Tampere University of Applied Sciences

Software Quality Assurance and Testing

Sudikshya Bhattarai

BACHELOR’S THESIS
May 2024

Bachelor’s Degree Programme in Software Engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Software Engineering

BHATTARAI, SUDIKSHYA:
Software Quality Assurance and Testing

Bachelor's thesis 34 pages, appendices 2 pages
May 2024

The purpose of this thesis was to explore the complexities of software quality assurance and testing in software development, emphasising the crucial role of Software Quality Assurance (SQA), and testing in ensuring the software meets high standards in today's competitive market. The goal of this research was to improve current QA and testing methods to meet the rapidly changing technological environment.

The research was conducted based on three primary areas: a literature study, a survey, and case studies on large technology companies. Initially, an in depth analysis of current literature revealed important patterns and topics in software quality assurance and testing. Additionally, quantitative data on current SQA and testing practices was collected from industry professionals and non-professionals through a survey conducted using Google Forms which helped to analyse the understanding level and ongoing trends. Finally, case studies of leading companies such as Google, Microsoft, and Apple were examined to comprehend their approaches to quality assurance. This mix of approaches offered a thorough perspective on SQA and testing methods, guaranteeing dependable and precise results.

The finding indicates that as the software industry progresses, there is an increasing requirement to incorporate automation and utilise AI/ML for better efficiency in SQA practices. Through focusing on these developments and adjusting to a technology-centered future, SQA can guarantee the provision of high-quality, dependable software that satisfies the needs of a competitive market.

Key words: software quality assurance, software testing, evolution, quality models

CONTENTS

ABBREVIATIONS AND TERMS

SQA	Software Quality Assurance
IT	Information Technology
QA	Quality Assurance
ISO	International Organization for Standardization
IEEE	Institute of Electrical and Electronics Engineers
ANSI	American National Standards Institute
CMM	Capability Maturity Model
IEC	International Electrotechnical Commission
SDET	Software Development Engineer in Test
SE	Software Engineer
AI	Artificial Intelligence
ML	Machine Learning
CI/CD	Continuous Integration/ Continuous Deployment

1 INTRODUCTION

In today’s interconnected world, software has become an essential tool embedded in every aspect of modern life, placing it in high demand. As technology progresses rapidly, maintaining software quality has become a significant challenge for developers and users alike. Customer demands for bug-free, high-quality software necessitate effective Software Quality Assurance (SQA) and testing. SQA and testing play vital roles in ensuring software meets user expectations by ensuring functionality and accuracy.

This thesis investigates the complexities of QA and testing within the context of software development. Innovative solutions are sought to address the challenges posed by the rapidly evolving technological landscape. Achieving high-quality, bug-free software requires a balance between creativity and practicality while keeping pace with new technology. In today's competitive world, companies need strategies to deliver high-quality software that meets user demands. Testing and QA processes are vital in ensuring software works correctly and accurately. Despite technological advancements, traditional testing methods remain important for proper software development. This thesis seeks to identify current challenges and trends in QA and testing practices.

According to Galin and Smith, SQA is a methodical and scheduled strategy intended to ensure that the software creation or upkeep process complies with approved functional and technical criteria. By combining ideas from Galin's research with the differentiation between external and internal quality, we obtain a thorough comprehension of the essential role played by SQA in guaranteeing the effectiveness of software development processes both functionally and operationally. This comprehensive comprehension serves as the foundation of our study, leading to improvement current methods and addressing new obstacles in the ever-changing field of SQA and testing (Smith, J, Wood, & B, 2012; Galin, 2004).

In the study by Smith, it was mentioned that QA and testing are crucial aspects of software development, yet they present unique challenges. Traditional approaches may struggle to adapt to the evolving landscape of modern application development, necessitating adaptability and flexibility. Companies often face tight time frames and resource limitations, complicating comprehensive testing processes. The pressure for speedy product delivery can result in incomplete testing and undiscovered weaknesses, with some issues only manifesting once the software reaches end-users (Smith, J, Wood, & B, 2012). Therefore, while QA and testing are important, overcoming these challenges requires innovative approaches and a willingness to adapt to the ever-changing environment of software development.

1.1 Research scope

This thesis explores the intricacies of SQA and testing in contemporary software development. It analyzes current approaches, emphasizing their advantages and drawbacks, and discusses how conventional testing techniques may find it challenging to keep pace with fast technological progress.

An essential factor involves comprehending the role of SQA in upholding functional and technical standards, as outlined by Galin and Smith, and examining internal and external quality aspects (Galin, 2004). The project also tackles obstacles like strict timeframes and scarce resources, as pointed out by (Smith, Johnson, & Williams, 2021) ,which may impede thorough testing.

The objective is to conduct a thorough evaluation of present and past QA and testing methods in the software industry. This study seeks to identify challenges and limitations in software quality methodologies over time and provide insights into current approaches. This effort is crucial for gaining a more thorough comprehension of the complexities required to guarantee software reliability, scalability, and meeting user standards in the current highly competitive environment.

1.2 Historical background

Kan (2003) discussed the historical background of software engineering's evolution and growing significance of software quality over time. In the 1960's and before, it could be seen as the period when Information technology (IT) catered to institutional requirements by connecting software to the daily functions of Institutions. The 1970s can be seen as when industries faced significant delays and increased cost in their projects, with an emphasis on planning and controlling of software projects. The 1980s can be seen as the cost era where costs continued to decline, and IT was widely spread among different organizations and Institutions and soon became available to Individual users. The 1990s can be viewed as a time of increased quality and productivity (Kan, 2003).

Earlier when software development was just introduced, the act of QA was not as meaningful as today. Back then QA was just a formality for making ensure that software meets the development purposes. Companies were doing QA simultaneously with software development, but it was just a post checking without the capacity of making any changes in case if an error occurred (Tummala, 2022).

Based on historical evidence, we can determine that the present time is considered a quality stage era, where SQA and testing have become dominant, and ‘quality’ is now the most crucial factor for corporate success and growth. Software companies are finding it increasingly difficult to create bug free software, and without the right tools, projects are at risk of failing. Even a minor flaw could result in significant software failures, resulting in substantial financial losses for businesses and detrimental effects on the quality of the software product. To achieve quality software, testing has become the prominent technique across the industry. Therefore, it is essential for software development to be more effective, ensuring that the quality of the delivered software meet customer expectations to achieve success (Chopde, 2011).

2 SOFTWARE QUALITY

The formal definition of quality given by International Standards organization (ISO) in 1986: “The total features and characteristics of product or service affecting its ability to meet needs” (Gillies, 2011).

Further emphasizes the 1986 definition by ensuring the product's quality and its ability to deliver the service as stated. It acknowledges that this is accomplished by the product's attributes and qualities. Quality is linked to possessing the necessary attributes and meeting performance expectations in each attribute (Gillies, 2011).

ISO9001:2008 expands this perspective: “The quality of an object can be assessed by comparing its inherent features with a set of criteria. When those innate qualities fulfill all criteria, high or outstanding level quality is achieved. If those characteristics fail to meet all requirements, a low or poor level of quality is achieved.”

