Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

Original article

Compliance of Software Developers and Engineers with the Formal Implementation of Software Testing at the General Electricity Company of Libya

Mohammed Hasni[1], Awraida Almesmari[2]*, Ousama Awad[3], Munsif Husayn[4]

[1]*University of Derna, Derna, Libya*
[2]*Department of Software Engineering, University of Derna, Derna, Libya*
[3]*College of Engineering, University of Derna, Derna, Libya*
[4]*College of Engineering technologies, Alqubbah, Libya*
Corresponding email. wardawardaabdrazeg@gmail.com

Abstract
Software testing is one of the essential parts of the Software Development Life Cycle (SDLC). It is a means to discover errors during the execution of the program, which ensures obtaining a defect-free software system. This testing evaluates the capabilities of the program and its usability. This study aims to analyze the extent of commitment of software developers and engineers to the formal implementation of software testing and to identify the factors affecting this commitment at the Libyan Electricity Company. The study relies on the descriptive analytical survey methodology by distributing an electronic questionnaire to a sample consisting of 30 participants representing different categories (developers, engineers, programmers, managers, and technicians) from different Libyan cities working in the General Electricity Company. The results showed that the commitment to software testing within the Libyan Electricity Company still lacks formality and that the level of commitment remains limited. The most prominent challenges are weak documentation, the absence of a testing environment, and the increase in problems after delivery. The most important influencing factors are the lack of specialized staff, the absence of a clear testing policy, and a low level of awareness of the importance of testing. Based on the results of the study, several recommendations were proposed to enhance the commitment to the implementation of software testing and to apply it in the present and future. It is recommended to provide training programs to raise awareness and skills, and provide specialized staff.
Keywords. *Software Testing, General Electricity Company, Unit Testing, Integration Testing.*

Introduction

Software testing plays a critical role in software engineering as it is essential to ensure the quality, performance, security, and reliability of software systems. Through testing, developers can identify and correct any errors or defects in the program. It improves the program's overall functionality and ensures that the software meets customer needs and expectations. Software plays a pivotal role in supporting administrative and technical operations within electricity companies. These companies have increasingly relied on integrated information systems to manage their internal affairs and operate their infrastructure with high efficiency. The systems include internally developed administrative platforms designed specifically for managing human resources, monitoring legal affairs, and automating various administrative procedures. They also include ready-made systems received from external providers that cover vital areas such as finance, operations, maintenance, and network control. Based on the importance of software testing in ensuring system quality, this study focuses on analyzing the commitment of software developers and engineers at Libyan electricity companies to the formal implementation of testing according to the approved standards. Besides, it analyzes the actual practices followed in the internal work environment. The study seeks to determine the level of commitment in Unit Testing, Integration Testing, System Testing, and Acceptance Testing, the strategies used, and the manual and automated execution methods. It also aims to monitor the obstacles that hinder the application of formal testing methodologies and to provide practical recommendations that contribute to improving software quality and efficiency in line with international quality standards. This study relies on the descriptive analytical survey methodology. An electronic questionnaire was used as a main tool for collecting data from the participants, including developers, engineers, managers, and technicians. This method describes the level of commitment to software testing implementation within the Electricity Company in Libya, and aims to uncover the shortcomings or strengths in the practices followed. Due to the dearth of in-depth local studies at Libyan electricity companies, this study aims to fill this gap by analyzing the actual level of commitment in a vital institutional environment, identifying the factors influencing

Received: *10-11-2025* - ***Accepted***: *09-01-2026* - ***Published***: *14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

it, and proposing applicable practical strategies to strengthen this commitment. The expected results are likely to support decision-makers, developers, and IT teams in adopting more effective testing policies. This would positively effect on the reliability of the digital infrastructure of electricity companies and their operational efficiency.

Objective

This study aims to analyze the level of commitment of software developers and engineers at Libyan electricity companies to the formal implementation of software testing during the development of internal administrative systems, particularly in critical departments such as Human Resources and Legal Affairs. This is examined in light of actual practices and internationally recognized standards, which require rigorous testing to detect defects in order to avoid as many errors as possible. Ultimately, testing reveals defects, improves system quality, estimates software reliability, reduces failures, and verifies whether the software meets the program requirements and client specifications, or functions correctly. It also helps build customer confidence by providing them with a high-quality product [1]. The study also seeks to understand the perspective of software developers on testing, their knowledge of it, and how the company views software testing. This objective arises from the need to ensure the accuracy and reliability of these systems, given their reliance on sensitive data such as employee personnel files, administrative information, and legal records. Furthermore, the study highlights the factors influencing the level of commitment to software testing, including the organizational, technical, and human factors. Identifying these factors contributes to a broader understanding of the extent of commitment to implementing software testing. It also provides recommendations to strengthen commitment to the formal implementation of software testing within the General Electricity Company.

Significance of the study

Improving the quality of internal administrative systems in electricity companies is of great importance. Software testing is considered a fundamental part of the Software Development Life Cycle, particularly in critical departments such as Human Resources and Legal Affairs. Any malfunction in these systems may lead to errors such as the loss of employee data or weaknesses in managing legal documents, which could negatively affect the institution and its operational efficiency. The formal implementation of software testing ensures the early detection of errors, thereby reducing repair costs and improving system performance. It also enables developers to verify that the software complies with the required functional and non-functional standards, such as security, speed, and reliability. It is especially critical for systems dealing with confidential and sensitive data. This study is the first study in the local context that analyzes the extent of commitment of software developers and engineers to the formal implementation of testing. It provides a valuable contribution to the software engineering field by shedding light on a vital sector, such as electricity, with a particular focus on internal administrative systems that have not received as much empirical study compared to operational or production systems. Hence, the success of testing the operational systems (i.e., all the software systems operating in the company), which help it to complete its work easily by meeting its needs from those systems; and supporting it in making decisions more accurately. Therefore, the weakness of the administrative system (i.e., Software-based administrative work) related to this software in general is caused by the lack of adequate tests for those software systems that have not been tested by software testing methods adopted in software engineering.

Research Problem

Electricity companies increasingly rely on internal administrative systems to manage human resources and legal operations. These systems involve the processing of sensitive data such as employee records, contracts, regulations, and official correspondence. However, practices in some institutions indicate that the development of these systems may not necessarily include strict adherence to the formal implementation of software testing. This increases the likelihood of functional errors or security vulnerabilities that could affect work efficiency and data reliability. The absence of systematic testing or reliance solely on informal testing may lead to several operational problems, such as the loss of legal documents or weak compliance with regulatory standards. Hence, there arises a need to study the extent of developers' commitment to implementing software testing in these systems and to understand the factors that hinder this commitment to propose practical solutions to improve software quality and ensure its reliability. Therefore, this study addressed the following research questions:

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

Q1 – To what extent are software developers and engineers in Libyan electricity companies committed to the formal implementation of software testing in internal administrative systems?
Q2 – What are the most common types of software testing applied in departments such as Human Resources and Legal Affairs?
Q3 – What are the factors that influence the commitment to implementing these tests?

