

Addis Ababa University

College of Natural and Computational Science

Department of Computer Science

"Data-Driven Quality Assurance in Agile Software project: Optimizing Scrum Checkpoints through a Conceptual Framework"

Nigusu Daniel GSE/1178/15

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia
June, 2025 G.C

Addis Ababa University
College of Natural Sciences

Nigusu Daniel GSE/1178/15

Advisor: *Ayalew Belay (Ph.D.)*

This is to certify that the thesis prepared by Nigusu Daniel, entitled: "*Data-Driven Quality Assurance in Agile Software project: Optimizing Scrum Checkpoints through a Conceptual Framework"* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Name	Signature	Date

Advisor: Ayalew Belay (Ph.D.)_____

Examiner: _____

Examiner: _____

Abstract

This thesis investigates the enhancement of quality assurance (QA) in agile software development, with a specific focus on the Scrum framework. Agile methodologies, while effective in promoting flexibility and fast delivery, often face challenges in maintaining consistent software quality due to limited time for testing and the absence of structured QA practices. Traditional quality models are often incompatible with the iterative and dynamic nature of agile processes.

To address this issue, the study proposes a data-driven conceptual framework that integrates quality assurance checkpoints into Scrum-based agile workflows. The framework is built upon two established quality models: the Plan–Do–Check–Act (PDCA) cycle and the Cross-Industry Standard Process for Data Mining (CRISP-DM). These models guide the systematic placement and evaluation of QA checkpoints across the Software Development Life Cycle and the Project Life Cycle.

The research followed a mixed-methods approach. The framework was initially developed through an in-depth literature review and expert consultations (qualitative phase), and then empirically validated using a quantitative survey of 51 agile professionals in Addis Ababa. The structured questionnaire captured data on QA practices across Scrum phases, and statistical analysis using SPSS including correlation analysis was applied to identify critical checkpoints and evaluate the framework's relevance.

Results show that integrating structured, data-driven QA checkpoints into Scrum significantly enhances defect tracking, test planning, and product quality. The proposed framework provides a scalable and practical solution for organizations seeking to align agile delivery with rigorous quality standards.

Keywords: Agile software development, Scrum, Quality assurance, Data-driven framework, Software quality, PDCA, CRISP-DM

Acknowledgments

First and foremost, glory be to the Almighty God.

Then I would like to express my sincere gratitude to my advisor Ayalew Belay (Ph.D.) for their invaluable guidance, encouragement, and support throughout the course of this research. I am also thankful to your insights and feedback helped shape this work.

Special thanks go to the software development companies and professionals who participated in the study and provided essential data. I am also grateful to my family and friends for their patience and continuous moral support.

Table of Contents

Acronyms and Abbreviations iv
List of figures v
List of tables v
Chapter One ..... - 1 -
1. Introduction ..... - 1 -
1.1 Background ..... - 1 -
1.2 Motivation ..... - 1 -
1.3 Statement of the Problem ..... - 2 -
1.4 Objectives ..... - 4 -
1.5 Methods ..... - 4 -
1.5.1 Research Design ..... - 4 -
1.5.2 Data Collection Method ..... - 5 -
Sampling Criteria ..... - 6 -
1.5.3 Data Analysis Techniques ..... - 7 -
1.5.4 Framework Development and Evaluation ..... - 7 -
1.6 Scope and Limitations ..... - 7 -
1.6.1 Scope: ..... - 7 -
1.6.2 Limitations: ..... - 8 -
1.7 Significance ..... - 8 -
1.8 Organization of Thesis ..... - 8 -
Chapter Two ..... - 9 -
2. Literature Review ..... - 9 -
2.1 Introduction ..... - 9 -
2.2 Conceptual and Theoretical Framework definitions ..... - 9 -
2.2.1 Agile Methodology ..... - 10 -
2.2.2 Agile Scrum ..... - 11 -
2.2.3 Agile and Software Quality Assurance ..... - 13 -
2.3 Current Practices and Challenges in Agile QA ..... - 14 -
2.4 Data-Driven Approaches in Quality Assurance for Agile Environments ..... - 20 -
2.4.1 The Role of Data-Driven Techniques in QA ..... - 20 -
2.4.2 Relevance to Agile Environments ..... - 21 -

2.4.3 Comparative Analysis: Data-Driven vs. Traditional QA ..... - 21 -
2.5 QA Checkpoints in SDLC and PLC ..... - 22 -
2.5.1 Defining SDLC and PLC ..... - 22 -
2.5.2 Checkpoints within the PLC and SDLC ..... - 22 -
2.5.3 Research on Critical QA Interventions ..... - 23 -
2.5.4 Frameworks and Guidelines for Effective QA Checkpoints ..... - 24 -
Chapter Three ..... - 26 -
3. Related Work ..... - 26 -
Chapter Four ..... - 31 -
4. Proposed solution ..... - 31 -
4.1 Introduction ..... - 31 -
4.2 CRISP-DM and PDCA Framework ..... - 31 -
4.2.1 Step 1: PDCA (Process Improvement & Execution) ..... - 31 -
4.2.2 Step 2: CRISP-DM (Data-Driven Insights & Optimization) ..... - 33 -
4.3 CRISP-DM + PDCA Model for Scrum QA ..... - 34 -
4.3.1 Checkpoint Mapping for QA across SDLC and PLC Stages in Scrum ..... - 36 -
4.3.2 Synthesis of the Integrated Framework ..... - 38 -
4.3.3 Mapping CRISP-DM and PDCA to Scrum QA Checkpoints ..... - 39 -
Chapter Five ..... - 41 -
5. Data Presentation and Analysis ..... - 41 -
5.1 Introduction ..... - 41 -
5.2 Demographic Profile of Respondents ..... - 41 -
5.2.1 Gender of Respondents ..... - 41 -
5.2.2 Age of Respondents ..... - 42 -
5.2.3 Education status ..... - 42 -
5.2.4 Position and years of experience ..... - 43 -
5.2.5 Scrum Team Vs Scrum Sprint Duration ..... - 44 -
5.2.6 Agile Adoption & Benefits ..... - 44 -
5.3 Correlation Analysis ..... - 46 -
5.3.1 Product Vision Processes Vs Project Success ..... - 47 -
5.3.2 Product Backlog Processes Vs Project Success ..... - 48 -
5.3.3 Release Backlog Planning Processes Vs Project Success ..... - 50 -

5.3.4 Sprint Backlog Processes Vs Project Success ..... - 51 -
5.3.5 Product Delivery Processes Vs Project Success ..... - 52 -
5.4 Discussion ..... - 54 -
5.4.1 Validating PDCA Integration in Scrum QA ..... - 54 -
5.4.2 CRISP-DM as a Data-Driven Enabler of QA ..... - 55 -
5.4.3 Tri-Layer QA Mapping: Scrum, SDLC, and PLC Alignment ..... - 56 -
Chapter Six ..... - 57 -
6. Conclusion and Recommendation ..... - 57 -
6.1 Summary, Conclusion, And Research Contributions ..... - 57 -
6.2 Recommendations ..... - 58 -
References ..... - 60 -
Annexes ..... - 65 -
Annex A: Data collection Instrument Questionnaire ..... - 65 -

Acronyms and Abbreviations

CI/CD	Continuous Integration / Continuous Deployment
CRISP-DM	Cross-Industry Standard Process for Data Mining
DAD	Disciplined Agile Delivery
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
LeSS	Large-Scale Scrum
PDCA	Plan–Do–Check–Act
PLC	Project Life Cycle
QA	Quality Assurance
SAFe	Scaled Agile Framework
SDLC	Software Development Life Cycle
SPSS	Statistical Package for the Social Sciences
TDD	Test-Driven Development
UAT	User Acceptance Testing

List of figures

Figure 2-1: Agile SDLC ..... - 10 -
Figure 2-2: scrum methodology, *Adapted from [6]* ..... - 12 -
Figure 2-3: Phases of SDLC and PLC ..... - 22 -
Figure 4-1: PDCA Cycle ..... - 32 -
Figure 4-2: CRISP-DM phases ..... - 33 -
Figure 4-3: Conceptual Framework of Proposed Solution ..... - 34 -
Figure 5-1: Gender of respondent ..... - 42 -

List of tables

Table 2-1: Summary of Agile QA techniques ..... - 17 -
Table 4-1: Application of PDCA in Scrum process ..... - 33 -
Table 4-2: Integration of PDCA with CRISP-DM ..... - 36 -
Table 4-3: SDCL stages mapping on Scrum QA checkpoints ..... - 37 -
Table 4-4: PLC stages QA checkpoints ..... - 38 -
Table 4-5: Key Scrum Checkpoints and QA Optimization Strategies ..... - 39 -
Table 4-6: Mapping of Scrum checkpoints ..... - 40 -
Table 5-1: Age of Respondents ..... - 42 -
Table 5-2 : Education status ..... - 43 -
Table 5-3: cross table of role in agile project with work experience ..... - 43 -
Table 5-4: size of Scrum team with duration of a Scrum sprint Cross table ..... - 44 -
Table 5-5: Respondent indication on benefit of adopting agile ..... - 45 -
Table 5-6: Respondent indication on success of agile projects ..... - 45 -
Table 5-7: Correlation Product Vision Processes Vs Project Success ..... - 48 -
Table 5-8: Correlation Product Backlog Processes Vs Project Success ..... - 49 -
Table 5-9: Correlation Release Backlog Planning Processes Vs Project Success ..... - 50 -
Table 5-10: Correlation Sprint Backlog Processes Vs Project Success ..... - 51 -
Table 5-11: Correlation Product Delivery Processes Vs Project Success ..... - 53 -

Chapter One

1. Introduction

1.1 Background

Software development has undergone a significant transformation over the past two decades, with the rise of agile methodologies such as Scrum, Kanban, and Extreme Programming (XP) [1], [2], [3].These iterative and incremental approaches have gained widespread adoption in the industry due to their ability to deliver working software more efficiently compared to traditional waterfall models [4]. Agile methodologies emphasize rapid prototyping, continuous feedback, and adaptability, enabling software teams to respond to changing requirements and market demands in a more nimble and effective manner [1], [2], [5].

Despite the numerous benefits of agile development, the integration of comprehensive quality assurance (QA) practices within this framework remains a challenge [6]. Agile's focus on speed and flexibility can sometimes come at the expense of thorough testing and quality control measures [1]. This can lead to the introduction of defects, technical debt, and increased maintenance costs as the software evolves [5].

However, QA on agile are more difficult especially those whose experience lies with more traditional and system development life cycle controls, as they struggle with how to perform on agile projects [1]. Project Quality Assurance throughout the project life cycle helps to improve the process to ensure product quality.

This study aims to investigate the practice of QA using data analysis techniques to enhance software quality assurance in agile projects.

1.2 Motivation

The widespread adoption of agile methodologies, particularly Scrum, has revolutionized the way software development teams approach project management, prioritizing speed and flexibility. However, as organizations race to meet deadlines and deliver software in rapid iterations, they often face challenges in ensuring consistent software quality. Although agile methods prioritize rapid delivery and high product quality, many projects still face recurring challenges such as defects, accumulation of technical debt, and inconsistent quality outcomes.

A significant challenge faced by teams is the difficulty in integrating robust QA practices into the fast-paced, iterative agile process. With frequent changes in scope, shifting priorities, and limited time for testing, QA can often become reactive rather than proactive. This leads to a higher risk of defects slipping through the cracks and causing delays in later stages of development. Additionally, teams struggle with identifying the right moments within the development cycle to implement effective quality checks, resulting in inconsistent quality across releases.

This research is motivated by the need to address persistent quality assurance challenges within agile frameworks. The need to identify key checkpoints within the SDLC and PLC that can ensure continuous and effective QA is critical. By employing data analysis techniques to uncover patterns and insights, this study goal is to provide practical, data-driven solutions that will help agile teams deliver software that meets high-quality standards, reduces defects, and improves long-term project sustainability.

1.3 Statement of the Problem

Agile software development methodologies have become the preferred approach for addressing the need for faster delivery and greater adaptability in the dynamic software industry. Dingsøyr *et al.* [7] highlight that the adoption of agile methods surged after the COVID-19 pandemic, increasing from 37% to 84% in software development and from 26% to 60% in IT departments, as reported in the 15th Annual State of Agile Report [8]. Notably, agile practices are extending beyond IT departments into sectors like finance, human resources, and marketing, where adoption rates have doubled.

Agile methodologies emphasize user feedback throughout the SDLC, favoring flexibility and faster delivery over exhaustive documentation [2]. They are particularly effective in managing unstable requirements during all development phases. However, despite their advantages, agile projects often struggle to meet quality objectives. For instance, only 45% of adopters report a positive impact on software quality despite enhanced quality being a primary reason for adopting agile [8]. This raises concerns about the integration of effective Quality Assurance (QA) practices within agile frameworks.

QA is inherently proactive, aiming to establish methods, processes, and standards that ensure quality from the outset [7], [8], [9]. It involves systematic activities such as process monitoring, testing, and continuous improvement to prevent defects. However, current QA approaches often

rely on manual, experience-based decision-making or generic techniques that are not well-suited to the unique challenges of agile environments [9]. These approaches, while historically effective, struggle to adapt to agile's iterative and people-centric nature, where traditional QA checkpoints often conflict with the flexibility and speed required in agile projects.

Traditional QA approaches face significant limitations in agile environments due to their reliance on linear processes, such as stage-gate reviews and detailed assessments, which are incompatible with agile's iterative and flexible nature [2]. These methods also lack adaptability to overlapping milestones and iterative cycles, making them less effective in maintaining quality within dynamic agile workflows [10]. Similarly, agile practices like Test-Driven Development (TDD), Acceptance Testing, and Continuous Integration (CI) come with their own challenges [11]. TDD demands a significant upfront investment in writing tests, making it difficult for developers to balance comprehensive coverage with tight deadlines [11]. Acceptance Testing heavily depends on clear and stable requirements, which are often unavailable in agile projects due to their evolving nature [12]. Continuous Integration, while essential, requires a robust infrastructure and toolset, which can be resource-intensive to establish and maintain, limiting its adoption in resource-constrained environments [10], [11].

Traditional Software Quality Assurance Plans (SQAPs) also present challenges in agile settings. These plans are often process-heavy and linear, lacking the agility to integrate seamlessly into Scrum workflows. They fail to address the proactive integration of QA measures within agile processes, such as sprint cycles, daily stand-ups, and sprint reviews, reducing their overall effectiveness [13].

In contrast, a data-driven approach to QA can address these limitations by leveraging insights from quantitative data to improve decision-making and quality assurance practices. By analyzing data at critical Scrum checkpoints such as sprint planning, daily stand-ups, and sprint reviews teams can identify potential issues early, track progress accurately, and make informed decisions to align deliverables with quality standards[10], [11], [14]. This approach focuses on iterative feedback loops and analytics to introduce a systematic and proactive framework for QA in agile development. By employing data analysis techniques, this research aims to identify key QA checkpoints within the SDLC and PLC. The ultimate goal is to adapt and show a conceptual

framework that enhances QA practices in agile methodologies, improving software quality and increasing the success rate of agile projects.

1.4 Objectives

General Objective:

The general objective of this research is to develop data-driven quality assurance approach for agile software projects and to optimize Scrum checkpoints through a conceptual framework.

Specific Objectives:

- To perform a comprehensive literature review to explore existing quality assurance practices, frameworks, and challenges in agile software development, with a particular focus on the Scrum methodology.
- To identify and define critical quality assurance checkpoints within the SDLC and PLC specific to agile projects, particularly within the Scrum framework.
- To develop a conceptual framework that integrates data-driven quality assurance practices into agile software development, tailored to Scrum methodology.
- To collect data and analyze current quality assurance practices in agile projects to identify challenges and limitations faced by teams when integrating effective QA methods.
- To evaluate the impact of implementing systematic quality assurance checkpoints on the overall success and quality of agile software development projects through quantitative and qualitative methods.
- To propose recommendations for software development companies on adopting data-driven quality assurance practices to enhance software quality and project success.