Thus, it can be said that “Quality is a very important necessity for the continuous growth of any organization in the worldwide competitive scenario and the same applies to software quality.”

Burnstein (2006), cited the IEEE standard glossary of software engineering terminology which defines software quality as the level at which a system, component, or process meets specified requirements and satisfies needs and expectations of customer or user.

Compliance with clearly defined performance requirements and functional, documented development standards, and implicit characteristics is expected in all professionally developed software (Pressman, 2005). Further elaborated on the importance of incorporating quality into software development, consistently assessing it, and making necessary corrections as needed (Godbole, 2004).

In software development, ensuring the quality is based on the various components of quality. Numerous software quality factors models and their classification have been proposed over time. It is crucial to assess software quality projects consistently with efficient quality assurance processes. The methods aim to stop defects and testing systems and processes for removing defects. This will allow the development of high-quality software with the desired features and successful delivery.

2.1 Software Quality Assurance

Gallin (2004) stated in his book, the most used definition of SQA is proposed by the IEEE glossary. According to this, software quality assurance involves two main aspects:

1. It is essential for ensuring that an item or product meets established technical requirements through a meticulously planned and organized set of actions.
2. It encompasses a series of tasks designed to evaluate the process of developing or producing products, distinguishing itself from quality control.

However, the author thinks that the scope of SQA is restricted by the IEEE definition, which doesn't include maintenance, timetable, and budget issues. A more detailed explanation of the SQA functions is expected to result in better results and higher levels of customer satisfaction. SQA goes beyond just the development process, which sets it apart from the IEEE definition. It must be expanded to offer various functions, including resolving software maintenance issues, and facilitating product delivery procedures. In addition, software quality assurance activities should cover more than just the technical aspects of functional requirements, including tasks related to setting schedules and budgets (Galin, 2004).

According to the author, SQA is defined as a systematic and structured process aimed at ensuring software development or maintenance meets functional and technical requirements, as well as adhering to deadlines and financial constraints(Galin, 2004).

2.2 Software Quality Models

Quality Models are developed to improve the quality of a product. Similarly, to ensure the quality of software, different quality models are used. Based on quality metrics and immense research, different software quality models are introduced by international organizations such as: IEEE, ISO etc. The model introduced by these organizations is accepted widely in the world and maintains the standard of their software which ensures trust among the customers. Models can be different to different organizations, e.g.: small organization may need a model that is feasible for its organization to improve quality and similar with medium and large organization. Models should be flexible enough for future scalability and maintainability.

2.2.1 McCall's Model

This is the initial quality model introduced by Jim McCall in 1997. This model identifies two tiers of quality factors. The first level of quality factors is outward and easily measurable. The second aspect quality includes criteria that are assessable either subjectively or objectively (Suman, Wadhwa, & Rohtak, 2014). The aspects of software quality encompass product functionality, product modifications, and product migration.

2.2.2 Boehm's Quality Model

Barry W. Boehm introduced this model to overcome and address the limitations of McCall's model. This model includes factors such as portability, maintainability, usability, human engineering, testability, understandability, and flexibility as indicated by (Suman, Wadhwa, & Rohtak, 2014).

2.2.3 ISO 9126 Standard Quality Model

The objective quality standard model aims to develop a quality framework for software and a set of criteria to measure its characteristics. It is an enhanced form of ISO 9000 and helps lessen the problem of initial release. Since this model is universal, comparing one product to another is simpler. ISO 9126 quality model was introduced as a global standard for measuring software quality. It outlines 21 quality factors that must be exhibited by a quality software product (Al-Qutaish & Rafa, 2010).

ISO 9126 describes as a standard comprising of four parts:

1. The revised model for quality is established by ISO/IEC 9126-1 (Al-Qutaish & Rafa, 2010).
2. A series of external measures are outlined by ISO/IEC 9126-2 (Al-Qutaish & Rafa, 2010).
3. A series of internal measures are outlined by ISO/IEC 9126-3 (Al-Qutaish & Rafa, 2010).
4. A range of metrics for quality during use is established by ISO/IEC 9126-4 (Al-Qutaish & Rafa, 2010).

In Figure 1, the two-part quality model includes the initial division into six characteristics, which are then further divided into sub-categories for both quality measures among (Al-Qutaish & Rafa, 2010).

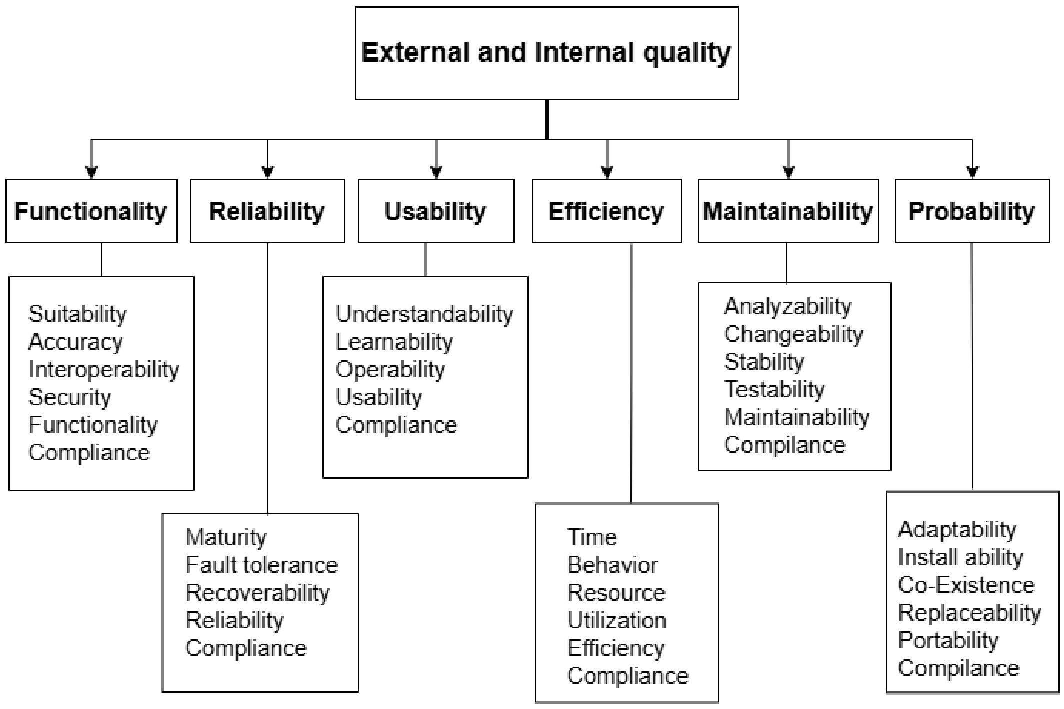


Figure 1. ISO 9126 quality model for external and internal quality characteristics (Al-Qutaish & Rafa, 2010).

As in Figure 2, the second part of the model represents four qualities in usage features (Al-Qutaish & Rafa, 2010).

