessary in this activity. A readiness check (smoke testing) of the provided environment is required to be performed by the test team [5].
  Activities
  - Determining the test cases to be created.
  - Evaluate and establish a standard for test cases and scripts.
  - Test data and test scenarios that will be utilized in the test cases are created.
  - Generate test data if the test environment is accessible.
- **Environment setup:** Setting up the test environment is a crucial aspect of the Software Testing Life Cycle (STLC). Essentially, the test environment determines the conditions under which the software is tested. This is a separate and independent activity that can commence concurrently with test case development. During this process, the testing team is not directly engaged, and the responsibility for creating the testing environment lies with either the developer or the customer [1].
  Activities
  - Comprehend the necessary architecture, establish the environment, and compile a list of hardware and software requirements for the test environment.

  - Establish the test environment and configure the test data.
  - Conduct a smoke test on the software build.
- **Test Execution**: Following the completion of test case development and test environment setup, the test execution phase commences. During this phase, the testing team initiates the execution of test cases prepared in the preceding step [1]. The procedure involves executing test scripts, maintaining them, and reporting any identified bugs. In the event of reported bugs, the task is returned to the development team for correction, followed by retesting [5].

  Activities
  - Tests are executed as per the plan.
  - Test results are documented, and defects for failed cases are logged.
  - Defects are mapped to test cases in the Requirements Traceability Matrix (RTM).
  - Monitor the defects until they are resolved and closed.
- **Test Case Closure:** The concluding phase of the Software Testing Life Cycle (STLC) is marked by Test Case Closure. The primary objective of the test closure stage is to ensure that all activities related to testing have been concluded, and the software is deemed ready for release [1]. In this phase, the development strategy, testing procedures, and potential defects are assessed for the purpose of incorporating these practices into future use if there is software with the same specifications [2]. By the end of the test closure stage, a clear understanding of the software's quality and reliability should have been gained by the testing team, and any identified defects or issues during testing should have been addressed. The documentation of the testing process and any lessons learned is also part of the test closure stage, enabling their utilization for enhancing future testing processes [1].

  Activities
  - Assess the fulfillment of cycle completion criteria, considering factors such as time, test coverage, cost, software, critical business objectives, and quality.
  - Record the insights gained from the project.
  - The quality of the work product is reported to the customer through both qualitative and quantitative means.
  - The test results are analyzed to identify the distribution of defects by type and severity.

5. Testing methods

The various strategies or approaches used to test an application, ensuring it behaves and appears as expected, are referred to as software testing methodologies. Various methods can be employed for software testing, and this chapter provides a brief overview of the available options [6].

- White Box Testing
- Black Box Testing
- Gray Box Testing

The below **figure3** shows the testing methods [designed by using basic word art.]

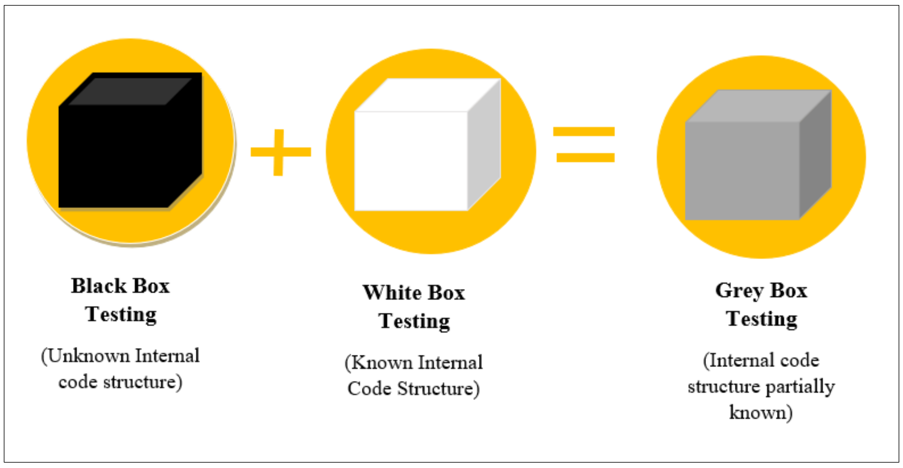

Black Box Testing
(Unknown Internal code structure)
+
White Box Testing
(Known Internal Code Structure)
=
Grey Box Testing
(Internal code structure partially known)

Figure 3 Testing Methods

5.1. White box testing

White box testing is a method for designing test cases that relies on the procedural design's control structure to derive them. Implementation errors, like inadequate key management, can be identified through this method by examining the internal workings and structure of software. White box testing is applicable across various levels of the software testing process, including integration, unit, and system levels. In white box testing, the source code is examined by the tester to identify which units of code are behaving inappropriately [15]. The below **figure4** shows the White Box Testing [designed by using basic word art.]