1.5 Methods

1.5.1 Research Design

This study employed a **mixed-methods research approach** to investigate and enhance quality assurance (QA) practices in Scrum-based agile software development. The mixed-methods design integrates both quantitative and qualitative techniques to provide a comprehensive understanding of the problem. Quantitative data were collected through a structured questionnaire administered to agile practitioners. Qualitative insights were obtained through an extensive literature review and expert consultations during the design and validation of the proposed conceptual framework.

The quantitative component involved collecting structured data through a Likert-scale questionnaire and analyzing it using statistical tools. The qualitative component focused on identifying best practices and challenges in agile QA through a detailed review of scholarly literature and domain-specific standards (e.g., PDCA, CRISP-DM, ISO/IEC 25010, and IEEE 730). Additionally, feedback from software engineering experts was used to guide the formulation and refinement of the framework and the questionnaire itself.

The research process was structured into multiple phases: literature synthesis, primary data collection, statistical analysis, framework development, and framework evaluation. Each phase built upon the previous to ensure the development of a robust and practical QA model for agile environments.

1.5.2 Data Collection Method

Data collection Tool

The research will target software development companies using agile methodologies, specifically Scrum. By focusing on different project sizes, aim to understand how QA practices scale in various agile environments.

For each company, Data collection takes place for project managers, Scrum Masters, QA specialists, and developers. The research will also analyze ongoing agile projects per company to identify key QA practices and optimize checkpoints in Scrum-based development.

A structured questionnaire was used as the primary data collection instrument. The questionnaire was designed to measure the implementation of QA practices across different Scrum processes and assess their impact on project success. It consisted of 32 closed-ended items using a 5-point Likert scale (1 = Strongly Disagree, 5 = Strongly Agree), grouped under five categories: Product Vision, Product Backlog, Release Planning, Sprint Backlog, and Product Delivery.

The questionnaire was developed using Google Forms and distributed through both online and printed formats to maximize accessibility. Expert reviews and pre-testing were conducted to enhance the clarity, relevance, and reliability of the questions.

The questionnaire items were derived from relevant QA literature and agile standards, ensuring alignment with theoretical models such as PDCA and CRISP-DM. Experts in agile project

management and software quality reviewed the instrument to confirm its content validity. A pilot test was conducted prior to full deployment, and necessary revisions were incorporated.

The internal consistency of the instrument was assessed using Cronbach's alpha, Cronbach's alpha coefficient was used to test the reliability of the questionnaire and was calculated for the five audit processes [15], with all five thematic areas **scoring above 0.7**, confirming acceptable reliability.

Sample Method and sample Size

Sampling is a procedure of selecting some part of the population to represent the entire population [54]. Sample is some portion of a population selected to participate in a study. There are different type of sampling methods and techniques.

In this study, non-probability sampling techniques were used. Among the non-probability sampling, purposive sampling technique was applied to select the eleven software companies. Purposive sampling which is preferable to take out vital information from concerned body employed systematically for the achievement of the researches objective. In this sampling, the researcher intentionally selected companies to understand more about the core phenomena of the issue under investigation mainly through getting information from resourceful companies or informants.

In addition, Convenience sample, 8 of them were project managers, 8 Scrum Masters of software companies, 5 of them were QA specialists and the remaining 26 were software developers.

Sampling Criteria

In this research, the following criteria were applied:

Inclusion/ exclusion criteria:

- Only companies that have implemented Scrum for at least one full project cycle be considered
- Companies working on projects of varying sizes (small, medium, and large)

For this thesis 11 companies were selected by purposive sampling method. 8 were selected by excluding based on criteria. Among those 8 software companies who have experience and have willingness to participate in this project 51 personnel answer the Questioner.

1.5.3 Data Analysis Techniques

Quantitative data collected from the survey were analyzed using SPSS software. Descriptive statistics were used to summarize the participants' responses, while Pearson correlation analysis was conducted to examine the relationships between quality assurance (QA) practices and perceived project success. The analysis focused on several key areas, including the frequency and placement of QA activities during different Scrum phases, the relationship between specific QA checkpoints and project performance, and the identification of patterns that support the applicability of the proposed QA framework. To better interpret and contextualize the quantitative findings, qualitative insights from existing literature were also incorporated. These insights helped guide the mapping of QA checkpoints and provided a deeper understanding of their role in agile project environments.

1.5.4 Framework Development and Evaluation

The conceptual framework was developed by synthesizing findings from the literature and empirical data. Specifically, the framework integrates the PDCA cycle for continuous process improvement and the CRISP-DM methodology for data-driven QA decision-making. It proposes the mapping of QA checkpoints across the Scrum process, SDLC, and PLC stages.

To evaluate the framework, the researcher interpreted the data in light of key performance indicators such as defect detection effectiveness, backlog prioritization accuracy, and overall sprint quality. The alignment between theoretical constructs and statistical results supports the feasibility and relevance of the framework.

1.6 Scope and Limitations

1.6.1 Scope:

This study focuses on enhancing SQA in agile software development projects, particularly within the Scrum framework. The research aims to identify critical QA checkpoints within the SDLC and PLC and to develop a conceptual framework for improving the effectiveness of quality assurance in agile methodologies. The scope of the study covers agile software projects in IT industries that follow Scrum, with a focus on both the quality of the development process and the quality of the final software product.

1.6.2 Limitations:

Potential limitations of the study include restricted access to historical project data, which may result in a smaller sample size for the quantitative analysis. Additionally, the variability in agile practices across different organizations could affect the generalizability of the findings.

1.7 Significance

This study is significant as it addresses the gap in effective QA practices within agile software development, specifically in the Scrum framework. By identifying critical QA checkpoints in the SDLC and PLC, the research offers a proactive approach to improving software quality. The findings can help agile teams reduce defects, enhance product quality, and improve project success rates. Additionally, the study provides valuable insights for both industry practitioners and academic researchers, contributing to more efficient and standardized QA processes in agile environments.

1.8 Organization of Thesis

This study investigates the practice of quality assurance (QA) using data analysis techniques to enhance software quality in agile projects. It focuses on developing and validating a conceptual framework for QA within agile methodologies. The thesis is organized into seven chapters. Chapter 2 provides the background of the study and offers a detailed review of the literature related to the research problem. Chapter 3 presents an overview of related works, highlighting previous studies and frameworks in the field of software quality assurance in agile environments. Chapter 4 introduces and explains the conceptual framework in detail, demonstrating how it addresses the research objectives. Chapter 5 presents a comprehensive discussion and analysis of the results obtained from the study. Finally, Chapter 6 concludes the thesis by summarizing the key findings and offering recommendations for future research and practical implementation.

Chapter Two

2. Literature Review

2.1 Introduction

This Chapter describes about the background related to the problem in detail and also provides the literature related to the problem so that this literature can be used for further analysis in future to gain more knowledge over the problem. This chapter also explains about the literature related to the methodology of the problem. With the purpose of literature to get understanding and theoretical background of Agile Software development methodology, researcher accessed different journals, conference papers and articles using the following key words. Agile software development, Agile Practices, Challenges in Agile Software Development, popular-agile-software development technologies, Factors that Significantly Impact Agile practices, benefits and challenges of agile method, Factors that Significantly Impact the Implementation of an Agile Software Development Methodology.

2.2 Conceptual and Theoretical Framework definitions

A project is a temporary endeavor undertaken to create a unique product, service, or result. Operations, on the other hand, is work done in organizations to sustain the business [20]. So, IT projects involve using hardware, software, and networks to create a product, service, or result [20].

PLC focuses on the project management phases that maps the life of a project from the beginning to its end, processes (i.e. Define Project Goal, Plan Project, Execute Project Plan, Close Project, and Evaluate Project), knowledge, skills, tools and techniques for effective managing a project. And SDLC focuses on the software engineering phases (i.e. Planning, Analysis, Design, Implementation, and Maintenance & Support), processes, tools and techniques for building and/or implementing IT solutions [20], [21].

There are several SDLC models like agile, waterfall, RAD, spiral which has their own advantages. SDLC is a framework that describes the activities performed at each stage of a software development life cycle [22]. It depends on the various applications to choose the specific model. In this paper, however, we will study the agile processes and its methodologies.

2.2.1 Agile Methodology

Agile process is an iterative approach in which customer satisfaction is at highest priority as the customer has direct involvement in evaluating the software [23]. The agile process follows the SDLC which includes requirements gathering, analysis, design, coding, testing and delivers partially implemented software and waits for the customer feedback [2]. Figure 2-1 shows the SDLC of Agile Process.



Figure 2-1: Agile SDLC

Agile methodologies are group of software development method that all are based on incremental and iterative development. There are four major characteristics that are main for all agile metrologies are: adaptive planning, iterative and incremental development, short and flexible response for change and highly promote communication.

The Agile Manifesto [24] defines Agrees for Agile software development in 2001 by setting up meeting and signed by the entire seventeen participants. The manifesto Declares: “We are revealing better ways of developing software by doing it and helping others to do it”. The twelve principles behind the agile manifesto also present the characteristics of Agile Software Development Methods (ASDMs). The Principles of agile manifesto presented by [24] stated following:

1. Our highest priority is to satisfy the customer through early and continuous Delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes tackle change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a reference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximizing the amount of work not done is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

As stated by [20], [25] there are various models available for agile process which helps in the development of the software. For example, Extreme Programming (XP), Scrum, Feature Driven Development (FDD) and others. But in this paper, we focus only on Scrum methodology.

2.2.2 Agile Scrum

Scrum is another popular agile method for the development of software. Scrum is different from other methods due to the idea of practical experience rather than theories that is known as "empirical process control" [9]. Agile scrum methodology is an agile development method which concentrates specifically on how to manage tasks within a team based development environment. As shown in Figure 2-2, a product owner creates a prioritized wish list called a product backlog. During sprint planning, the team pulls a small chunk from the top of that wish list, a sprint backlog, and decides how to implement those pieces. The team has a certain amount of time—a sprint (usually 2 to 4 weeks)—to complete its work, but it meets each day to assess its progress (daily Scrum). The sprint ends with a sprint review and retrospective [6].

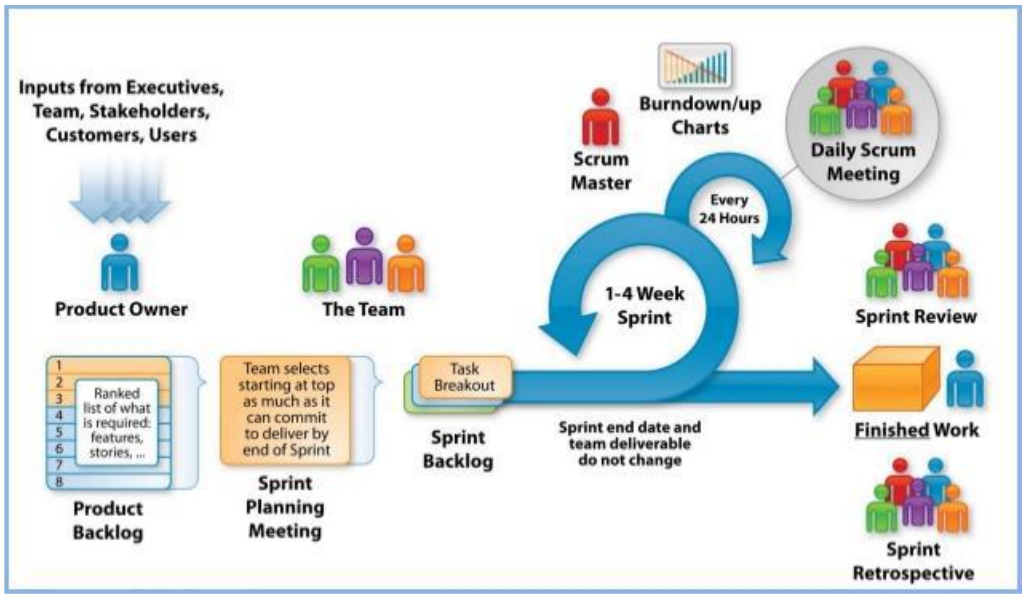

Figure 2-2: scrum methodology, *Adapted from [6]*

As Mkoba and Marnewick [15] Scrum framework comprises of the following processes:

- **Product Vision:** describe the vision for the product which is aligned with the organization's business needs.
- **Release Planning:** The purpose of a release planning is to establish the schedule for the product releases. The meeting is conducted to determine the length of the iterations and sprint meetings. This includes release plans for multiple sprints.
- **Product Backlog:** The list of the product requirements (user stories) are included in the product backlog. The user stories are selected for development based on the priority levels assigned by the customer. Items listed in the product backlog support the delivery of the product vision.
- **Sprint Backlog:** Sprint backlog includes the list of requirements selected from the product backlog that are included in a particular sprint. Sprint backlog has sprint (iteration) which comprises of sprint planning meeting, daily scrum meeting, sprint review meeting and sprint retrospectives. The duration of sprint iteration is two weeks to one month which are used to build a potentially shippable product increment to a customer.
- **Shippable Product (Product Deliverable):** At the end of each sprint, the product deliverable that met acceptance criteria is shipped to the customer.

2.2.3 Agile and Software Quality Assurance

As ISO 24765:2017 definition [26], Software Quality is the degree to which a software product meets established requirements; however, quality depends upon the degree to which those established requirements accurately represent stakeholder needs, wants, and expectations.

As IEEE Standard for Software Quality Assurance Processes (IEEE Std 730-2014) definition [27] , SQA is a set of activities that define and assess the adequacy of software processes to provide evidence that establishes confidence that the software processes are appropriate for and produce software products of suitable quality for their intended purposes. A key attribute of SQA is the objectivity of the SQA function with respect to the project. The SQA function may also be organizationally independent of the project; that is, free from technical, managerial, and financial pressures from the project.

As Ampler reported that [11], software quality can be met in agile methods through the development of test. Because there is strong pressure to develop test cases before writing code in agile improvement, it seems logical to agree with Ampler that "quality is an inherent factor of agile" [11]. Testing and quality are interconnected because the goal of testing is to find the bugs before delivering the product, which gives the chance to the developer to ensure the quality of the product by rectifying the bugs [28]. From an agile view, quality has been identified by McBreen [29] as the ability to respond to alterations in development software. Which means that much the regularly delivery of tested, running, and client-certified software program at the cease about every new release is a vital issue of agile quality assurance [29]. Quality assurance (QA) is a method for avoiding errors or defects during developing software and staying away from issues while delivering solutions or services to clients; such as ISO 9000 which characterizes as a component of value administration concentrated on offering confidence that quality necessities will be fulfilled. This bug prevention in quality confirmation varies carefully from bug detection and dismissal in quality control and has been alluded to as a move left as it concentrates on quality earlier in the procedure [30].

As it defined above Scrum is one of the agile approaches, which can be used as a solution for quality issues [31] since it has long been acknowledged as the most prevalent agile method, and the interest in Scrum is increasing over. Scrum works well with complex projects [31]. To deal with complexity using the Scrum, the quality is considered as the main concern while developing

a project. Jeon *et al.*[32] further mentioned that despite the known advantages of Scrum, it has three disadvantages: First, the backlogs of Scrum focus only on functional features, making it difficult to effectively reflect the quality attributes; second, these backlogs focus on implementing functional backlogs without a traceability analysis of the relationships among them; third, current agile methods are focused on individual teams or project needs, making it difficult to use the method in organizations with multiple cooperating teams. Therefore, it is important to address the quality issues in Scrum. The main backbones of the Scrum are its iterative process and continuous feedback. Li *et al.*[33] observed that continuous daily feedback received during sprint retrospective meetings proceed with more focus on software quality.