Theoretical Framework

1. Generic Software Testing Terms

Software Testing

Software Testing is an essential activity to discover all the errors and bugs in the software before actual deployment of the product [2]. Software Testing is a process to evaluate the software and identify defects [3]. Software must perform as per requirements; however, it is very common to have bugs or defects in software. The bugs can be generated during development, bug fixing, feature addition, code refactoring, and even during software maintenance [4]. Software testing seeks to analyze a product's qualities or capabilities and decide whether or not it meets the required requirements, and how to improve them. Software testing is the process of running tests to find vulnerabilities and generate defect-free software. Software testing is a well-researched topic that has experienced a lot of development work and will become increasingly significant in the future [5].

Verification

is the checking of software documents, code, design, and program. It does not involve test execution. It uses methods like auditing, inspections, and walk-through [2]. Verification is done at the beginning of the development process. It includes reviews of all customer requirements and meetings, inspections to evaluate documents, code, and specifications [6]. It is the process of checking the software concerning the specification [7].

Validation

is the dynamic method of validating and testing the actual product. It does not include executions. It uses methods like white box, black box, etc [2]. Validation is about determining if the system complies with the requirements and performs functions for which it is designed, and meets [6]. It is the process of checking software concerning the customer's expectation [7].

Quality Assurance (QA)

is a group of activities to make sure that the maintenance or/and development process is adequate to make sure that the system meets its objectives. A standardized and planned set of activities necessary to provide adequate confidence that requirements are properly established and services or products conform to specified requirements. It does not involve executions [2].

Software Quality

is defined as the ability of the developed software to meet all users' requirements as well as the company's stakeholders' requirements. Software quality can be achieved by different approaches throughout the software development lifecycle (SDLC) phases [8]. Nowadays, software quality is being prioritized, and there is a strong emphasis on developing high-quality software solutions. Before creating a high-quality software product, several software quality attributes need to be determined [5]. These quality attributes can be assured via the software quality assurance (SQA) [8].

Software quality assurance (SQA)

consists of many activities to ensure that the produced software is adequate in terms of software services and fulfills all the described requirements by the client [8]. SQA encompasses a wide range of elements, from those that arise during certain stages of software development to the majority of them. SQA requires a broad range of skills and is essential to a project's overall success. An already-existing core set of competencies is expanded to include new data fields like software and dependability. Therefore, an independent frame is necessary for SQA to function properly [5].

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

Quality Standards

Achieving software quality can be done by following the well-known international standards such as ISO/IEC and IEEE. The ISO/IEC 9000 standard family can be followed by the software manufacturer organization to achieve an optimal software quality management system. This family of standards provides guidelines to reach a good level of quality [8].

Software testing standard ISO/IEC/IEEE 29119 ISO 29119

is an international standard in the software testing process to support software testing. An organization can adapt to use in a software development life cycle, an organization's development process, or their existing. When using international standards, users will receive standards accepted by many people around the world, and the high quality of the testing process in the organization [9].

2. Types of Testing

There are three broad types of testing, as shown in (Figure 1) [7]. The type of testing that checks that every function of a software application works in accordance with the requirement specification is known as functional testing. It basically includes unit testing, integration testing, user acceptance testing, etc. Functionality of the system is checked, providing some input (valid and invalid) and observing the respective output produced. This type of testing can be easily carried out manually. The type of testing that checks the non-functional conditions (scalability, usability, endurance, etc.) of a software application is known as non-functional testing. Non-functional testing is equally important as functional testing. It basically includes performance testing, load testing, and usability testing. This type of testing is difficult to carry out manually. When the software application has been deployed, and some enhancements or changes have been made in the application, then the maintenance testing is done. It basically includes regression testing and maintenance testing [2].

3. Levels of Testing

Unit testing

This testing emphasizes the individual unit or module in isolation [10]. This type is performed at the lowest level in testing stages [6], [11]. It tests the basic unit level of functionality of a software product or application and is often called “unit”, “module”, or “component” testing [6], [7], [10], [11]. It is the most modest collection of lines of code that can be tested. Unit testing is considered a white-box testing class because it is meant to evaluate the code as implemented rather than assessing conformance to some set of requirements [7]. As this testing type is performed by developers, they need to have proper knowledge about code design. A quality assurance team member can also perform this type of testing. It is cost effective testing type, and also it is not time consuming activity. Several other testing techniques are performed under unit testing effectively, like functional testing, structural testing, etc. Unit testing does not depend on the whole system. This testing can also be performed while fixing issues in parallel [11].

Unit testing uses several effective testing techniques. The testing techniques are categorized into three types:

a. Functional Testing.
b. Structural Testing.
c. Heuristic or Intuitive Testing [7].

Integration Testing

It is performed when two or more components or modules are integrated into a larger structure [6] or when two or more tested units [11] are combined. That must work together to ensure an error-free flow of control and data among combined units and their overall correct design and integration [10]. Testing is often done on both the interfaces of the modules. It occurs after unit testing and before validation [6]. The main aim of integration testing is to combine these small units and test them together. Every time a tester needs to perform each test after adding a new unit to the existing group. On performing this testing type continuously, fewer errors may occur at the time of regression testing. If immediate bugs are reported while performing integration testing, then less effort will require for regression testing. If a new module gets added in the structure, testers then need to verify each test case repeatedly from start to end. Integration testing is an upper-level testing [11]. Integration testing is an efficient technique for constructing the program structure as well as for performing tests to uncover errors related to interfacing. The objective of integration

Received: *10-11-2025* - ***Accepted***: *09-01-2026* - ***Published***: *14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

testing is to integrate the unit tested components and test them as a group. Integration testing strategies can be broadly categorized into two principal approaches: top-down and bottom-up. The top-down strategy is an incremental procedure that begins with the main control module and progressively incorporates subordinate modules into the overall program structure. This integration may proceed in either a depth-first or breadth-first manner, ensuring that the system is gradually assembled while maintaining the hierarchical control flow. In contrast, the bottom-up strategy initiates development and testing with atomic modules, which represent the smallest functional units of the system. As integration proceeds upward, the processing required for higher-level modules is consistently supported by the availability of subordinate elements. Together, these strategies provide complementary pathways for verifying system functionality, with top-down emphasizing control and structural coherence, and bottom-up ensuring that foundational components are thoroughly validated before higher-level integration [7].

