JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

An Analysis of Quality Assurance Practices Based on Software Development Life Cycle (SDLC) Methodologies

Kishan Patel
Independent Researcher
kpatelscholarnetwork@gmail.com

***Abstract*—Software project development is a large, massive, and conspicuous phenomenon in the PC industry. The testing phase is also considered as one of the sub-processes of the software development life cycle (SDLC). The overall goal of the SDLC is to enhance the quality of the software product and minimise the risks of failure or customer disillusion. Quality is one of the key factors that determine quality end-result to a large extent. The effectiveness of the final program in every project can be ensured by quality assurance since it should meet the requirements of the project as well as the expectations of the client. This work aims to develop a qualitative and quantitative understanding of QA practices applied with respect to different SDLC frameworks, including Waterfall, V-Model, Scrum, XP, and Kanban. The goal of the study is to evaluate the advantages and limitations of a variety of QA strategies, report the findings of the study, and describe how QA functions should intersect with those of SDLC. Since the study focuses on effects of QA on software quality and project success, it offers insights on how to apply the insights to the specific needs of software projects. These are underscores the significant need to integrate adequate QA measures in the SDLC to enhance an efficiency, reliability and product quality of software products.**

Keywords*—*Software engineering, software development life cycle, Agile Methodology, V model, Waterfall model, Quality Assurance.

I. INTRODUCTION

Software engineering is a field that covers every facet of software development, from system maintenance to the first phases of system definition. In general, the purpose of this engineering is to produce software that performs high and on time. In order to do this, different branches of knowledge come together to develop disciplines that will work together to create a practical result. To that end, there is a framework known as the SDLC that helps with the planning, creating, and testing of software programs [1][2][3]. For their roles in identifying business concerns, initiating and overseeing projects, and ensuring their smooth execution, system analysts must choose a suitable SDLC methodology [4].

Software quality is defined as the degree to which the product satisfies the stated business needs, is reasonably free of bugs or defects, is delivered within the stipulated time frame and budget, and can be easily maintained. There are two parts to software quality when it comes to software engineering: functional quality and structural/non-functional quality. The goal of software functional quality is to guarantee that designs meet the needs of services and products depending on their functional specification. Software quality assurance checks that a product or service meets non-functional criteria. It backs up the functional needs or specifications of services or products, including robustness, maintainability, and the degree of accuracy delivered by the services or software. Any software engineering project worth its salt would prioritise quality[5][6].

The term "software quality assurance" (SQA) refers to a collection of procedures used to guarantee the accuracy and completeness of software development processes. SQA is a processing-based activity that evaluates and organises the processes that generate software services or products. SQA focuses on preventing services or products from faults with a collection of proactive staff functions, while software quality control identifies and fixes flaws using a set of reactive line functions. In reality, software quality control (SQC) is software testing. Products undergo testing in software testing before being published to the live environment[7][8].

The five most popular SDLC approaches are covered in detail in this study paper. QA, or quality assurance, is a collection of procedures for verifying that software engineering processes and methods are up to snuff. There are several methods that QA performs. The process of quality assurance may be divided into two sections. As a developer, your first priority should be to ensure that no defects occur throughout the development cycle. The second phase of inspection is the quality control that involves identifying of the flaws or mistakes that may have been done in the end product. This research focuses on integrating the more conventional waterfall model with features from the modern agile model to enhance the development processes' effectiveness and reliability while maintaining quality.

A. Research Motivation and Significance/Aim

The primary purpose of this research is to explore and analyse the QA processes of various SDLC models. The rationale for this research is informed by the centrality of QA in the accomplishment of software development endeavours. It is now more imperative than ever to ensure the quality of software products because of the many methodologies employed in the SDLC and the increasing sophistication of software systems today. This research will seek to review and compare QA practices across the various SDLC methodologies in order to understand how QA activities are conducted and implemented for the improvement of the quality, reliability and performance of software products. The research will focus on evaluating traditional and agile methodologies to find out the best practices and approaches that would facilitate the incorporation of effective QA practices in developing high-quality software that meets user expectations and business requirements.

B. Research contribution

The following research paper contribution is as:

- Comprehensive Comparative Analysis of QA Practices: The findings of this research offer an effective comparison of QA practices in a range of traditional SDLC models and contemporary agile frameworks. Thus, by highlighting the manner in which QA activities are integrated and performed in each model, the study reveals the advantages and disadvantages associated with distinct approaches that can be beneficial for teams involved in software development.
- Identification of Best Practices in QA: This work also determines and records the best practices of QA as regards the various varieties of the SDLC methodology. These guidelines can provide software development teams with a blueprint for efficiently pursuing an effective QA strategy, thus increasing the overall quality and reliability of software products.
- Impact of QA on Software Quality and Project Success: The direct effects of QA practices on the total quality of software products and successful software development projects are the subjects of this research. In this respect, the project-based study establishes the connection between the QA activities and their positive impact on the achievement of the project objectives and customer requirements.
- Framework for Integrating QA in Agile and Traditional Models: The research develops a framework for effectively integrating QA practices into both agile and traditional SDLC models. This framework provides practical steps and considerations for embedding QA activities seamlessly into the development process, enhancing the efficiency and effectiveness of QA efforts.

II. SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) METHODOLOGIES

Projects are developed using the models of the SDLC. Software development process models are another common name for these types of models [9]. A SDLC is a fundamental methodology for creating software in a methodical fashion. The process consists of many steps, the first of which is determining the software's functional requirements. The next steps are development, testing, and design.

Figure 1: Development process of the SDLC approach

Important stages of the SDLC methodology's development process are shown in Figure 1:

Step 1: Requirements analysis

This step is crucial for analysing and documenting client requirements in a clear and natural way. An expert team gathers and records project information in a document. This paper describes project development in detail. A document providing client needs is called the Software Needs Specification (SRS)[10][11]. After gathering relevant data, developers formulate a plan for the project and check its viability in light of the product's intended technical and economic uses [12][13].

Step 2: Designing

After analysing project data, design shows the requirements' structure in a computer language. This stage involves project form design and implementation. The Software Design Description document describes this design. To ensure the best product, specialists examine the SDD document under specific criteria to assess risk, product quality, design, cost, and timeline.

Step 3: Implementation and unit testing

The code portion that was carefully documented in SDD during the previous design stage is included in this step. Each of the project's modest, functional pieces is put through testing. If required, coding changes may be made at the product inspection step [14].

Step 4: Testing

A software testing process helps find and fix faults to improve product quality. Some unit testers can't effectively test the program interface between units. Thus, a thorough system test and software testing is required. SDLC software testing uses most of the software development expenditure. This stage builds developer trust before the product is given to customers or published [15].

Step 5: Maintenance

The last step in creating a program before it is delivered, run, and published is called program maintenance. User Acceptance Testing (UAT) tests the new product in a genuine business context. This stage gradually corrects coding flaws to improve the product and eliminate old parts after use

A. Types of Traditional SDLC Approach

To guarantee the effectiveness of the model creation process, each model has certain procedures that must be followed. The conventional model and the agile model, which will be covered in more depth in this section, are the two primary categories of development models.

1) Waterfall Model OF SDLC

The waterfall model is the inaugural paradigm in the chronicles of software development, as shown in Figure 2. The Linear Sequential Life Cycle Model, often known as the LSLC model, is characterised by its simplicity in both comprehension and application. The waterfall paradigm follows a sequential approach where each step is fully accomplished before progressing to the subsequent phase. The SDLC began with the waterfall model, which is the oldest and original method used to create software products without any stages that overlap.

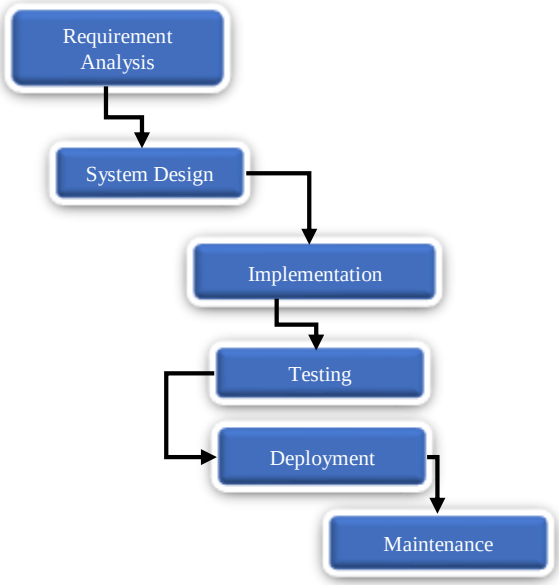

Figure 2: Sequential phase of Waterfall Model

The waterfall technique, the first approach in the SDLC, was widely used to create software products and had a significant role in the project's success.

2) V-Model

Similar to the waterfall model, the V-model operates by a succession of stages that cascade from the top layer down, as seen in Figure 3. Each phase of the development cycle is linked to the testing process in this paradigm. This methodology is very structured since each step doesn't start until the previous one is finished [16]. In most cases, validation follows verification on one side of the model, and the two sides are connected in terms of the model's coding architecture.

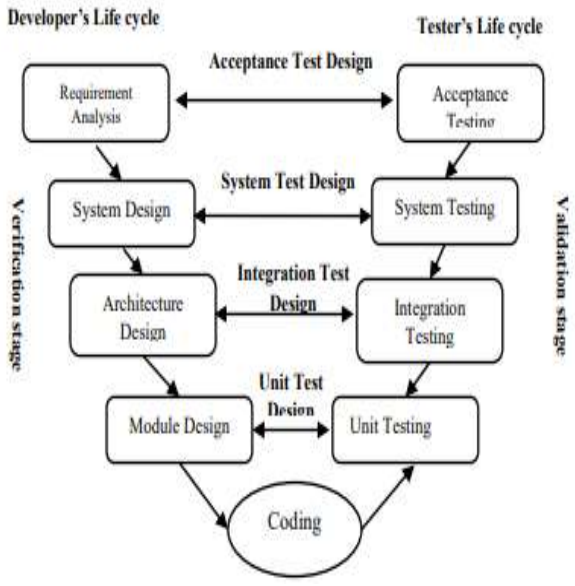

Figure 3: Phase of V-Model

Each level is finished individually since this is a highly declined model. With well-stated criteria, the V-Model performs effectively in small projects. Because each step has a distinct deliverable and review procedure, the model's rigidity makes it easy to manage.