2.3 Current Practices and Challenges in Agile QA

SQA plays a vital role in ensuring the success of software products. To understand the practice of SQA within Agile methodologies, it is essential to grasp the concepts of software quality, quality assurance, and the SDLC. Agile practices inherently support QA by embedding quality assurance processes within the development cycle. Scholars have identified and evaluated specific Agile quality assurance techniques, including refactoring, test-driven development (TDD), acceptance testing, continuous integration (CI), pair programming, face-to-face communication, on-site customer engagement, frequent customer feedback, and system metaphors [11]. These practices, while effective, present unique implementation challenges that are worth examining.

Refactoring is a technique used to improve the internal structure of code without altering its external functionality. It enhances code readability, reduces technical debt, and facilitates future modifications. Refactoring is implemented iteratively, particularly after adding new features or resolving bugs. However, it demands a high skill level to avoid introducing new defects. Tight deadlines often limit the scope of refactoring, and poorly written code can make the process more challenging [11], [34], [35].

Test-Driven Development (TDD) involves writing automated tests before coding, ensuring that the code meets predefined requirements. Developers iteratively write a test, implement the necessary code, and refactor as needed. While TDD promotes a robust development process, it has a steep learning curve for developers unfamiliar with the technique. Additionally, it requires significant time to write and maintain tests, and its effectiveness heavily depends on the availability of reliable tools [11], [36].

Acceptance testing ensures that the software meets user expectations and business requirements. This process involves creating test cases based on user stories and acceptance criteria and is often conducted during sprint reviews or continuous delivery pipelines. However, translating requirements into effective tests is complex. Collaboration between QA teams and stakeholders can be hindered by communication barriers, and unclear acceptance criteria may lead to ineffective testing [37].

Continuous Integration (CI) integrates code changes into a shared repository regularly, followed by automated builds and testing. This practice allows teams to identify and resolve issues promptly. Developers commit code frequently, often daily, ensuring that integration issues are minimal. However, CI requires a disciplined approach from the team, and setting up and maintaining CI infrastructure can be resource-intensive. Automated tests also need to be reliable to catch errors efficiently [10], [37], [38].

Pair programming is a collaborative coding technique where two developers work together on the same task. One developer writes the code while the other reviews it, fostering knowledge sharing and code quality improvement. Despite its advantages, some developers may resist this practice due to discomfort with collaboration. The perceived inefficiency of allocating two resources to a single task and the challenge of finding effective pairings can also hinder its adoption [37], [39].

Face-to-face communication is central to agile methodologies, enabling quick decision-making and clarity among team members. Regular stand-up meetings, planning sessions, and ad-hoc discussions facilitate direct communication. However, geographically distributed teams face challenges in maintaining real-time communication. Additionally, cultural and language differences may lead to misunderstandings [37, 40].

On-site customer engagement ensures that customer representatives are available for quick decision-making and clarifications. Customers participate in daily stand-ups and sprint planning sessions to provide real-time feedback. However, their continuous availability is often impractical, and misalignment between business and technical priorities can slow down progress [37].

Frequent customer feedback ensures that the product aligns with evolving user needs and expectations. Agile teams collect feedback during sprint reviews, demos, and user testing sessions.

However, delayed feedback can disrupt development timelines, and changing customer priorities often necessitate rework, impacting the overall progress [37], [41].

System metaphors are used to create a shared understanding of the system's architecture and functionality. These metaphors are often employed during design discussions to align the team on project goals. However, abstract metaphors may lead to varied interpretations among team members, and not all individuals may be comfortable using them as design tools [37].

In table below the summarized understanding of these practices and addressing their challenges are placed.

Table 2-1: Summary of Agile QA techniques

Technique	Description	Challenges
Refactoring	Make small changes to code, Code behavior must not be affected, Resulting code is of higher quality	• Requires skilled developers to identify and implement effective refactoring without introducing defects • Lack of sufficient time allocation due to tight deadlines can hinder its adoption. • Poor initial code quality makes refactoring efforts complex and time-consuming.
Test-driven development	Create a test, Run the test, Make changes until the test passes	• High learning curve for developers new to TDD • Requires substantial time to write and maintain test cases, which might delay delivery in fast-paced projects. • Inadequate tool support in some environments.
Acceptance testing	Quality assurance test done on a finished system, Usually involves the users, sponsors, customer, etc.	• Difficulty in translating user requirements into comprehensive acceptance tests • Collaboration barriers between QA teams and business stakeholders. • Lack of clear acceptance criteria can lead to ambiguous test cases.

Continuous integration	Done on a daily basis after developing a number of user stories. Implemented requirements are integrated and tested to verify them. This is an important quality feature.	• Requires consistent discipline to integrate code frequently. • Infrastructure setup and maintenance for CI tools can be resource-intensive • Ensuring automated tests are comprehensive and reliable is challenging.
Pair programming	Two developers work together in turns on one PC, Bugs are identified as they occur, Hence the product is of a higher quality	• Resistance from developers uncomfortable with collaborative coding • Higher perceived costs due to two developers working on the same task. • Requires effective pairing to ensure productivity and avoid conflicts.
Face-to-face communication	Preferred way of exchanging information about a project as opposed to use of telephone, email, etc. Implemented in the form of daily stand-up meetings of not more than twenty minutes. This is similar to the daily Scrum in the Scrum method. It brings accountability to the work in progress, which vital for quality assurance.	• Geographically distributed teams face challenges in achieving effective face-to-face communication • Cultural and language barriers can complicate understanding.
On-site customer	A customer who is a member of the development team, Responsible for clarifying requirements	• Availability of a customer representative throughout the project lifecycle is often impractical

		• Misalignment of technical and business perspectives can delay decisions.
Frequent customer feedback	Each time there is a release the customer gives feedback on the system, and result is to improve the system to be more relevant to needs of the customer. Quality is in fact meeting customer requirements.	• Customers may not always provide timely or actionable feedback. • Feedback loops can be disrupted by changing customer priorities
System metaphor	Simple story of how the system works. Simplifies the discussion about the system between customer/ stakeholder/ user and the developer into a nontechnical format. Simplicity is a key to quality	• Abstract metaphors can lead to misinterpretations among team members. • Not all teams are familiar with using metaphors as a guiding principle in system design

To solve the challenges on around the current practices Data-driven approaches offer powerful solutions to address the challenges associated with Agile QA techniques. By leveraging analytics, predictive models, and automation, Agile teams can enhance decision-making, improve process efficiency, and achieve higher quality outcomes in software development [10], [11], [14]. The use of data-driven strategies provides teams with actionable insights that help in identifying inefficiencies and quality bottlenecks early in the development cycle. Furthermore, by continuously monitoring metrics and utilizing historical data, teams can proactively address issues and optimize workflows, ultimately leading to improved software quality and faster delivery times. Incorporating data analysis into the Agile QA process not only enhances the effectiveness of the techniques but also enables a more adaptable and responsive development environment.

2.4 Data-Driven Approaches in Quality Assurance for Agile Environments

Data Analysis in Software Development allows businesses to make more informed decisions, create data-driven applications, and deliver enhanced user experiences. Analytics can help answer important questions mangers ask about their projects. The goal of analytics to assist decision makers in extracting important information and in-sights from data sets that would otherwise be hidden [42].

Data-driven techniques have gained prominence in recent years for improving software quality assurance (QA), particularly in agile environments. The integration of data analytics into QA processes allows for more precise decision-making, real-time feedback, and the ability to anticipate potential issues. This literature review examines the role of data-driven approaches, their application in agile methodologies, and their advantages over traditional QA practices.

2.4.1 The Role of Data-Driven Techniques in QA

Data-driven techniques leverage quantitative and qualitative data to enhance the QA process. These methods involve using metrics, historical data, and predictive analytics to monitor and improve software quality. The biggest contribution of using Data-driven approach is the Real-time Monitoring and Feedback. Continuous integration and continuous deployment (CI/CD) pipelines generate vast amounts of data. Techniques like predictive analysis are used to predict defects and their root causes. Metrics like defect density and test coverage guide resource allocation.

Defect density, defined as the number of defects per unit of code (e.g., per KLOC), enables agile teams to identify modules with high defect trends and prioritize them for rework or intensified testing [10], [14], [37]. In contrast, test coverage measures the proportion of code or functionalities tested by automated or manual test cases. High test coverage indicates broader testing scope and lower risk of undetected defects, while low coverage flags areas needing attention [11], [13], [17].

When used together, these metrics allow project managers and QA leaders to make data-driven decisions about where to focus limited resources whether by assigning additional testers, increasing test automation, or refactoring risky modules. As Bhanushali [10] and Wakili et al. [14] explain, continuous monitoring of test coverage and defect metrics in agile cycles enables real-time quality tracking, often supported by tools such as SonarQube or Jenkins integrated into CI/CD pipelines. Moreover, Martinez-Fernández et al. [17] emphasize that these metrics are essential

components of software analytics frameworks used to continuously assess and improve software quality. Wong et al. [13] also highlight the importance of embedding such QA checkpoints across both the SDLC and the PLC, ensuring continuous improvement through iterative QA feedback loops. Ultimately, these metrics support proactive risk management and contribute to more consistent product quality and sprint success.

2.4.2 Relevance to Agile Environments

Agile methodologies emphasize adaptability, continuous improvement, and iterative development, making data-driven quality assurance (QA) highly compatible with agile principles [7]. One key advantage is enhanced collaboration. Shared dashboards and real-time analytics promote transparency and foster effective communication among cross-functional teams an essential component of agile workflows. Additionally, data-driven QA enables faster feedback loops by reducing the time between defect detection and resolution, which is critical in the short and frequent iterations typical of agile projects. As agile initiatives scale, maintaining consistent quality across multiple teams and iterations becomes increasingly complex. Data-driven approaches help address this challenge by standardizing QA practices and ensuring uniform performance evaluation, thereby supporting scalability and long-term project success.

2.4.3 Comparative Analysis: Data-Driven vs. Traditional QA

Traditional QA relies on predefined test cases, often leading to redundant testing. Data-driven approaches focus on high-risk areas, optimizing resource use. A study by Wong et al. [13] showed a 30% reduction in defect rates using predictive analytics in agile projects. Manual methods are prone to human error. In contrast, data-driven techniques use empirical evidence, improving reliability. On the Adaptability side Traditional methods struggle to keep up with dynamic requirements, whereas data-driven approaches thrive on continuous updates and feedback. According to a survey by Gartner [44], 70% of organizations using data-driven QA reported faster time-to-market compared to traditional methods.

Data-driven techniques represent a paradigm shift in software QA, particularly in agile environments. By utilizing data and analytics, these methods overcome the limitations of traditional QA approaches, leading to improved efficiency, accuracy, and scalability

2.5 QA Checkpoints in SDLC and PLC

This section examines critical QA checkpoints within the SDLC and PLC, their relationship with QA, and the scientific frameworks underpinning their implementation.

2.5.1 Defining SDLC and PLC

SDLC provides a systematic approach to software development, enhancing reliability and reducing costs [45]. Standards such as ISO/IEC 12207:2017 [45] outline lifecycle processes to improve consistency and efficiency.

The PLC, in contrast, as Project Management Body of Knowledge (PMBOK) [26] encompasses the broader spectrum of activities required to deliver a project from initiation to closure. As shown in figure 2-3 below, SDLC focuses on software-specific tasks, PLC integrates these into the larger context of project management, addressing resource allocation, risk management, and stakeholder

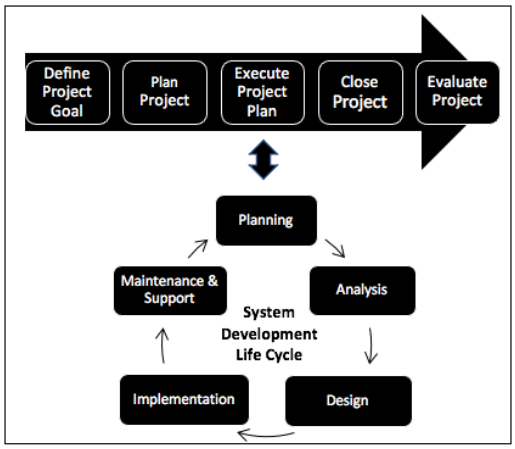

Figure 2-3: Phases of SDLC and PLC

engagement. Frameworks such as the PMBOK guide [46] PLC practices to achieve project objectives effectively.

2.5.2 Checkpoints within the PLC and SDLC

QA checkpoints are strategic stages in SDLC and PLC where quality assurance activities are performed to evaluate deliverables against predefined standards. These checkpoints serve as quality gates, identifying and resolving issues early to minimize rework and ensure alignment with project requirements. For example, during the design phase of SDLC, a QA checkpoint might involve reviewing design documentation for completeness and technical feasibility [47]. Similarly, in the execution phase of PLC, QA audits may verify that deliverables conform to quality standards and project scope [48].

Agile Scrum has five processes such as product vision, product backlog, release planning, spring backlog and potentially shippable product to customer. In Agile SDLC, checkpoints align with sprint reviews (e.g., backlog grooming, sprint demos) to validate incremental deliverables. For example, A design sprint checkpoint assesses user story feasibility before coding begins [13].Also Agile PLC uses daily stand-ups and retrospectives as informal checkpoints to adjust priorities and improve processes. Agile PLC leverages daily stand-ups and retrospectives as dynamic checkpoints to adapt priorities and improve processes [49]. Brown *et al.* [50] research highlights that Agile's frequent checkpoints reduce defect rates by 40% compared to traditional methods. For QA, this translates to early defect detection through automated testing in CI/CD pipelines, scope control via sprint planning, and risk mitigation via regression testing. A case study by TechSolutions Inc. [51] demonstrated that combining PLC's release planning checkpoints (ensuring business-goal alignment) with SDLC's User Acceptance Testing (UAT) checkpoints (validating functionality) improved project success rates by 25%. These strategies collectively ensure quality while accommodating Agile's flexibility, making checkpoints indispensable for modern project and software delivery.

The integration of QA checkpoints enhances the traceability and accountability of deliverables, fostering collaboration among teams and aligning outputs with organizational goals. Research indicates that QA interventions at critical points can reduce defect propagation and improve overall project efficiency [13].

2.5.3 Research on Critical QA Interventions

Recent studies underscore the importance of QA interventions at key phases in the SDLC and PLC. For example, requirement validation during the early stages of SDLC significantly reduces ambiguities, leading to better-defined project scopes [52]. Similarly, findings demonstrate that incorporating QA in the development and testing phases mitigates risks associated with integration and deployment failures [53].

Empirical evidence supports the use of continuous QA processes in Agile environments. Iterative testing and feedback mechanisms as critical components for adaptive quality management. These interventions align with the principles of continuous integration and deployment, enabling rapid identification and resolution of defects [13], [52].

2.5.4 Frameworks and Guidelines for Effective QA Checkpoints

QA checkpoints in SDLC and PLC are indispensable for ensuring product quality and project success. By strategically embedding QA activities at critical stages, organizations can detect defects early, reduce rework, and align deliverables with business objectives. Leveraging established standards such as ISO/IEC 25010, IEEE 730, and Agile QA frameworks enhances the effectiveness of these checkpoints [35], [36].