System Testing

It involves testing an integrated complete software to check against its compliance with its requirements [7] [10]. It verifies the overall interaction of components to ensure the unanimous working of all modules and programs without error [10]. It executes to ensure the end-to-end quality of the whole system, and it is often based on the functional specification of the system [6] (tests the functionality of software). Non-functional quality attributes (tests quality of software), testing such as performance, reliability, usability, security testing and maintainability [6] [10]. Validation of the whole architecture is the main aim of system testing. The quality of software can be ensured by performing system testing. From the specified requirements, this testing is performed on the production environment. System testing falls under black box testing [11]. System testing encompasses several distinct approaches, each designed to evaluate specific aspects of software performance and reliability. Recovery testing focuses on the ability of an application to withstand and recover from unexpected failures, such as crashes or hardware malfunctions. In this process, mechanisms like re-initialization, checkpointing, data recovery, and restart procedures are examined to ensure that the system can reconstruct itself correctly after forced failures. Security testing, by contrast, assesses the robustness of protection mechanisms embedded within the system. The tester actively attempts to penetrate defenses, whether by acquiring passwords through external means, deploying custom software to bypass safeguards, or overwhelming the system to deny service. The overarching goal is to identify vulnerabilities and evaluate the system's resilience against threats. Graphical user interface testing is directed toward verifying that the product's interface conforms to its specifications. This involves checking the functionality and usability of menus, buttons, icons, toolbars, dialogue boxes, and windows to ensure that the user experience is consistent and reliable. Compatibility testing, meanwhile, examines the system's ability to operate harmoniously within its broader environment. This includes verifying integration with hardware, supporting software, database management systems, and operating systems to confirm that the developed system functions seamlessly across diverse configurations. Collectively, these testing strategies provide a comprehensive framework for validating system reliability, security, usability, and interoperability [7].

Acceptance Testing

Acceptance testing is called the final stage of testing. It is a very important type of testing and is performed before delivering the system to the end user. It tests whether the product meets all specified criteria given by the customer or client [11]. Acceptance testing is known as Quality assurance (QA) testing, final testing, verification testing, and validation testing [11]. The user carries this type of testing where the product is developed externally by a third party. Acceptance testing falls under the black-box testing approach, where the user is not very much involved in the internal working of the scheme [7]. The main aim of doing this testing is to check whether the system functions properly as per the specified requirement, and no bugs should occur at this stage [11]. Acceptance testing may be executed at two different levels: one at the system provider level and another at the end-user level [7].

Classification of Acceptance Testing

User Acceptance Testing

is an essential step before the system is finally deployed to the end-user. User acceptance testing is generally done by the actual software user to ensure that it can handle the specified task in the real world scenarios [7].

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

Alpha Testing and Beta Testing

QA teams or developers generally have done this type of testing. Alpha testing is conducted in the presence of developers and in the absence of users. In this testing, the following criteria are examined, such as spelling mistakes, broken lines, and cloudy direction [7]. After completing the alpha testing successfully, beta testing is performed. Beta testing is conducted by real users who actually handle the software in real-world scenarios. The customers provide their assessment to the developer for the outcome of the experiment. It is also known as field testing. Feedback from the users is used to improve the performance of the system/product before it is released to other users/customers [7].

Operational Acceptance Testing

also known as functional preparedness testing, is an approach of assuring all the specified processes and procedures of the system are in place to allow the user/tester to use it [7].

Contact and Regulation Acceptance Testing

The system is tested against the required criteria as mentioned in the contract document. It is also proven to check if it meets all the government and local authority rules and regulations, even all the essential standards [7].

4. Software Testing Strategies

Manual Testing

Manual testing is the most established and most rigorous type of software testing [12]. Manual Testing may be a kind of code take a look acting during which test cases area unit dead manually by a tester while not victimisation any machine-driven tools. The aim of Manual Testing is to spot bugs, issues, and defects within the code application [3]. Manual code testing is the most primitive technique of all testing varieties [3,12], and it helps to seek out essential bugs within the code application. Any new application should be manually tested before its testing is machine-driven. It needs additional effort; however, it is critical to ascertain whether automation is practicable. Manual testing ideas do not need information about any testing tool. It requires a tester to perform manual test operations on the software application without the help of test automation [12].

Automation testing

Automation testing could be a code testing technique that performs exploitation of special machine-driven testing code tools to execute a test suit suite. On the contrary, Manual Testing is performed by somebody sitting in front of a PC, fastidiously capital punishment the check steps. The automation testing code may also enter test knowledge into the system under testing, compare expected and actual results, and generate careful check reports. Check automation demands extended investments of cash and resources. Successive development cycles would require the execution of the same check suite repeatedly. Employing a check automation tool, it is possible to record this check suite and replay it as needed. Once the check suite is machine-driven, no human intervention is needed. This improved the ROI of check Automation. The goal of Automation is to scale back the number of check cases to be run manually and not eliminate Manual testing altogether [3]. Every organization has a unique reason for automating software quality activities. Automated testing tools are able to execute tests, describe outcomes, and estimate results with earlier test runs. Tests completed with these tools can be run over and over again at any time. The procedure used to implement automation is called a test automation framework [12].

Software Testing Tools

Selenium is a widely used testing tool designed to automate tests performed on web browsers. Its versatility lies in its ability to execute across multiple browsers while maintaining compatibility with a range of programming languages, making it a flexible option for developers and testers [2]. Ranorex, by contrast, offers an integrated solution for mobile, web, and desktop testing. It combines an intuitive click-and-go interface suitable for beginners with a powerful integrated development environment (IDE) tailored for automation experts, though it is available only as licensed software [2]. Lambda Test represents another significant advancement in cross-browser test automation, enabling users to run Selenium-based automation tests on a secure, scalable, and reliable cloud-based Selenium Grid. This approach enhances efficiency by providing a robust infrastructure for distributed testing across diverse browser environments

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

[2]. Would you like me to expand this into a comparative academic-style discussion that highlights the relative strengths and limitations of each tool? That could make it more suitable for inclusion in a research manuscript or technical report.

5. Software Testing Techniques

The importance of software testing to software quality cannot be overemphasized. After the development of the code, it is necessary to test the software to identify all the errors, and they must be fixed before releasing the software. Although it is impossible to identify and fix all the errors in the software, every phase attempts to remove as many errors as possible. Testing helps in finding the errors; however, it cannot be concluded that the software is free of errors [13]. Thus, testing techniques can be categorized into two parts:

a. Static Testing.
b. Dynamic Testing.

Static Testing

It refers to the method of testing where the code is not executed. It does not require highly skilled professionals since the actual execution of the code is not done in this process. It starts with the initial phase of SDLC; hence, it is also known as verification testing. The main objective of static testing is to enhance the quality of software products by helping software professionals to identify and resolve their own errors that occur early during the development process. Static testing is performed on documents like SRS, design documents, source code, test suites, and web page content. It is performed before code deployment. As a result, it provides the evaluation of code and documentation. The static testing techniques include: inspection, walkthrough, technical reviews, and informal reviews [13].

Dynamic Testing

Dynamic Testing is also known as validation, and it is a kind of software testing technique in which the dynamic behavior of the code is analyzed. It requires a highly skilled professional with the proper domain knowledge. It involves testing the software for the input and output values. The Dynamic Testing is divided into two categories [13]:

- Functional Testing or Black box testing.
- Structural testing or White Box testing.

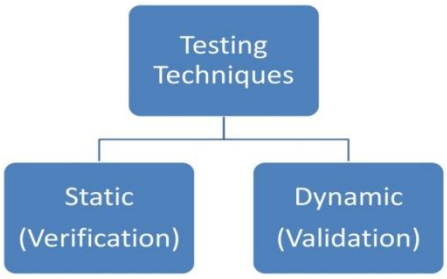

Fig.1. Testing Techniques

For the commencement of the Testing process, the first step is to generate test cases. The test cases are developed using various testing techniques for effective and accurate testing. The major testing techniques are Black box testing, White Box testing, and Grey Box testing [14].