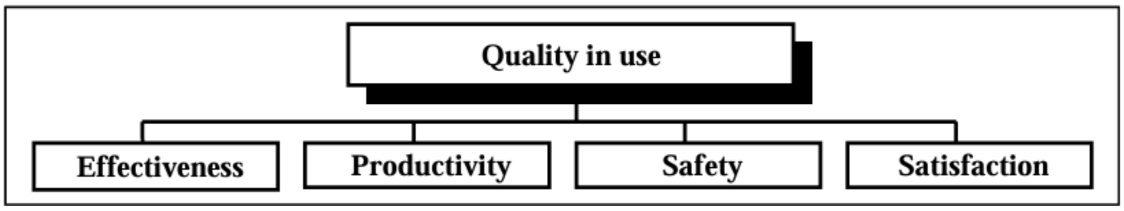


Figure 2. ISO 9126 quality model for quality in use characteristics (Al-Qutaish & Rafa, 2010).

The initial document in the ISO 9126 series, known as the Quality Model, presents a two-parts quality model for assessing software product quality:

I. Model of both internal and external quality.
II. Model of quality in use.

2.2.4 Capability Maturity Model

The Capability Maturity Model (CMM) is a technique employed to enhance a company's software development approach. It describes five levels of processes that mature in a structured and systematic way as they progress. CMM outlines five levels: initial, repeatable, defined, managed, and optimizing. This model is like ISO 9001, part of the ISO 9000 standards. ISO 9001 has a basic standard for software process quality, whereas the CMM creates a framework for ongoing enhancement and is more detailed than ISO in outlining the approaches to reach this objective (Al-Qutaish & Rafa, 2010).

2.2.5 IEEE Model

IEEE is fundamentally a guideline for software upkeep designed to provide a top-notch framework. It includes extra regulations such as SQA, verification, and validation. This model shows the resulting aspects: ease of use, dependability, ease of transport, and ease of maintenance.

2.3 Software Testing

Testing is described in the ANSI/IEEE 1059 standard as examining a software item to detect differences between present and desired states (defects, errors, bugs) and evaluating the functionalities of the software product (Singh & Singh, 2012).

Software testing is the primary stage of software development. It is utilized for assessing the caliber, performance, and dependability of the software product. Conducting functional tests on the software helps uncover flaws or inconsistencies that may exist between user expectations and the actual results. It is crucial for this approach that the software meets the designated requirements and functions correctly to avoid potential risks of failure and vulnerability. Testing that is thorough leads to a user experience improvement through delivering a product that has a stable performance and is both time and energy efficient. With the help of testing, the overall cost of developing software and product release time can be significantly reduced by early identification of errors. Be it conducted manually or the automated way, software testing is vital for validating functionality, improving quality, and safeguarding reputation so that users can have high confidence in the software performance resulting in customer satisfaction and business success.

2.3.1 Software testing methods

The software testing process comprises different software testing types and techniques that are performed during the testing time to ensure the code is corrected from the defects before the software is released to the market.

Verification and validation are commonly utilized in software development and testing; however, they have distinct meanings. These variances play a crucial role in software testing. Verification involves ensuring that a software adheres to its specifications, while validation involves confirming that it meets the user's requirements. Verification and validation involve assessing whether a software system meets specified requirements and standards, ensuring it meets its intended purpose.

2.3.2 Functional testing

Functional testing is a form of software testing that confirms the functionality of a software system or application. Its primary focus is on guaranteeing that the system operates in line with defined functional requirements and fulfills the necessary business requirements(Bath & McKay, 2014).

Functional testing is carried out to verify the system's features, abilities, and interactions with different components. This involves assessing the software's input and output, processing data, engaging with users, and examining how the system responds to various situations and conditions. Functional testing is centered on verifying that a system functions as intended.

Various types of functional testing include:

1. Unit Testing: This includes testing individual software modules or components to make sure they function as intended. Developers typically conduct out unit testing, which is frequently automated.
2. Integration Testing: After Integration of each component and modules, integration testing is conducted to ensure they work correctly when integrated. It ensures that these components function together as a system.
3. System Testing: It assesses the performance of the complete software system. It ensures that the system complies with the specified criteria and operates as planned.
4. User Acceptance Testing: User Acceptance Testing involves confirming that the software meets both the end users' needs and the business objectives. Stakeholders or end users typically finalize the software before it is approved for distribution.

2.3.3 Non-Functional Testing

Non-functional testing does not evaluate the functional aspects of the software system but rather on its performance, reliability, load testing and other quality aspects.

Various types of non-functional testing are:

1. Performance Testing: In this testing, software is tested based on its speed, responsiveness, scalability, and stability. It includes load testing, stress testing and scalability testing to determine system behavior under different scenarios.
2. Security Testing: To protect data, maintain confidentiality, integrity and availability, security testing is done in software. Also, to avoid unauthorized access, vulnerabilities and threats security testing must be done.
3. Compatibility Testing: It ensures that software works without any issues in different platforms, devices, operating systems, browsers, and network environments. It ensures that the system does not have any compatibility issues while using a wide range of devices.
4. Scalability Testing: It evaluates how well the software can handle the workload and user load by scaling up the infrastructure. It helps to ensure that

the system can handle future growth and maintain performance levels as the demand increases**.**

5. Load Testing**:** Through load testing, the functionality of the software is evaluated under both peak and expected load scenarios. It assists in locating resource usage, reaction times, and performance bottlenecks to make ensuring the system can manage the expected workload.

2.3.4 Black-box testing and white-box testing

As stated by Patton (2006), during black-box testing, the tester is only aware of the expected functionality of the software. He is unable to peek inside the box to learn how to use it. If he enters a specific input, he receives a specific output. He is unaware of the how or why, only that it occurs. Testers design test cases based on the system's specifications and requirements, focusing on uncovering defects related to incorrect functionality, missing features, or usability issues. This method is advantageous when the internal implementation details are complex or inaccessible, as testers do not need knowledge of the underlying code to conduct effective testing (Patton, 2006),.

In white-box testing, the software can review the program's code for assistance in testing. He can view the contents of the box. Testers can adjust their testing methods based on their observations to focus on numbers that are deemed to have a higher or lower likelihood of failure. White-box testing is beneficial for identifying defects related to logic errors, boundary conditions, and performance bottlenecks, as well as improving code maintainability and readability. It provides insights into the quality of the code and helps enhance the overall reliability and robustness of the software (Patton, 2006).

2.3.5 Static and dynamic testing

Patton (2006) discusses in his book that two alternate terms for describing software testing are static testing and dynamic testing. Static testing involves examining and evaluating code without running it. Contrastingly, dynamic testing consists of running the software and monitoring how it behaves to evaluate its accuracy, efficiency, and other traits of quality.

Static testing looks at the software documentation, requirements, design documents, code, and other related artifacts to find defects, inconsistencies, and quality problems. Static testing methods consist of code reviews, walkthroughs, inspections, and static code analysis. Static testing plays a key role in identifying problems early in the development process, thus cutting down on the expense of addressing defects and enhancing the overall quality of software (Patton, 2006).

Dynamic testing verifies the software's functionality by testing it against requirements and specifications using different test cases. It includes different types such as unit testing, integration testing, system testing, acceptance testing, and regression testing. Dynamic testing is useful for identifying issues regarding functionality, performance, security, and reliability by examining how the software acts in various scenarios (Patton, 2006).