To ensure effective implementation of QA checkpoints, organizations can leverage established frameworks and guidelines:

a) **ISO/IEC 25010**: This standard defines a model for evaluating software quality based on attributes such as functionality, reliability, usability, and security. QA checkpoints aligned with these attributes provide comprehensive coverage of quality aspects [48].

b) **IEEE 730 Standard**: The IEEE 730 standard outlines the components of a Software Quality Assurance Plan (SQAP), including guidelines for reviews, audits, and defect tracking. It provides a robust foundation for formalizing QA processes [27].

c) **Agile Quality Assurance**: Agile methodologies embed QA activities within iterative cycles. Techniques like Test-Driven Development (TDD) and Continuous Integration/Continuous Deployment (CI/CD) emphasize frequent validation, reducing time-to-market while maintaining quality [24]. Studies presented at Agile Alliance conferences illustrate the effectiveness of these techniques in dynamic environments.

d) **Risk-Based Testing**: This approach prioritizes QA activities based on risk assessment. For instance, high-risk functionalities identified during the planning phase are allocated additional resources and rigorous testing [34]. Risk-based testing enhances resource efficiency and reduces the likelihood of critical defects.

e) **Metrics-Driven QA**: Metrics such as defect density, test coverage, and mean time to failure (MTTF) provide quantitative insights into the effectiveness of QA checkpoints. These metrics enable continuous improvement and informed decision-making during the project lifecycle [47].

The literature highlights the integration of Agile methodologies, particularly Scrum, with data-driven QA techniques to enhance software quality. Agile methodologies emphasize adaptability,

iterative development, and customer satisfaction. However, challenges such as limited traceability of quality attributes, lack of scalability for organizations with multiple teams, and difficulty addressing non-functional requirements within Scrum have been identified. These limitations underscore the need for innovative approaches to improve Agile QA practices.

Data-driven approaches, leveraging analytics and predictive models, offer significant potential to address these challenges. They enable real-time monitoring, defect prediction, and efficient resource allocation, aligning seamlessly with Agile's iterative nature. Agile QA techniques such as Test-Driven Development (TDD), Continuous Integration (CI), and iterative testing are essential components of the methodology but face hurdles such as skill demands, resource intensity, and organizational resistance. Scrum, as a widely used Agile framework, supports continuous quality improvement through its iterative and feedback-oriented processes. However, its current practices often overlook quality attributes in backlogs, highlighting the need for optimization.

Embedding QA checkpoints in the Software Development Life Cycle (SDLC) and Project Life Cycle (PLC) ensures early defect detection and alignment with project objectives. These checkpoints are guided by frameworks like ISO/IEC 25010 and IEEE 730 standards, providing systematic approaches to improving software quality.

Chapter Three

3. Related Work

The integration of QA practices into agile software development has become increasingly significant with the widespread adoption of methodologies like Scrum. Traditional QA techniques, designed for linear development models, face challenges when applied to the iterative and adaptive nature of agile frameworks. This section reviews existing studies to identify their approaches, strengths, weaknesses, and lessons learned, providing a foundation for addressing gaps in QA practices within agile methodologies.

Agile-Quality Relationship Studies

This subsection synthesizes foundational research on agile-quality correlations. Abdalhamid *et al.* [1] conducted a systematic mapping study to evaluate the relationship between agile methodologies and software quality. Their findings reveal that agile practices positively influence specific quality attributes, such as flexibility and customer satisfaction, through continuous feedback. However, they also identified persistent challenges, including maintainability and the lack of formal QA techniques. This study highlights a significant gap in empirical evidence, as most conclusions are based on conceptual insights rather than real-world case studies, emphasizing the need for further research into the long-term effects of QA practices in agile settings.

Wakili *et al.* [14] examined how agile practices inherently support QA by embedding quality assurance within the development phase or as supplementary practices. Their analysis highlights the higher frequency of QA activities in agile methodologies compared to waterfall approaches, due to agile's iterative nature. However, they noted that comparing the software quality achieved through agile versus waterfall is challenging, as initial conditions such as cost are not directly comparable. The study also suggests that while agile practices can be tailored to suit team requirements and available time, achieving this balance can be difficult and sometimes unrealistic.

Continuous Quality Practices

The respective papers focuses on automated QA mechanisms in agile environments. Bhanushali [10] delineated CI/CT's role in agile workflows, identifying test environment consistency as a key hurdle despite proposing containerization solutions. Sagheer et al. [16] extended this by theorizing parameter-based QA enhancement frameworks, though lacking scalability proofs.

Bhanushali [10] explored the roles of Continuous Integration (CI) and Continuous Testing (CT) in agile development, emphasizing their importance in ensuring routine code integration, automated testing, and quick feedback. The study identified challenges, such as managing test data, maintaining a consistent test environment, and integrating CI/CT practices into agile workflows. While the paper proposes solutions such as synthetic test data, containerization technologies, and automated tools, it lacks empirical data and real-world case studies to validate these approaches. Additionally, the scalability of the proposed solutions for large, complex projects remains unaddressed.

Sagheer *et al.* [16] analyzed how agile methodologies can be leveraged to ensure software quality. Known for their iterative and incremental approach, agile methods adapt to changing requirements and deliver products efficiently. However, maintaining quality in agile-developed software remains challenging. The paper proposes enhancing QA practices within agile by analyzing various quality parameters, emphasizing the importance of quality assurance activities to ensure high-quality software products. Despite offering valuable theoretical insights, the paper lacks empirical data or real-world applications to validate its framework. Furthermore, it does not address scalability for large, complex projects, leaving a critical gap for future research.

Checkpoint-Based Quality identification and Frameworks

In this section the researchers Examines temporal QA approaches. Wong *et al.* [13] proposed a comprehensive SQAP that aligns with both the PLC and SDLC. Their approach advocates for systematic quality checkpoints across various project phases, such as planning and execution, to improve compliance with quality standards. Their list of checkpoints created at different stages of PLC and SDLC based on the R&A principle from ISO9001:2015 and CMMI. While their findings demonstrate the potential of early and well-defined checkpoints to reduce defects and enhance software quality, the framework's applicability in agile contexts is limited. Notably, it does not address the role of data analysis in determining the placement and frequency of these checkpoints nor explicitly consider agile methodologies like Scrum.

Paper by S.Pargaonkar [18] explores the importance of software quality in architecture design and introduces a survey-based methodology to enhance software quality attributes like Performance, Dependability, and Safety. However, it also sheds light on historical limitations, such as the focus on singular stakeholder views and the delayed generation of detailed insights post-design phase,

emphasizing the need for a more proactive approach to ensure software usability from the outset. Additionally, the document hints at potential challenges with data-driven approaches, suggesting a requirement for substantial historical data for effectiveness, which could pose difficulties for organizations with limited data resources. To address these gaps, organizations should prioritize data accessibility, automate processes for data collection and analysis, foster continuous learning, encourage collaboration among teams, and regularly validate the effectiveness of data-driven quality assurance methods through testing and feedback for continuous improvement and alignment with software quality goals.

Emerging Data-Driven Paradigms

M. Cardarelli *et al.* [19] introduces a data-driven approach, Micro-Quality, aimed at evaluating the quality of micro service architectures. This innovative method emphasizes abstraction, reusable quality attributes, and continuous assessment. By focusing on architecture modeling and leveraging metric providers for automatic computation, Micro-Quality offers a structured way to measure and maintain system quality. However, the complexity of diverse quality assessment tools poses a challenge, making it difficult for developers to efficiently extract actionable insights. To enhance this approach, standardization of quality attributes, scalability of the ecosystem, thorough validation studies, user-friendly interfaces, and community engagement are recommended. These suggestions aim to strengthen the approach's effectiveness, applicability, and adoption within the realm of data-driven quality assurance for micro services. One key lesson learned from the document is the importance of addressing the complexity and diversity of quality assessment tools in the context of evaluating micro service architectures. The limitation highlighted is the potential challenge developer’s face in efficiently extracting actionable insights due to the varied internal quality models of existing frameworks and tools. This underscores the need for standardization, scalability, validation, user-friendly interfaces, and community engagement to overcome such limitations and enhance the effectiveness of data-driven quality assurance approaches for micro services.

The article by Silverio Martínez-Fernández *et al.* [17] explores the use of software analytics tools to improve software quality in Agile Software Development (ASD) environments. Part of the EU-funded Q-Rapids Project under Horizon 2020, the research investigates how integrating Quality Models (QMs) into software analytics tools can enhance real-time quality assessment and drive

continuous improvement. The Q-Rapids Tool, designed for ASD environments, incorporates QMs to provide strategic indicators for assessing both product and process quality. The study involved 22 practitioners from various industries, including telecommunications, public safety, and risk analysis, who evaluated the tool's understandability, reliability, and usefulness. The findings highlighted the importance of clear quality models, real-time data visualizations, customization to fit organizational needs, and improved traceability from high-level quality indicators to detailed metrics. These insights are highly relevant to Data-Driven Quality Assurance in Agile Software Projects, particularly in real-time quality assessment and the integration of QMs to provide actionable insights at Scrum checkpoints, addressing challenges in continuous quality improvement.

Mkoba and Marnewick [15] highlighted the necessity of effective auditing practices tailored to agile methodologies. They observed that traditional auditing methods are ill-suited to the iterative and collaborative nature of agile projects. Their proposed framework emphasizes stakeholder engagement, iterative auditing processes, and the establishment of QA metrics. However, this framework lacks empirical validation and does not sufficiently address the complexities of large-scale agile projects, leaving room for further development.

The paper D. F. Oliveira and M. A. Brito [57] proposes an innovative framework that integrates the PDCA (Plan-Do-Check-Act) cycle with the CRISP-DM methodology to improve quality assurance (QA) and quality control (QC) in deep learning (DL) system development, particularly in data science projects. The paper discusses the challenges associated with ensuring and controlling the quality of deep learning (DL) systems. Given the increasing adoption of DL in various domains, the need for robust quality assurance and control mechanisms is crucial, especially in safety-critical applications.

This dual-layered approach aims to address the unique challenges of DL systems, such as their data-driven nature and lack of transparent development practices, by embedding QA and QC practices into each stage of the CRISP-DM pipeline. The framework is developed using Design Science Research (DSR), which involves six phases: problem identification, definition of objectives, design and development, demonstration, evaluation and validation, and communication of results. Although no specific datasets were used or evaluated in this work, the framework is

theoretical and intended for future application and validation through expert review and case studies.

The strengths of the paper lie in its structured methodology, practical integration of two well-established models (PDCA and CRISP-DM). The framework is structured into two levels, with the primary level focusing on quality assurance and control activities based on the PDCA cycle, and the secondary level comprising the technical activities of developing DL systems as outlined by CRISP-DM, and its focus on a critical gap in current DL engineering practices. However, a key limitation of the study is the lack of empirical validation or real-world application, which restricts the framework's immediate practical relevance. Despite its conceptual strength, the absence of implementation data raises questions about its feasibility and adaptability in industrial environments.

Recent studies highlight a growing need for innovative, data-driven solutions to improve quality assurance (QA) in agile methodologies especially in large-scale or complex software projects, where traditional practices often fall short. This research responds to that need by investigating how data analysis techniques can be used to identify key QA checkpoints and by proposing a scalable conceptual framework tailored for agile environments.

Although various efforts have been made to improve QA in agile, the literature still points to a critical gap: there is no systematic approach for optimizing Scrum checkpoints using data-driven QA methods. Many current practices struggle to fully capture essential quality attributes or to integrate advanced analytics into daily Scrum activities.

To address this, the study introduces a conceptual framework adapted from the PDCA (Plan–Do–Check–Act) cycle and the CRISP-DM (Cross-Industry Standard Process for Data Mining) model. This adapted framework integrates data-driven QA practices directly into the Scrum process, using real-time analytics and predictive techniques to enhance decision-making at each checkpoint. The goal is to improve software quality and make Scrum more proactive, efficient, and suitable for today's fast-paced development environments.

Chapter Four

4. Proposed solution

4.1 Introduction

This chapter delineates the architecture of the proposed framework, designed to bridge gaps identified in existing QA and agile development practices. As a conceptual model in this research combine and adapt the Framework for Quality Assurance and Quality Control designed by D. F. Oliveira and M. A. Brito [57] and work of Elizabeth Mkoba and Carl Marnewick [15] on defining checkpoints on agile scrum framework.

Considering a framework as a set of strategies and actions aimed at solving a specific problem, it enables the optimization of results while ensuring quality and productivity [55], [56]. The strategy used to construct the framework involved first analyzing critical concepts related to the development process. Secondly, it sought to integrate two widely used tools for quality assurance and control in software development: CRISP-DM and PDCA.

4.2 CRISP-DM and PDCA Framework

Based on the strategy of D. F. Oliveira and M. A. Brito [57], the framework was developed and structured into two process steps. The first step corresponds to PDCA activities, which are subsequently applied in the second step, corresponding to CRISP-DM activities.

4.2.1 Step 1: PDCA (Process Improvement & Execution)

The Plan-Do-Check-Act (PDCA) cycle plays a crucial role as a quality management system that facilitates the continuous improvement of processes and products. It ensures the achievement of required goals through a flow of activities designed to adjust deviations occurring in the process. According to Isniah *et al.* [58] PDCA serves as a systematic approach to quality improvement.

PDCA ensures continuous improvement in QA planning, execution, and validation within Scrum. The focus of this first phase is to present all the practical aspects related to quality assurance and control, representing a set of steps based on PDCA activities to be integrated into the Scrum

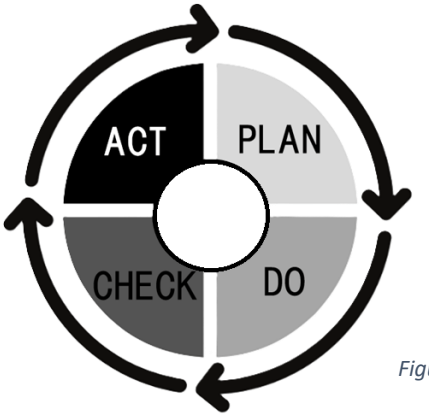

Figure 4-1: PDCA Cycle

process. CRISP-DM is used as a methodology to optimize QA strategies, providing strategic support and continuous improvement.

The PDCA cycle consists of four phases, as shown in figure 4-1. It begins with establishing a plan that outlines objectives and procedures to be implemented in the project. The next phase involves executing these procedures while documenting any problems encountered. The third phase focuses on reviewing these issues, identifying their causes, and analyzing the deviations from expected outcomes. Finally, necessary changes are made to procedures to eliminate or mitigate the identified issues. This retrospective evaluation and improvement approach form the foundation for modern software process improvement techniques.

The integration of the PDCA cycle within the Scrum process enhances quality assurance by ensuring continuous improvement at each stage of development. The table 4-1 below outlines how each phase of PDCA is applied within different Scrum processes to optimize QA strategies, improve defect tracking, and enhance test coverage.

Table 4-1: Application of PDCA in Scrum process

Scrum Process	PDCA Phase	Application in QA
1 Product Vision	Plan	Define QA goals, quality metrics, and risk areas. Identify defect-prone modules.
2 Product Backlog	Plan	Prioritize QA-related backlog items based on historical defect data and test coverage.
3 Release Planning	Do	Implement test plans, automated testing, and defect tracking processes.
4 Sprint Backlog	Check	Track sprint defects, measure test effectiveness, and analyze QA reports.
5Product Deliverables	Act	Optimize test coverage, improve automation, and refine QA strategies for the next sprint.

4.2.2 Step 2: CRISP-DM (Data-Driven Insights & Optimization)

The CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology enhances the QA process by integrating data-driven decision-making. It complements the PDCA cycle by

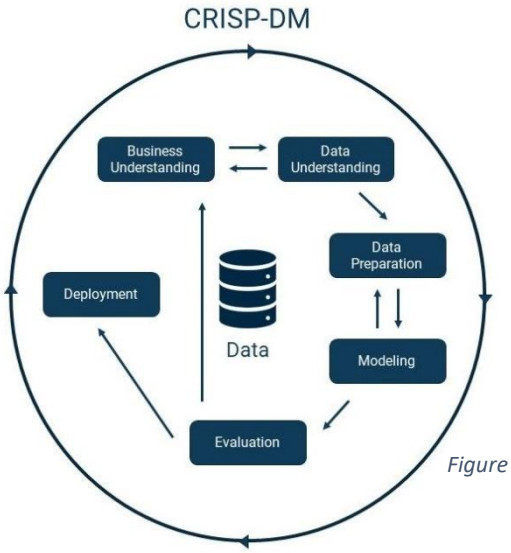

Figure 4-2: CRISP-DM phases

leveraging data analysis to improve defect detection, optimize test coverage, and refine quality assurance strategies.