3) Spiral Model

This is the evolving software operation model that combines the linear sequential model's controlled and methodical aspects with the ongoing prototyping process. Figure 4 depicts the spiral model that is developed utilising software in a progressive release sequence. Early incarnations of the incremental releases could have been paper models or prototypes. Later cycles result in an increasing number of absolute varieties of the designed systems [17].

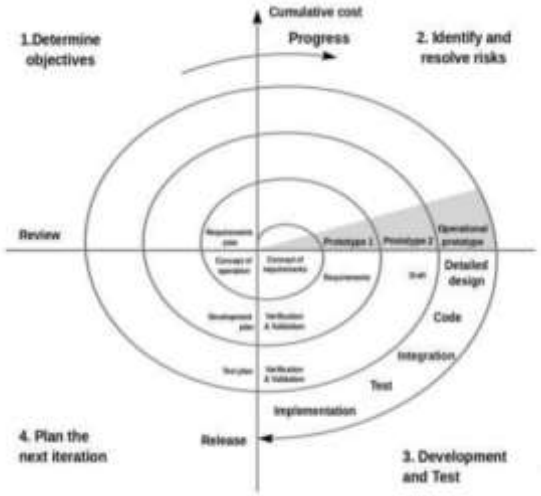

Figure 4: Spiral Model

This type is highly costly. Executing a precise risk analysis is essential for the successful completion of the project since it heavily influences the project's progress. This model is not suitable for minor jobs.

4) Iterative Model

Figure 5 depicts the iterative model, one of the SDLC approaches. This model relies on gradually implementing software development requirements via smaller units until a full system is achieved. To accommodate software development process needs, the iterative or incremental approach is made up of many architectural units [18]. The components of any architectural unit include design, development, testing, and implementation. Every iteration of the development process aims to produce a final version that satisfies the criteria of the proposed model and is ready for publication. Every time a new model is being developed, every need has to be verified and validated in order to guarantee the success of the iterative iteration of software development [19].

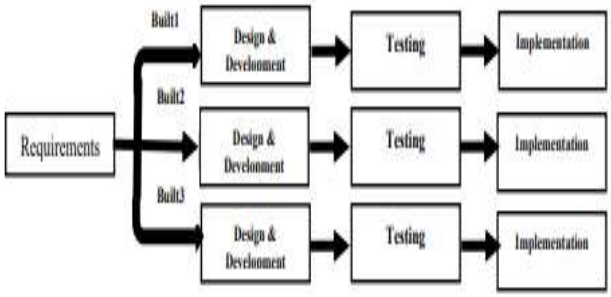

Figure 5: Iterative Model

This model is scalable quickly and early. Cost is only incurred when needs change. Simplicity of testing and error-handling via iterations

5) Big Bang Model

The Big Bang model is an SDLC paradigm commonly employed for small projects, without a specific set of phases shows in figure 6. This model is dependent on the inputs, which include financial resources and efforts, as well as the outputs, which refer to the final product. Nevertheless, it often falls short of meeting the customer's expectations because of its inherent vagueness regarding its specific objective. This type is renowned for its exceptional velocity and minimum scrutiny, as well as its compact development team. The Big Bang approach emphasises the distribution of project development resources without extensive or any preparation. Generally, changes in the model requirements have minimal influence on the whole process of project renewal. Thus, this

paradigm is particularly suitable for brief projects that do not necessitate an extensive comprehension of the requirements and include a small development team.

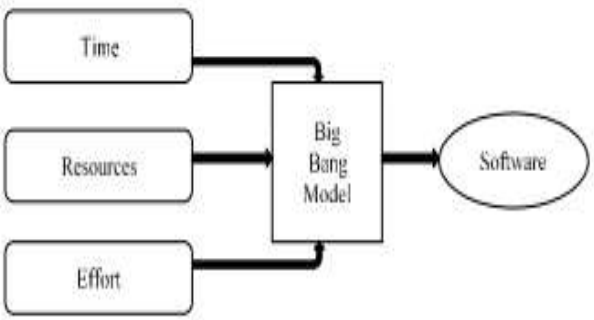

Figure 6: Big Bang Model architecture

Planning is not necessary or necessary at all, and managing the model is simple. The model is very adaptable and doesn't demand a lot of resources. For new pupils, it is an appropriate learning environment.

Every model needs its own distinct sequence of procedures to guarantee software development success. Processes including requirements gathering, design, development, testing, and maintenance were foundational to SDLC models.