Black Box Testing Technique

Black box testing is a technique of testing software based on output requirements and without any knowledge of the internal structure or coding in the program [15]. It tests the functionality of the application without going into its implementation details. This technique can be applied to every level of testing within the SDLC. It mainly executes the testing in such a way that it covers each and every functionality of the application to determine whether it meets the initially specified requirements of the user or not. It is capable of finding incorrect functionalities by testing their functionality at each minimum, maximum, and base case value. It is the simplest and most widespread testing process used worldwide [14].

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

White Box Testing Technique

White Box testing is called clear box or glass box testing [14], and it is significantly effective as it is the method of testing that not only tests the functionality of the software but also tests the internal structure of the application. While designing the test cases to conduct the white box testing, programming skills are required to design the test cases. This kind of testing can be applied to all levels, including unit, integration, or system testing. This type of testing fulfils the need to determine whether the information systems protect data and maintain the intended functionality. As this kind of testing process makes use of the internal logical arrangement of the software, it is capable of testing all the independent paths of a module. Every logical decision is exercised, all loops are checked at each boundary level, and internal data structures are also exercised. However, white box testing serves a purpose as a complex testing process due to the inclusion of programming skills in the testing process [14]. It is a strategy for software debugging in which the tester has excellent knowledge of how the program components interact. This method can be used for web services applications. This is done based on customers view point; only the tester knows the set of inputs and predictable outputs [15].

Grey Box Testing Technique

It is the combination of the White Box and Black Box Testing Technique serving the advantages of both [14]. It generally succeeds in combining the benefits of both black box and white-box testing. Grey-box testing takes the straightforward approach of black box testing [15]. The need for such kind of testing arose because, in this type of testing, the tester is aware of the internal structure of the application; hence, testing the functionality in a better way, taking the internal structure of the application into consideration [14]. However, it employs some limited knowledge of the inner workings of the application, and knows fundamental aspects of the system. Therefore, a tester can verify both the output of the user interface and the process that leads to that user interface output. Gray-box testing can be applied to most testing phases; however, it is mostly used in integration testing [15].

6. Software Testing Life Cycle (STLC)

The software testing lifecycle is a sequence of activities conducted to perform software testing in a systematic and planned manner. In STLC, different activities are carried out to improve the quality of the product. STLC is a subset of the Software Development Life Cycle (SDLC) [1]. The cycle of software testing comprises requirements gathering and analysis, preparing the test plan for the entire test process, scripting test cases, executing the test cases, comparing the results obtained from execution, and finally, test closure, providing all test deliverables [16].

The phases of STLC are explained below

Requirement Analysis

In this first phase of STLC, the test team studies the requirements and checks whether the requirements are testable or not. Business requirement specification and software requirement specification play a very important role in this phase [1]. In this phase of testing, the team and the Quality Assurance team study and analyze the requirements properly. The testing team and the QA team try to understand the requirements thoroughly. If the testing team and QA team are unable to understand any point from the requirements, then they can interact with stakeholders to solve the queries related to the requirements in detail. Once testers and the QA team understand the requirement properly, it gives them a clear idea about which type of testing will be required to test the product. Testing priorities can be set from this. Testers can set the environment for testing [11].

Test Planning

In this phase, the test plan and test strategies are determined [11]. Once the test team is clear with the requirements, the testers can start making test plans. A test plan is a proper strategy or approach that helps testers conduct their testing [1]. The testing manager calculates the efforts, time, and cost estimation required for testing. Hence, the testing team determines the required environment for testing, the types of testing, and the roles and responsibilities [11].

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

Test Case Development

Testers mostly determine requirements or test plans to create test cases. Testers start creating test cases and test scripts in this phase. All preconditions should be taken into consideration while creating the test case and the test script. Verification of the test case should be done in this phase. If a test environment is available, then the test team will create test data [11]. The test team also prepares the Requirement Traceability Matrix in this phase [1].

Test Environment Setup

This phase plays a vital role in the Software Testing Life Cycle. It decides the conditions under which software is tested [1][11]. This activity is independent. That is, testers can start performing this activity along with the development of the product. In this requirement, a list of hardware and software is prepared. The tester does not decide the test environment; however, it is created by the developer and customer [11]. This phase can run in parallel with the design phase. The deliverables in this phase are the test environment and the smoke test results [1].

Test Execution

After test environment setup, execution of test cases is performed based on the defined test plan [1]. Testers start executing test cases, and if any failure occurs in it, testers should report it [11]. All the positive and negative test cases must be executed and documented in a proper format. Defect report should be prepared for failed test cases and reported to the development team for rectification [1]. Then, the development team fixes that bug. Testers retest that bug and mark it as closed if it is fixed or reopen it if it is not fixed [11].

Test Closure

In this phase, evaluation of different activities, including completion time, quality, execution, business conditions, and quality of software, is carried out. Then, testers prepare the test closure report. Quantitative report should be prepared to give work product to customers. It is very important to find out how many defects have occurred, and their distribution according to their severity and priority should be done [11]. Once testing is completed, the matrix, reports, and results are documented. It is the last phase of STLC. STLC is a very important phase of SDLC, and the final product cannot be released without passing through these testing processes. The different phases of the Software Testing Life Cycle must be executed in the same order in which they are defined [1].

Fig.2. Phases of software testing life cycle.[16]

Methodology

This study is based on the descriptive-analytical survey methodology, where a questionnaire was used as the primary tool for collecting the data from the participants. The purpose of this methodology is to describe the reality of compliance with the implementation of software testing within the General Electricity Company of Libya. It also analyzed the obtained results in order to identify the potential shortcomings or strengths in the current practice and to determine the factors influencing the execution of the tests.

Received: *10-11-2025* - ***Accepted***: *09-01-2026* - ***Published***: *14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

Study Population

The study population included the employees involved in or affected by software development processes at the General Electricity Company across several Libyan cities. The sample consisted of various categories related to software, such as programmers, developers, systems engineers, managers, technicians, and quality assurance staff. These categories were selected to ensure comprehensive representation of different perspectives and to assess the extent of compliance with the formal implementation of software testing.

Data Collection Tool

An electronic questionnaire was employed as the primary data collection instrument and distributed to participants within the study population. The questionnaire consisted of close-ended questions specifically designed to assess the level of compliance with software testing practices. It addressed several key themes, including the types of tests applied (unit, integration, system, and acceptance testing), the entities responsible for conducting these tests, the extent of documentation prepared for test plans, test cases, and test results prior to system approval, and the adequacy of the testing environment before implementation. Additional areas of focus included the conduct of formal reviews of system quality prior to operation, the occurrence of software issues following delivery, the degree to which system requirements were met, and the factors influencing compliance with software testing implementation.

