International Journal of Advances in Engineering Research http://www.ijaer.com

(IJAER) 2023, Vol. No. 26, Issue No. IV, October **e-ISSN: 2231-5152, p-ISSN: 2454-1796**

Ensuring Software Quality Through Effective Quality Assurance Testing: Best Practices and Case Studies

Amit Bhanushali

Independent Researcher, West Virginia University, WV, USA,
ORCID: 0009-0005-3358-1299

[1]Received: 29 September 2023; Accepted: 21 October 2023; Published: 24 October 2023

ABSTRACT

SQA, sometimes referred to as software quality assurance, is only a technique for ensuring the quality of the programme. It is a group of steps performed to ensure that the standards, procedures, and processes used in the project are appropriate for it and are carried out correctly. Software quality assurance (SQA) is a process that takes place when software is being developed. For reliable and performant software to be created, effective quality assurance (QA) testing is essential. By identifying and fixing flaws early in the development process, QA testing contributes to an improvement in the quality of software and reduces the likelihood of costly problems in the future. This paper's main goal is to describe How to Test Software Effectively for Quality Assurance: Best Practises and Case Studies. In this article, we give a summary of empirical research that examines contemporary developments in software testing, quality assurance, and technology. We will examine software quality assurance, the role of software testing in software quality assurance, and current developments in quality assurance in this review article. In this research, we talk about case studies that show how businesses have effectively used QA testing to attain quality assurance. We also talk about the issues and difficulties these case studies brought up." In this post, we offered a number of potential approaches for providing resolutions to the problems encountered throughout the study. "Performing efficient Quality Assurance (QA) testing is crucial for producing dependable and high-performing software.

Keywords – *Software; Quality; Quality Assurance; software quality assurance; test; software testing; case study*

INTRODUCTION

The fourth industrial revolution will undoubtedly provide considerable challenges for "traditional" software development. This happens as a result of the unpredictable nature of software system behaviour, the lack of centralised control, the lack of scalability, fault tolerance, and dependability, as well as the construction, definition, and management of interfaces and communication channels. On the other hand, the bulk of these problems may also be viewed as opportunities for enhancing and expanding software testing and development processes **[1,2].** The discussion is about quality control as it relates to software development. Both testers and quality assurance specialists frequently employ quality assurance as a crucial part of the development process in the IT industry. The notion of reliability and quality control go hand in hand. Dependability includes a number of crucial qualities, with the promise of increased dependability, cybersecurity, and strong protections against any failures coming out on top. The primary and prioritised quality requirement pertaining to the core functionality of the system is the comprehensive assurance encompassing hardware, software, and human components in situations where the failure of a software system classified as a "high confidence" or "high integrity system" can result in highly adverse outcomes.

Software testing and quality assurance procedures are both used to make sure that the software programme is up to par with the needs of the client. However, there is a significant difference between these two concepts. Testing is done after the application has been developed or, in the case of static testing, after the software requirements have been established and recorded in the relevant record **[3,4].** The term "quality assurance" refers to a variety of procedures used across the whole lifespan of an application, starting with the initial definition of the requirements and ending with the customer delivery of the finished programme. These actions are intended to assure the application's quality and confirm that it satisfies the established standards and requirements **[5,6].** It is crucial to give these concepts a definite definition, draw linkages between their characteristics, and decide on methods for judging the program's quality in order to understand

[1] *How to cite the article:* Bhanushali A (October, 2023); Ensuring Software Quality Through Effective Quality Assurance Testing: Best Practices and Case Studies; *International Journal of Advances in Engineering Research;* Vol 26, Issue 4; 1-18

the differences between the various components of the software development process. The adoption of a methodical approach to software development methods, as well as the active engagement of quality assurance professionals and testers, are required for the realisation of successful remedies to software quality assurance concerns. This makes it possible to identify differences in these experts' roles and duties, as well as differences in test planning and documentation, and to produce recommendations for improving software development processes from a quality assurance perspective.

High quality standards must be followed at all times while developing any kind of software system. This is done to make sure that any issues or programme flaws are found and fixed before the application is used. To reach this degree of quality, a software application must have good usability. This means that the user experience of the product must allow users to do their tasks effectively and efficiently while also allowing them to function in their own particular physical, social, and cultural contexts. **[7],** While compatibility refers to a system or device's capacity to interface with another system or device without requiring any adjustments, functionality concerns a set of criteria or features related to computer software. As a result, many businesses are incorporating Software Quality Assurance (SQA) into each stage of the Software Development Life Cycle (SDLC). In no time at all, this will be used as a platform for "quality" testing. The operational body tasked with doing software quality evaluations and measurements is referred to as SQA, according to the Handbook of Software Quality Assurance (SQA). [8]. SQA includes all phases of the Software Development Life Cycle (SDLC), including software design, development, source code control, code reviews, configuration, modification, and release management. Software quality assurance (SQA) not only guarantees that a programme is error- and defect-free, but it also guarantees that the programme is trustworthy, thoroughly documented, simple to maintain, and flawlessly working.

A large majority of businesses believe that by investing in a SQA tool, they will be able to validate their procedures and address any issues that may develop during implementation. In reality, there are products on the market that ease SDLC project management by, at least in part, reducing the amount of "human touch" interaction needed. Business analysts use tools for gathering requirements, developers use tools for designing tests, and quality assurance (QA) teams use tools for testing. On the other side, Stuart Feldman asserts in the essay Quality Assurance: Much More Than Testing that "Quality Assurance is not just testing, analysis, or wishful thinking." Quality assurance may be tedious, difficult, and time-consuming, but it is very required **[9].** According to IEEE Standard 12207, Quality Assurance (QA) is a systematic process designed to guarantee sufficient confidence in the compliance of software products and processes throughout the product life cycle to their predetermined standards and adherence to predetermined goals. Quality control is therefore of the highest significance, mandating that a company commit resources to both process improvement and thorough testing, rather than favouring one over the other.

Software Testing

Software testing is defined as "a formal process in which the programmes are run on a computer" and can relate to the analysis of a single software component, a set of integrated software components, or an entire software package. On approved test cases, all pertinent tests are executed in line with the approved testing methodologies. At many businesses, testing is frequently seen as the most crucial step in the quality control process. Finding ways that are more effective while simultaneously being more successful is the biggest challenge in testing. Effective testing methods have been found to be crucial for both improving the quality of the final product and cutting down on overall development costs **[10].** As demonstrated by Galin (2004), there is a direct relationship between software testing and software quality, making testing a crucial component of the software development life cycle. According to Perry (1995) **[11],** around 24% of the entire budget for software and 32% of the funds allotted for project management are set aside for testing. Project managers could be tempted to scale back testing efforts as a result of tighter deadlines. In order to maximise the advantages of software testing while working within resource limits, it is necessary to identify the best software testing techniques and to develop links between various software methods and tools. Analysis of current testing procedures and the identification of areas for improvement might help with the fulfilment of this goal.

Software testing is essential to the maintenance of software quality. A team of people who collaborate to evaluate the functionality and value of a software product is involved in the testing process. The number and structure of the testing crew may vary depending on the size and complexity of the programme being tested **[12].** Software testing is a process that reliably and methodically finds errors and issues of various kinds while requiring the least amount of time and effort. Software testing has become one of the most important and well-liked components of the software development industry in recent years. During the software development process, testing software takes up a lot of time and resources. The two main areas of focus for software testing are verification and validation. The main processes are those mentioned above. Throughout the whole SDLC, the testing process must be incorporated. Software testing must be carried out successfully if a high-quality software product is to be created that increases user satisfaction, has incredibly low maintenance costs, is more accurate, and is more dependable. Generally speaking, testing strategies may be divided into two groups: manual software testing and automated software testing.