III. AGILE SDLC TYPES

Agile software development is widely known as the "moving quickly" technique due to its lightweight nature. This strategy involves developing software in little portions termed ‘iterations’ or ‘increments’, each representing a project stage [20][21]. According to, the agile methodology prioritises client participation and interaction in product development over traditional methods like requirements research and planning. Many models will be discussed in this method. Figure 7 shows agile architecture[22].

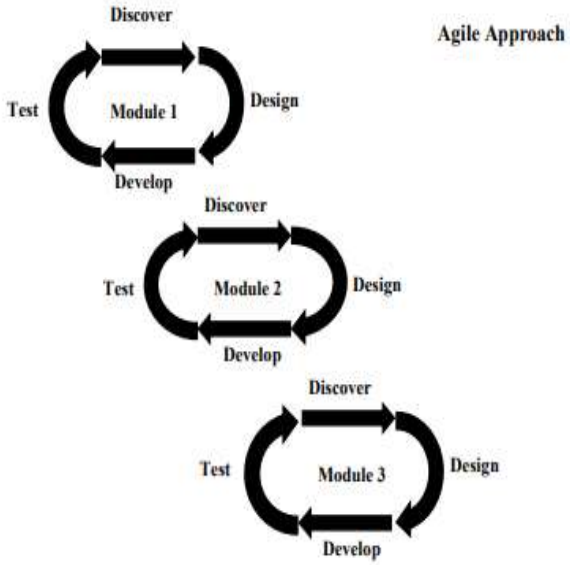

Figure 7: Agile SDLC types

This method of software development is too legalistic, especially considering how much easier projects are to communicate when using the agile paradigm. From an expenditure standpoint, project financing is greatly simplified by its pay-as-you-go structure. Little to no preparation is necessary, making it simply controllable and providing developers with flexibility.

A. XP (EXTREME PROGRAMMING)

The Extreme Programming (XP) methodology in the middle of the 1990s. XP is iterative and delivers little versions at the conclusion of each iteration, much like other agile methodologies. The twelve principles upon which XP is built provide an emphasis on the technical parts of agile software development [23].

B. SCRUM Model

The current and most important agile model is Scrum. Credit for its creation goes to Ken Schwaber and Jeff Sutherland in 1993. It claims that developers may work through difficult project difficulties and come up with innovative solutions by following the scrum methodology. Product development may be managed and regulated in a way that progressively and realistically meets customer expectations using this process, which is clear and easy to understand [24]. This concept organises all activities into sprints. Sprints categorise tasks according to the complexity and scale of the design. Scrum adapts to all required changes. This model organises teams to work more efficiently and produce high-quality results [5]. Figure 8 depicts Scrum model architecture.

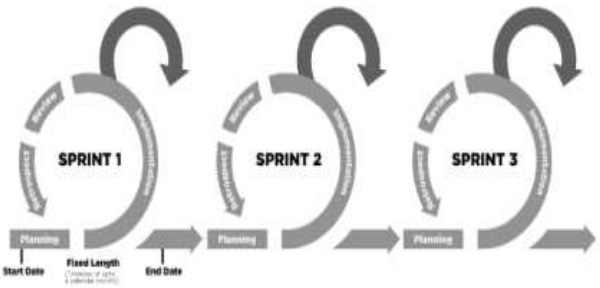

Figure 8: Scrum model architecture

Effective and efficient communication is among team members. Frequently get feedback from customers on the product. To satisfy the demands of the consumer, it produces high-quality items.

C. Kanban Model

Toyota Company created Kanban to track industrial operations. Around the globe, systems development organisations generally agree on this approach. The foundation of the Kanban methodology is just-in-time (JIT) concepts. The Kanban technique informs of required items and quantities for timely delivery. The tool can identify constraints in development that could impact workflow [25][26]. Additionally, it ensures transparency within the Kanban board management work team. Additionally, it can track tasks at any given time, enabling process monitoring. This method involves using the limit option to organise tasks to the maximum [27]. Kanban aims to shorten project completion time. In summary, three key aspects include worker awareness of present tasks, system development by the computing developer, and employee leadership in program development through the Kanban model. Refer to Figure 9 for the Kanban model design.

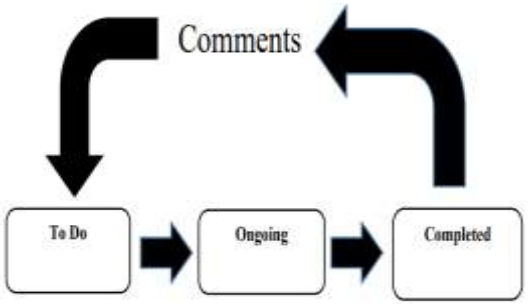

Figure 9: Kanban model architecture

Model that is simple to use and applicable to a wide range of businesses. Adaptable with Cooperation Support. Minimal overheads.

IV. ROLE OF QUALITY ASSURANCE IN SDLC

QA is an integral part of any effective SDLC process. QA plays a much more substantial part in the SDLC than is often believed. One critical component of the SDLC is QA. It guarantees the best way to create a high-quality product that satisfies all requirements, is free of defects, and performs as expected. Planning, implementing, testing, deploying, and maintaining are all stages of the SDLC in which QA may play an essential role[28].