As shown on figure 4-2, it has 6 phases, In the Business Understanding phase, QA goals are defined based on Scrum metrics and defect history, ensuring a clear direction for quality improvement. The Data Understanding phase involves collecting and analyzing test reports, defect logs, and user feedback to identify patterns and potential issues. Following this, the Data Preparation phase focuses on cleaning and preprocessing data for statistical analysis using tools like SPSS, enabling defect prediction and process optimization.

The Modeling phase applies machine learning or statistical models to predict defect-prone areas, allowing teams to proactively address quality concerns. During the Evaluation phase, insights are validated, test effectiveness is assessed, and comparisons are made against past sprints to measure progress. Finally, in the Deployment phase, findings are implemented into new sprint cycles, automation is improved, and backlog items are refined to ensure continuous improvement in Scrum-based QA processes.

This structured approach ensures that QA strategies are data-driven, continuously optimized, and aligned with agile development goals.

4.3 CRISP-DM + PDCA Model for Scrum QA

The integration of CRISP-DM (Cross Industry Standard Process for Data Mining) and PDCA (Plan–Do–Check–Act) into Scrum-based quality assurance (QA), as shown on diagram below that proposed by D. F. Oliveira and M. A. Brito [57], introduces a structured and data-driven approach

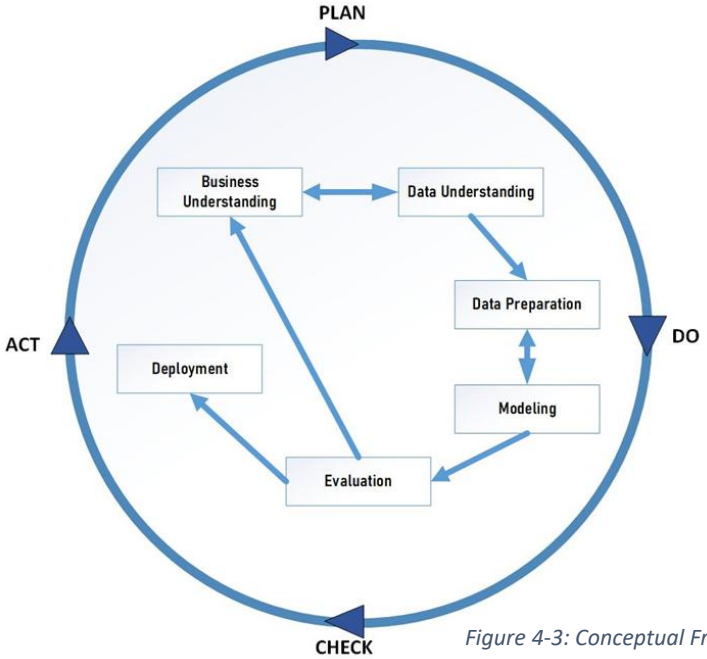

Figure 4-3: Conceptual Framework of Proposed Solution

to continuous improvement. PDCA offers an iterative framework that promotes process refinement through four key stages: Plan, Do, Check, and Act. In the Plan phase, potential improvements are identified and changes are proposed. The Do phase involves the implementation of these changes on a small or pilot scale to minimize risk while collecting data. During the Check phase, the results are analyzed to evaluate the impact of the changes and extract insights. In the Act phase, the successful changes are adopted for wider implementation, while ineffective ones are revised or discarded. This cyclic approach enables ongoing refinement and supports adaptive decision-making.

CRISP-DM complements PDCA by providing a systematic methodology for leveraging data throughout the QA process. The model consists of six phases: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. These stages guide the transformation of raw QA data such as test logs, bug reports, and team performance metrics into actionable knowledge. In a Scrum environment, this analytical structure enhances the team's ability to identify defect patterns, prioritize testing efforts, and make informed decisions about quality risks.

As shown on table 4-2 below when integrated, PDCA serves as the continuous improvement backbone, while CRISP-DM supplies the analytical insights needed to guide improvement decisions. This hybrid model aligns with Scrum's iterative nature and supports the "inspect and adapt" principle by embedding continuous feedback loops into the development process. Each phase of PDCA maps to specific stages of CRISP-DM, allowing quality improvements to be guided by real-time data and empirical results. As a result, Scrum teams benefit from improved defect detection, reduced rework, and higher-quality deliverables, ultimately enhancing sprint performance and product reliability.

Table 4-2: Integration of PDCA with CRISP-DM

Scrum Process	CRISP-DM Phase	PDCA Phase	Application in QA
Product Vision	Business Understanding	Plan	Define quality goals, risk areas, and key Scrum checkpoints for QA.
Product Backlog	Data Understanding	Plan	Collect historical defect data, customer feedback, and test case failures.
Release Planning	Data Preparation, Modeling	Do	Apply defect prediction models, optimize test plans, and allocate QA resources.
Sprint Backlog	Evaluation	Check	Track real-time sprint defects, test coverage, and sprint velocity impact.
Product Deliverables	Deployment	Act	Implement QA improvements, refine testing strategies, and enhance automation.

4.3.1 Checkpoint Mapping for QA across SDLC and PLC Stages in Scrum

To establish a comprehensive and data-driven QA framework in Scrum environments, this study integrates structured checkpoint validation that placed by Mkoba and Marnewick [15] using the PDCA cycle and CRISP-DM model. While Scrum provides iterative checkpoints through its ceremonies (e.g., Sprint Planning, Reviews, and Retrospectives), these can be further optimized by aligning them with the stages of the SDLC and the PLC. This mapping enhances traceability, accountability, and quality measurement throughout the development process.

a) Scrum QA Checkpoints to SDLC Stages

The SDLC includes six key stages Requirements, Design, Development, Testing, Deployment, and Maintenance. By aligning Scrum QA checkpoints with these SDLC phases, quality assurance becomes an embedded process throughout the development lifecycle. The table below illustrates the integration:

Table 4-3: SDCL stages mapping on Scrum QA checkpoints

SDLC Stage	Mapped Scrum Checkpoint	QA Focus and Activities
Requirements	Sprint Planning	Requirements validation, Acceptance Criteria (AC), Definition of Ready (DoR).
Design	Backlog Refinement	Technical clarification, risk assessment, test planning, and defect prediction.
Development	Daily Stand-ups	Progress tracking, early defect detection, CI/CD test status monitoring.
Testing	Sprint Review	Validation of Definition of Done (DoD), user acceptance testing (UAT), regression tests.
Deployment	Sprint Retrospective	Release readiness, incident analysis, and deployment QA strategy refinement.
Maintenance	Post-Sprint Monitoring	Bug monitoring, production feedback, planning hotfixes and support tasks.

This alignment ensures that each SDLC stage includes a quality checkpoint informed by Scrum's empirical process and supported by CRISP-DM's analytical capabilities.

b) QA Checkpoints across the Project Life Cycle (PLC)

The Project Life Cycle (PLC), comprising Initiation, Planning, Execution, Monitoring, and Closure, provides a broader management-level view of project progression. Incorporating QA checkpoints at each PLC stage guarantees that quality considerations are addressed from project start to completion. The following table outlines this mapping:

Table 4-4: PLC stages QA checkpoints

PLC Stage	QA Checkpoint	Focus and Deliverables
Initiation	Project Charter QA Review	Define quality scope and stakeholder expectations, establish baseline metrics.
Planning	QA Sprint Planning	Test strategy definition, risk analysis, AC development, and resource allocation.
Execution	QA Sprint Activities	Execution of planned tests, defect tracking, TDD/BDD practices, automation testing.
Monitoring	QA Metrics Evaluation	Analyze defect trends, test effectiveness, performance indicators using CRISP-DM insights.
Closure	Final QA Review	Test report compilation, final validation, documentation of lessons learned and retrospective insights.

By embedding QA validation points within each PLC phase, teams ensure proactive risk management, improved process control, and consistent delivery of high-quality software.

4.3.2 Synthesis of the Integrated Framework

This dual mapping of QA checkpoints to both SDLC and PLC stages, supported by Scrum events, provides a tri-dimensional view of quality management. The PDCA cycle enables iterative refinement of QA activities, while CRISP-DM transforms test and defect data into actionable insights. Together, they drive continuous improvement, support decision-making, and align development efforts with quality goals at every level of the software project.

In Agile software development, particularly within Scrum frameworks, checkpoints serve as critical control stages to assess progress, detect defects early, and ensure product quality. While Scrum is traditionally structured around iterative events Sprint Planning, Daily Stand-ups, Sprint Review, and Retrospective these events can be further enhanced when formally classified as quality assurance (QA) checkpoints. Such classification enables systematic validation, accountability, and process transparency, directly aligning with continuous improvement goals.

As shown on table 4-5 below, Each Scrum event can be integrated with QA practices by assigning specific checkpoint objectives and measurable outcomes. For example, Sprint Planning becomes

not only a planning session but also a QA checkpoint for validating user requirements and preparing test strategies through clearly defined Acceptance Criteria (AC) and a Definition of Ready (DoR). Similarly, Daily Stand-ups can serve as checkpoints for ongoing defect identification and defect log updates, especially when integrated with bug-tracking tools. Sprint Reviews and Retrospectives represent formal validation and process feedback stages that benefit from structured QA frameworks like PDCA and CRISP-DM.

Table 4-5: Key Scrum Checkpoints and QA Optimization Strategies

Scrum Event	Checkpoint Focus	QA Optimization Strategy
Sprint Planning	Requirements validation and test strategy	Define Acceptance Criteria and prepare a DoR checklist to ensure stories are testable.
Daily Stand-up	Progress review and early defect detection	Use real-time defect logs; integrate test status updates through CI tools and bug trackers.
Sprint Review	Stakeholder validation and UAT	Validate working software against Definition of Done (DoD); conduct user acceptance testing.
Sprint Retrospective	QA process improvement	Apply PDCA to evaluate testing bottlenecks, root causes of defects, and team performance.
Backlog Refinement	Prioritization and risk-based testing	Identify technical debt, refine test cases, and groom backlog for test efficiency.

This structured transformation of Scrum events into QA checkpoints ensures early validation, systematic feedback, and ongoing process enhancement cornerstones of quality in agile software projects.

4.3.3 Mapping CRISP-DM and PDCA to Scrum QA Checkpoints

Integrating CRISP-DM and PDCA into Scrum introduces a hybrid framework that merges data-driven analysis with iterative quality improvement. CRISP-DM offers a well-defined six-phase methodology for exploring, modeling, and applying knowledge from project data. PDCA complements this by ensuring changes and insights are implemented iteratively and consistently refined over time. Together as shown in table 4-6 below, they enhance Scrum QA checkpoints by enabling both strategic insight and operational improvement.

The integration of PDCA and CRISP-DM into Scrum introduces a holistic and structured framework for managing quality throughout the software lifecycle. By formally identifying QA checkpoints at key Scrum events and aligning them with SDLC and PLC stages, teams can ensure both process accountability and data-driven improvement. This approach transforms agile quality assurance from a reactive function into a proactive, strategic component that enhances software delivery and team performance sprint by sprint.

This model ensures that each QA checkpoint is both analytically informed and continuously improved, resulting in enhanced defect prevention, reduced rework, and higher product reliability.

Table 4-6: Mapping of Scrum checkpoints

CRISP-DM Phase	PDCA Stage	Mapped Scrum Checkpoint	QA Focus & Strategy
Business Understanding	Plan	Sprint Planning	Define QA objectives, assess risk areas, and clarify AC based on business goals.
Data Understanding	Plan	Backlog Refinement	Analyze defect history, prioritize stories based on quality risk.
Data Preparation	Do	Daily Stand-ups	Collect defect/test metrics, validate readiness and blockers with real-time insights.
Modeling	Do	Development (TDD/CI phase)	Apply test-driven development, integrate QA checkpoints in automated CI/CD pipelines.
Evaluation	Check	Sprint Review	Conduct UAT, verify functional coverage, and assess regression results against DoD.
Deployment	Act	Sprint Retrospective	Analyze defect trends and metrics; improve QA processes for future sprints.

Chapter Five

5. Data Presentation and Analysis

5.1 Introduction

This section presents the primary data collected for the study. Responses from the administered questionnaires were analyzed to identify patterns and trends using descriptive statistical methods, including tabulation, graphical representations, and frequency distributions. A comprehensive analysis is then conducted in alignment with the principles outlined in the Agile Manifesto [24].

To ensure the accuracy and reliability of the analysis, data were processed using IBM SPSS (version 26) and Microsoft Excel (2016). Relevant summaries are also provided to support the interpretation of results, and key findings are highlighted to serve as a foundation for the subsequent sections of the research.

The initial part of this section provides an overview of the demographic characteristics of the respondents and summarizes essential information directly related to the research questions. The following section explores the interrelationships among the five core components of agile processes that are central to the study's objectives.

These results will be further elaborated in the concluding sections, where the broader implications of the findings will be discussed in detail.

5.2 Demographic Profile of Respondents

5.2.1 Gender of Respondents

The respondent's gender profile indicates, as shown in the figure below, the majority of the respondents are male which accounts 62.75 percent of the respondents whereas female respondents account for 37.25 percent of the respondents.

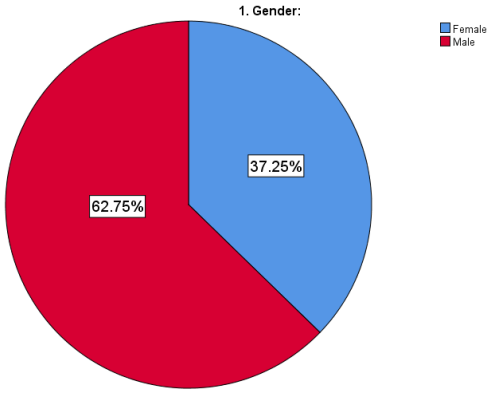

Figure 5-1: Gender of respondent

Source: Own Primary Survey, 2025

5.2.2 Age of Respondents

The age of majority of the respondents as shown from table is between 25-35 years of age numerically 84.3 percent. Whereas 2 percent falls under 25 that means more than 86.3 percent of the project team are within the age of 25-35 years that may contribute to the acceptance of agile method with less resistance to change. The fact that the majority of the respondents are young and adult implies it is an opportunity for project agility in the project environment.

Table 5-1: Age of Respondents

Age range:

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	25 to 35	43	84.3	84.3	84.3
	36 to 45	7	13.7	13.7	98.0
	under 25	1	2.0	2.0	100.0
	Total	51	100.0	100.0	

Source: Own Primary Survey, 2025

5.2.3 Education status

Education level attained by respondents as indicated in table below shows majority of the respondents were first degree holders which is 74.5 percent of the total respondents which is followed by Post graduates 25.5 percent. That means most of the respondent for the data is collection have higher educational status which is an opportunity to know and implement advanced

contemporary project management methods such as data driven agile since level of understanding and educational background of team will play an important role in adopting new approaches.

Table 5-2 : Education status

Educational status:

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Bachelor Degree	38	74.5	74.5	74.5
	Post graduate	13	25.5	25.5	100.0
	Total	51	100.0	100.0	

Source: Own Primary Survey, 2025

5.2.4 Position and years of experience

Looking at table below about the respondents work experience, most of them joined the project in less than 5 years. As illustrated in table, most of the respondents had been using an agile approach in managing and implementing projects for between 3 and 6 years. Most of the project managers were involved in managing agile projects in their organizations.