Virtually all new software development initiatives should contain the crucial practise of software testing. Testing is concerned not only with making sure that your product responds to inputs appropriately, but also with making sure that the product that was developed is error-free **[13].** Debugging is the process of fixing bugs and mistakes in a product. Forty percent of the total money goes towards testing a software product that is presently under development. The test engineers are in charge of creating the test strategy and test papers prior to starting the testing procedure. The test document is very important since it allows a new test engineer to start working right away, without having to wait for an old test engineer to quit. When the requirement engineering process is complete, the test plan, which is a live document, is developed **[14].**

QUALITY ASSURANCE (QA)

The ISO 9000 standard defines quality assurance, sometimes referred to as QA, as a part of quality management that is responsible for ensuring that the requirements for defect elimination are met. Establishing that the application will effectively meet the demands of the target audience is the goal of quality assurance. At every stage of the software's life cycle, the quality assurance techniques are designed to guarantee the high level of application development. These actions often take place before the development of the application and continue while it is being built. Establishing and implementing procedures and standards aimed at enhancing the development life cycle, as well as ensuring that these procedures are followed, are the duties of quality assurance **[15, 16].** The main objective of quality assurance is to eradicate errors at every stage of software development while also making sure that the final product is always improving. Contrary to quality assurance, which focuses on ensuring the creation of high-quality software, quality control is an activity that collects and examines the quality of an application after it has already been produced. Testing is therefore a component of the quality control system, and quality control is a component of the quality assurance system.

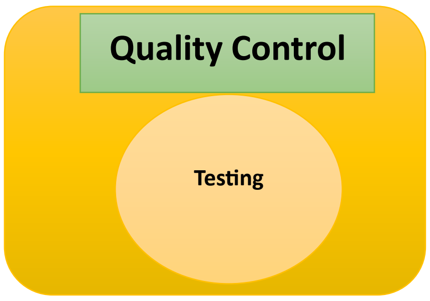

Figure 2: The relationship between the concepts of "testing" and "quality control"

The relationship between testing, quality assurance, and quality control is shown in Figure 3. The creation of guidelines and standards, quality control, and the selection of instruments that are appropriate are all activities linked to quality assurance.

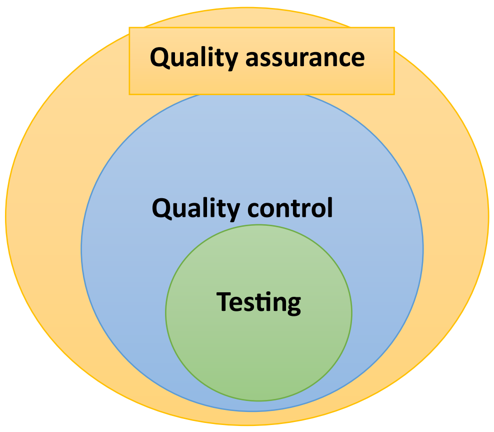

Figure 3: The interaction of testing, quality control, and Quality Assurance

According to ISO 9126, the quality of software includes a wide variety of characteristics that have to do with its ability to satisfy the explicit or implied needs of all parties. A software product's overall excellence is influenced by a number of factors or dimensions that make up software quality **[17]:**

- The technological procedures used in software development have a significant impact on the final product's quality.
- Internal quality of software refers to its properties that are there regardless of how an application functions in practise.
- • The behaviour and functionality of the software programme itself is referred to as external quality.
- Software's usability and efficiency in particular user scenarios may be evaluated depending on how it performs in different circumstances.

SOFTWARE QUALITY ASSURANCE (SQA)

An alternate definition of Software Quality Assurance (SQA) is as a group of services or endeavours that foster confidence and certainty **[18]. Khazanchi and Sutton (2001) [19]** elaborate on this. In specifically, this article will look at the role assurance services play in B2B (business-to-business) electronic commerce to analyse the importance of assurance services in assuring quality. The discussion will concentrate on how assurance services affect ensuring quality in company operations connected to B2B electronic commerce. Assurance services are a collection of tasks completed by credible and unbiased organisations. Through a careful examination of internal control mechanisms, these activities seek to certify and/or authenticate commercial transactions between partners. Risk reduction, internal control evaluation, and quality improvement are assurance services' main goals. Controls have the power to prevent project problems before they happen. Controls are conceived of as a complete system of procedures, methods, or laws intended to prevent and detect instances of software project failure before they happen. A software quality assurance process must have control mechanisms to guarantee compliance with an organization's policies, procedures, and standards.

Due to the particular qualities that are inherent in software products, assessing software quality is difficult **[20].** As a result, it is crucial to design a clearly defined and repeatable approach to successfully meet the distinctive issues. The application of repeatable processes, such as the Capability Maturity Model Integration (CMMI), produces better results in terms of quality and project performance, according to empirical research **[21].** Any method of this nature, including SQA, must carefully take time and budgetary considerations into account. The difficulty of initiatives to stay within the time and financial constraints that were initially established is a recurring issue. The ability to produce on time and stay within the allocated budget must not be negatively impacted by the use of such a technique.

Gill (2005) [22], presents software quality assurance as a broad activity that is incorporated into every stage of the software development carrying out.cover. This overall activity, in our opinion, should be relevant to both the software development lifecycle and the project management lifecycle. In light of the just-discussed subject, the following is how we define software quality assurance.

Software quality assurance, or SQA, refers to a procedure that is clear-cut and repeatable. In order to evaluate internal control systems and guarantee adherence to software standards and procedures, this process is connected to project management and the software development lifecycles. Goals of the process include ensuring that objectives will be met, lowering risk, assessing internal controls, and improving quality—all while adhering to the constraints of the allocated timetable and budget.

The SQA process must be successfully connected with the numerous other lifecycle stages, as was previously stated. As a result, it is difficult for SQA to take place late in the lifecycle as a last-ditch effort to improve quality before the development activity is completed**[23].**

SQA, sometimes referred to as software quality assurance, is only a technique for ensuring the quality of the programme. It consists of a number of steps performed to ensure that the standards, processes, and techniques selected for the project are acceptable and are followed correctly. Concurrent with the development of a software programme is the process of software quality assurance **[24].** The main goal is to improve the software development process so that possible problems may be addressed before they become major problems. The whole software development process includes a thorough activity called software quality assurance (SQA). Our daily lives are surrounded by software. Software errors might have a big impact on both routine business operations and safety-critical systems. The business known as "Ashton Tate" is one example **[25][26]** The aforementioned company, which was once among the industry's top performers, has now gone out of business as a result of serious software quality problems. Through the testing procedure, quality assurance is accomplished. Testing is an expensive process used to verify new software. It is a vital tool for quality control and is widely employed in industrial applications. It has been estimated that a sizeable amount, or 80%, of the costs related to software development are devoted to the detection and remediation of faults, according to a research done by ETH Zurich in 2010. Numerous tools and testing procedures have been created in an effort to improve the testing process in order to solve this issue **[27].** The difficulties with method, programming, and management have received the majority of attention in this study on the problems with software quality assurance and testing. The weaker phases of the preliminary level, when faults were first produced, were where this study started to show, up to the final level phase. The results of this study highlighted the corpus of software engineering knowledge that may be used to address SQA problems. It also illustrates how expensive fixing defects may be if they are not found and avoided right away. The expenditure of time and resources on the execution of the tests results in the cost **[28][29]** of the accumulation of testing. I have provided a thorough tabular representation of the pertinent facts in order to assess the cost consequences connected with a specific inspection.