5.1.1. White box testing

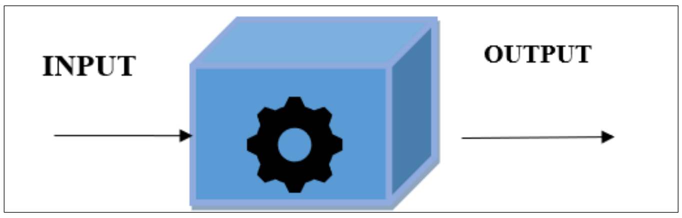


Figure 4 White Box Testing

Advantages

- Ideal and effective for extensive code segments.
- There is no need for access to the code.
- Clearly distinguishes the user's viewpoint from the developer's perspective by defining roles visibly.

Disadvantages:

- Limited coverage occurs as only a selected number of test scenarios are actually performed.
- Designing the test cases is challenging.

5.2. Black box testing

Black box testing is a software testing technique utilized to determine the functionality of an application. The primary focus is on the available input for an application and the expected outputs for each input value. This method relies on software requirements and specifications, and the internal workings of the item being tested are not known by the tester. It is also known as specification-based testing or behavior testing. In this testing approach, the internal code implementation of the application is not required to be known by the tester. Both valid and invalid inputs are handled according to customer requirements [16]. The below **figure5** shows the Black Box Testing [designed by using basic word art.]

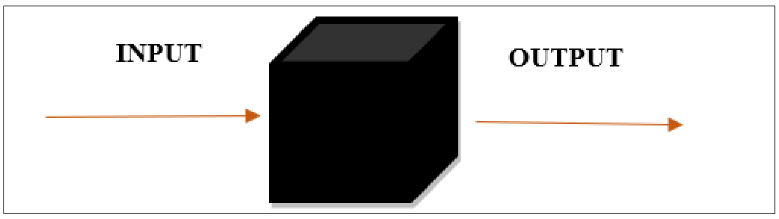


Figure 5 Black Box Testing

Advantages

- Ideal and effective for extensive code segments.
- There is no need for access to the code.
- Clearly distinguishes the user's viewpoint from the developer's perspective by defining roles visibly.

Disadvantages:

- Limited coverage occurs as only a selected number of test scenarios are actually performed.
- Designing the test cases is challenging.

- Testing efficiency is compromised as the tester possesses only limited knowledge about the application.

5.3. Grey box testing

Grey Box Testing is a product testing approach that combines elements of both Black Box Testing and White Box Testing. In Black Box Testing, the internal structure of the item under scrutiny is unknown to the tester, whereas in White Box Testing, the internal structure is known. In Grey Box Testing, partial knowledge of the internal structure is possessed.This entails having access to internal data structures and algorithms for the purpose of designing test cases, while still conducting testing at the user, or black-box, level [19]. The below **figure6** shows the Gray Box Testing. [designed by using basic word art.]

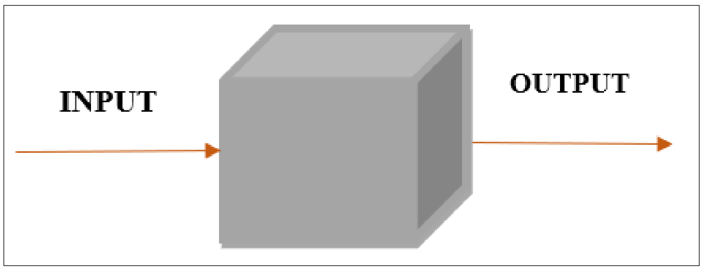


Figure 6 Gray Box Testing

Advantages

- Combined benefits of black-box and white-box testing are offered wherever possible.
- The test is conducted from the user's perspective rather than the designer's.
- Relying on interface definition and functional specifications, grey-box testers do not depend on the source code.