Based on its methodology, the study can be classified as a survey study in terms of data collection, and as a descriptive-analytical study in terms of the presentation and interpretation of results. This dual classification enables a realistic depiction of the level of compliance with software testing practices within the Libyan electricity sector. A total of 30 valid responses were obtained. To ensure the relevance of the sample, participants were initially asked whether they possessed knowledge or familiarity with software testing. If the response was negative, the questionnaire was terminated. This filtering mechanism ensured that the final sample consisted exclusively of individuals directly engaged in testing activities, including testers, developers, engineers involved in system development, and software quality specialists

Results

This section aims to present the results obtained through the analysis of the questionnaire distributed to a group of employees at the General Electricity Company across several Libyan cities. These employees represent different professional categories, including programmers, developers, systems engineers, managers, technicians, and quality assurance teams. The results focus on multiple aspects related to the level of compliance with software testing implementation, such as the types of tests applied, the entities responsible for conducting them, the level of documentation, the availability of a testing environment, the presence of a formal quality review, and the identification of software issues that emerge after system delivery. The data are presented in the form of percentages that illustrate the variation in practices among the target groups. This presentation seeks to provide a comprehensive picture of the current state of software testing practices within the company. It was found that females represented 7% in the developer community within the company, while males accounted for 93% of the participants in the company, as illustrated in (Table1).

Table1. Distribution of participants according to their gender

Gender	Respondents	Percentage
Female	2	7%
Male	28	93%

Table2. Distribution of participants by years of experience in the Electricity Company

Years of experience	Respondents	Percentage
Less than 5 years	8	27%
5-10 years	6	20%
10-15 years	4	13%
More than 15 years	12	40%

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

The results indicate that the most represented group among the participants is those with long experience (more than 15 years), accounting for 40% of the total sample. This distribution reflects the study's strong reliance on the opinions of well-experienced experts who possess extensive practical knowledge in the field of software development and systems management. The high proportion of this group adds greater credibility to the findings, given that long years of experience are associated with increased awareness of the importance of formal software testing and its application in accordance with established quality standards.

Table.3. Distribution of participants by job position in the Electricity Company

Job Roles	Respondents	Percentage
Developers	4	13%
System engineers	6	20%
Programmers	10	33%
Managers	6	20%
Quality Assurance	2	7%
Technicians	2	7%
Total	30	100%

The results showed that the most represented category was the programmers, represented by 33% of the sample, followed by systems engineers and managers, each with an equal share of 20%. In contrast, developers constituted only 13% of the total sample. Technicians and quality assurance staff were the lowest category, represented by 7%. These findings indicate that the study relied heavily on the perspectives of practitioners (programmers and systems engineers) compared to other categories. That is, developers were not significantly represented in the survey (13%) compared to programmers or systems engineers. The small number of developers may reflect the actual situation within the Electricity Company, where work is mainly concentrated among programmers and systems engineers, while developers play a smaller or more limited role. It may also suggest that development tasks are distributed among programmers rather than assigned to the development department, or that developer participation in the survey was relatively weak.

Table 4. The respondents' knowledge of software testing, their involvement in software development, and their direct experience with software testing

	Familiar with testing		Participate in development		Experience directly in testing	
	Respondents	Percentage	Respondents	Percentage	Respondents	Percentage
Yes	22	73.3%	20	66.6%	24	80%
No	6	20%	10	33.3%	6	20%
Not sure	2	6.7%	----------	----------	----------	----------

The results showed that the majority of participants (73.3%) know software testing, while 20% indicated that they had no knowledge, and 6.7% stated that they were unsure. This distribution reflects that most members of the sample possess a cognitive awareness of the fundamental concepts of software testing. The majority of participants (66.6%) had direct experience in participating in software development processes, while 33.3% had not previously been involved in such activities. This distribution indicates that two-thirds of the sample possess practical experience in system and software development, which provides the study with an important applied dimension and enhances the credibility of the findings related to compliance with software testing procedures. Moreover, most of the participants had direct practical experience in software testing, with 80% confirming that they had engaged in actual testing practices, while 20% reported lacking such experience. These finding highlights that the majority of the sample do not rely solely on theoretical knowledge but also possess applied practical experience, which strengthens the validity of the collected data. However, the presence of 20% of participants without direct experience reflects that adherence to practical testing practices is not comprehensive across all categories.

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

Table 5: The frequency of respondents across the testing levels being implemented

Testing levels	Respondents	Percentage
Unit Testing	12	20.7%
Integration Testing	8	13.8%
System Testing	20	31.0%
Acceptance Testing	22	34.5%

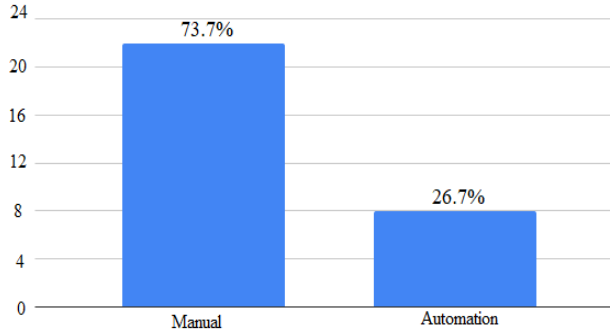

Fig 3. The proportion of manual and automated testing usage

Since the participants were allowed to select more than one type of test, the survey results revealed a noticeable variation in the levels of compliance with different software testing practices. The highest percentage was for acceptance testing (34.5%), followed by system testing (31.0%). In contrast, unit testing and integration testing were less frequently implemented (20.7%) and (13.8%), as illustrated in (Figure 4). This distribution reflects that the participants place greater emphasis on the final testing stages related to verifying system readiness and user acceptance, rather than on the initial tests of individual units or their integration. This may indicate that the company's working environment tends to prioritize overall system outputs and end-user satisfaction, while relatively neglecting early-stage testing, which is essential for ensuring software quality and reducing failures in later phases. The weak implementation rates of unit and integration testing may represent a gap in the software testing lifecycle, as these two types of tests constitute the first line of defense against software defects and contribute significantly to reducing the cost of later fixes. Therefore, the current findings highlight the need to strengthen the culture of early testing among developers and engineers and to adopt stricter policies to ensure comprehensive implementation of all testing levels. The results showed that developers' compliance with testing is still largely confined to manual execution, with a high rate of 73.3%, while reliance on automated testing did not exceed 26.7%. This reveals a gap in the level of professionalism in practice, as automated testing is expected to constitute a fundamental component of developers' adherence to modern testing standards, as illustrated in (Figure 5).