Table 5-3: cross table of role in agile project with work experience

Agile approach in managing and implementing projects

		How many years have you been using an agile approach to manage and implement projects?			
		0-2 years	3-6 years	7+ years	Total
Your role in managing agile projects	Business Analyst	2	0	0	2
	Other	2	0	0	2
	Project Manager	0	2	6	8
	QA specialists	2	3	0	5
	Scrum Master	3	5	0	8
	Software developer	10	13	3	26
Total		19	23	9	51

Source: Own Primary Survey, 2025

5.2.5 Scrum Team Vs Scrum Sprint Duration

As table 5-4 presents the size of the Scrum team against the average number of sprints in a release. The results indicate that approximately all of the Scrum teams had below 10 members. Scrum teams most could also have between 1 and 4 weeks sprints duration in a release.

Table 5-4: size of Scrum team with duration of a Scrum sprint Cross table

size of your Scrum team with average duration of a Scrum sprint Cross tabulation

		average duration of a Scrum sprint				
		1-4 weeks	12+ weeks	5-7 weeks	8-12 weeks	Total
size of your Scrum team	1-4 members	15	1	6	3	25
	11-15 member	1	0	0	0	1
	5-10 members	13	0	8	4	25
Total		29	1	14	7	51

Source: Own Primary Survey, 2025

5.2.6 Agile Adoption & Benefits

In this section respondent answers why they use agile software project management approach in your company to identify benefit and project success criteria. As the table below shows, most respondents indicated that the reasons for adopting agile were accelerated software delivery in their organization (94.1%), enhanced delivery predictability and enhanced ability to manage changing abilities (92.2%). As table 5-5 shown below, most respondents indicated that the benefits of adopting agile were delivery speed/time to market, ability to manage changing priorities and project visibility. The results also indicate that the benefits of adopting agile were managing distributed teams and reducing project cost (90.2%).

Table 5-5: Respondent indication on benefit of adopting agile

		Count	Layer N %	Row Valid N %
1. My organization adopted agile to accelerate software delivery.	No	3	5.9%	100.0%
	Yes	48	94.1%	100.0%
2. Agile adoption has improved our ability to manage changing requirements.	No	4	7.8%	100.0%
	Yes	47	92.2%	100.0%
3. Agile methodologies have increased stakeholder collaboration.	No	2	3.9%	100.0%
	Yes	49	96.1%	100.0%
4. Our agile projects have reduced project risks.	No	6	11.8%	100.0%
	Yes	45	88.2%	100.0%
5. Agile methods have helped in cost reduction.	No	5	9.8%	100.0%
	Yes	46	90.2%	100.0%

Source: Own Primary Survey, 2025

The results in Table 5-6 below on how organizations measure success of agile projects, respondents indicated that customer/user satisfaction (90.2%), business value delivered (90.2%), velocity (80.4%), planned vs. actual story release dates (74.5%) and planned vs. actual stories per iteration (74.5%) were the most measures of agile project success. These results mean that the most common criteria used to measure agile project success are customer/user satisfaction, business value delivered, velocity, planned vs. actual story release dates and planned vs. actual stories per iteration.

Table 5-6: Respondent indication on success of agile projects

		Count	Layer N %
6. Customer satisfaction is the most important measure of our agile project success.	No	5	9.8%
	Yes	46	90.2%
7. Business value delivered is a key indicator of agile project success.	No	5	9.8%
	Yes	46	90.2%
8. We track project velocity to measure agile project performance.	No	10	19.6%
	Yes	41	80.4%
9. We compare planned vs. actual stories per iteration to measure success.	No	13	25.5%
	Yes	38	74.5%
10. Budget vs. actual cost tracking is important in our agile projects.	No	10	19.6%
	Yes	41	80.4%

Agile project success was measured using the weights in percentage (%) for success criteria against audit processes. These weights (%) were obtained from respondents' perceptions on relationship between agile project success criteria and audit processes (46 items). That indicated that Customer/ User satisfaction (90.2%) had the highest weight (%) between agile project success criteria and audit processes followed by Revenue/ Sales Business value delivered impact, Budget vs. Actual cost (80.4%), and Individual hours per iteration/week.

5.3 Correlation Analysis

In an effort to explore whether there is a meaningful link between auditing practices and the success of agile projects, a Pearson correlation analysis was carried out. The data analysis was conducted following the procedures outlined in the SPSS Survival Manual [59], the strength of a correlation whether positive or negative can be interpreted as follows: correlations between 0.10 and 0.29 are considered weak, those from 0.30 to 0.49 are viewed as moderate, and values from 0.50 to 1.00 reflect a strong relationship.

This study concentrated on examining five specific audit areas within the Scrum framework: product vision, product backlog, release backlog, sprint backlog, and final product deliverables. These areas were identified through a careful review of relevant literature and aligned with standard Scrum processes.

To measure these aspects, 32 items were created, each reflecting a particular dimension of the auditing process in agile project environments. A five-point Likert scale, ranging from "strongly disagree" to "strongly agree," was used to gauge the respondents' views on each item. Participants were asked to respond based on their experience with their most recent agile project, ensuring that their feedback was grounded in actual practice.

The responses aimed to capture both how extensively each audit process was applied and how critical each one was perceived to be in ensuring project success. The correlation analysis was then used to evaluate whether a stronger application of these audit mechanisms was associated with higher levels of success in agile project delivery. The outcomes of this analysis are presented and discussed in the sections that follow.

5.3.1 Product Vision Processes Vs Project Success

In the Agile product vision process, it is important to align quality assurance (QA) efforts with the overall business objectives. Table 5-7 below, presents the results of the correlation analysis between product vision audit processes and agile project success.

The study looked at how the processes involved in defining an agile product vision relate to the success of agile projects. A Pearson correlation analysis was used to find out whether there were any significant relationships between specific audit items and how successful agile projects turned out to be a weak but positive correlation was observed between project success and the creation of a product vision agreed upon by the Scrum team ($r = 0.257$, $p > 0.05$), suggesting that establishing a shared vision contributes positively, though modestly, to project outcomes. A moderate and statistically significant correlation was found between project success and the identification of key Scrum roles such as Product Owner, Scrum Master, and Scrum Team ($r = 0.344$, $p < 0.05$), indicating that clearly defining team roles is vital for successful delivery. The analysis also showed a stronger correlation between agile project success and training provided to the Scrum team and stakeholders on Agile values and principles ($r = 0.420$, $p < 0.01$), emphasizing the importance of continuous learning and alignment with Agile philosophy. Similarly, a moderate correlation ($r = 0.356$, $p < 0.05$) was identified between project success and the definition of software quality objectives aligned with business goals. Establishing a steering committee for strategic oversight also showed a significant positive relationship with project success ($r = 0.418$, $p < 0.01$), highlighting the value of governance structures in Agile initiatives. Furthermore, the development of epics and user stories with clearly defined acceptance criteria correlated moderately with success ($r = 0.395$, $p < 0.01$), underscoring the importance of clarity in requirements and expectations. Finally, a notable correlation was found between project success and ensuring alignment with Agile values and principles ($r = 0.407$, $p < 0.01$), as well as holding user group meetings to discuss appropriate epics ($r = 0.418$, $p < 0.01$), the latter being the highest among the observed coefficients. These findings collectively suggest that integrating structured visioning activities, stakeholder engagement, role clarity, and adherence to agile values within the product vision phase significantly enhances the likelihood of agile project success.

Table 5-7: Correlation Product Vision Processes Vs Project Success

Checkpoint items influencing the QA		Project Success Variable
1.1 Created a product vision agreed upon by the Scrum team	Pearson Correlation	.257
	Sig. (2-tailed)	.068
	N	51
1.2 Identified critical stakeholders (Product Owner, Scrum Master, Scrum Team)	Pearson Correlation	**.344***
	Sig. (2-tailed)	.013
	N	51
1.3 Conducted training on Agile values and guiding principles	Pearson Correlation	**.420****
	Sig. (2-tailed)	.002
	N	51
1.4 Defined software quality objectives (aligned with business goals)	Pearson Correlation	**.356***
	Sig. (2-tailed)	.010
	N	51
1.5 Established Steering Committee for strategic oversight	Pearson Correlation	**.418****
	Sig. (2-tailed)	.002
	N	51
1.6 Developed epics and user stories with clear acceptance criteria	Pearson Correlation	**.395****
	Sig. (2-tailed)	.004
	N	51
1.7 Ensured alignment with agile values and guiding principles	Pearson Correlation	**.407****
	Sig. (2-tailed)	.003
	N	51
*. Correlation is significant at the 0.05 level (2-tailed).		
**. Correlation is significant at the 0.01 level (2-tailed).		

5.3.2 Product Backlog Processes Vs Project Success

As shown in table 5-8 below, the correlation results show several key relationships between backlog-related audit practices and project success. First, developing product requirements that align with the product vision showed a moderate to strong positive correlation with project success ($r = 0.490$, $p < .01$), suggesting that ensuring consistency between initial vision and detailed requirements plays a significant role in guiding project outcomes. Maintaining meeting minutes for reviewing and approving backlog items had a slightly stronger correlation ($r = 0.500$, $p < .01$), indicating that good documentation and collaborative review contribute positively to transparency and shared understanding across the team.

An even higher correlation was observed for establishing a formal change request and approval process for updating backlog items ($r = 0.509$, $p < .01$), which underlines the importance of controlled change management in preserving backlog integrity and supporting project agility. Providing training for the Scrum Master and team on backlog management also had a significant impact ($r = 0.439$, $p < .01$), reinforcing that capacity-building and knowledge transfer are essential in maintaining a well-managed, responsive backlog.

Finally, the formal ownership of the product backlog by the Product Owner showed a smaller but statistically significant positive relationship with project success ($r = 0.289$, $p < .05$). While weaker than the other variables, this result still emphasizes the value of having clear accountability in backlog management. In total, these findings demonstrate that structured backlog practices rooted in communication, documentation, training, and governance have a measurable and often strong influence on the success of agile projects.

Table 5-8: Correlation Product Backlog Processes Vs Project Success

Checkpoint items influencing the QA		Project Success Variable
2.1 Developed product requirements aligned with the product vision	Pearson Correlation	**.490****
	Sig. (2-tailed)	.000
	N	51
2.2 Maintained meeting minutes reviewing and approving backlog items	Pearson Correlation	**.500****
	Sig. (2-tailed)	.000
	N	51
2.3 Established a change request and approval process for backlog updates	Pearson Correlation	**.509****
	Sig. (2-tailed)	.000
	N	51
2.4 Conducted training for Scrum Master & Team on backlog management	Pearson Correlation	**.439****
	Sig. (2-tailed)	.001
	N	51
2.5 Product Owner formally owns the product backlog	Pearson Correlation	.289*
	Sig. (2-tailed)	.040
	N	51
**. Correlation is significant at the 0.01 level (2-tailed).		
*. Correlation is significant at the 0.05 level (2-tailed).		

5.3.3 Release Backlog Planning Processes Vs Project Success

As table 5-9 shown, the correlation analysis between release backlog practices and agile project success highlights the importance of structured planning and alignment across development phases. Developing a release backlog aligned with the product backlog showed a moderate positive correlation with project success ($r = 0.407$, $p < .01$), indicating that maintaining consistency between these two backlogs helps ensure smoother transitions from planning to delivery and supports clearer prioritization of features. Creating a release schedule that spans multiple sprints had a weaker but still statistically significant correlation ($r = 0.349$, $p < .05$), suggesting that planning for long-term iterations contributes positively to predictability and overall project coordination.

Additionally, documenting meeting minutes that capture and approve release planning decisions also demonstrated a moderate positive relationship ($r = 0.398$, $p < .01$) with project success, highlighting the value of formal documentation and team consensus in release planning. The highest correlation in this set was found in the practice of defining iteration lengths and sprint planning structures ($r = 0.460$, $p < .01$), reflecting that clear, consistent planning cycles significantly support Agile project delivery and success.

Table 5-9: Correlation Release Backlog Planning Processes Vs Project Success

Checkpoint items influencing the QA		Project Success Variable
3.1 Developed a release backlog aligned with the product backlog	Pearson Correlation	.407**
	Sig. (2-tailed)	.003
	N	51
3.2 Created a release schedule covering multiple sprints	Pearson Correlation	.349*
	Sig. (2-tailed)	.012
	N	51
3.3 Documented meeting minutes approving release planning decisions	Pearson Correlation	.398**
	Sig. (2-tailed)	.004
	N	51
3.4 Defined iteration length and sprint planning structure	Pearson Correlation	.460**
	Sig. (2-tailed)	.001
	N	51
**. Correlation is significant at the 0.01 level (2-tailed).		
*. Correlation is significant at the 0.05 level (2-tailed).		

5.3.4 Sprint Backlog Processes Vs Project Success

The analysis, as shown table 5-10 below, found a statistically significant and strong positive correlation between project success and the use of a Scrum board to track progress during sprints ($r = 0.463$, $p < 0.01$), highlighting the importance of transparent task management and team visibility. Defined sprint requirements aligned with the product backlog also showed a strong and significant correlation ($r = 0.423$, $p < 0.01$), suggesting that clear and well-aligned requirements serve as a foundation for successful project execution.

Several other checkpoint practices were moderately but significantly associated with project success. These include the existence of a sprint block list ($r = 0.422$, $p < 0.01$), the generation of sprint review reports and retrospective findings ($r = 0.418$, $p < 0.01$), and the existence of a sprint progress report ($r = 0.387$, $p < 0.01$). These findings underscore the importance of ongoing reflection, progress tracking, and proactive problem-solving in Agile QA processes.

Documented sprint planning meetings, including task breakdowns and effort estimations, also showed a significant positive correlation ($r = 0.375$, $p < 0.01$), reinforcing the value of structured planning and workload transparency. The presence of a sprint burn down chart ($r = 0.350$, $p < 0.05$) and the regular maintenance of daily standup meeting minutes ($r = 0.339$, $p < 0.05$) were also positively associated with success, though slightly weaker, indicating that consistent communication and progress monitoring contribute meaningfully to QA efforts.

Table 5-10: Correlation Sprint Backlog Processes Vs Project Success

.Checkpoint items influencing the QA		Project Success Variable
4.1 Defined sprint requirements aligned with the product backlog	Pearson Correlation	.423**
	Sig. (2-tailed)	.002
	N	51
4.2 Documented Sprint Planning meetings, task breakdowns, and estimates	Pearson Correlation	.375**
	Sig. (2-tailed)	.007
	N	51
4.3 Maintained daily standup meeting minutes	Pearson Correlation	.339*
	Sig. (2-tailed)	.015
	N	51
4.4 Existence of sprint progress report	Pearson Correlation	.387**
	Sig. (2-tailed)	.005
	N	51

4.5 Existence of a sprint burn down chart to monitor the progress of the Scrum team	Pearson Correlation	.350*
	Sig. (2-tailed)	.012
	N	51
4.6 Existence of scrum board used by scrum team to track its progress during sprint	Pearson Correlation	.463**
	Sig. (2-tailed)	.001
	N	51
4.7 Existence of sprint block list	Pearson Correlation	.422**
	Sig. (2-tailed)	.002
	N	51
4.8 Generated Sprint Review Reports and Retrospective Findings	Pearson Correlation	.418**
	Sig. (2-tailed)	.002
	N	51
**. Correlation is significant at the 0.01 level (2-tailed).		
*. Correlation is significant at the 0.05 level (2-tailed).		

5.3.5 Product Delivery Processes Vs Project Success

A strong and significant relationship was found between project success and the documentation of issues encountered during the project, as the table 5-11 show ($r = 0.407$, $p < 0.01$), indicating that recognizing and addressing challenges is critical to outcome quality. Similarly, lessons learned ($r = 0.381$, $p < 0.01$) and the formal documentation of product deliverables acceptance ($r = 0.372$, $p < 0.01$) were also strongly correlated, highlighting the importance of organizational learning and clear acceptance criteria in QA.