A. Role of QA in Planning from SDLC

The responsibility of ensuring that a software project satisfies all quality criteria, including scope, budget, deadline, and standard compliance, falls on Quality Assurance throughout the planning phase. In order to assess if user needs are feasible within the parameters of the project, QA may examine and evaluate them. This helps in making sure that the right people are involved in setting expectations and allocating resources at the start of a project.

- The development process is guided by the precise plans created by Quality Assurance.
- Performing tasks include writing documentation detailing procedures and quality assurance requirements and developing test strategies.
- QA also helps in risk reduction by seeking out any issues in advance and fixing them before they escalate.

B. Role of QA in Implementation from SDLC

QA ensures that specific measures regardless of their development or during the implementation phase, standard measures of quality in the services to be rendered are met. Besides building and conducting tests and ensuring that all the quality standards are met, there is a need to document the process. Besides the enhancement of the quality aspect and the reduction of mistake risks in this phase, QA also provides early access to higher maintenance and support solutions. The following summarises QA's responsibilities throughout the installation phase:

- Code Reviews: A major activity of code reviews involves ensuring that all subprograms and components conform to quality assurance criteria as performed by QA specialists. This implies evaluating coding techniques and ensuring that the code is correct to the standard practices in the industry.
- System Integration Testing: Moreover, the interoperation of each component of the software with the other components is also verified by the QA engineers. The concept of system integration testing is directed at verifying many aspects at once to ensure their high quality and compatibility.
- User Acceptance Testing: To ensure that the system meets the user requirements as described in the development documents, the QA engineers perform user acceptance testing, abbreviated as UAT. UAT involves testing a number of specific user stories that technical writers have developed.

C. Role of QA in Testing from SDLC

Testing quality assurance (QA) may appear self-evident at first. To be sure, QA spends a lot of time developing test cases and reporting bugs, but it also does much more than merely "test" throughout the testing process. Instead, QA focusses on comparing the product or service to predetermined criteria of quality. Functionality, usability, dependability, performance, and adherence to industry standards are just a few of the elements that QA concentrates on. to guarantee that, prior to the program being made available to users, all of its criteria are satisfied. In order to carry out these tests successfully:

- QA teams usually employ automated testing methods to find functional problems early in the development phase.
- Analyse the application's performance in comparison to other comparable apps on the market and look for potential conflicts between various components.
- To make sure the user is comfortable, you should do user experience (UX) testing.
- Before launching the product, ensure it satisfies all applicable regulations and industry standards.

D. Role of QA in Deployment from SDLC

QA makes ensuring that every component of custom software development is correctly implemented, tested, confirmed, and delivered during the deployment stage. This makes it possible to deploy the product confidently and without any problems or bugs:

- QA also performs a number of activities, such as:
- Verifying that all tests have been executed without error.
- Making sure the product's necessary parts have been deployed correctly and efficiently.
- The process of fixing problems that crop up during testing or after deployment.
- To make sure no modifications were performed accidentally during deployment, it is recommended to do regression tests.

E. Role of QA in Maintenance from SDLC

Product maintenance also entails ensuring that the product is accurate and has no flaws. The objective of QA is to make sure that new versions of the application do not bring new bugs to the system. It has been helpful to get their perception on the effectiveness of these changes and satisfaction of customers with Application Maintenance Services. As part of their maintenance responsibilities, QA must ensure:

- Checking that software upgrades are functioning correctly
- Making sure the changes work as expected by testing them
- Identifying potential problems with updates
- Making sure clients are satisfied with changes and enhancements by checking in with them
- Keeping a record of every release-related problem so that it may be addressed in upgraded versions.
- Companies can ensure that their goods are dependable and free of bugs for a long time by investing in quality assurance during maintenance!

V. PRACTICES OF QUALITY ASSURANCE (QA)

QA which is also referred to as Quality assurance has been defined by the ISO 9000 standard as the element of quality management that ensures that the preconditions for eliminating defects are met. The purpose of quality assurance is to determine if the application will successfully satisfy the needs of the intended audience. The purpose of QA methods is to ensure a high standard of application development at every step of the software life cycle. These steps usually begin before the application is even considered for development and go on throughout its creation. The responsibility of quality assurance is to establish and execute standards and processes with the goal of improving the development life cycle and ensuring that these procedures are followed. The overarching goal of QA is to guarantee that software is consistently becoming better and better by eliminating defects at every level of development. Quality control, in contrast to quality assurance, deals with the collection and examination of application quality after production has already occurred rather than before its formation. As a result, testing is an element of both quality control and quality assurance structures. To achieve quality, one must set reasonable expectations.

Ensure that items can be objectively measured against those specifications. Several organisations lack the ability or

motivation to establish detailed specifications. Evaluation of a finished product cannot determine quality. Avoiding quality compromise is the goal. Figure 10 shows an interaction of testing, quality control, and QA.