2.3.6 Test Automation

To overcome the limitations of manual testing, test automation has been introduced. Test automation basically means using special tools and scripts to run tests automatically, instead of having people do it all by hand manually.

Though automation of tests takes a lot of investment at initial phase but over time it saves a lot of cost which can be seen in Figure 3. Manual testing takes a lot of time to test the different scenarios in a software, but automated testing takes less time and evaluate possible scenarios of tests at no time (Nextgen Technology Ltd., 2023).

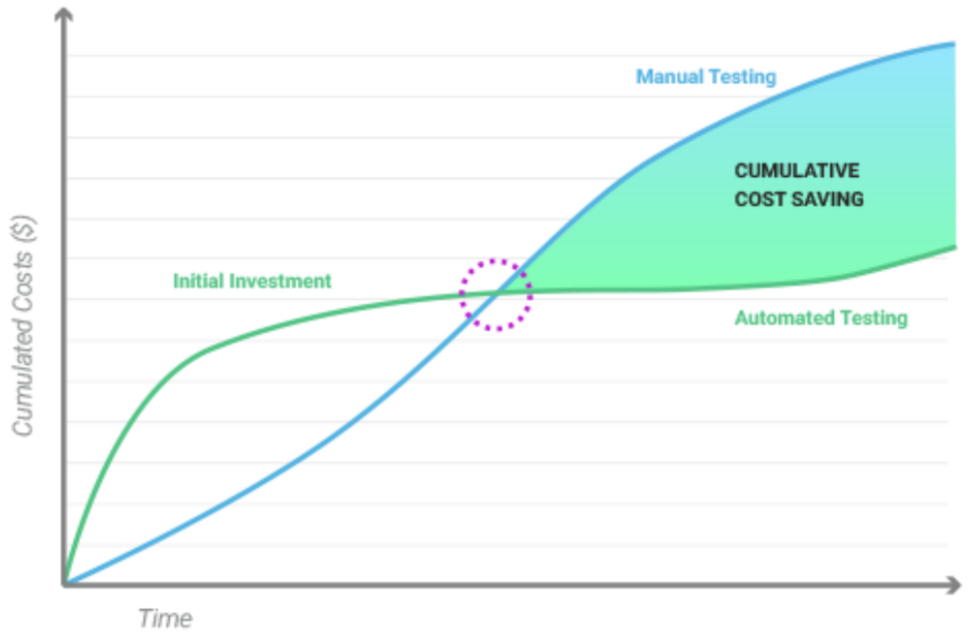


Figure 3. Costing in automated testing vs Manual testing (Nextgen Technology Ltd., 2023)

Automated testing is an ongoing and future trend in SQA field. It will make significant adjustments to guarantee the software's quality. Switching to automated testing has range of benefits:

1) Automated tests provide a uniform testing method, ensuring consistent results each time.
2) Automated is costly at upfront investment but has significant long-term savings.
3) Automation ensures the speed of tests without compromising its quality.
4) It offers the depth of the testing by covering all the possible scenarios.

3 RESEARCH METHODOLOGY

Three different research methods were used to investigate SQA and testing. First, a comprehensive review of the existing literature was conducted to identify key trends and themes in the field. Secondly, case studies from major companies such as Google, Microsoft, and Apple are analyzed to provide insight into their quality assurance strategies. Lastly data was collected through a survey conducted using Google Form. These methodologies collectively provided important insights into SQA and testing practices, enriching the understanding of the subject matter.

3.1 Data collection from existing research

The first step in the data collection process was taken by gaining a deep understanding of the research landscape surrounding our study. To accomplish this, data was gathered through trusted sources such as textbooks, journals, articles, and conference research papers. Additionally, useful information was collected from authoritative websites and other authoritative sources with deep industry knowledge.

Thorough reviews of previous studies and current progress were aimed at building a strong fundamental understanding. This involved both consolidating existing information and identifying common themes and emerging trends within the field.

This thorough examination not only prepared for the research but also provided important insights into the future direction of this field. By analyzing current progress, future trends and future paths within this field were predicted.

3.2 Case Studies

Case study analyses the quality assurance strategies at big companies like Google, Microsoft, and Apple were also looked at. Valuable insight into how large software development environments implement quality assurance strategies was provided by these real-world case studies.

3.2.1 Quality Assurance at Google

Google, a top technology firm globally, is recognized for its innovative products and services that have transformed the way we access and utilize information and technology. Hence, delivering high-quality software to users is extremely crucial.

Google's approach to software testing is based on three fundamental principles as outlined by Whittaker. To start with, the company highly prioritizes speed, focusing on quickly developing and launching products. This method focuses on shorter release cycles while maintaining quality, as Google understands that in the competitive world of software development, being the first to launch can result in success. Additionally, Google depends greatly on automation to guarantee the effectiveness and precision of its testing procedure. The company prioritizes automation infrastructure to optimize testing procedures, with manual testing being used only as a final option. Finally, Google promotes a culture of testing, urging developers to assume responsibility for the quality of their code and play an active role in the testing procedure. This culture makes sure that all members of the organization are accountable for upholding superior quality standards during the entire development process. (Whittaker, et al., 2012).

3.2.2 Quality Assurance at Microsoft

In the middle of 2014, Microsoft replaced the Software Development Engineering Test (SDET) position with the SE position, taking inspiration from the larger web team at Bing. The shift transferred testing duties to developers, with QA concentrating on testing for end-users. This change happened at the same time as Microsoft's biggest layoff ever, impacting 18,000 workers, including numerous SDET positions. Although there were initial worries, the transition occurred without any issues, especially helping teams that have daily shipment schedules. In

2017, Brian Harry from the Visual Studio Team Services team observed notable enhancements in testing productivity and responsibility after the integration of development and testing departments. This merger removed the separation of coding and testing responsibilities, making each team member responsible for the quality of the product. The revamping of testing methodologies led to a more agile and responsive quality assurance approach within Microsoft (orosz, 2023).

3.2.3 Quality Assurance at Apple

At the core of Apple's reputation lies a rigorous quality assurance and testing approach that permeates every aspect of product development, from hardware to software. At the heart of this approach lies an unwavering commitment to user experience, which is demonstrated through extensive usability testing. By including real users in the testing process, Apple can identify interface inconsistencies and areas for improvement, creating a smooth and intuitive user experience.

Beyond usability testing, Apple's testing strategy extends to functional, performance and security. Functional testing evaluates each feature's functionality under a variety of conditions. Performance testing evaluates speed, stability, and responsiveness under pressure. Apple's stringent security testing seeks to identify and correct vulnerabilities, protecting user data and privacy, while tight integration of QA and development ensures rapid issue resolution.

Apple's tiered testing approach involves internal QA teams and employee testing, as well as public beta programs. By continuously soliciting and analyzing user feedback, Apple drives iterative improvements throughout the product's life cycle (Beta Breakers, 2023).

3.2.4 Comparative Study of Approaches Employed by: Apple, Microsoft, and Google

Table 1. SQA Approaches in Apple, Microsoft, and Google (Beta Breakers, 2023; orosz, 2023; Whittaker, et al., 2012).