Post-project support and maintenance agreements demonstrated a significant positive correlation ($r = 0.369$, $p < 0.01$), suggesting that ongoing support mechanisms are associated with more successful outcomes. Additionally, taking formal actions to resolve project issues ($r = 0.362$, $p < 0.01$), having a working deliverables agreement ($r = 0.360$, $p < 0.01$), and generating forward-looking recommendations ($r = 0.349$, $p < 0.05$) were all moderately but significantly associated with project success.

Other reflective practices, such as documenting improvements for future projects ($r = 0.340$, $p < 0.05$), were also found to have a meaningful positive impact, reinforcing the value of continuous improvement. Notably, the existence of a formal project closure contract ($r = 0.264$, $p = 0.061$) showed a positive but statistically non-significant correlation, suggesting that while closure documentation may be beneficial, its influence may vary depending on project context.

Table 5-11: Correlation Product Delivery Processes Vs Project Success

Checkpoint items influencing the QA		Project Success Variable
5.1 Existence of a formal working deliverables agreement	Pearson Correlation	.360**
	Sig. (2-tailed)	.010
	N	51
5.2 Existence of a product deliverables acceptance report	Pearson Correlation	.372**
	Sig. (2-tailed)	.007
	N	51
5.3.1 Lessons learned from the project	Pearson Correlation	.381**
	Sig. (2-tailed)	.006
	N	51
5.3.2 Improvements identified for future projects	Pearson Correlation	.340*
	Sig. (2-tailed)	.015
	N	51
5.3.3 Issues encountered during the project	Pearson Correlation	.407**
	Sig. (2-tailed)	.003
	N	51
5.3.4 Actions taken to resolve project issues	Pearson Correlation	.362**
	Sig. (2-tailed)	.009
	N	51
5.3.5 Recommendations for upcoming projects	Pearson Correlation	.349*
	Sig. (2-tailed)	.012
	N	51
5.4 Existence of a formal project closure contract	Pearson Correlation	.264
	Sig. (2-tailed)	.061
	N	51
5.5 Existence of a post-project support and maintenance agreement	Pearson Correlation	.369**
	Sig. (2-tailed)	.008
	N	51
**. Correlation is significant at the 0.01 level (2-tailed).		
*. Correlation is significant at the 0.05 level (2-tailed).		

The correlation analysis across five major Scrum components product vision, product backlog, release backlog, sprint backlog, and product delivery revealed that structured QA checkpoints are positively associated with project success. The results showed statistically significant moderate to strong correlations in many areas, emphasizing the critical role of systematic auditing in agile practices.

A number of product vision-related processes were moderately or strongly correlated with project success. Training on Agile values ($r = 0.420$, $p < 0.01$), forming a strategic steering committee ($r = 0.418$, $p < 0.01$), and user group meetings ($r = 0.418$, $p < 0.01$) highlight the importance of vision alignment, stakeholder involvement, and Agile literacy. Among all areas, product backlog management exhibited some of the highest correlations, with the strongest being the change request and approval process ($r = 0.509$, $p < 0.01$), followed by meeting documentation ($r = 0.500$, $p < 0.01$) and alignment with product vision ($r = 0.490$, $p < 0.01$). These findings stress that controlled change management and transparent documentation are central to quality. Release backlog items such as clearly defined iteration lengths ($r = 0.460$, $p < 0.01$) and sprint planning structures demonstrated meaningful impact. Sprint backlog practices also showed high significance, especially the use of Scrum boards ($r = 0.463$, $p < 0.01$), aligned sprint requirements ($r = 0.423$, $p < 0.01$), and sprint review reports ($r = 0.418$, $p < 0.01$). These findings point to the necessity of transparency, structured planning, and proactive review cycles in maintaining high-quality standards across sprints.

From the product delivery phase, documenting project issues ($r = 0.407$, $p < 0.01$), lessons learned ($r = 0.381$, $p < 0.01$), and actions taken to resolve problems ($r = 0.362$, $p < 0.01$) were significantly linked to project success. This indicates that retrospective reflection and organizational learning contribute meaningfully to Agile QA. While the formal project closure contract was not statistically significant ($r = 0.264$, $p = 0.061$), its positive trend suggests that formal closure mechanisms still play a supportive role.

5.4 Discussion

This discussion interprets the empirical findings presented in the data analysis and connects them directly to the conceptual framework proposed in Chapter 5. The integration of CRISP-DM and PDCA into Scrum QA practices is supported by statistically significant correlations found between key checkpoint activities and agile project success. These results validate the framework's strategic direction and demonstrate the relevance of embedding structured QA checkpoints throughout the Scrum lifecycle.

5.4.1 Validating PDCA Integration in Scrum QA

The PDCA cycle Plan, Do, Check, and Act forms the operational backbone of the proposed framework. This structure is confirmed by the strong and moderate correlations found in the data:

- **Plan Phase**: Activities such as defining a product vision ($r = 0.257$) and aligning sprint requirements with the product backlog ($r = 0.423$, $p < 0.01$) support the importance of early planning. The strong relationship between clearly defined team roles ($r = 0.344$, $p < 0.05$) and training on Agile principles ($r = 0.420$, $p < 0.01$) further underscores the significance of upfront clarity, a core component of the Plan phase in PDCA.
- **Do Phase**: Strong positive correlations were observed in backlog refinement and execution practices, including the change request process for backlog updates ($r = 0.509$, $p < 0.01$) and sprint execution tools like Scrum boards ($r = 0.463$, $p < 0.01$). These findings justify the "Do" phase's emphasis on structured execution, traceability, and defect tracking.
- **Check Phase**: Correlations such as those for maintaining sprint progress reports ($r = 0.387$, $p < 0.01$) and retrospective documentation ($r = 0.418$, $p < 0.01$) reinforce the need for regular inspection. These practices directly support the Check phase's role in performance evaluation and metrics-based validation.
- **Act Phase**: Practices such as issue resolution ($r = 0.362$, $p < 0.01$), lessons learned ($r = 0.381$, $p < 0.01$), and implementing forward-looking recommendations ($r = 0.349$, $p < 0.05$) indicate the need for responsive action and continuous refinement, which are central to the Act phase of PDCA.

Together, these findings empirically validate the PDCA integration in the framework and support the cycle's application across Scrum ceremonies to drive quality assurance and improvement.

5.4.2 CRISP-DM as a Data-Driven Enabler of QA

The CRISP-DM methodology complements PDCA by anchoring QA in evidence-based insights. This is substantiated by the data:

- The strong and moderate correlations for activities involving documentation (e.g., sprint planning meetings, backlog meeting minutes) reflect the necessity of the Data Understanding and Preparation phases, where project data is analyzed and structured for decision-making.
- Modeling activities, such as the prediction of defect-prone components, align with structured QA tasks like backlog prioritization ($r = 0.490$, $p < 0.01$) and sprint planning with aligned requirements ($r = 0.423$, $p < 0.01$).

- The Evaluation and Deployment phases are reflected in practices such as sprint review reports ($r = 0.418$, $p < 0.01$) and stakeholder collaboration ($r = 0.418$, $p < 0.01$), which demonstrate the value of continuous learning and adaptive implementation.

These insights confirm CRISP-DM's role in transforming raw QA data into actionable knowledge, making it an essential analytical layer in the proposed hybrid framework.

5.4.3 Tri-Layer QA Mapping: Scrum, SDLC, and PLC Alignment

The dual mapping of QA checkpoints to SDLC and PLC stages further enhances the practicality and comprehensiveness of the proposed framework. The data analysis revealed meaningful correlations across checkpoints representing all stages of the software and project life cycles. In terms of SDLC integration, the requirements and design stages were strengthened by activities such as backlog refinement and vision alignment, while the testing and deployment phases showed strong associations with sprint review practices and post-project support mechanisms (e.g., $r = 0.369$, $p < 0.01$). Similarly, within the PLC mapping, closure activities such as documenting lessons learned ($r = 0.381$, $p < 0.01$) and defining support and maintenance agreements ($r = 0.369$, $p < 0.01$) validated the importance of post-delivery quality assurance. These findings support the idea that quality assurance should not be treated as an isolated activity but rather as a continuous thread woven throughout the entire project lifecycle. This tri-dimensional alignment linking Scrum events, SDLC phases, and PLC stages ensures that QA becomes a pervasive, structured, and data-informed process, enhancing consistency, accountability, and delivery quality across agile projects.

The empirical findings lend strong support to the proposed PDCA + CRISP-DM Scrum QA Framework, affirming its potential to improve quality assurance in agile environments. Organizations can leverage this integrated model to enhance visibility and accountability across Scrum checkpoints, predict and prevent defects through the use of backlog data and modeling techniques, and implement continuous feedback loops that enable real-time quality control. Furthermore, the framework helps align quality goals with broader business objectives and stakeholder expectations, ensuring that quality assurance is both strategic and operational. By grounding the framework in statistically measurable results, it becomes a replicable and adaptable model that agile teams can apply to embed QA effectively into every phase of the software development lifecycle.

Chapter Six

6. Conclusion and Recommendation

6.1 Summary, Conclusion, And Research Contributions

This study addressed the growing need for structured quality assurance in agile software development by proposing a data-driven QA framework optimized for Scrum environments. The research was motivated by persistent challenges such as inconsistent QA implementation, technical debt, and insufficient proactive testing practices in agile teams. To solve these issues, a conceptual framework was designed by integrating PDCA and CRISP-DM, two well-established models known for their emphasis on iterative improvement and data-driven decision-making.

A mixed-methods approach was employed. First, a comprehensive literature review was conducted to identify gaps in current agile QA practices. Then, data were collected from 51 agile practitioners across 7 software development companies in Addis Ababa, Ethiopia, including project managers, Scrum Masters, QA specialists, and developers. The quantitative analysis was performed using IBM SPSS, and the results revealed strong positive correlations between the implementation of structured QA checkpoints and improvements in software quality, defect reduction, and sprint performance.

The research confirmed that the proposed framework effectively enhances agile QA by promoting early defect detection, improved traceability, and continuous feedback. By applying statistical reasoning to QA activities, agile teams can proactively manage quality issues rather than reactively addressing those after defects are deployed to production. The integration of CRISP-DM facilitates the use of historical data and predictive analytics to support QA decision-making, while PDCA reinforces continuous process improvement throughout the sprint cycle.

The primary contributions of this study are as follows:

- A validated QA framework that integrates Scrum with PDCA and CRISP-DM models.
- Empirical evidence supporting the placement of QA checkpoints across SDLC and PLC stages through a structured survey and statistical analysis of 51 agile practitioners.
- Measurable results linking QA checkpoints to quality outcomes using SPSS data.
- Practical insights for organizations seeking to align agile speed with product quality using data-informed strategies.

In conclusion, this study provides a scientifically justified and empirically validated framework that enhances quality assurance in agile Scrum environments. By aligning QA activities with agile values and leveraging data analytics, the research contributes both theoretical insight and practical tools for improving software quality in dynamic development contexts. The results underscore the necessity of transforming QA from a reactive function into a proactive, data-driven discipline that supports agility without compromising reliability.

6.2 Recommendations

Based on the findings of this study and the validation of the proposed data-driven QA framework, several recommendations are offered to improve quality assurance practices in agile software development environments, particularly those using Scrum.

First, organizations should institutionalize data-driven quality assurance by integrating it into their agile governance structures. This involves embedding quality checkpoints within standard Scrum ceremonies such as sprint planning, backlog refinement, daily stand-ups, and retrospectives. Establishing policy mandates for these checkpoints can promote consistency, facilitate proactive defect prevention, and reinforce organizational commitment to quality.

Second, agile teams are encouraged to invest in QA analytics and automation tools that support real-time quality monitoring and continuous integration workflows. Platforms such as SonarQube, Jenkins, and SPSS can help track metrics like defect density, code coverage, and sprint-level rework frequency. These tools enhance visibility into potential risks and enable teams to respond to quality issues more rapidly and with greater precision.

Third, continuous training and capacity building should be prioritized. Scrum Masters, QA engineers, and developers should receive targeted training on interpreting QA metrics, implementing checkpoints, and using relevant analytics and testing tools. Strengthening team members' competence in these areas ensures that quality assurance becomes an integral and effective part of the agile process rather than an afterthought.

Additionally, organizations should adopt recognized industry standards such as ISO/IEC 25010 for software quality requirements and IEEE 730 for quality assurance plans. These standards can serve as a foundation for internal QA audits and assessments. Monitoring key metrics—including defect density, user acceptance testing (UAT) success rates, and frequency of sprint-level

rework—will support continuous process improvement and decision-making based on empirical evidence.

Finally, fostering cross-functional collaboration is essential. Quality should not be the sole responsibility of QA personnel. Instead, it should involve coordinated efforts among developers, business analysts, Scrum Masters, and product owners. Encouraging collaborative QA discussions during sprint planning and retrospectives can help align expectations, improve requirement clarity, and reduce the likelihood of quality issues later in the development cycle.

By adopting these recommendations, agile organizations can better integrate structured QA into their workflows, elevate product quality, and align their practices with the evolving demands of modern software development. These actions will also support the practical adoption of the adapted PDCA and CRISP-DM-based QA framework proposed in this study.

References

[1] S. Abdalhamid, A. Mohammed, and A. Mishra, "Agile and quality: A systematic mapping study," in *International Conference on Computational Science and Computational Intelligence (CSCI)*, 2019.

[2] S. Sharma, D. Sarkar, and D. Gupta, "Agile processes and methodologies: A conceptual study," *International Journal on Computer Science and Engineering,* vol. 4, no. 5, p. 186–190, 2012.

[3] T. Lautert, A. Neto, and N. Kozievitch, "A survey on agile practices and challenges of a global software development team," in *International Conference on Software Process Improvement and Capability Determination*, 2019.

[4] U. Ameta, M. Patel, and N. S. Rathore, "Fusing artificial intelligence with Scrum framework," SSRN, Udaipur,india, 2023.

[5] C. N. Egwim and H. Alaka, "A comparative study on machine learning algorithms for predicting construction projects delay," *IEEE Access,* vol. 9, p. 112968–112978, 2021.

[6] M. A. Amani, A. Behdinian, and M. Sheikhalishahi, "Evaluating factors affecting project success: An agile approach," in *Proceedings of the 2023 International Conference on Industrial Engineering and Operations Management (IEOM)*, 2023.

[7] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software,* vol. 85, no. 6, p. 1213–1221, 2012.

[8] R. Knaster, "15th Annual State of Agile Report," Digital.ai, North Carolina,USA, 2021.

[9] R. Hanslo and M. Tanner, "Machine learning models to predict agile methodology adoption," in *the 2020 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2020.

[10] A. Bhanushali, "Challenges and solutions in implementing continuous integration and continuous testing for agile quality assurance," *International Journal of Engineering Research and Technology,* vol. 12, no. 1626, p. 1–8, 2023.

[11] S. Ambler, "Quality in an Agile World," *Software Quality Professional,* vol. 7, no. 4, pp. 34-40, 2005.

[12] R. M. Fontana, V. Meyer, S. Reinehr, and A. Malucelli, "Progressive Outcomes A framework for maturing in agile software development," *Journal of Systems and Software,* vol. 102, p. 88–108, 2015.

[13] W. Y. Wong, W. F. Pok, T. H. Sam, and C. W. Too, "Software quality assurance plan: Setting quality assurance checkpoints within the project life cycle and system development life cycle," *2022 IEEE 18th International Colloquium on Signal Processing & Applications (CSPA),* vol. 1, pp. 1-6, 2022.

[14] A. A. Wakili, L. N. Alhassan, and A. Kamagata, "Quality Assurance Practices in Agile Methodology," *International Jornal of Advanced in Science and Engineering ,* vol. 9, no. 10, pp. 141-149, 2020.