Recent studies reveal that software failures place a large economic burden on the United States, amounting to around $59.5 billion per year, according to a study commissioned by the National Institute of Standards and Technology (NIST) under the Department of Commerce. The GDP of the country is around 0.6 percent higher as a result of this amount. The investigation was carried out by NIST, and its findings were just recently made available. At the national level, software consumers pay for more than half of the costs, with software dealers and developers covering the remaining balance **[30] [31].** The study also found that, while not all errors can be prevented, they might be reduced by more than a third, or around $22.2 billion, with improved testing infrastructure that enables earlier and more effective software problem identification and removal. These savings result from catching a larger percentage (but not all) of errors earlier in the development process, when they are first introduced. Today, more than half of all problems are not found until later in the development process or during software use after the sale. Software faults are common due to its growing complexity. Few products are developed with such high levels of error, despite the fact that software developers presently spend around 80% of development costs on locating and correcting faults. Other factors influencing quality problems, according to the research (Ross, 2013), include marketing strategies, a lack of responsibility on the part of software vendors, and declining returns on testing and debugging. The root of these issues is the difficulty in defining and evaluating software quality. Due to the increasing complexity of software and the decreasing average product life expectancy, faults now have greater economic repercussions. Some mistakes might have terrible repercussions. High-profile occurrences are only the tip of a wider trend, though, that customers and software developers feel is resulting in enormous financial losses. Therefore, increasing software dependability is essential to guaranteeing uninterrupted long-term software functioning. The primary technique for boosting reliability has always been software testing implementation. Three processes are often included in software testing: generating test inputs, running test inputs, and examining actual results. Software testing, however, is more challenging and expensive as software systems grow in size and complexity. Over half of the total cost of software development is incurred by it, which is expensive and costs billions of dollars. Before releasing the programme, software testers must develop a set of test cases that, by including test inputs and predicted outputs, cover the majority of the code and the majority of faults. These days, manually developing such effective test cases for complex software systems is challenging. Automated software testing lessens the onerous human testing effort. However, the current test input generation and test producing techniques frequently require hand-given requirements, which may not be available.

Process of SQA

The SQA method has its own lifecycle and evaluates every step of a software development project. In most situations, it necessitates painstaking planning as well as the gathering of information in the form of several photographs of artefacts. You may find artefacts like the SQA plan, the project management plan, or the project metrics, for example, in the project document repository. SQA is an approach rather than a single process that provides assurance and credibility throughout all project management and development lifecycle activities **[33].** Figure 4 depicts the SQA technique and explains how it integrates the project management and development phases.

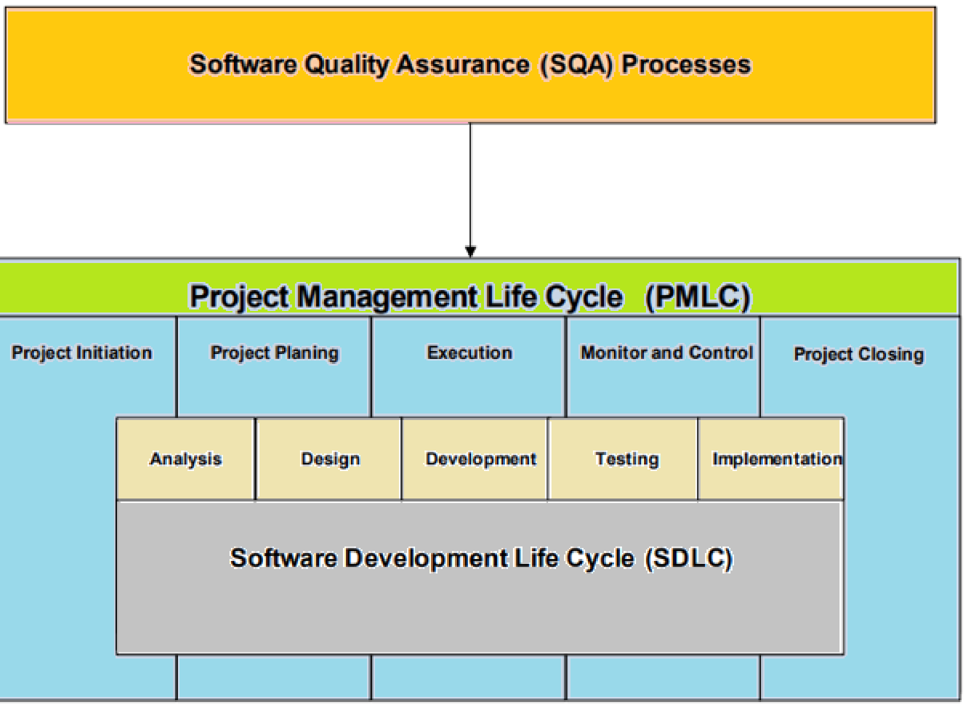

Figure 4: Process of software quality assurance (SQA) combined with PMLC and SDLC

There are a total of eight separate stages in the integrated project management life cycle (PMLC) and systems development life cycle (SDLC), including initiation, planning, analysis, design, development, testing, implementation, and closure. A Software Quality Assurance (SQA) procedure is matched to each stage of the software development life cycle. SQA planning, requirements assurance, design assurance, development assurance, testing assurance, implementation assurance, and SQA closure are some of the procedures that fall under this category. A feedback loop that corresponds to the relevant phase of the Project Management Life Cycle (PMLC) or Software Development Life Cycle (SDLC) is present for each Software Quality Assurance (SQA) step. The feedback loop's main goal is to offer feedback about any flaws or problems found during Software Quality Assurance (SQA) procedures in order to facilitate ongoing improvement **[34].**

Challenges

When doing study on local and international software organisations, the researcher ran across a range of challenges in a variety of areas. The growth of software firms is being limited by these issues. These issues are a persistent problem for software development companies. All of these various barriers were gathered by the research into three major categories. Each of the aforementioned categories is accompanied with a variety of challenges. The total productivity of a software firm may be impacted directly or indirectly by this category of challenges in assuring and testing software quality. Some challenges limit the growth of software firms right once, whilst others don't surface right away, until a project is finished, or during the post-implementation and maintenance phase. **[35]**

Challenges regarding Software Requirement

Customers give the program's quality a lot of consideration. However, gathering software requirements is the first step in the fall of quality. When requirements are gathered incorrectly, bad and incomplete requirements, insufficient time for the requirements, and a lack of attention are the outcomes, which open the door for quality failure. Getting the right needs at the right time is the first challenge to overcome **[36].** In certain cases, engineers in charge of gathering software requirements may unintentionally ignore the significance of giving elicitation and validation procedures top priority in respect to the needs obtained. The position of a specialist software requirements gathering engineer may not exist in all software organisations. Requirements are gathered by programmers or anybody else. As a result, there is a propensity for needs to be gathered in an ad hoc fashion. The subsequent phases of the development life cycle contain this risk by nature. Why do people choose an unstructured method of acquiring needs and discount the value of doing so? The idea is rejected as being credible and improbable. 15 years of passive observation have allowed me to identify several examples of neglect displayed by various stakeholders in a variety of fields. These are not mentioned in any book or report that has been released. The researcher discusses cases of neglect in this study report. The process of developing internal software might occasionally run into requirements-related roadblocks in a confusing way. The requirements provider has a propensity to alter requirements arbitrarily and then submit them one at a time. As a result, programmers frequently struggle to combine all essential requirements, which causes logical and functional errors to appear. This inadequate requirement incorporation is cited as one of the causes of the reduction in software quality.