Disadvantages

- Due to the unavailability of access to the source code, the capability to review the code and achieve comprehensive test coverage is restricted.
- The tests may become redundant if a test case has already been run by the software designer.
- It is impractical to test every conceivable input stream as it would require an unreasonable amount of time.

6. Levels of testing

Before it can be deemed bug-free and ready for deployment, software must undergo various testing stages. The different methodologies that can be employed during software testing are involved in the levels of software testing. They are typically four levels of software testing, as outlined below

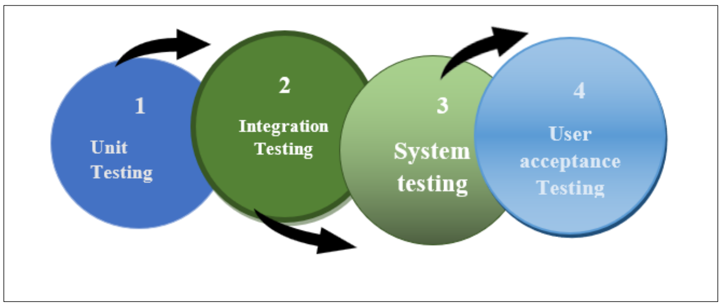


Figure 7 Levels of Testing

The above figure 7 shows the Levels of Testing [designed by using basic word art].

6.1. Unit testing

Unit testing is a code-based testing process performed by developers, primarily aimed at testing each individual unit separately. This testing can be conducted for small code units, typically no larger than a class A unit is the smallest testable portion of a system or application that can be compiled, linked, loaded, and executed. At the code level, unit testing is performed to assess each component individually, ensuring their independence and examining their functionality. Automation of unit tests is not only possible but also highly recommended in the current fast-paced development environment. To create a unit test, it is necessary to define the expected behavior of the code and write corresponding code to verify if the expectations are met [18]. The unit test should then be executed to confirm that everything operates as anticipate. Unit testing is done in various individual components of a system. And this is also used for the validation of whether each component is working is same as user requirements or not. This test is done by three approaches i.e., Black box testing and with the help of white box testing, and unit test is done by the developers of the software.

6.2. Integration testing

Integration testing is the second level of software testing, following unit testing in the testing process. In this testing, two or more unit-tested modules are integrated for testing. The technique involves interacting with components, and the integrated modules are then verified to determine if they function according to expectations. Additionally, interface errors are detected during this process.

The testing of data flow from one module or component to other modules is primarily done through integration testing. The primary objective of integration testing is to detect defects at the interaction points between integrated components or units. When each component or module operates independently, the examination of data flow between the dependent modules is carried out, and this process is referred to as integration testing.

6.3. System testing

System testing is focused on testing an entire system according to its specifications and encompasses various activities like functional testing (testing based on behavioral descriptions of the system) and performance testing (evaluating response time and resource usage). Essentially, the implementation being tested is compared to its intended specification. In this article, emphasis is placed on functional system testing, and therefore, deriving test cases from the analysis stage is of interest [19].

System Testing is a testing phase that validates the entire and thoroughly integrated software product. It aims to assess the end-to-end system specifications. Typically, the software constitutes just one component of a broader computer-based system. In the end, the software interacts with other software and hardware systems. System Testing is characterized by a series of diverse tests specifically designed to thoroughly exercise the entire computer-based system. System testing involves the examination of fully integrated software, where all the elements of the system are tested together as a unified whole to ensure they collectively meet the system requirements.

6.4. Acceptance testing

Acceptance testing involves having the work checked by a user for the purpose of being accepted. Confidence in the software product's fitness for purpose is established through acceptance testing, which performs validation on the software product. Although other stakeholders may be involved, acceptance testing is primarily conducted by the user or customer. Defects can arise at any stage of software development, and if left unfixed early, they become increasingly costly to address. The quality of the product in terms of identified defects is measured through testing, which is carried out at various levels including Component Testing, Integration Testing, System Testing, and Acceptance Testing. Acceptance testing involves a user evaluating another's work with the aim of approving it. The purpose of Acceptance Testing is to instill confidence in the user that the software product is suitable for its intended purpose. Therefore, Acceptance Testing validates the software product. While conducted by the user or customer, it may involve other stakeholders as well [20].