[15] E. Mkoba and C. Marnewick, "Conceptual framework for auditing agile projects," *IEEE Access,* vol. 8, pp. 126460-126476, 2020.

[16] M. Sagheer, T. Zafar, and M. Sirshar, "A Framework For Software Quality Assurance Using Agile Methodology," *International Journal of Scientific & Technology Research,* vol. 4, no. 2, pp. 44-49, 2015.

[17] Silverio Martínez-Fernández, Anna Maria Vollmer, Andreas Jedlitschka, Xavier Franch, Lidia López, Prabhat Ram, Pilar Rodríguez, Sanja Aaramaa, Alessandra Bagnato, Michał Choraś, and Jari Partanen, "Continuously assessing and improving software quality with software analytics tools: a case study," *IEEE Access,* vol. 7, pp. 68219 - 68239, 2019.

[18] S. Pargaonkar, "Enhancing Software Quality in Architecture Design: A Survey- Based Approach," *International Journal of Scientific and Research Publications,* vol. 13, no. 8, pp. 16-19, 2023.

[19] Mario Cardarelli,Ludovico Iovino,Paolo Di Francesco,Amleto Di Salle,Ivano Malavolta,Patricia Lago, "An Extensible Data-Driven Approach for Evaluating the Quality of Microservice Architectures," in *the 34th ACM/SIGAPP Symposium*, Limassol, Cyprus, 2019.

[20] K. Schwalbe, InformatIon technology, Boston,USA: Cengage, 2018.

[21] H. Kerzner, Project Management A Systems Approach to Planning, Scheduling, and Controlling, Hoboken, New Jersey: John Wiley & Sons, Inc, 2017.

[22] Tobin J Lehman, Akhilesh Sharma, "Software Development as a service: Agile Experiences," in *annual SRII Global Conference*, 2011.

[23] Barry Boehm,Richard Turner, Balancing Agility and Discipline: A Guide for the Perplexed, Boston: Pearson Education, Inc., 2003.

[24] Kent Beck,James Grenning,Robert C. Martin, "Manifesto for Agile Software Development," agilemanifesto, 2001. [Online]. Available: https://agilemanifesto.org/. [Accessed 15 December 2024].

[25] Shevali Agarwal, Anurag Punde, "A Study of Agile Software Development," *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY,* vol. 2, no. 11, pp. 3085-3087, 2013.

[26] ISO/IEC/IEEE24765:2017, "Systems and software engineering — Vocabulary," ISO, Geneva, Switzerland, 2017.

[27] IEEE730-2014, "IEEE Standard for Software Quality Assurance Plans," IEEE, NJ,USA, 2014.

[28] Parvez Mahmood Khan, M.M. Sufyan Beg, "Measuring Cost of Quality (CoQ) on SDLC Projects is Indispensible for Effective Software Quality Assurance," *International Journal of Soft Computing and Software Engineering,* vol. 2, no. 9, 2014.

[29] P. McBreen, "McBreen.Consulting," McBreen.Consulting, 2003. [Online]. Available: http://www.mcbreen.ab.ca/talks/CAMUG.pdf. [Accessed 10 December 2024].

[30] Ernest Mnkandla,Barry Dwolatzky, "Defining Agile Software Quality Assurance," in *International Conference on Software Engineering Advances*, Tahiti, 2006.

[31] Deepti Mishra, Samia Abdalhamid, "Software Quality Issues in SCRUM:A Systematic Mapping," *Journal of Universal Computer Science,* vol. 24, no. 12, pp. 1690-1716, 2018.

[32] Sanghoon Jeon, Myungjin Han, Eunseok Lee, Keun Lee, "Quality Attribute driven Agile Development," in *International Conference on Software Engineering Research, Management and Applications*, Baltimore, MD, USA, 2011.

[33] Li. J., Moe. N.B., Dybå. T., "Transition from a plan-driven process to Scrum - A longitudinal case study on software quality," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, Bolzano/Bozen, Italy, 2010.

[34] Tom Mens and Tom Tourwe, "A Survey of Software Refactoring," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,* vol. 30, no. 2, pp. 1-15, 2004.

[35] M. Fowler, Refactoring: Improving the Design of Existing Code, Book News, Inc., 1999.

[36] David Janzen,Hossein Saiedian, "Test-Driven Development:Concepts, Taxonomy,and Future Direction," *IEEE Computer Society,* pp. 43-50, 2005.

[37] Ming Huo, June Verner, Liming Zhu, Muhammad Ali Babar, "Software Quality and Agile Methods," in *28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, New South Wales, Australia, 2004.

[38] Michael Hilton,Timothy Tunnell,Kai Huang, "Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects," in *ACM/IEEE International Conference on Automated Software Engineering*, Singapore Singapore, 2016.

[39] Laura Plonka, Helen Sharp, Janet van der Linden, "Knowledge Transfer in Pair Programming: An In-depth Analysis," *International Journal of Human-Computer Studies,* vol. 73, no. 1, pp. 66-78, 2015.

[40] Gary M. Olson and Judith S. Olson, "Distance Matters," *International Journal of HUMAN-COMPUTER INTERACTION,* vol. 15, pp. 139-178, 2000.

[41] Rashina Hoda; James Noble, "Becoming Agile: A Grounded Theory of Agile Transitions in Practice," in *IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, Buenos Aires, Argentina, 2017.

[42] Raymond P.L. Buse,Thomas Zimmermann, "Analytics for Software Development," in *ACM International Conference on Software Engineering (ICSE)*, New Mexico, USA, 2010.

[43] Kim, S., Zimmermann, T., & Nagappan, N., "Data-driven software engineering," in *Proceedings of the 36th International Conference on Software Engineering.*, Hyderabad,India, 2014.

[44] L. Goasduff, "Gartner," Gartner, 19 may 2021. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2021-05-19-gartner-says-70-percent-of-organizations-will-shift-their-focus-from-big-to-small-and-wide-data-by-2025. [Accessed 19 Dec 2024].

[45] ISO/IEC/IEEE12207:2017, "Systems and software engineering — Software life cycle processes," ISO, Geneva, Switzerland, 2017.

[46] PMBOKGuide, "A Guide to the Project Management Body of Knowledge," Project Management Institute, PA, USA, 2017.

[47] I. Sommerville, Software Engineering, Pearson, 2015.

[48] ISO/IEC25010:2011, "Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE)," ISO, Geneva, Switzerland, 2011.

[49] Ram Srinivasan and Bernie Maloney, "Scrumalliance," Scrumalliance, 2024. [Online]. Available: https://www.scrumalliance.org/about-scrum. [Accessed 12 2024].

[50] Brown, A., Smith, J., & Lee, R., "Agile QA efficiency," *Journal of Software Engineering,* vol. 39, no. 2, p. 115–130, 2024.

[51] M. Nakhate, "Tech Solutions Inc.," 22 10 2023. [Online]. Available: https://meghna-nakhate.medium.com/business-case-study-tech-solutions-inc-908892f85f73. [Accessed 5 2 2025].

[52] M. Alphonce, "Enhancing Software Quality through Early-Phaseof SoftwareVerification and Validation Techniques," *International Journal of Technology and Systems,* vol. 9, no. 1, pp. 1-15, 2024.

[53] Amran Hossain, Dr. Md. Abul Kashem, Sahelee Sultana, "Enhancing Software Quality Using Agile Techniques," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 10, no. 2, pp. 87-93, 2013.

[54] J. W. Creswell, Research design: Qualitative, quantitative, and mixed methods approaches, Thousand Oaks, CA: Sage Publications, 2013.

[55] S. Singh, "researcher.life," researcher.life, 20 07 2023. [Online]. Available: https://researcher.life/blog/article/what-is-a-conceptual-framework-and-how-to-make-it-with-examples/. [Accessed 22 dec 2024].

[56] D. M. L. Smith, "blainy," blainy, 5 07 2024. [Online]. Available: https://blainy.com/conceptual-framework/. [Accessed 22 Dec 2024].

[57] Domingos F. Oliveira and Miguel A. Brito, "Framework for Quality Assurance and Quality Control in DL Systems," in *International Conference on Electrical, Computer and Energy Technologies (ICECET)*, Cape Town-South Africa, 2023.

[58] S. Isniah, H. Hardi Purba, and F. Debora, "Plan Do Check Act (PDCA) Method: Literature Review and Research Issues," *Jurnal Sistem dan Manajemen Industri ,* vol. 4, no. 1, p. pp. 72–81, 2020.

[59] J. Pallant, SPSS Survival Manual: A Step by Step Guide to Data Analysis Using IBM 5th ed., Crows Nest, NSW, Australia: Allen & Unwin, 2013.

Annexes

Annex A: Data collection Instrument Questionnaire

Addis Ababa University

College of Natural and Computational Science

Department of Computer Science

Questionnaire for Thesis Fulfillment of Master of Computer Science Degree

Dear respondents,

This questionnaire is prepared by Mr.Nigusu Daniel a graduate student of Addis Ababa University, department of computer science who currently working thesis in title of "*Data-Driven Quality Assurance in Agile Software project: Optimizing Scrum Checkpoints through a Conceptual Framework*" to identify and assess practice on Quality Assurance in Agile Software project especially on Scrum Checkpoints through a Conceptual Framework. It is designed to collect information about your experience with agile methods and practices **related to Quality assurance and Quality Checkpoints.** Your response to the items of this questionnaire will be used to analyze the existing agile practices and to investigate the practice of QA using data analysis techniques to enhance software quality assurance in agile projects. Additionally, it aims to develop and validate a conceptual framework for QA in agile methodologies.
I hope you will be able to take some time and carefully complete this questionnaire. You are kindly requested to fill the questionnaire as frank and reasonable as possible. You don't need to mention your name and also you are assured that responses for these questions remain confidential and used for academic purpose only.

Thank you in advance for your cooperation!

Section 1: General Information

The following questions will not be used for identification purposes. Participant ID: _____
Just tick in a check box☞ ☑

1 **Gender**: Male ☐ Female ☐
2 **Your age:** Under 25 ☐ 25 to 35 ☐ 36 to 45 ☐ Over 45 ☐
3 **Educational status:** Diploma ☐ Post graduate ☐ Bachelor Degree ☐ PhD ☐
4 What is your role in managing agile projects?
☐ Project Manager ☐ Scrum Developer ☐ Business Analyst
☐ Other (please specify): _____

5 How many years have you been using an agile approach to manage and implement projects?

☐ 0–2 years ☐ 3–6 years ☐ 7+ years

6 What is the size of your Scrum team?

☐ 1–4 members ☐ 5–10 members ☐ 11–15 members ☐ 16+ members

7 What is the average duration of a Scrum sprint?

☐ 1–4 weeks ☐ 5–7 weeks ☐ 8–12 weeks

Section 2: Agile Adoption & Benefits

Please complete the following questionnaire with specific regard to the Agile software project management approach in your company to identify benefit and project success criteria. These questions are answer "Yes" or "No"

Note:

For this section Question 6-10 use your current or previous project as reference.

Tick on your response ☞ ☑

1 My organization adopted agile to accelerate software delivery. **Yes** ☐ **No**☐

2 Agile adoption has improved our ability to manage changing requirements. **Yes**☐ **No** ☐

3 Agile methodologies have increased stakeholder collaboration. **Yes**☐ **No**☐

4 Our agile projects have reduced project risks. **Yes**☐ **No**☐

5 Agile methods have helped in cost reduction. **Yes**☐ **No**☐

6 Customer satisfaction is the most important measure of our agile project success. **Yes**☐ **No**☐

7 Business value delivered is a key indicator of agile project success. **Yes**☐ **No**☐

8 We track project velocity to measure agile project performance. **Yes**☐ **No**☐

9 We compare planned vs. actual stories per iteration to measure success. **Yes**☐ **No**☐

10 Budget vs. actual cost tracking is important in our agile projects. **Yes**☐ **No**☐

Section 3: Auditing Agile Projects

Instruction

Please complete the following questionnaire with specific regard to the Agile software project management approach focusing on identifying **factors influencing the QA, a**nalyze ongoing agile projects per company to identify key QA practices and identify and optimize checkpoints in Scrum-based development. These questions have a 5-point Likert scales ranging from strongly disagree to strongly agree.

Please indicate your response by ticking on the appropriate box among option that best describes your experience on the agile methods with your project endeavor.

1. SD: **Strongly Disagree** 3. N: **Neutral** 5. SA: **Strongly Agree**
2. D: **Disagree** 4. A: **Agree**

Tick on your response ☞ ☑

No	ITEMS	SD	D	N	A	SA
		1	2	3	4	5
1	Product Vision (Plan)					
1.1	Created a product vision agreed upon by the Scrum team					
1.2	Identified critical stakeholders (Product Owner, Scrum Master, Scrum Team)					
1.3	Conducted training on Agile values and guiding principles					
1.4	Defined software quality objectives (aligned with business goals)					
1.5	Established Steering Committee for strategic oversight					
1.6	Developed epics and user stories with clear acceptance criteria					
1.7	Ensured alignment with agile values and guiding principles					
2	Product Backlog (Plan)	**SD**	**D**	**N**	**A**	**SA**
		1	**2**	**3**	**4**	**5**
2.1	Developed product requirements aligned with the product vision					
2.2	Maintained meeting minutes reviewing and approving backlog items					
2.3	Established a change request and approval process for backlog updates					
2.4	Conducted training for Scrum Master & Team on backlog management					
2.5	Product Owner formally owns the product backlog					
3	Release Planning (Do)	**SD**	**D**	**N**	**A**	**SA**
		1	**2**	**3**	**4**	**5**
3.1	Developed a release backlog aligned with the product backlog					
3.2	Created a release schedule covering multiple sprints					
3.3	Documented meeting minutes approving release planning decisions					
3.4	Defined iteration length and sprint planning structure					
4	Sprint Backlog (Check)	**SD**	**D**	**N**	**A**	**SA**
		1	**2**	**3**	**4**	**5**
4.1	Defined sprint requirements aligned with the product backlog					
4.2	Documented Sprint Planning meetings, task breakdowns, and estimates					
4.3	Maintained daily standup meeting minutes					
4.4	Existence of sprint progress report					
4.5	Existence of a sprint burn down chart to monitor the progress of the Scrum team					

4.6	Existence of scrum board used by scrum team to track its progress during sprint					
4.7	Existence of sprint block list					
4.8	Generated Sprint Review Reports and Retrospective Findings					
5	Product (Act)	**SD**	**D**	**N**	**A**	**SA**
		1	**2**	**3**	**4**	**5**
5.1	Existence of a formal working deliverables agreement					
5.2	Existence of a product deliverables acceptance report					
5.3	Existence of a project retrospective, including:					
5.3.1	Lessons learned from the project					
5.3.2	Improvements identified for future projects					
5.3.3	Issues encountered during the project					
5.3.4	Actions taken to resolve project issues					
5.3.5	Recommendations for upcoming projects					
5.4	Existence of a formal project closure contract					
5.5	Existence of a post-project support and maintenance agreement					

Declaration

I hereby declare that the study which is being presented in this thesis entitled "*Data-Driven Quality Assurance in Agile Software project: Optimizing Scrum Checkpoints through a Conceptual Framework*" is original work of my own effort and that all sources of materials used for this thesis have been fully acknowledged. The information presented in this thesis is true to the best of my knowledge and understanding. I, the undersigned, declare that this project work is carried out under the supervision of Ayalew Belay (Ph.D.). Indeed, I would also declare that this thesis is not submitted to this university or any other academic institution anywhere for the award of any academic diploma or degree.

Nigusu Daniel Signature_____ Date:_____

Candidate

This is to certify that the above declaration made by the candidate is correct to the best of my knowledge.

Ayalew Belay (Ph.D.) Signature_____ Date:_____

Advisor