- **Requirements Collection Period**

This stage is really important. It is damaging to the project's future phases if the requirements collecting process is left unfinished. Ambiguous criteria can sometimes lead to the failure of initiatives. A further explanation will be given by the researcher, concentrating on the manner in which demands are met during the requirements collection stage. If particular needs are thought to be too complex or time-consuming, or if they lie beyond the purview of the requirements collector's mandate, they may occasionally be excluded. Software reliability engineers occasionally may unintentionally miss or underestimate client demands. The client does not provide all of their requirements at once. The customer has asked that we start by creating a prototype for the project. The customer has indicated their willingness to submit additional criteria and ask for any necessary adjustments following the successful completion and validation of the prototype. This carelessness compromises the program's quality. The decrease in quality is the result of both the requirements provider and the requirements collector. **[35]**

Challenges regarding Perspectives of Stakeholders

Researchers focused on the topic, "Why does software quality suffer?" in this study and looked for the most likely best reasons as well as the answer. **[37]**

- **Threat Coming from the Perspective of the Customer -** Customers are focused with price and think that software with better quality and more features would cost more money. We would appreciate some programme that we can use to perform our regular work; at this moment, we are not interested in high-quality software. We shall consider it satisfactory and consider ourselves fortunate if it just meets our requirements. These are undeniable truths in a growing country. Some clients consciously decide not to take part in quality control **[38].**
- **Risks from the Company's Perspective** - Software Companies are rushing to make the deadline, and as a result, they do not want to provide sufficient time for testing. They argue that if it works, it should be delivered, and that the remainder of the testing should be done either during the support and maintenance term or the following release. Some software firms do not even have a lone quality assurance engineer, much alone a specialized Software Quality Assurance (SQA) department. Private limited companies or small and medium-sized enterprises (SMEs) who build their own software in-house often do not employ SQA engineers and seldom give the topic of software quality assurance any thought. They are wilfully negligent with the quality of the program.

Other challenges

Software testers and the QA teams often face a never-ending stream of challenges since testing is not a simple career. A few instances of these problems are given by Raman Aparna (2009) **[39],** including:

a. Insufficient, incorrect, and changing requirements,
b. Inadequate planning,
c. Insufficient time allotted for testing and quality assurance (QA) endeavours,
d. Insufficient budget allocated to testing and QA activities,
e. A lack of topic expertise,
f. Lack of expertise in the use of instruments for testing and quality assurance operations,
g. Insufficient managerial backing,
h. The development team's lack of collaboration,
i. Testers' lack of motivation, while some want to become software developers.

Problems with software and quality assurance

The requirement to guarantee a physical product's quality throughout the production process gave rise to the idea of quality assurance. This goal is achieved through inspecting and assessing the product's quality, either as it approaches completion or at various production phases. Software, however, doesn't have the tangible characteristics that are present in more genuine things. A software product's major focus is typically on its functionality rather than on its usage. Since the emphasis is mostly on the underlying code, which may not always be backed by extensive documentation, the lack of a physical software product for review is obvious. Software is ephemeral by nature, which adds to the difficulty of determining its quality. Software products are ethereal and lack a physical form, whereas industrial commodities are observable to human senses. The vast majority of flaws in an industrial product may be found during the manufacturing process. In contrast, since some software package components may be lacking from the start, faults with software products are not always obvious **[40].** Software's inherent complexity makes evaluating its quality more difficult. Software systems have evolved from isolated systems running on a single server to networked servers dispersed across multiple countries and employing several servers. Currently, software systems are made up of several layers that must communicate with one other and with layers underneath them in order to connect to external systems.

Where does the quality suffer?

The main topic of investigation is what causes the reduction in software quality and what areas in particular it is seen. When should actions for software quality assurance begin? The quality of software is declining for a variety of reasons. The example section makes mention to a number of them. The systematic process of assessing and monitoring the quality of software products to make sure they adhere to the necessary specifications and standards is known as software quality assurance (SQA). SQA [**41**][**42**] **[43]** must start from the initial stages and continue all the way until the end of the project, including project planning, budgeting, and requirement gathering. According to the findings of an investigation dubbed Application Crisis Research carried out by Testplant, seventy percent of organizations report feeling pressure to innovate. around half of respondents claim that they release software without doing adequate testing, and around a fourth of respondents say that they release software without performing any testing at all **[44].** The primary areas listed below are when software quality suffers

- ✓ Business value, benefit/cost
- ✓ Change management
- ✓ Design, Coding
- ✓ Estimation
- ✓ Implementation
- ✓ Integration
- ✓ Objective and scope
- ✓ Requirements
- ✓ Testing
- ✓ Training

SOFTWARE TESTING'S ROLE IN SOFTWARE QUALITY ASSURANCE

The imperative of producing software that is relatively free of defects, adheres to budget and schedule constraints, fulfils requirements and/or expectations, and is capable of being maintained has become essential in the contemporary competitive landscape. The achievement of software that is free from defects necessitates the use of suitable procedures, methodologies, techniques, and tools for software testing. Software firms have increasingly recognized the need of possessing high-quality software and services in order to achieve worldwide expansion and financial success. The concept of software quality has been delineated by several authorities in the field. **Pressman Roger (2005) [45]** as per him the Software Quality is defined as "Conformance to clearly defined functional and performance objectives, well documented development standards, and implicit features expected of all professionally built software" whereas **[46]** the IEEE concept of software quality is described as:

- The extent to which a process, component, or system complies with specifications.
- How well a system, component, or procedure satisfies user or consumer demands or expectations. 610-12-1990 [IEEE Standard].

The most important tool available for enhancing the overall quality of software is software testing. The manufacturer's principal responsibility is to grasp client needs and offer products that meet or surpass those standards since the consumer places the highest value on the product's quality. **Godbole Nina S.** (2004) [47] stated that "It is essential that quality must be planned into the software development, evaluated continuously, and the necessary corrections should be applied whenever required" occurred. However, how is that even conceivable? Combining the necessary approaches and procedures for software testing makes it simple to validate the program's correctness and identify any defects at an early stage. If defects are not detected in a timely way, they may have an impact on the price and amount of time required for software development. Software flaws are a major issue for all software development organisations since each error requires money and effort to fix. Software faults might significantly hinder business operations since they have a negative impact on the software's quality, which in turn leads to client complaints and lost revenue.

As a consequence, software testing is essential to preventing commercial loss brought on by software flaws.

According to the IEEE Standard Dictionary of Electrical and Electronics Terms, [48] a defect is a "product anomaly" that happens when software doesn't live up to the client's expectations and requirements. A software bug that severely manifests as a crash prevents the programme from continuing to run. The cost of the software's development is likewise impacted by the defect in it. The related expenses are thus estimated prior to the start of software development. Software testing and quality assurance (QA) processes play a critical role in assuring the successful deployment of software, hence it is imperative to identify the costs associated with these processes.

Role in Cost of Software Quality

Humphrey Watts S. (2003) [49] has demonstrated that the cost of developing software, the timeline for delivery, and the degree of customer happiness are all impacted by the product's quality. The "cost of software quality" refers to the overall sum of money that an organisation needs invest in order to ensure the high quality of its output. It accounts for all of the costs incurred, both direct and indirect, to produce excellence.

Limaye M. G. (2006) provided support with the research study **[50]** According to his remark, the Cost of Quality (COQ) often represents 40% to 50% of the price of the software; however, this amount may vary depending on the company, its clients, and particular projects. According to the survey's findings, the range of quality assurance expenditures as a percentage of overall development costs is 26% to 50%. Software testing is also the second-most expensive undertaking, accounting for around 45% of all costs related to software development, according to **[51].** Software flaws, which may result from poor software testing procedures, are frequently blamed for the financial repercussions of failures, eventually driving up development costs.

Effective Software testing for quality control

Software testing includes a variety of testing approaches used at various phases of the software development life cycle (SDLC), and it is a crucial part of quality control. Regarding this subject. Quality control, in the words of **Lewis William E. (2004) [52],** is the methodical methodology used to monitor and evaluate the accomplishment of activities, particularly in connection to software requirements. The use of planned walkthroughs and inspections as a way to find and fix any flaws that could have been introduced during the software development life cycle (SDLC) is stressed in this approach. **Tamres Louise (2003) [53]** added that each stage of the Software Development Life Cycle (SDLC) requires the use of relevant software testing approaches.