7. Conclusion

This review paper extensively discusses the software development life cycle and software testing life cycles. Emphasizing the crucial role of software testing in enhancing the quality of software development, the conclusion drawn from numerous papers is that software testing is a fundamental aspect of the software development life cycle. The primary purpose of testing is to identify defects, bugs, and errors within a system or developed software. It serves as a vital component of software quality control (SQC), focusing on maintaining the quality of software products through various tests, including Unit testing, Integration testing, System testing, and Acceptance testing. Despite the testing process, there is no guarantee that no faults remain in the system. Different teams are assigned distinct responsibilities, such as a separate team handling System testing, and component developers responsible for component testing.

Interface testing is conducted to pinpoint faults in the interfaces of complex components. Test automation plays a significant role in cost reduction for testing, providing support for automated testing software tools.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] MishraApoorva, DubeyDeepty, A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios,International Journal of Advance Research in Computer Science and Management Studies, Volume 1, Issue 5, October 2013(https://www.geeksforgeeks.org/software-testing-basics/).

[2] JainRitu, Ugrasen Suman, A Systematic Literature Review on Global Software Development Life Cycle, ACM SIGSOFT Software Engineering Notes, March 2015 Volume 40 Number 2 (https://www.javatpoint.com/software-testing-tutorial)

[3] SHAFIQ SAAD, MASHKOOR ATIF, MAYR-DORN CHRISTOPH, AND EGYED ALEXANDER, A Literature Review of Using Machine Learning in Software Development Life Cycle Stages,Digital Object Identifier 10.1109/ACCESS.2021.3119746, VOLUME 9, 2021(https://stackify.com/what-is-sdlc/)

[4] Jiantao Pan, Software Testing,© 2019 JETIR January 2019, Volume 6, Issue 1,www.jetir.org (ISSN-2349-5162)

[5] SinghGuddi, A STUDY ON SOFTWARE TESTING LIFE CYCLE IN SOFTWARE ENGINEERING, Globus An International Journal of Management & IT A Refereed Research Journal Vol 9 / No 2 / Jan-Jun 2018 ISSN: 0975-721X, (https://www.guru99.com/software-testing.html)

[6] Raju Ranjan, Department of Computer Science and Engineering, A Review Paper on Software Testing, © 2019 JETIR January 2019, Volume 6, Issue 1

[7] Bassil Youssef, A Simulation Model for the Waterfall Software Development Life Cycle, International Journal of Engineering & Technology (iJET), ISSN: 2049-3444, Vol. 2, No. 5, 2012

[8] Khan Mohd. Ehmer, Khan Farmeena, A Comparative Study of White Box, Black Box and Grey Box Testing Techniques, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No.6, 2012

[9] Verma Akanksha, KhatanaAmita, ChaudharySarika C, A Comparative Study of Black Box Testing and White Box Testing, © 2017, IJCSE All Rights Reserved, Volume-5, Issue-12.

[10] Hussain Tariq,Dr.Satyaveer Singh, A Comparative Study of Software Testing Techniques Viz. White Box Testing Black Box Testing and Grey Box Testing, IJAPRR International Peer Reviewed Refereed Journal, Vol. II, Issue V, p.n. 01-08, 2015.

[11] Nidhra1Srinivas, Dondeti2 Jagruthi, BLACK BOX AND WHITE BOX TESTING TECHNIQUES –A LITERATURE REVIEW, International Journal of Embedded Systems and Applications (IJESA) Vol.2, No.2, June 2012.

[12] Briand Lionel, Labiche Yvan, A UML-Based Approach to System Testing, Carleton University TR SCE-01-01-Version 4 Revised June 2002.

[13] Pandit Pallavi, Tahiliani Swati, AgileUAT: A Framework for User Acceptance Testing based on User Stories and Acceptance Criteria, International Journal of Computer Applications (0975 – 8887) Volume 120 – No.10, June 2015.