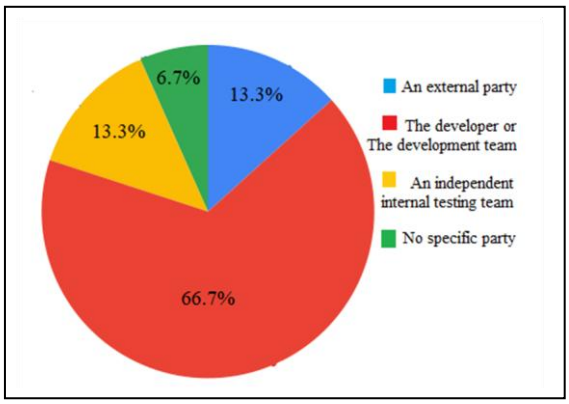

Fig.4.The person responsible for unit and integration testing

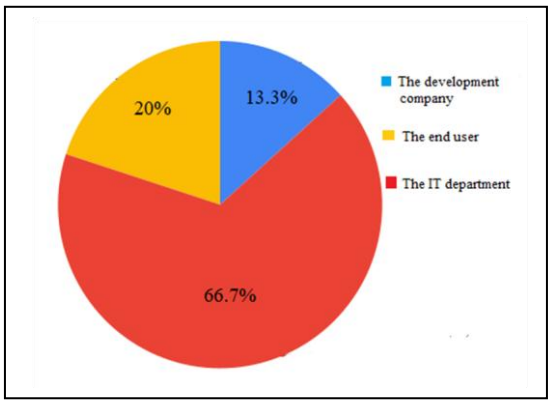

Fig.5. The person responsible for system and acceptance testing

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

The survey results showed that the responsibility for conducting unit and integration tests mostly falls on the developer himself or the development team (66.7%), while this task was assigned to an independent internal testing team (13.3%) or to an external party (13.3%). Meanwhile, 6.7% of the participants indicated that there is no specific entity responsible for these tests. The main responsibility for conducting system and acceptance tests lies with the internal IT department, accounting for 66.7% of the total participants, while 20% indicated that the end user plays this role. In contrast, only 13.3% confirmed that the developing company is responsible for performing these tests.

Table.6. illustrates the occurrence of software problems after the system delivery

issues	Respondents	Percentage
Yes, many	18	60%
Yes, but few	8	26.7%
No	2	6.7%
Not sure	2	6.7%

Furthermore, the results showed that a large proportion of participants (60%) reported recurring software problems after system delivery, while 26.7% indicated the presence of problems, but to a limited extent. In contrast, only 6.7% confirmed the absence of problems or stated that they were unaware of their existenc. This distribution reflects that the majority of developed systems still suffer from software errors after delivery, which indicates the weak effectiveness of the testing procedures applied, particularly in the early stages (unit testing and integration testing). The high percentage of participants (60%) who confirmed the existence of frequent problems after delivery highlights real challenges in quality control and ensuring the efficiency of the system's lifecycle. Neglecting the systematic application of unit and integration tests causes the accumulation of errors, which are only discovered in the final stages or even after the system is deployed. Therefore, strengthening the culture of early testing and adopting stricter methodologies for quality assurance should be considered among the top priorities in the current work environment.

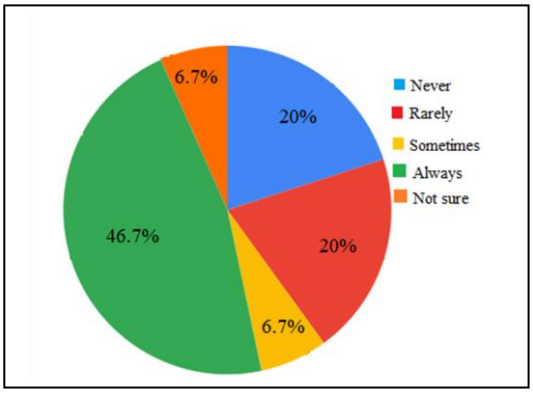

Fig.6. Documentation of the test plan and test cases

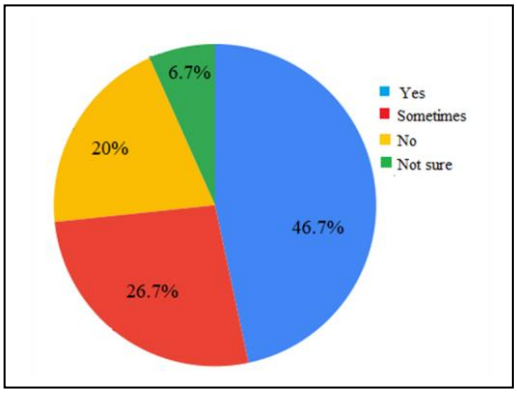

Fig.7. Documentation of test results

As for the level of documentation of testing activities, the survey results revealed that the level of documentation is still inconsistent among the participants, as 46.7% reported that they always document the test plan and test cases, while 20% rarely do so, and only 6.7% mentioned that they sometimes document them. Meanwhile, 20% of the participants do not carry out the documentation process at all, and 6.7% confirmed that they were unaware of the existence of such a procedure, as illustrated in (Figure 6). This finding is an indicator of weak institutional commitment to quality standards. The literature emphasizes that documenting the test plan and its cases constitutes the cornerstone for ensuring error traceability, evaluating test coverage, and facilitating subsequent review processes.

It was also found that the documentation of test results before system approval varies among participants, as 46.7% confirmed that they document the results regularly, while 26.7% indicated that they sometimes do so. In contrast, 20% stated that they never document the results, and 6.7% reported that they were unaware of the existence of this practice, as illustrated in (Figure 7). This distribution reflects that less than half of the sample adheres to the regular

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

documentation of test results, which indicates an institutional shortcoming in the application of systematic quality practices. Documenting test results represents a pivotal step in ensuring transparency in the acceptance process and providing a reference record to rely on in the event of future problems. Moreover, the absence or limitation of documentation may help explain the high percentage of issues that arise after system delivery due to the lack of official evidence to prove that the system has successfully passed all testing stages.