Company	Approach to Quality Assurance	Key Strategies and Outcomes
Google	Emphasizes expeditious software development with stringent quality standards. Implements abbreviated release cycles for rapid deployment. Relies heavily on automation for efficient testing processes. Cultivates a testing-centric culture among developers.	• Implementation of abbreviated release cycles for rapid deployment. • Heavy reliance on automation for efficient testing processes. • Cultivation of a testing-centric culture among developers.
Microsoft	Decentralizes testing responsibilities from dedicated SDET roles to developers. Integrates development and testing departments for holistic quality assurance. Enhances testing productivity and accountability.	• Decentralization of testing responsibilities to developers. • Integration of development and testing departments. • Enhancement of testing productivity and accountability.
Apple	Prioritizes user experience excellence through extensive usability testing. Conducts functional, performance, and security testing for robust feature functionality. Implements a tiered testing approach involving internal QA teams and beta programs based on user feedback.	• Emphasis on extensive usability testing for user experience refinement. • Conduct of comprehensive functional, performance, and security testing. • Continuous iteration based on user feedback.

3.3 Data Collection from survey

A questionnaire was created using Google Forms for participants to complete on their own. The survey was created to be brief and easy to understand, aiming at both IT experts and non-experts with different levels of SQA expertise. It consisted of the following questions:

1. *How would you rate your familiarity with Software Quality Assurance (SQA) principles and practices?*
2. *Have you been involved in software testing activities before?*
3. *What type of software testing methodologies are you familiar with? (Check all that apply)*
4. *What challenges do you face in ensuring the quality of software products during the development process?*
5. *How important do you think it is to implement Continuous Integration/Continuous Deployment (CI/CD) practices in software development for ensuring quality?*
6. *How do you measure the success of your software testing efforts?*
7. *How do you think Artificial Intelligence (AI) and Machine Learning (ML) can impact software testing in the future?*

A convenience sample of 25 IT professionals and non-professionals received the distributed survey. A 60% response rate was achieved with a total of 15 responses received. Although a bigger sample size would be preferable for making broader conclusions, the information obtained from this specific group offers valuable data for this thesis. The survey link was distributed electronically through email invitations. Participants were informed about the study's purpose, the voluntary nature of their participation, and the anonymity of their responses.

The quantitative data (Likert scale and multiple-choice questions) will be analyzed using descriptive statistics to identify trends and central tendencies. The open-ended responses will be subjected to thematic analysis to extract recurring themes and insights.

4 RESULTS AND ANALYSIS

This chapter reviews the outcome of the study on SQA and testing, drawing on foundational details from a writing review as well as case studies on notable tech firms. It begins by analyzing existing studies, focusing on critical trends, methods, and successful techniques, and pinpointing regions not yet covered by these studies. The section later analyzes case studies of leading tech companies, providing practical insights into real-world SQA practices, identifying effective strategies and common challenges. Finally, survey results from professionals and nonprofessionals in software development and testing are documented, capturing their experiences with current SQA procedures. This structure offers a comprehensive understanding of the topic.

4.1 Results from existing research

The existing research highlights a notable shift in companies' perspectives towards software quality and testing in today's digital landscape. Previously, the focus was primarily on speedy market deployment without much consideration for software quality. However, a significant realization has emerged, acknowledging that software quality is pivotal for customer satisfaction, cost reduction, and overall business efficiency.

One key finding is the recognition that upfront investment in software quality can yield substantial cost savings in the long run. Early identification and resolution of issues during development can mitigate the need for costly rework, minimize the risk of errors, and streamline operational processes. This proactive approach not only reduces the financial burden associated with post release defect fixes but also fosters a culture of continuous improvement within organizations.

However, despite these advancements, existing research also reveals several limitations and challenges (Smith, J, Wood, & B, 2012). These include:

1. **Resource Constraints:** Many companies still face challenges in allocating adequate resources for comprehensive quality assurance and testing processes. Limited resources can hinder the thoroughness and effectiveness of testing efforts.

2. **Complexity of Modern Systems:** The increasing complexity of modern software systems poses challenges for traditional testing methodologies. Testing such systems requires innovative approaches to ensure thorough coverage and accuracy.

3. **Time Pressure:** Tight project schedules and deadlines often lead to compromises in testing, with a focus on meeting release timelines rather than ensuring comprehensive quality assurance. This can result in overlooked defects and reduced software reliability.

4. **Adaptability to Emerging Technologies:** As technology evolves rapidly, existing testing methodologies may struggle to adapt to new platforms, frameworks, and architectures. Ensuring compatibility and effectiveness across diverse technological landscapes remains a challenge.

5. **Integration with Development Processes:** Effective integration of quality assurance and testing processes with agile and DevOps practices is essential for seamless software development. However, achieving this integration can be complex and requires careful coordination and collaboration among teams.

4.2 Results from case studies

The case studies of Google, Microsoft, and Apple provide valuable insight into the different approaches of leading technology companies to quality assurance and software testing. These companies share commonalities and distinctive strategies that reflect the evolution of quality assurance practices in the technology industry.

Table 2. Comparative Analysis of case studies (Beta Breakers, 2023; orosz, 2023; Whittaker, et al., 2012).

Aspect	Google	Microsoft	Apple
Speed vs. Quality	Emphasizes rapid development with quality maintenance	Shifts focus to end-user-centric testing	Upholds unwavering commitment to user experience
Testing Automation	Relies heavily on automation	Utilizes automation for productivity	Utilizes automation alongside extensive manual testing
Developer Involvement	Cultivates a testing-centric culture	Integrates developers and testers for accountability	Iterates based on user feedback for continual improvement
Testing Methods	Abbreviated release cycles, automation	Decentralization of testing responsibilities	Extensive usability testing, functional, performance, and security testing
Key Outcomes	Rapid deployment without compromising quality	Improved productivity and accountability	Enhanced user experience, comprehensive product quality
Notable Strategies	Shorter release cycles, automation infrastructure	Integration of development and testing departments	Tiered testing approach, integration with user feedback

4.3 Results from survey

To understand familiarity with SQA principles and practices options were given and the survey revealed that most respondents (53.3%) possessed a moderate level of familiarity with SQA principles and practices. Approximately a third

(33.3%) indicated high familiarity, while a smaller portion (13.3%) reported no familiarity at all as shown in Figure 4. This highlights the need for continued education and awareness initiatives in the field of SQA.

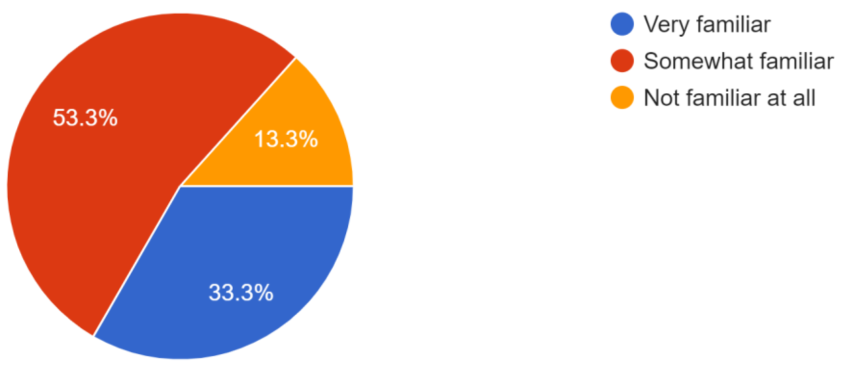


Figure 4. Pie chart representing familiarity with SQA practices and principles.