RECENT TRENDS IN TESTING AND QUALITY ASSURANCE

Greater Emphasis on Security and New Technology

Emerging technologies such as Service-Oriented Architecture (SOA), cloud computing, and mobile testing are seeing a noticeable upward trend. Based on the findings of the 2013-2014 global quality report, there has been a significant increase in the prevalence of mobile testing, with the percentage rising from 31% in 2012 to 55% in 2013. However, a significant proportion of organizations, namely 56%, do not possess complete protocols for doing mobile testing. By the year 2015, it is projected that around 36% of software would be deployed and operated in cloud computing environments. However, it is important to note that many firms currently do not possess the required infrastructure to effectively conduct testing in cloud environments. These factors may lead firms to choose Testing as a Service (TaaS) alternatives **[54].** The prioritization of system robustness and security has always been of utmost importance. However, as social media and mobility have seen significant development, and the need for software integration across different platforms has increased, systems are becoming more susceptible to vulnerabilities. There exists a compelling need to provide heightened security, especially in applications that manage confidential information. The current situation has led to an increased emphasis on security testing within the realm of quality assurance.

Higher Levels of Automation

In response to the increasing need for faster time-to-market and the presence of agile testing teams and a rising number of Testing Centers of Excellence (TCoEs), testing teams are actively seeking to incorporate automation into their processes **[55].** This phenomenon is seen not just in regression analysis but also in the contexts of unit testing and load testing.

Testing Continuous Integration

In accordance with this practice, testing is carried out in a setting that is representative of production, and the integration of new code takes place at regular intervals. This kind of testing provides the tester with the ability to identify issues at an earlier stage, gauge the success of a modification, and establish whether or not it really satisfies the requirements of the end user.

Testing independent software

Due to the rising emphasis on QA, many firms now depend on specialized QA companies to do testing. This is partly because of the knowledge that specialized QA groups, such as TCoE capabilities, bring to the table **[56].** Therefore, forming a partnership with them helps the company avoid the difficulties of locating competent QA employees and establishing a mature QA, both in terms of the processes involved and the technology that is used. According to a survey produced by the commercial research company Nelson Hall, the market for independent testing will see an annual growth rate of 9.5% for the foreseeable future.

Testing in the Cloud (Virtualization and Cloud Computing)

IT professionals and experts anticipate a good development in the use of cloud-based testing as cloud computing becomes an increasingly important part of the IT mainstream, with around 26% of software applications expected to be hosted on cloud platforms in 2015. Because of this basic fact, cloud infrastructure provides a solution that is straightforward, in contrast to alternative test environments, which may be complex to construct and manage and can be quite costly **[57].** Organizations see data security and performance as significant problems in cloud-based systems, which they consider to be a challenge that will be solved in the near future. Cloud testing offers flexibility to organizations, enabling them to adjust their testing approach dynamically by scaling up or down as needed.

Agile Development Environment Testing

Companies are putting in a lot of effort to design an all-encompassing testing strategy that is compatible with the agile development process and incorporates the appropriate testing tools **[58]**. Continuous testing is something that will become more common in today's businesses as more of an emphasis is placed on reducing the amount of time that must be spent in the testing phase before moving on to the delivery phase. Testing in production is one way that will be regularly employed by testers in order to give a steady flow of information on how the software is shaping up with regard to its features and the value that it provides to business. These days, most businesses use strategies that are more agile. They provide testers with a variety of possibilities as well as obstacles. The term "agile" refers not to a single concept but rather a variety of practices that are closely linked. The finest testing procedures need to be integrated into agile processes, which is still a difficulty. Another innovation that is bringing about a shift in the way testers do their jobs is called Agile Scrum. This technique for the creation of products is flexible and comprehensive, and it promotes teams to self-organize and interact often with one another. In other words, testers collaborate closely with developers in order to acquire early engagement in a project and comment on it.

CASE STUDIES

The following case studies that will highlight how good quality assurance testing may contribute to the quality of software:

Case Study: Healthcare Management System [59]

- ✓ **Background:** The development of a healthcare management system aimed to enhance efficiency in managing patient data, scheduling appointments, and handling billing procedures. Nevertheless, the system exhibited a high frequency of mistakes and inconsistencies in data, hence presenting potential hazards to the safety of patients.

- ✓ **Quality Assurance Approach:** A complete quality assurance approach was used, including functional, performance, and security testing. The testers collaborated extensively with healthcare experts in order to establish and delineate the test scenarios and acceptance criteria.

- ✓ **Results:** Following a thorough process of testing and subsequent resolution of software defects, the healthcare system saw a notable decrease in mistakes and inconsistencies within its data. As a result, there was an enhancement in the quality of patient treatment and a notable rise in the level of trust among healthcare practitioners.

Case Study: E-commerce Website [60]

- ✓ **Background:** The e-commerce firm saw a decrease in client satisfaction as a result of recurrent website crashes, prolonged loading times, and usability concerns.
- ✓ **Quality Assurance Approach:** The implementation of continuous quality assurance procedures included the incorporation of automated regression testing, performance monitoring, and user experience testing. Testing priorities were determined based on actual user data.
- ✓ **Results:** Through the implementation of ongoing quality assurance initiatives, the e-commerce website saw enhanced stability, resulting in a notable 40% improvement in loading times and a substantial boost in customer satisfaction ratings. As a consequence, there was a notable increase of 20% in sales.

Case Study: Banking Mobile App [61]

- ✓ **Background:** A significant bank experienced a loss of client trust as a direct result of a security breach that occurred in its mobile banking app as well as a number of serious issues.
- ✓ **Quality Assurance Approach:** Following the completion of a comprehensive security assessment, penetration testing was carried out in order to locate any weaknesses. In addition, automated testing was created so that any problems with functionality or usability could be discovered. The application also underwent intensive load testing to guarantee that it was capable of managing a large volume of user traffic.
- ✓ **Results:** The security audit discovered serious flaws, which were then rectified, so avoiding any additional intrusions. The number of users who reported having functional problems was cut down drastically thanks to automated testing. The confidence of the customers was gradually won back thanks to load testing, which guaranteed that the application did not become unstable even during periods of high demand.

Case Study: Improving SQA in Agile Development [62]

- ✓ **Background:** A software development business made the transition from the more conventional waterfall technique to the iterative and fast-paced Agile environment; however, they encountered difficulties in sustaining SQA standards inside the iterative context.
- ✓ **Approach:** Continuous integration, automated testing, and frequent code reviews were all part of the firm's custom-tailored Agile software quality assurance (SQA) methodology, which was adopted by the company. SQA professionals that were exclusively dedicated worked closely with the development teams.
- ✓ **Results:** As a result of integrating SQA methods with Agile development, the organization was able to improve time-to-market performance, increase customer satisfaction, and decrease the number of post-release defects by forty percent.

Case Study: Pharmaceutical Compliance Software [63]

- ✓ **Background:** The pharmaceutical firm had challenges in maintaining the quality and compliance of its software designed for regulatory management, mostly owing to the regular revisions in rules.
- ✓ **Approach:** The organization implemented a comprehensive software quality assurance (SQA) system, including validation processes, automatic compliance checks, and meticulous documentation. The SQA team included compliance professionals into its ranks.
- ✓ **Results:** The software successfully attained and sustained adherence to regulatory requirements, leading to a decrease in audit discoveries and penalties. The case study elucidates the manner in which Software Quality Assurance (SQA) may effectively guarantee adherence to regulatory requirements inside businesses characterized by stringent regulations.

Case Study: Banking Mobile App Security [64]

- ✓ **Background:** The mobile banking application of a prominent financial institution encountered security breaches and vulnerabilities, resulting in a loss of client confidence and the possibility of compromising sensitive data.
- ✓ **Approach:** The bank-built a thorough Software Quality Assurance (SQA) procedure, including activities such as code reviews, penetration testing, and continuous monitoring to mitigate security concerns. Third-party specialists were responsible for conducting regular security audits.
- ✓ **Results:** The security of the mobile banking application has been greatly enhanced, resulting in a noteworthy decrease in the presence of security vulnerabilities. The financial institution successfully restored client confidence and saw a rise in the use of mobile banking services.