Figure 10: The interaction of testing, quality control, and Quality Assurance.

A. Software Quality Assurance (SQA) Factor

There are three main categories into which the factors influencing software quality fall. There are three groups: organisational, technological, and human. What follows is an in-depth description of each of these:

1) Organization Factors

According to IS0 DIS 13407, user-centered design enhances system quality by enhancing working circumstances while maintaining mental and physical health. By preserving human health from work-related pressure, higher performance and faster creation and testing can lead to high-quality products. Human variables impact product quality, cost, and length. An organisation should consider it when creating a software product. Human factors are influenced by software development company's culture, training, and team dynamics [10].

2) Technological factors

The individual tasked with creating a product's need specifications has a direct impact on the output's quality. A poor level of requirement specification will have a negative impact on product quality because of the direct correlation between the two. Based on Garvin's theory, a high-quality end product verifies that the suitable requirement specification describes the production quality [29].

3) Human factors in SQA

Software quality affects user experience and utilisation. The product's functionality depends on user type and number. This encompasses end-user usability, efficiency, dependability, portability, and maintainability. Customer happiness and product performance determine quality. Both measuring degree and care are important for quality. Care is provided by generalising quality measurement impacts across contexts. Product quality can be improved by changing intermediate and end-product processes. There are various tasks to obtain results.

Human quality influences user-centred design and usability. A user-centered design incorporates prototypes and mockups, while interface impacts usability. Usability and user-centred design improve product quality.

VI. LITERATURE REVIEW

This section provides the existing work on QA from SDLC. Software development has exploded in popularity during the last 20 years.

In, Laaraib et al., (2021) Ensuring that a software product satisfies its realistic business criteria and quality standards is the goal of QA in software development. For each phase of the SDLC, the best software quality assurance procedures have been determined by conducting a thorough review of all software quality requirements and quality assurance practices. This research paper proposes a technique and details how to use it within the SDLC. Software development teams will be able to accomplish their quality assurance objectives with the aid of the suggested technique[30].

In, Akbar et al., (2017) study on software quality development because it impacts client satisfaction, which is crucial to project success. The software process model does a lot of things: it manages project time, enhances the process, assures software quality, uses process knowledge effectively, and covers a number of work situations. Furthermore, this article introduces the "AZ-Model" SDLC model, which incorporates additional steps into the procedure. A high-quality product may be developed on time with its help, and it outperforms traditional methods. An extensive comparison and statistical studies examine the AZ model's importance in software development in this study[31].

In, Karim et al., (2016) shows that the Software changes are complicated and require a systematic way to monitor, control, and track. As a result, in order to optimise business benefits, product quality, process routines, and performance profitability, it is essential that all members of the project team adhere to change control standards and principles. This paper also gives a broad review of IT Project Management software change control. The document discusses SDLC and PLC project activities step-by-step[32].

In, Laaraib et al., (2021) software development, quality assurance ensures that a product fulfils their practical business requirements and quality standards and reviewed several software quality requirements and assurance procedures to determine the most effective practices at various stages of the SDLC. A proposed technique and implementation during SDLC are presented in this research article. This methodology aims to aid software development teams in meeting quality assurance goals[30].

In, Wong et al., (2022) study develops PLC and SDLC milestones to assist the IT sector in implementing software project management as official software process initiatives. An effort-driven SQAP documents SPI methodology, workflows, strategies, and tools. Meeting client needs, contract duties, and SLA compliance requires project teamwork. SQAP misbehaviour will affect customer interactions, brand and reputation, operational and financial expenses, competitiveness, and IT industry confidence. Thus, SQAP deployment requires major PLC and SDLC checkpoints[33].

In, Mohino et al., (2019) research a technique is used to demonstrate how, if security is taken into account from the outset, vulnerabilities may be quickly found and fixed within the project's scheduled timeframe, saving the customer money and effort and boosting the likelihood of functionality and quality success. Software security, quality, and, often, project value are largely disregarded in such an atmosphere. Agile models are trendy right now. This environment, a new software development model that prioritises security across the program life cycle and uses agile approaches is appropriate[34].

In, Mohammed et al., (2017) the purpose of this article is to catalogue the SDLC's software security methodologies. The next step was to use a systematic mapping approach to locate the primary SDLC software security studies. A total of 118 main studies were categorised. According to the findings, the two most popular methods for checking the security of code are static and dynamic analysis[35].

The related work on quality assurance in software development highlights various methodologies and models aimed at ensuring software products meet business specifications and quality requirements. Key contributions include the integration of AI and machine learning into QA

processes, the introduction of new SDLC models like the AZ-Model for improved task management and quality control, and the development of comprehensive Software Quality Assurance Plans (SQAP) to formalise software process initiatives. Studies emphasise the importance of combining methodologies such as Agile and Waterfall to leverage their strengths, the critical role of security from the project's inception, and the necessity for robust change control management. However, challenges remain in ensuring consistent application of QA practices, managing the complexity of software changes, achieving seamless integration of security measures, and maintaining balance between different development methodologies to meet diverse project needs.