Nearly half of the participants (46.7%) confirmed prior involvement in software testing activities, suggesting a practical understanding within the sample group. The remaining half (53.3%) lacked such experience as shown in Figure 5, indicating a potential target audience for introductory SQA training.

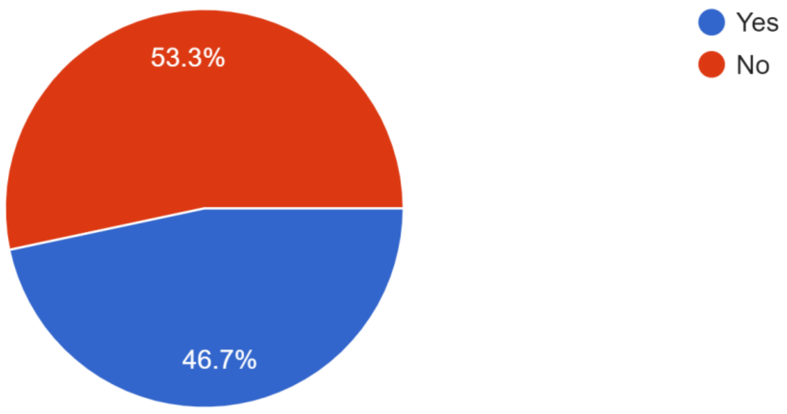


Figure 5. Pie chart representing involvement in testing activities.

In Figure 6, to understand the familiarity with software testing methodologies, most practiced testing methods were on the option where manual testing emerged as the most recognized methodology (78.6%), followed by automated testing (57.1%). Black-box testing (42.9%) and white-box testing (28.6%)

received less recognition, suggesting a potential focus area for enhancing testing expertise. Regression testing also garnered similar awareness (28.6%)

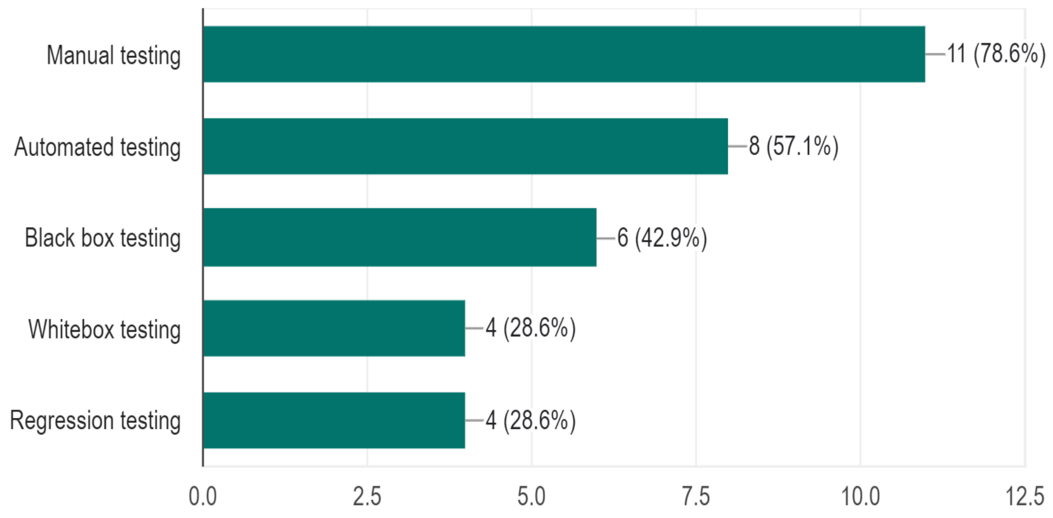


Figure 6. Chart on familiarity with software testing methodologies.

As shown in Figure 7, the majority (93.3%) of respondents recognized the critical importance of implementing CI/CD practices for maintaining software quality. This signifies a strong understanding of the benefits of CI/CD within the sample group.

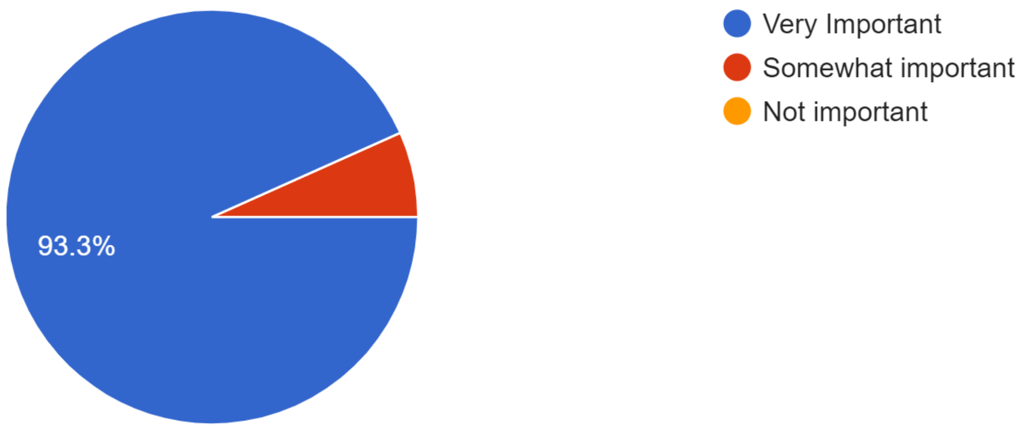


Figure 7. Pie chart representing importance of CI/CD

The survey revealed various significant obstacles in ensuring software quality. These include difficulties in managing changing requirements and maintaining clear communication within development teams, resource constraints hindering thorough testing efforts, handling the complexity of software systems, ensuring security throughout the development process, and limitations in testing to simulate real world scenarios. These observations pinpoint areas that require enhancements to improve SQA practices and results.

The survey results emphasize keyways to evaluate the effectiveness of software testing. This involves gathering feedback from end users to evaluate usability and reliability, monitoring test case pass/fail rates for stability, keeping track of identified defects, and ensuring adherence to both functional and non-functional requirements. These observations not just give important standards for assessing testing efficiency but also present useful ways for improving SQA practices.

The survey respondents predict that AI and ML will transform software testing by automating everyday tasks and potentially fixing bugs. AI powered tools are also noted for their capacity to automate the verification of software visuals across various platforms, offering enhanced efficiency, dependability, and scalability in testing procedures. However, they emphasize the crucial importance of human supervision in efficiently overseeing and enhancing AI powered procedures. These observations emphasize the revolutionary possibilities of AI and ML in influencing the future of software testing, emphasizing the importance of CI/CD practices to accompany these technological progressions.

5 CONCLUSIONS

This thesis has offered a thorough examination of the development of SQA and testing techniques, emphasizing their crucial importance in the current competitive market. The success and growth of a business is highly dependent on creating sturdy and dependable products, highlighting the critical need for error free software development. SQA is essential for guaranteeing the quality of software products, and software testing is crucial for identifying and addressing bugs in the early stages of development. To satisfy customer demands and remain competitive, different software quality models have been created to provide organized methods for assessing and enhancing software quality.