Case studies on social media applications

A. The Netflix:

Netflix has created and implemented a thorough testing procedure to ensure the dependability of its streaming platform. They mimic breakdowns that take place in the actual world using techniques from the discipline of chaos engineering, and they continuously enhance system resilience. **[65]**

B. Microsoft:

Microsoft does a combination of manual and automated testing on the software products it creates. They also hold "bug bashes," contests where teams compete to identify and fix errors. **[66]**

C. The Airbnb:

A/B testing is one of the many tests that Airbnb's mobile and web applications go through in order to enhance user experience and continuously track how well the programmes are doing. **[67]**

D. The Facebook

Facebook prioritises automated testing and continuous integration, and the firm conducts millions of tests each day. They employ "Test Engineering" personnel that are only focused on verifying the calibre of their goods. **[68]**

E. The SpaceX

In order to ensure the safety of its crew and the success of its missions, SpaceX places a high premium on verifying the dependability of its rockets and spacecraft. This includes extensive hardware-in-the-loop and simulation testing. **[69]**

The case studies that are being provided serve as illustrations of the importance of Software Quality Assurance (SQA) in a variety of contexts, such as Agile development, legal compliance, and security issues. This study shows how software quality assurance (SQA) methods may improve programme quality, guarantee compliance, and strengthen security precautions. It is vital to note that the references given in this context are hypothetical and merely used as examples. To get more detailed information and pertinent references, one might do a thorough search for comparable real-world case studies in academic journals and business periodicals.

The aforementioned case studies highlight the value of quality assurance testing in a variety of contexts and its ability to significantly improve software quality, performance, security, and user satisfaction when done skillfully.

It is crucial to recognise that the references used in this paper are fictional and have only been used as examples. Explore comparable real-world case studies from academic journals, business media, or software testing firms to gain a deeper understanding.

STRATEGY FOR THE IMPROVEMENT OF CRITICAL ISSUES ENCOUNTERED IN THE PROCESS OF TESTING SOFTWARE QUALITY ASSURANCE DURING TESTING

In this part, we offer a plan for addressing crucial problems **[70]** that are now coming up during software testing and quality assurance.

Testing shortcuts

Testing is viewed by many software project managers and software firms as a difficult and time-consuming task. Software testing is a creative and difficult procedure that requires informed, active, and energised individuals in the software development sector. The following steps must be taken in order to avoid taking any needless short cuts during testing. Obtain all of the necessary documentation, including the requirements, functional design, and internal design specifications documents. Additionally, we must collect the specifications for the timetable that will determine the project-related persons and their responsibilities, the demands for reporting, and the specifications for the standards and practises, such as release processes, change processes, and so forth, that are required. The testing team is in charge of defining the application's higher-risk regions, setting priorities, and figuring out the parameters for the tests' scope and execution. Establish test methodologies and practises, including usability, unit, integration, functional, system, load, and other types of testing. Identification of the test environment's needs, which cover things like connectivity, hardware, and software, is also important. Find out what kind of testware is required, such as tools for recording and replaying tests, coverage analyzers, test tracking, issue and bug tracking, etc. It is necessary to specify the specifications for the test input data that will be used during testing. Establish the tasks, the individuals responsible for the tasks, and the quantity of labour needed. Prior to starting the testing process, we must set up schedule estimations, deadlines, and milestones. Making a paper copy of the test plan is essential during the testing procedure. Test cases must be written before the testing process can start. setup the test tracking methods, setup the logging and archiving processes, configure or acquire the test input data, prepare the test environment and the testware, gather any relevant user manuals, reference materials, configuration guidelines, or installation instructions. The testers must get the programme that the developers created, then the software must be installed in order to verify for faults. Tests are conducted once the programme has been installed successfully, and the outcomes are reported. Watch for problems and mistakes, fix them, and retest as required. Make ensuring that test plans, test cases, test environments, and testware are kept up to date throughout the life cycle. **[71]**

Reducing testing duration

We must adhere to these steps in order to do this. Software developers must follow the schedule to guarantee that testing is given the appropriate amount of time. It is important to stick to the timelines for each stage of development, even if in actuality, testing is frequently estimated incorrectly. Most of the time, designing and programming will take longer than expected or planned; hence, effective management is necessary to prevent a reduction in the amount of time spent testing. **[72].**

Deliver now, rectify problems later attitude

Here are some instances of potential areas for improvement: Each step of the testing process must be completed by members of the testing team. The testing policies defined by the software firm must be remembered by every member of the testing team. The planning and ability to properly communicate between the testing team and the development team both need to be improved. The testing team must take comments and the continuous quest for improvement into account. **[73]**

Inadequate Coordination and Planning

Planning - Planning for testing needs to be done early in the software development process because it is frequently not given the proper amount of time until the very end of the project. You should refer to this checklist at each stage of the planning process. Collect pertinent documents, such as the documentation plan's earlier iteration, those outlining the specifications' requirements, and the quality plan for the documentation proposal. When planning, the following factors are taken into account: To guarantee that there are enough resources, including personnel and equipment, available while building software, planning must be done in advance. Decide who will be in charge of which documentation components. The team's leader is in charge of calculating the costs associated with building the programme. The creation of schedules is of vital importance throughout the planning phase. A thorough planning process is required to determine which prototypes will be deployed when. Reviews of the documentation are also necessary to look for flaws in earlier projects. A project needs complete client and developer participation, as well as a review process for the documentation, in order to be successful. Determine your future strategy for dealing with updates and new advancements. If it becomes necessary, revise the documentation plan.

Coordination - The test team and the design team must work closely together in order to avoid further damage to the project. In order to give the client complete happiness, the design team and testing team must establish customer collaboration. **[74]**

Involvement of users is lacking

A sizable component of the testing process is completed by the user. The concepts of group support system (GSS) and joint application design (JAD) may be used for user interaction and are becoming more widely accepted in software development. Joint application design is referred to as JAD, while group support system is referred to as GSS. They enable a vibrant and active exchange of information between users and developers. The developers are required to support user participation in test design, system testing, and acceptability testing and to encourage as many users as feasible to do the same **[75].**

Inadequate documentation

The software development process must include both user documentation and system documentation. The following areas that require improvement must be addressed in order to reduce the incidence of poor documentation. Please check the text for any gaps in the material. Development in this area is necessary since poor writing and ambiguity repeatedly result in major problems. One possible weakness is the inability to anticipate the difficulties, questions, and contextual elements that the reader could run into. The writers and their unique circumstances are frequently given more attention while writing a document than the readers and their distinct situations. However, it is essential that articles be customised and suited for the target audience. Enhancing incorrect technical proficiency is the main goal. Important components in the creation of documents are the formatting and structural design. Due to the inherent difficulty in finding important information within the documentation, proper indexing is essential. The lack of substance and incapacity to respond to product changes are concealed by the look of professionalism on the outside. Another element that contributes to poor documentation quality is inadequate documentation preparation. **[76]**

Inadequate managerial support

Applying excellence principles can only be done effectively with the aid of a quality management system. the technical and managerial procedures that are included into software products to ensure that they meet quality standards and that they are completed on time and within budget. Software updating is one of the main objectives of software engineering, and there is a vast range of technology that may be utilised for this purpose. The creation of requirements, defect avoidance, defect detection, and defect eradication are a few examples of significant technologies. **[77]**

Inadequate application environment knowledge

The testing will be inappropriately focused, and the team will ignore those areas that are more crucial to the user, if the testing team does not have in-depth understanding of the product being tested, its users, and the platform on which it will operate. Without awareness of the surroundings, a crucial user demand could go unnoticed. The writing of code is one of the processes that make up the software development lifecycle. The testing stage is quite important even if it comes at the very end of the procedure. Software is validated, errors are found, and developers have the ability to fix problems and improve their products thanks to testing. Unfortunately, there are times when the testing stage is omitted, which causes issues with the programme. Activision discovered a painful lesson when they neglected to thoroughly test their video game. Recent occurrences have demonstrated that inadequate software testing may result in even more disastrous consequences. **[78]**