VII. CONCLUSION AND FUTURE WORK

SQA strategies incorporated by many people of quality software engineers. SQA in context to SDLC has been analysed with respect to various aspects. Thus, to contribute to the creation of high-quality products, the SQA team should be engaged in the process starting from the SDLC's first phase. In conclusion, this study establishes that QA is an essential component of the SDLC and drives the success of software projects. The comparison of the traditional and agile SDLC processes shows that the incorporation of quality assurance practices into development maintains product quality and ensures that it meets the user and business expectations. A study outlines approaches and offers a guide as to how QA activities should be conducted at different stages of the SDLC ranging from the planning and design phase to the implementation, testing, deployment, and maintenance phases. Moreover, a research reveals that it's crucial to adjust QA approaches to meet the needs of projects and take into account the size, complexity, and people involved. The software development teams will be in a position to improve on the quality assurance practices as suggested in this study and thus increase the rate of successful projects and customer satisfaction.

Future research lies in the implementation of AI and machine learning into the QA process for automation and improved outcomes. Studying QA in new SDLC approaches like DevOps and continuous delivery can improve more rapid, dependable releases. Enhancing real-time monitoring techniques for QA activities and investigating the effect of organisational culture on QA practices will lead to refining them. Further research can include qualitative analyses of the effects of the different established QA approaches across industries and over time with a view to enhancing sustainable quality management.

REFERENCES

[1] D. Doshi, L. Jain, and K. Gala, "REVIEW OF THE SPIRAL MODEL AND ITS APPLICATIONS," *Int. J. Eng. Appl. Sci. Technol.*, 2021, doi: 10.33564/ijeast.2021.v05i12.053.

[2] N. Mohammed, A. Munassar, and A. Govardhan, "A Comparison Between Five Models Of Software Engineering," *Int. J. Comput. Sci.*, 2010, doi: 10.1.1.403.3201.

[3] Y. Bassil *et al.*, "Making Sense of Software Development and," *Int. J. Eng. Adv. Technol.*, 2012.

[4] C. Larman and V. R. Basili, "Iterative and incremental development: A brief history," *Computer*. 2003. doi: 10.1109/MC.2003.1204375.

[5] M. A. Subih *et al.*, "Comparison of agile method and scrum method with software quality affecting factors," *Int. J. Adv. Comput. Sci. Appl.*, 2019, doi: 10.14569/ijacsa.2019.0100569.

[6] P. Jain, A. Sharma, and L. Ahuja, "A customized quality model for software quality assurance in agile environment," *Int. J. Inf. Technol. Web Eng.*, 2019, doi: 10.4018/IJITWE.2019070104.

[7] R. T. Kaleem Ullah, Majid Ali Tunio, Zahid Ullah, Muhammad Talha Ejaz, Muhammad Junaid Anwar, Muhammad Ahsan, "Ancillary services from wind and solar energy in modern power grids: A comprehensive review and simulation study," *J. Renew. Sustain. Energy*, vol. 16, no. 3, [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=G4XzELUAAAAJ&citation_for_view=G4XzELUAAAAJ:qjMakFHDy7sC

[8] M. Tuteja and G. Dubey, "A Research Study on importance of Testing and Quality Assurance in Software Development Life Cycle (SDLC) Models," *Int. J. Soft Comput. Eng.*, no. 2, pp. 2231–2307, 2012.

[9] I. H. Sarker, F. Faruque, U. Hossen, and A. Rahman, "A survey of software development process models in software engineering," *Int. J. Softw. Eng. its Appl.*, 2015, doi: 10.14257/ijseia.2015.9.11.05.

[10] V. P. Vinita Rohilla, Saksham Arora, Priyansh Singh Nirwan, "Recommendation System Using Location-Based Services," *Proc. Int. Conf. Innov. Comput. Commun.*, [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=zlcFgwEAAAAJ&citation_for_view=zlcFgwEAAAAJ:qjMakFHDy7sC

[11] Tutorials Point, "SDLC Tutorial," Tutorials Point.

[12] A. Coronato, "Agile software development life cycles," in *Engineering High Quality Medical Software: Regulations, standards, methodologies and tools for certification*, 2018. doi: 10.1049/pbhe012e_ch13.

[13] M. K. Vinita Rohilla, Sudeshna Chakraborty, "Artificial Intelligence and Metaheuristic-Based Location-Based Advertising," *Sci. Program.*, no. 1, 2022, [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=zlcFgwEAAAAJ&citation_for_view=zlcFgwEAAAAJ:Tyk-4Ss8FVUC

[14] T. T and M. Prasanna, "Research and Development on Software Testing Techniques and Tools," 2018. doi: 10.4018/978-1-5225-7598-6.ch109.

[15] T. Jindal, "Importance of Testing in SDLC," *Int. J. Eng. Appl. Comput. Sci.*, 2016, doi: 10.24032/ijeacs/0102/05.