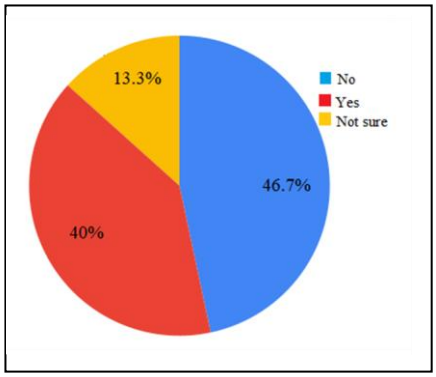

Fig.8. provision of a testing environment

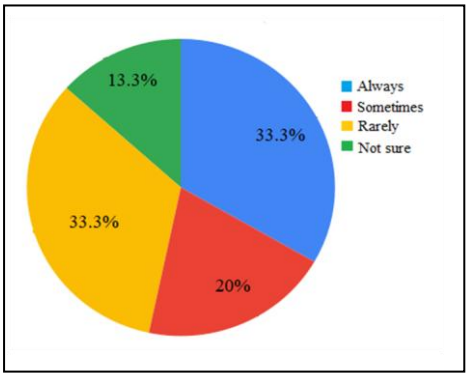

Fig.9. Conducting a formal review

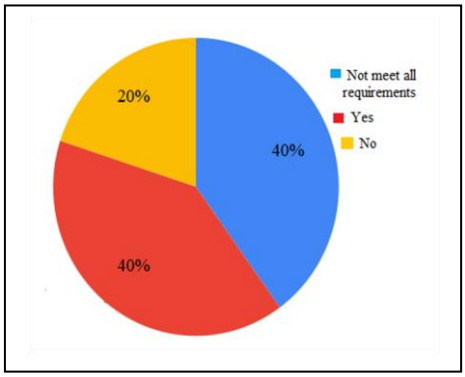

Fig.10. Functionality of delivered systems according to requirements

The survey revealed that 40% of participants confirmed the existence of a prepared testing environment before system implementation, while 46.7% indicated the absence of such an environment, and 13.3% stated that they were unaware of its existence, as illustrated in (Figure 8). This result reflects that only a small number of participants prepare a formal testing environment before implementation, while the majority of participants neglect this practice. This is a clear indicator of a fundamental shortcoming in the testing lifecycle, as the absence of an independent testing environment increases the likelihood of post-delivery failures and undermines the reliability of the results. In addition, the results showed that the formal review of system quality before operation is not carried out systematically in most cases, as 33.3% of participants indicated that this review is always conducted, while another 33.3% stated that it is rarely conducted, and 20% mentioned that it is sometimes conducted. Meanwhile, 13.3% reported that they were unaware of the existence of such a practice, as illustrated in (Figure 9). As for the functionality of the system, 40% of the participants reported that the delivered systems do not function according to all the agreed specifications and requirements, while another 40% confirmed that the systems actually meet all specifications and requirements. Meanwhile, 20% indicated that the systems do not fulfill these specifications, as illustrated in (Figure 18). This distribution reflects that nearly half of the systems suffer from gaps or shortcomings in meeting the agreed requirements, which indicates the presence of deficiencies in the testing process and quality control before delivery.

Hence, it is concluded that:

- Documentation of the test plan and test cases is always 46.7%
- Documentation of test results before approval is only 46.7%
- Preparation of a testing environment is absent by 46.7%.
- Formal review before operation is always only 33.3%.

The current result aligns with these indicators to show that the absence of a formal methodology for testing and documentation has directly affected the compliance of systems with specifications. In other words, the weak formal commitment to testing leads to the delivery of systems that do not fully meet the requirements.

- **Unit and Integration Testing**: carried out by 66.7% of developers or development teams.
- **System and Acceptance Testing**: mainly the responsibility of the internal IT department (66.7%), with a limited role for the developing company (13.3%) and the end user (20%).
- **Post-delivery Issues**: 60% confirmed the existence of many software problems after delivery.

Accordingly, it can be concluded that the commitment of developers and engineers to testing in the studied work environment does exist, but it remains selective and incomplete, as it is concentrated on the final stages while neglecting

Received: *10-11-2025* - ***Accepted***: *09-01-2026* - ***Published***: *14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

the early stages and supporting quality elements. This negatively affects the quality of the systems and leads to the emergence of problems after approval. The results also showed that the commitment to software testing within the Libyan electricity company still lacks a formal and systematic character. It was found that the main focus is placed on final tests (system and acceptance), while early tests (unit and integration) are not carried out with the same level of commitment. The results also indicated that developers play a central role in the early stages of testing, whereas responsibility for the final stages is mostly transferred to the IT department, with limited involvement from the developing company and end users. In addition, the findings revealed weaknesses in documentation (whether for the test plan or for results before approval), the absence of an independent testing environment for nearly half of the participants, and low rates of formal review before operation. These factors were reflected in the high rate of software problems observed after delivery. Therefore, it can be concluded that current practices reflect a partial and informal commitment to testing. It highlights the need to develop institutional policies and mandatory standards that would enhance the quality of software systems and reduce operational problems.

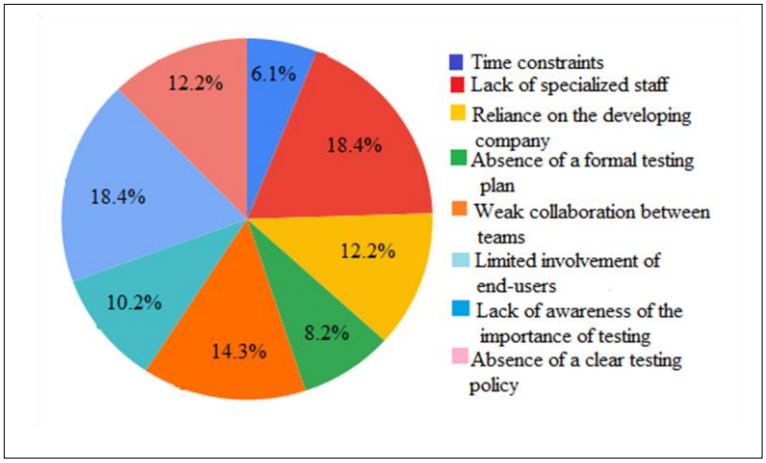

Fig.11. Factors influencing the implementation of software testing

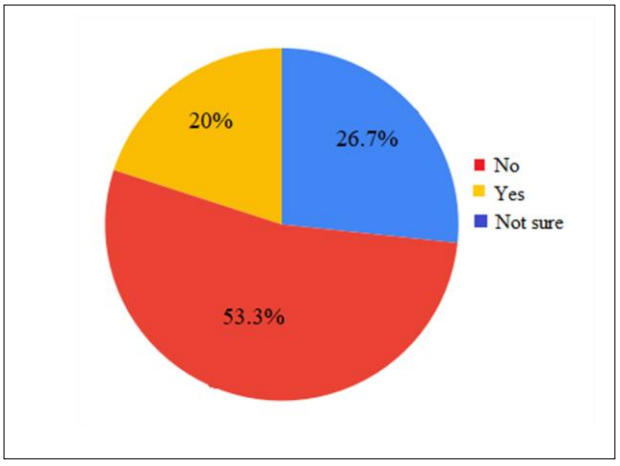

Fig.12. Implementation of quality standards

The study results revealed a set of factors that hinder the commitment to implementing software testing. For example, time constraints posed a challenge for developers (6.1%), while 8.2% of the participants indicated that the absence of a formal test plan reduces the effectiveness of testing. The lack of specialized personnel emerged as one of the most prominent human challenges by 18.4% of the participants, whereas excessive reliance on the developing company was identified as an influential factor by 12.2% of the participants. At the organizational level, 12.2% of the participants stated that the absence of a clear testing policy is considered an obstacle, while weak collaboration between teams limited implementation efficiency 14.3%. The results showed that insufficient involvement of end users constituted an additional factor by 10.2% of the participants. Finally, weak awareness of the importance of testing emerged as a major determinant, 18.2%, reflecting the need to strengthen a culture of quality and ensure the integration of testing into institutional practices (Figure 11).

In addition, the results showed that the commitment to implementing tests is also affected by the extent of applying the adopted quality standards. A number of participants indicated the absence of clear adherence to unified software testing standards, which was reflected in the variation of output quality. This mirrors the need to integrate quality standards such as ISO/IEC/IEEE 29119 or similar within institutional testing procedures in order to improve the level of commitment and increase the reliability of the developed systems. There are only 20% of the participants committed to standards, while 53.3% reported non-compliance, and 26.7% indicated that they were not aware of the existence of quality standards. This result indicates a clear gap in adopting international standards regulating software testing (Figure 12).