The survey findings have provided insight into existing knowledge in SQA, pinpointing areas that need enhancement in approaches, with a particular focus on the importance of practices like CI/CD. Additionally, barriers and the growing significance of AI and ML in testing indicate possible improvements in SQA. Illustrative instances from top organizations demonstrate different strategies centered around quickness, connectivity, user interface, and protection, all supported by a mutual priority for automation. The focus of SQA is increasingly moving towards a tech centered setting, with a greater dependence on AI testing tools and the requirement for SQA experts to acquire appropriate skills.

This thesis aims to improve QA and testing methods by examining present and previous strategies, grasping common practices, and tackling limitations. Challenges and ongoing trends in SQA and testing practices have been identified through thorough exploration and analysis of these approaches. The goal of putting these solutions into action is to produce high quality software products without any bugs that align with the demands of the present competitive market.

REFERENCES

Al-Qutaish, & Rafa, E. (2010). Quality models in software engineering literature: an analytical and comparative study. *Journal of American Science, 6*, 166--175. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=77b51002b53d002278997c71c72eaf2300a87ec7

Bath, G., & McKay, J. (2014). *The software test engineer's handbook: a study guide for the ISTQB test analyst and technical test analyst advanced level certificates 2012.* Rocky Nook, Inc.

Beta Breakers. (2023, December 26). *The Journey of Apple Software Application Quality Assurance*. Retrieved from Beta Breakers: https://www.betabreakers.com/the-journey-of-apple-software-application-quality-assurance/

Burnstein, I. (2006). *Practical software testing: a process-oriented approach.* Springer Science \& Business Media.

Chopde, M. B. (2011). *A study of the role of software testing in quality assurance in selected software enterprises in Pune.* Savitribai Phule Pune University, Indian Institute of Cost and Management Studies and Research (INDSEARCH). Pune: Shodhganga@INFLIBNET. Retrieved from http://hdl.handle.net/10603/204163

Galin, D. (2004). *Software quality assurance: from theory to implementation.* Pearson education.

Gillies, A. (2011). *Software quality: theory and management.* Lulu. com.

Godbole, N. S. (2004). *Software quality assurance: Principles and practice.* Alpha Science Int'l Ltd.

Kan, S. H. (2003). Metrics and models in software quality engineering. Addison-Wesley Professional.

Nextgen Technology Ltd. (2023, September 04). *Calculating the ROI of Test Automation for Product Development*. Retrieved from nextgen: https://resources.nextgen-technology.com/roi-of-automated-testing

orosz, G. (2023, September 19). *How Big Tech does Quality Assurance (QA).* Retrieved 2024, from The Pragmatic Engineer: https://newsletter.pragmaticengineer.com/p/how-big-tech-does-qa?ref=blog.pragmaticengineer.com

Patton, R. (2006). *Software testing.* Pearson Education India.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach.* Palgrave macmillan.

Singh, S. K., & Singh, A. (2012). *Software testing.* Vandana Publications.

Smith, J, D., Wood, & B, K. (2012). Engineering quality software: a review of current practices, standards and guidelines including new methods and development tools.

Smith, J., Johnson, A., & Williams, B. (2021). Software Quality Assurance and Testing: A Review. *Journal of Software Engineering*.

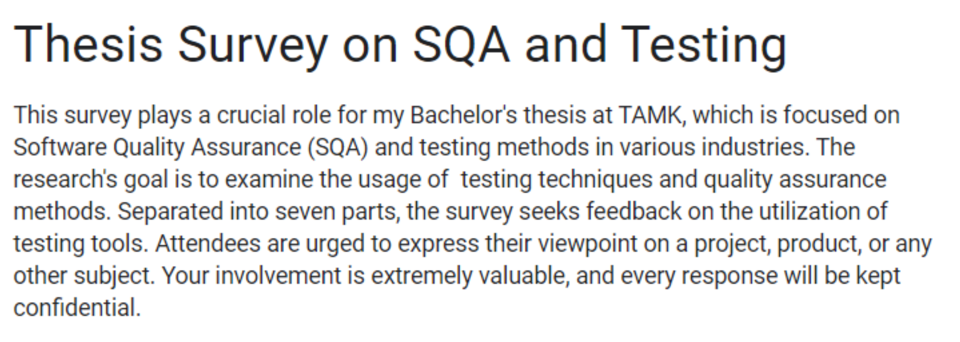
Suman, Wadhwa, M., & Rohtak, M. (2014). A comparative study of software quality models. *International Journal of Computer Science and Information Technologies, 5*, 5634--5638. Retrieved from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a9bd1630952307a7f1fc9e8f5f0b35135dfe85ef

Tummala, M. (2022). *An Evolution from Software Quality Assurance to Test Engineering*. Retrieved from Get Software Services.

Whittaker, A, J., Arbon, Jason, Carollo, & Jeff. (2012). *How Google Tests Software.* Addison-Wesley. Retrieved from https://www.linkedin.com/pulse/how-google-has-improved-quality-its-products-qa-my-from-sarkisian/?trackingId=f1VBirN6QomBTgUdOyK%2F2g%3D%3D