[16] T. Chittagong and T. Islam, "Introducing a New Sdlc Trigon Model for," in *Proceedings of the International Conference on Sustainable Development in Technology for 4th Industrial Revolution 2021 (ICSDTIR-2021)*, 2021.

[17] P. Trivedi and A. Sharma, "A comparative study between iterative waterfall and incremental software development life cycle model for optimizing the resources using computer simulation," in *Proceedings of the 2013 2nd International Conference on Information Management in the Knowledge Economy, IMKE 2013*, 2013.

[18] P. Seema Suresh Kute and P. Surabhi Deependra Thorat, "A Review on Various Software Development Life Cycle (SDLC) Models," *Int. J. Res. Comput. Commun. Technol.*, 2017.

[19] O. J. Okesola, A. A. Adebiyi, A. A. Owoade, O. Adeaga, O. Adeyemi, and I. Odun-Ayo, "Software Requirement in Iterative SDLC Model," in *Advances in Intelligent Systems and Computing*, 2020. doi: 10.1007/978-3-030-51965-0_2.

[20] R. P. Pawar, "A Comparative study of Agile Software Development Methodology and traditional waterfall model," *IOSR J. Comput. Eng.*, 2015.

[21] V. Rohilla, S. Chakraborty, and M. Kaur, "Artificial Intelligence and Metaheuristic-Based Location-Based Advertising," *Sci. Program.*, 2022, doi: 10.1155/2022/7518823.

[22] V. R. Rinky Dwivedi, "Empowering Agile Method Feature-Driven Development by Extending It in RUP Shell," *Adv. Comput. Comput. Sci. Proc. ICCCCS 2016*, vol. 1, 2016, [Online]. Available: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=zlcFgwEAAAAJ&citation_for_view=zlcFgwEAAAAJ:u5HHmVD_uO8C

[23] L. Williams and A. Cockburn, "Agile software development: It's about feedback and change," *Computer*. 2003. doi: 10.1109/MC.2003.1204373.

[24] V. Hema, S. Thota, S. Naresh Kumar, C. Padmaja, C. B. Rama Krishna, and K. Mahender, "Scrum: An Effective Software Development Agile Tool," in *IOP Conference Series: Materials Science and Engineering*, 2020. doi: 10.1088/1757-899X/981/2/022060.

[25] W. Zayat and O. Senvar, "Framework Study for Agile Software Development Via Scrum and Kanban," *International Journal of Innovation and Technology Management*. 2020. doi: 10.1142/S0219877020300025.

[26] N. Shibu, sini sunny, A. Rajkumar, A. Sayed, P. Kalaiselvi, and R. Namakkal-Soorappan, "N-Acetyl Cysteine Administration Impairs EKG Signals in the Humanized Reductive Stress Mouse," *Free Radic. Biol. Med.*, 2022, doi: 10.1016/j.freeradbiomed.2022.10.122.

[27] K. Risener, "A Study of Software Development Methodologies," *Comput. Sci. Comput. Eng.*, 2022.

[28] S. I. Services, "Role of Quality Assurance in SDLC," solzit.com. [Online]. Available: https://www.solzit.com/role-of-quality-

assurance-in-sdlc/

[29] Z. A. Maher, H. Shaikh, M. S. Khan, A. Arbaaeen, and A. Shah, "Factors Affecting Secure Software Development Practices among Developers-An Investigation," in *2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences, ICETAS 2018*, 2018. doi: 10.1109/ICETAS.2018.8629168.

[30] E. Laaraib *et al.*, "A Methodology for Incorporating Quality Assurance Practices during Software Development Life Cycle," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 10, no. 3, pp. 2296–2301, 2021, doi: 10.30534/ijatcse/2021/1141032021.

[31] M. A. Akbar *et al.*, "Improving the quality of software development process by introducing a new methodology-Az-model," *IEEE Access*, 2017, doi: 10.1109/ACCESS.2017.2787981.

[32] N. S. A. Karim, A. Albuolayan, T. Saba, and A. Rehman, "The practice of secure software development in SDLC: an investigation through existing model and a case study," *Secur. Commun. Networks*, 2016, doi: 10.1002/sec.1700.

[33] W. Y. Wong, T. Hai Sam, C. W. Too, and W. Fong Pok, "Software Quality Assurance Plan: Setting Quality Assurance Checkpoints within the Project Life Cycle and System Development Life Cycle," in *2022 IEEE 18th International Colloquium on Signal Processing and Applications, CSPA 2022 - Proceeding*, 2022. doi: 10.1109/CSPA55076.2022.9782044.

[34] J. de V. Mohino, J. B. Higuera, J. R. B. Higuera, and J. A. S. Montalvo, "The application of a new secure software development life cycle (S-SDLC) with agile methodologies," *Electron.*, 2019, doi: 10.3390/electronics8111218.

[35] N. M. Mohammed, M. Niazi, M. Alshayeb, and S. Mahmood, "Exploring software security approaches in software development lifecycle: A systematic mapping study," *Comput. Stand. Interfaces*, 2017, doi: 10.1016/j.csi.2016.10.001.