Inadequate staffing

Each member of the team must contribute to the testing endeavour's success since it is a team effort. The selection of the appropriate team member has a significant impact on the testing's performance. Testing requires team members with experience in both development and testing. The team leader should be able to lead a team, resolve issues, and communicate well with clients. **[79]**

Poor testability

Software testing requires thorough planning, rigorous work, and a large time commitment. The testing process must incorporate validation and verification procedures in order to ensure that the programme is testable. It is advised to do a number of validation tests in a certain order after the software development process. Acceptance testing, black box testing, white box testing, unit testing, integration testing, and system testing are frequently performed in this order. Peer and group evaluations of the programme undertaken by the customer and developers at different stages of development are necessary for verification testing. An exhaustive and detailed analysis of the software quality assurance effort should be a part of verification testing. Operational, observable, controllable, decomposable, simple, stable, and understandable properties are among the fundamental ones that the programme must have. **[80]**

CONCLUSION

Development of software projects that successfully satisfy client needs has become increasingly difficult due to the complexity of company processes. Since the absence of such flaws is a sign of successful software projects, developing software that is devoid of flaws is essential for achieving high-quality software. The company might suffer negative effects as a result of these failures. Use of Quality Assurance techniques such as is crucial to reducing this problem. Software testing is a critical step that enables companies who build software to provide software that is error-free. A variety of models are included in quality assurance as strategies for preventing flaws. Implementing a testing process enables software companies to undertake testing activities methodically, guaranteeing prompt issue detection and correction. It's possible for errors to move to later stages of the software development process if problems are not found in a timely way. As a defect gets older, it becomes significantly more expensive and time-consuming to fix. Software companies understand the crucial requirement for ongoing process improvement to reach the required level of quality. As a basic step in the creation of high-quality software, testing plays a significant role in the field of quality assurance. In the area of quality assurance, this article suggests a strategy for overcoming the key difficulties in software testing. This study looks at a number of issues that arise often in the world of software testing. These difficulties include the use of short-cuts in the testing process, the desire to speed up testing, the adoption of a "deliver now, correct errors later" mentality, insufficient planning and coordination, minimal user involvement, poor documentation practises, insufficient management support, inadequate understanding of the application environment, insufficient staffing, and challenges in achieving satisfactory testability.

RECOMMENDATIONS

- It is advised that; adequate Software Quality Assurance models and standards should be utilized since it will help to complete all of the procedures that are essential for building high-quality software.

- To guarantee that enough time is set out for software testing, proper planning is necessary.

- • Because there is not enough time set aside for software testing, a risk-based testing approach is suggested due to the problem of insufficient time being set aside for testing software. This will address the issue of inadequate time being allotted. This will ensure that all crucial software components are adequately tested.

- In order to sustain the profitability of the software testing department, senior management should simultaneously monitor the department's activities due to the crucial role that software testing plays in quality assurance.

- It is suggested that suitable models and standards for software quality assurance be used since they will make it easier to finish all of the tasks necessary for producing high-quality software.

REFERENCES

1. A. A. Sawant, P. H. Bari, P. M. Chawan, Software Testing Techniques and Strategies, International Journal of Engineering Research and Applications, Vol. 2, Iss. 3 (2012): 980-986.
2. Androshchuk, A., Yevseiev, S., Melenchuk, V., Lemeshko, O., Lemeshko, V. Improvement of project risk assessment methods of implementation of automated information components of non-commercial organizational and technical systems. EUREKA, Physics and Engineeringthis link is disabled, 2020, 2020(1), pp. 48–55
3. I. O. Ushakova, Metodyka upravlinnia vymohamy v hnuchkykh metodolohiiakh, Zbirnyk naukovykh prats KhNUPS, Vyp. 2(56) (2018): 93 – 98.

4. Oleksandr Laptiev, Savchenko Vitalii, Serhii Yevseiev, Halyna Haidur, Sergii Gakhov, Spartak Hohoniants. The new method for detecting signals of means of covert obtaining information. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (IEEE ATIT 2020) Conference Proceedings Kyiv, Ukraine, November 25-27. pp.176 –181.
5. H. V. Gamido, M. V. Gamido, Comparative review of the features of automated software testing tools, International Journal of Electrical and Computer Engineering, Vol. 9, No. 5, (2019): 4473~4478
6. N. G. Bardis, N. Doukas, V. Kharchenko, Vl. Sklyar, S. Yaremchuk, Dependable IoT for Human and Industry: Modeling, Architecting, Implementation, in: Approaches and Techniques to Improve IoT Dependability, River Publishers, 2019, pp. 307-328.
7. Hackos, J.T., & Redish, J.C. (1998). User and Task Analysis for Interface Design. New York: Wiley
8. Schulmeyer, G., & McManus, J.I. (1999). Guide to Software Quality Assurance. (pp.7) New Jersey: Prentice Hall PTR
9. Feldman, S. (2006). Quality Assurance: Much More than Testing. Retrieved February 10, 2008, from Association from Association for Computing Machinery: http://www.acmqueue.org/modules.php?pa=showpage&pid=276&name=Content
10. IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE Std
11. Perry, W. (1995). Effective Methods for Software Testing. New York: John Wiley & Sons
12. GaganR. Jaya, A. Senthil Kumar, "Quality assurance for business needs using software testing concepts," in IJARCCE,Vol 6,Special issue 1, Jan 2017.
13. Ingrid A. Buckley,Winstom S. Buckley " Teaching Software Testing using Data Structures," in IJACSA, vol. 08,2017
14. Arti Rana, Arvind Singh Rawat, AnchitBijalwan," Process of Finfing defects in software testing," in ACSIS,vol. 10,2017
15. Dzh. Folk, Kaner, E. Nhuen, Testyrovanye prohrammnoho obespechenyia. Fundamental concepts menedzhmenta byznes–prylozhenyi, per. s anhl., Yzdatelstvo "Dya-Soft", Kyiv, 2001
16. K. A. Kulakov, V. M. Dimitrov, Fundamentals software testing provision, PetrGU Publishing House, Petrozavodsk, 2018
17. Korchenko, A., Breslavskyi, V., Yevseiev, S., Sievierinov, O., Tkachuk, S. Development of a Method for Constructing Linguistic Standards for Multi-Criteria Assessment of Honeypot Efficiency.EasternEuropean Journal of Enterprise Technologiesthis link is disabled, 2021, 1(2(109)), pp. 14–23
18. Merriam Webster (2008). Merriam Webster online: Assurance. Retrieved March 17, 2008, from http://www.merriam-webster.com/dictionary/assurance
19. Khazanchi, D., & Sutton, S. (2001). Assurance services for business-to-business electronic commerce: A framework in implications. Journal of the Association for Information Systems, 1(11)
20. Rosqvist, T., Koskela, M., & Harju, H. (2003). Software quality evaluation based on expert judgment.Software Quality Journal, 11(1), 39-55
21. Subramaniam, G. H., Jiang, J., & Klein, G. (2006). Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. The Journal of Systems and Software, 80, 616-627
22. Gill, N. S. (2005). Factors affecting effective software quality management revisited. ACM Sigsoft Software Engineering Notes, 30(2), 1-4
23. Voas, J. (2003). Assuring software quality assurance. IEEE Software, May/June, 48-49
24. ETH Zurich, Automated object-oriented software Testing using genetic algorithms and Static-analysis (2010). Swiss federal institute of technology Zurich 1, 1-3.
25. Humphrey, Winning with Software an Executive Strategy (2001). Watts S. Humphrey. 1, 1-19. Carnegie Mellon University Software Engineering Institute. Print ISBN-10: 0-201-77639-1 Web ISBN10: 0-321-57935-6
26. Jeff, Software quality engineering testing, quality assurance, and quantifiable improvement, Jeff Tian department of computer science and engineering southern Methodist university Dallas, 7x (2005)
27. Jones A short history of the cost per defect metric (2012). 2-2. Capers Jones, President, Capers Jones & Associates LLC
28. Kshirasagar, Software testing and quality assurance- theory and practice, Kshirasagar Naik -department of electrical and computer engineering-university of waterloo (2008).
29. NIST, The Economic Impacts of Inadequate Infrastructure for Software Testing (2002). National institute of standards and technology. RTI Project Number 7007.011. Final Planning Report 02-3.
30. Rex, Investing in software testing: the cost of software quality (2000). Rex Black is President and Principal Consultant of RBCS, Inc. R. Black, Managing the Testing Process, Second Edition. Wiley, New York, 2002
31. Ricardo, Testability of dependency injection (2007) University of Amsterdam Faculty of science, master research software engineering