The results also showed that the implementation of security tests is characterized by noticeable variation. A total of 46.7% of participants stated that these tests are always conducted, while 40% reported that they sometimes conduct the

Received*: 10-11-2025 -* ***Accepted****: 09-01-2026 -* ***Published****: 14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

test, 6.7% indicated that they never conduct the test, and 6.7% mentioned that they were unaware of their existence. These results highlight that the application of security tests within the company is still inconsistent. Although a considerable percentage regularly carry them out, the partial reliance or complete absence of such tests reflects a shortcoming in ensuring the protection of systems and data from potential security risks, which requires further organization of these practices (Figure 13).

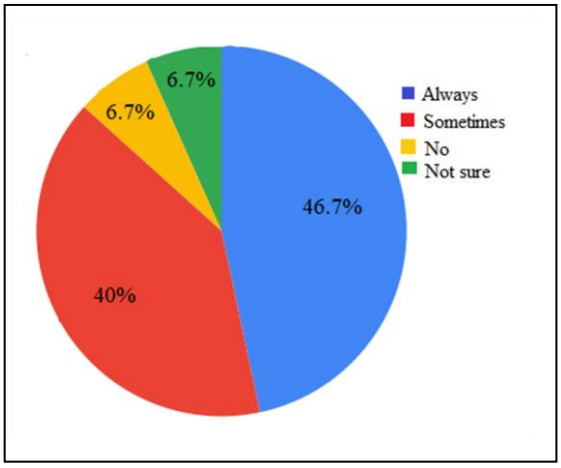

Fig.13. Implementation of security testing

These results are not limited to a particular city, but were obtained from several branches of the electricity company in different cities across Libya. This gives the study additional strength in terms of representation and makes the findings more comprehensive and objective. They reflect the reality of software testing practices in the electricity sector at the national level. Moreover, this geographical diversity reduces the likelihood of bias in the results toward a specific work environment or management. It highlights that the observed challenges (such as the absence of a testing environment, weak documentation, and the high rate of post-delivery problems) are general characteristics of the software work environment in this sector, rather than exceptional cases.

Recommendations

Based on the findings of the study, several recommendations can be advanced to strengthen organizational commitment to software testing and ensure its systematic implementation. First, the employment of specialized staff—including developers, quality engineers, and dedicated software testers—is essential to address the shortage of expertise and guarantee the structured execution of testing activities. Equally important is the enhancement of awareness and continuous professional development, which can be achieved through training programs and workshops that emphasize the significance of software testing and the use of automated methods to reduce knowledge gaps. The establishment of a formal test plan is also critical, as it provides a clear, written framework aligned with international quality standards and prevents randomness in the testing process. Complementing this, the adoption of comprehensive institutional testing policies is necessary to formally define responsibilities, roles, and mechanisms, thereby ensuring accountability and consistency. Strengthening collaboration among development, quality assurance, and technical support teams through improved communication and integration mechanisms further reduces inefficiencies and weak coordination. End-user involvement should be increased, particularly during acceptance testing, to ensure that systems meet practical requirements and expectations. Effective time management is another priority, requiring sufficient allocation of testing activities within the software project lifecycle and their inclusion in the overall work schedule. Finally, adherence to international quality standards, such as ISO/IEC/IEEE 29119, is recommended to standardize procedures, document requirements, and foster a culture of quality across the organization. Collectively, these measures provide a structured pathway toward enhancing compliance with software testing practices and improving overall system reliability.

Received: *10-11-2025* - ***Accepted***: *09-01-2026* - ***Published***: *14-01-2026*

Alq J Sci. 2026;2(1):27-43
https://doi.org/10.69667/ajs.26103
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

Conclusion

This study concluded that software testing at the Libyan electricity company still faces multiple challenges related to human, organizational, and technical aspects (such as the absence of a testing environment, weak documentation, and the high rate of post-delivery problems). The results showed that reliance on manual testing far exceeds automated testing. The commitment to quality standards and security testing is limited and inconsistent. A set of influencing factors also emerged, including the lack of specialized employees, the absence of a formal test plan, weak collaboration between teams, and limited awareness of the importance of testing. These findings indicate that software quality within the company is exposed to risks associated with software errors and frequent post-delivery modifications, which undermine the efficiency and credibility of the systems. Enhancing commitment to testing requires the adoption of clear formal policies, the provision of specialized training programs, the expansion of automated tools usage, and greater involvement of end users in the testing process. Furthermore, the study suggests that future research should expand on examining the impact of applying modern automated testing tools and linking them to international standards such as ISO/IEC/IEEE 29119, thereby increasing the opportunities to improve software quality in the local context.

Conflict of interest. Nil

References

1. AlbaUmar MA. Comprehensive study of software testing: Categories levels techniques and types. Int J Adv Res. 2019;7(6):9-16.
2. Anand A, Uddin A. Importance of software testing in the process of software development. Int J Sci Res Dev. 2019;12(6):1-5.
3. Anwar N, Kar S. Review paper on various software testing techniques & strategies. Glob J Comput Sci Technol. 2019;19(2):43-9.
4. Arumugam AK. Software testing techniques & new trends. Int J Eng Res Technol. 2019;8(12):2278-0181.
5. Ashiq S, Masood AB, Fakhar MH, Iqbal MW, Nazir Z, Muhammad HAB, et al. Challenges and barriers to software testing. Bull Bus Econ. 2024;13(1):357-66.
6. Bajjouk M, Rana ME, Ramachandiran CR, Chelliah S. Software testing for reliability and quality improvement. J Appl Technol Innov. 2021;5(2):40-6.
7. Bhatt D. A survey of effective and efficient software testing technique and analysis. Iconic Res Eng J. 2017;1(8):326-30.
8. Islam M, Khan F, Alam S, Hasan M. Artificial intelligence in software testing: A systematic review. In: TENCON 2023-2023 IEEE Region 10 Conference (TENCON); 2023 Oct 31-Nov 3; Chiang Mai, Thailand. IEEE; 2023. p. 524-9.
9. Patidar R, Sharma A, Dave R. Survey on manual and automation testing strategies and tools for software application. Int J Adv Res Comput Sci Softw Eng. 2017;7(4):424-31.
10. Raksawat C, Charoenporn P. Software testing system development based on ISO 29119. J Adv Inf Technol. 2021;12(2):128-34.
11. Rana I, Goswami P, Maheshwari H. A review of tools and techniques used in software testing. Int J Emerg Technol Innov Res. 2019;6(4):262-6.
12. Rani S, Gupta D. A comparative study of different software testing techniques: a review. J Adv Shell Program. 2018;5(1):1-8.
13. Sethi MA. A review paper on levels, types & techniques in software testing. Int J Adv Res Comput Sci. 2017;8(7):269-71.
14. Srivastava N, Kumar U, Singh P. Software and performance testing tools. J Inform Electr Electron Eng. 2021;2(1):1-12.
15. Sundaram A. Technology based overview on software testing trends, techniques, and challenges. Int J Eng Appl Sci Technol. 2021;6(1):94-8.
16. Taley DS, Pathak B. Comprehensive study of software testing techniques and strategies: a review. Int J Eng Res. 2020;9(8):817-22.
17. Umar MA. A study of software testing: categories, levels, techniques, and types. Authorea Preprints. 2023 Jul 11.

Received: *10-11-2025* - ***Accepted***: *09-01-2026* - ***Published***: *14-01-2026*