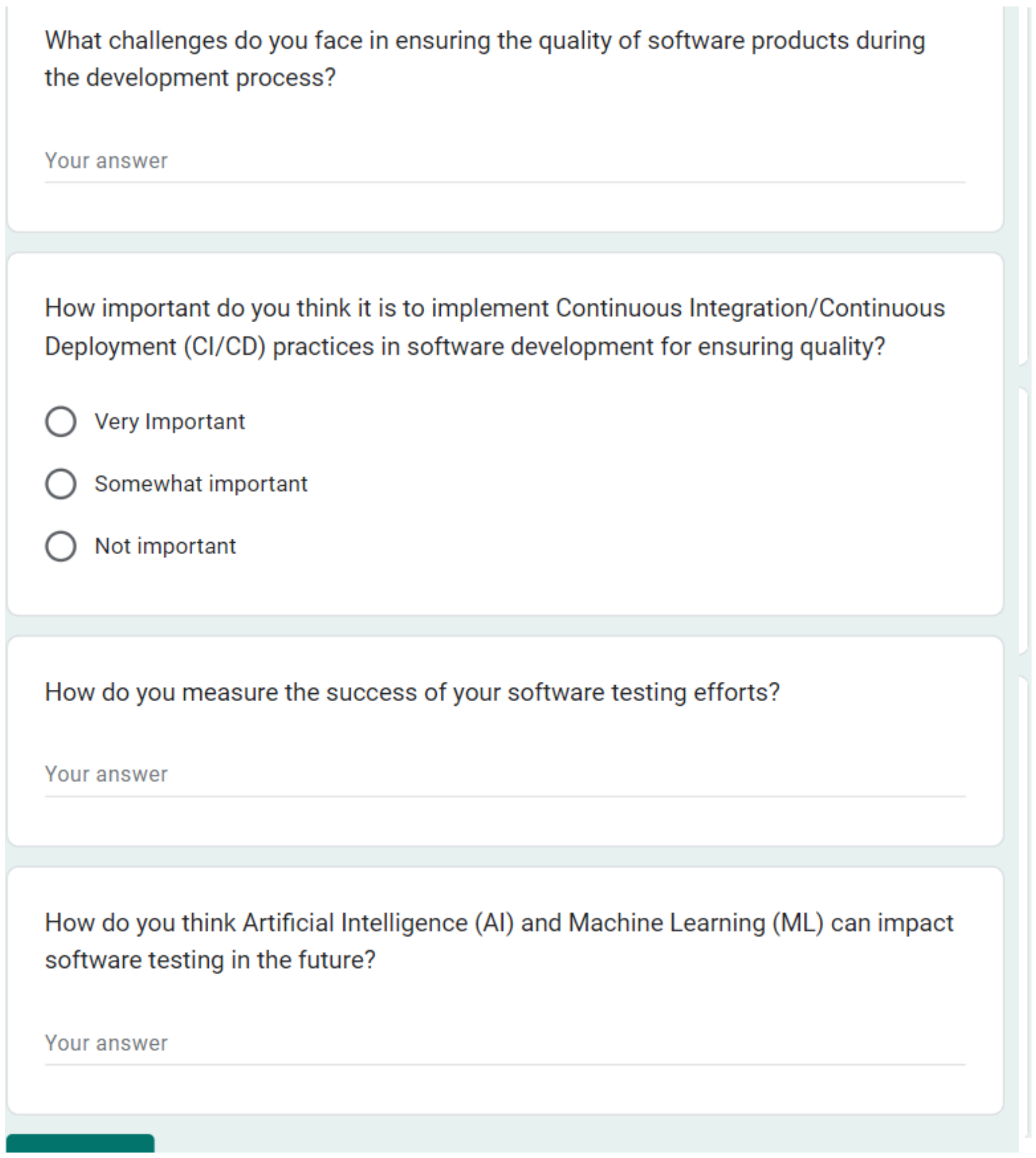
APPENDICES

Appendix 1. Questionnaire for the participants of the survey



Thesis Survey on SQA and Testing

This survey plays a crucial role for my Bachelor's thesis at TAMK, which is focused on Software Quality Assurance (SQA) and testing methods in various industries. The research's goal is to examine the usage of testing techniques and quality assurance methods. Separated into seven parts, the survey seeks feedback on the utilization of testing tools. Attendees are urged to express their viewpoint on a project, product, or any other subject. Your involvement is extremely valuable, and every response will be kept confidential.



What challenges do you face in ensuring the quality of software products during the development process?

Your answer

How important do you think it is to implement Continuous Integration/Continuous Deployment (CI/CD) practices in software development for ensuring quality?

○ Very Important

○ Somewhat important

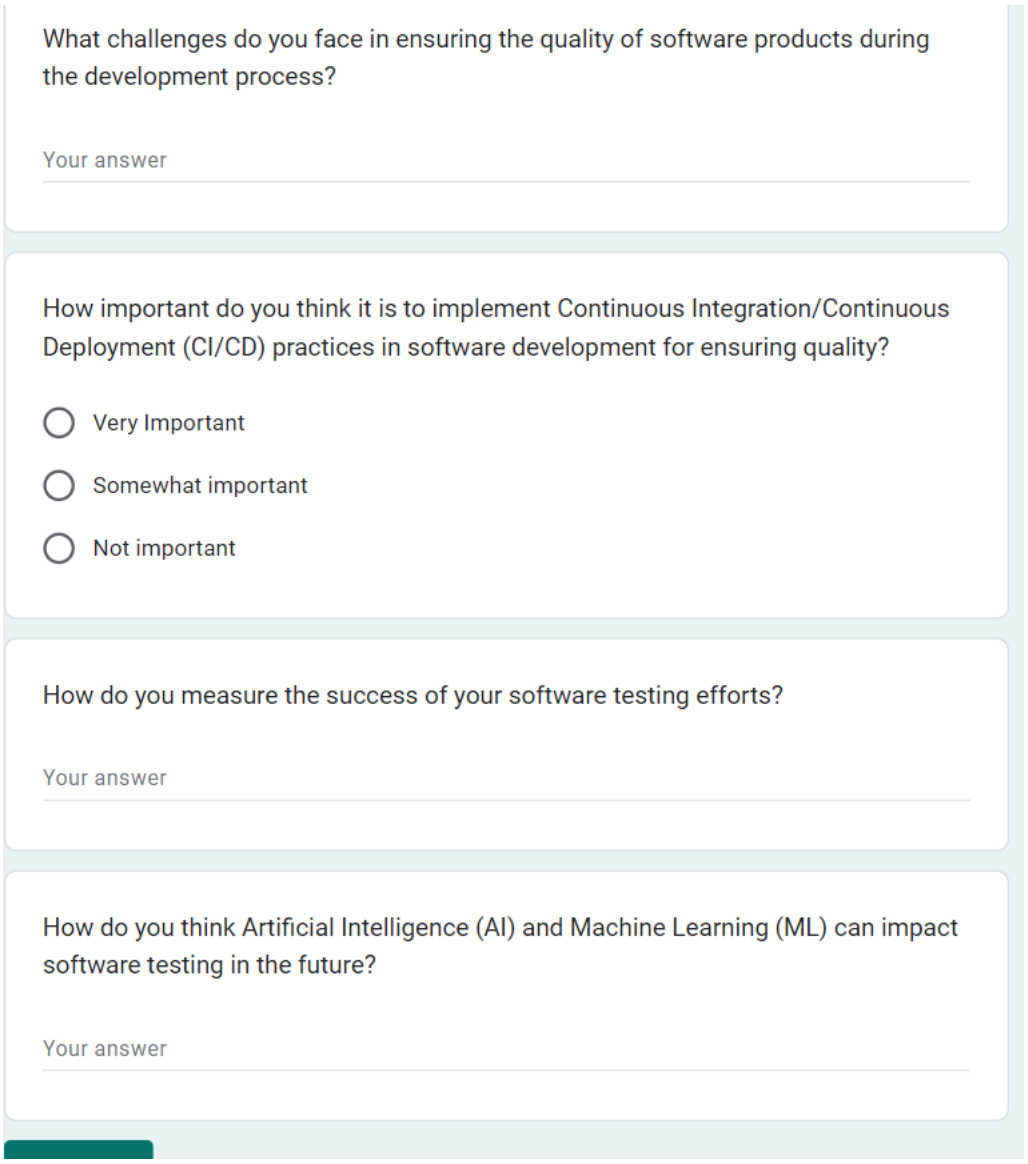
○ Not important

How do you measure the success of your software testing efforts?

Your answer

How do you think Artificial Intelligence (AI) and Machine Learning (ML) can impact software testing in the future?

Your answer



What challenges do you face in ensuring the quality of software products during the development process?

Your answer

How important do you think it is to implement Continuous Integration/Continuous Deployment (CI/CD) practices in software development for ensuring quality?

○ Very Important

○ Somewhat important

○ Not important

How do you measure the success of your software testing efforts?

Your answer

How do you think Artificial Intelligence (AI) and Machine Learning (ML) can impact software testing in the future?

Your answer