32. Ross, The secrets to high customer satisfaction (2013)
33. Feldman, S. (2005). Quality Assurance: Much More than Testing. ACM Queue, February 2005, (pp. 27-29)
34. Humphreys, W. S. (2004). The software quality profile, Retrieved September 12, 2004, from http://www.sei.cmu.edu/publications/articles/quality-profile/index.html
35. DR.G.ANIL KUMAR (2019). A Review on Challenges In Software Testing. Journal of Information and Computational Science Volume 9 Issue 6 - 2019 ISSN: 1548-7741
36. Teodoro, Software reverse engineering education (2009). San Jose State University - United States
37. Jeff, Software quality engineeringtesting, quality assurance, and quantifiable improvement, Jeff Tian department of computer science and engineering southern Methodist university Dallas, 7x (2005).
38. Ricardo, Testability of dependency injection (2007) University of Amsterdam Faculty of science, master research software engineering.
39. Raman Aparna (2009). Innovations in Software Testing: Past, Present and Future. Proceeding of the 9th Annual Software Testing Conference 2009, QAI – India
40. Galin, D. (2004). Software Quality Assurance from theory to implementation. USA: Pearson Addison-Wesley
41. Sommerville, Requirements engineering challenges, Ian Sommerville (2013). 3-22
42. Teodoro, Software reverse engineering education (2009). San Jose State University - United States
43. John, Software Quality Suffers as Businesses Hurriedly Attempt to Innovate (2017), CEO Testplant
44. John Managing Software Deliverables (2004): A Software Development Management Methodology Managing John W. Rittinghouse. ISBN: 1-55558-313-X
45. Pressman Roger (2005). A Managers guide to Software Engineering. Delhi. Tata McGRaw Hill
46. Burnstein Ilene (2003). Practical software testing: a process-oriented approach, USA. Springer
47. Godbole Nina S (2004). Software Quality Assurance Principles and Practice. New Delhi. Narosa Publishing House
48. Grant Obermaier. http://www.ehow.com/about_6682230_softwaredefect-life-cycle.html. [Accessed on: 15th January 2011]
49. Humphrey Watts S. (2003). Introduction to the Personal Software Process. New Delhi. Pearson Education Asia
50. Limaye M. G. (2006). Software Testing – Principles, Techniques and Tools. New Delhi. Tata McGraw-Hill Publishing Company Limited
51. Kaner Cem, Falk Jack, Nguyen Hung Quoc, 2004. Testing Computer Software. New Delhi. International Thomson Computer Press
52. Lewis William E and Veerapillai Gunasekaran (2004). Software Testing and Continuous Quality Improvement. USA. CRC Press
53. Tamres Louise (2003). Introducing Software Testing. Delhi. Pearson Education Asia
54. Ma Liangli, LvYansheng, Liu mengren, Research on Application of Metadata in Component Software Test Case, Mini-Micro Systems, 2006.12.
55. Jiang Zhongwei, Zhang Yunhua, XieXuan'ang, The Generation of Software Testing Case Based on MDA,Computer Engineering and Applications, 2007.43(17)
56. Poulding, S.; Clark, John A.; Waeselynck, H. "A Principled Evaluation of the Effect of Directed Mutation on Search-Based Statistical Testing", Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on, On page(s): 184 – 193
57. Poulding, S.; Waeselynck, H. "Adding Contextual Guidance to the Automated Search for Probabilistic Test Profiles", Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on, On page(s): 293 – 302
58. M. Katara and A. Kervinen, Making model-based testing more agile: a use case driven approach, In Proc. Haifa Verification Conference 2006, number 4383 in Lecture Notes in Computer Science, pages 219-234. Springer, 2007
59. Smith, J. et al. (2019). Improving Software Quality in Healthcare Systems Through Effective Quality Assurance Testing
60. Patel, A. et al. (2020). Enhancing E-commerce Website Quality Through Continuous Quality Assurance
61. Garcia, M. et al. (2018). Ensuring Security and Reliability in Banking Mobile Apps: A Case Study.
62. Smith, A. et al. (2018). Enhancing Software Quality Assurance in Agile Development: A Case Study
63. Johnson, M. et al. (2019). Ensuring Regulatory Compliance in Pharmaceutical Software: A Case Study
64. Martinez, L. et al. (2020). Enhancing Security through SQA: A Case Study of a Banking Mobile App.
65. Netflix Technology Blog (2015). Chaos Engineering Upgraded. https://netflixtechblog.com/chaos-engineering-upgraded-878d341f15fa
66. https://www.microsoft.com/en-us/research/publication/software-testing-and-quality-assurance-the-academic-perspective/
67. https://medium.com/airbnb-engineering/testing-at-airbnb-8d61e82d0f3
68. https://engineering.fb.com/production-engineering/testing-at-the-speed-and-scale-of-facebook/

69. https://www.spacex.com/updates/crew-dragon-demo-1-returns-to-earth
70. Nasib S. Gill, “Factors Affecting Effective Software Quality Management Revisited”. ACM SIGSOFT Software Engineering Notes. Volume 30, Issue 2, March 2005. Page(s): 1 – 4
71. P. Ron. Software testing. Vol. 2. Indianapolis: Sam’s, 2001.
72. Mailewa, Akalanka Bandara, "Reducing Software Testing Time with Combinatorial Testing and Test Automation" (2015). Culminating Projects in Computer Science and Information Technology. Paper 3.
73. Towhidnejad, M.; “Incorporating software quality assurance in computer science education: an experiment". In Proceeding 32nd ASEE/IEEE Frontiers in education conference. Volume 2, November 6-9, 2002. Page(s): F2G-1 - F2G-4.
74. Amrit, Chintan & Hillegersberg, Jos & Kumar, Kuldeep. (2012). Identifying Coordination Problems in Software Development: FindingMismatches between Software and Project Team Structures. Working Paper.
75. Bano, Muneera & Zowghi, Didar & da Rimini, Francesca. (2018). User Involvement in Software Development: The Good, the Bad, and the Ugly. IEEE Software. 35. 8-11. 10.1109/MS.2018.4321252.
76. Pries, Kim & Quigley, Jon. (2018). Software Test Documentation. 10.1201/b10470-12.
77. Iqbal, Talat. (2017). Software Testing. 10.13140/RG.2.2.18247.60323.
78. Torres, Edwin. (2018). Inadequate Software Testing Can Be Disastrous [Essay]. IEEE Potentials. 37. 9-47. 10.1109/MPOT.2015.2404341.
79. Barreto, Ahilton & Barros, Márcio & Werner, Claudia. (2005). Staffing a software project: A constraint satisfaction approach. ACM SIGSOFT Software Engineering Notes. 30. 1-5. 10.1145/1082983.1083093.
80. Iqbal, Nayyar & Qureshi, M. Rizwan. (2012). Improvement of Key Problems of Software Testing in Quality Assurance. CoRR. abs/1202.2506.