Aalto University
School of Electrical
Engineering

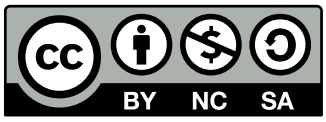
Master's Programme in Automation and Electrical Engineering

The impact of Quality Assurance and Security measures on software projects

Nico Renlund

Master's Thesis

2024



Aalto University
School of Electrical
Engineering

Author Nico Renlund

Title The impact of Quality Assurance and Security measures on software projects

Degree programme Automation and Electrical Engineering

Major Control, Robotics and Autonomous Systems

Supervisor Prof. Valeriy Vyatkin

Advisor MSc Musa Jallow

Collaborative partner Huld Oy

Date 13 December 2024 **Number of pages** 71 **Language** English

Abstract

The software industry is constantly evolving, demanding constant awareness of new threats and compliance with updated guidelines in the management of software projects. This thesis examines the integration of Quality Assurance (QA) and security measures within software development projects, highlighting their essential role in creating robust and reliable end products. This was achieved through the analysis of qualitative data collected from interviews with Huld QA and security experts, as well as with product managers and owners of Huld customer organizations. Additional insights were obtained from a review of internal communication channels. The obtained data allowed the identification of best practices, challenges, and areas of improvement in various software project environments.

This thesis highlights the key aspects of implementing QA and security measures in software development, including their early integration and the importance of clearly defined requirements. Regarding working practices within and across projects, strong collaboration and the utilization of feedback loops are essential for improving the development process. In addition, the incorporation of a security-focused individual in each development team is emphasized. Finally, software tools are identified as critical components to ensure consistent and effective practices.

Despite the valuable findings, this thesis acknowledges its limitations, particularly the dependence on qualitative data. Although it offers valuable context, it simultaneously lacks the completeness of quantitative validation. Future research could investigate the development of industry-specific metrics and methodologies to assess the impact of QA and security measures, thus closing this gap.

By presenting implementable strategies, this thesis provides actionable approaches for improving software quality, reducing vulnerabilities, and promoting sustainable development practices that can be adapted in a variety of industries and organizational settings.

Keywords Quality Assurance, Software Security, Software Development, Software Testing, Huld, IEC 62443

Aalto-universitetet
Högskolan för
elektroteknik

Författare Nico Renlund

Titel Effekten av kvalitetssäkring och säkerhetsåtgärder på mjukvaruprojekt

Utbildningsprogram Automation och Elektroteknik

Huvudämne Kontroll, Robotik och Autonoma system

Övervakare Prof. Valeriy Vyatkin

Handledare MSc Musa Jallow

Samarbetspartner Huld Oy

Datum 13 December 2024 **Sidantal** 71 **Språk** engelska

Sammandrag

Mjukvaruindustrin utvecklas ständigt och kräver konstant medvetenhet om nya hot och efterlevnad av uppdaterade riktlinjer för hantering av mjukvaruprojekt. Detta examensarbete undersökte integrationen av kvalitetssäkring och säkerhetsåtgärder inom mjukvaruutvecklingsprojekt, vilket belyser deras viktiga roll för att skapa robusta och pålitliga slutprodukter. Detta uppnåddes genom analys av kvalitativ data som samlats in från intervjuer med Huld kvalitetsansvariga och säkerhetsexperter, samt med produktchefer och ägare från Huld kundorganisationer. Ytterligare insikter erhölls från en granskning av interna kommunikationskanaler. Tillsammans möjliggjorde de erhållna uppgifterna identifiering av bästa praxis, utmaningar och förbättringsområden i olika mjukvaruprojektmiljöer.

Detta examensarbete belyser nyckelaspekterna av att implementera kvalitetssäkring och säkerhetsåtgärder i mjukvaruutveckling, inklusive deras tidiga integration och vikten av tydligt definierade krav. Ett nära samarbete och utnyttjande av feedbackloopar är avgörande för att förbättra utvecklingsprocessen inom och mellan projekt. Dessutom att inkorperera en säkerhetsfokuserad individ i varje utvecklingsteam betonas. Slutligen identifieras mjukvaruverktyg som kritiska komponenter för att säkerställa konsekventa och effektiva utevecklingsmetoder.

Trots de värdefulla fynden medger denna avhandling i synnerhet dess begränsningar i beroendet av kvalitativa data. Även om den erbjuder ett värdefullt sammanhang, så saknar den det fullständiga kvantitativa valideringen. Framtida forskningar skulle kunna undersöka utvecklingen av branschspecifika mått och metoder för att bedöma effekterna av kvalitetssäkring och säkerhetsåtgärder, vilket minskar denna lucka.

Genom att presentera implementerbara strategier ger denna avhandling handlingsbara tillvägagångssätt för att förbättra mjukvarukvaliteten, minska sårbarheter och främja hållbara utvecklingsmetoder som kan anpassas i en mängd olika branscher och organisatoriska inställningar.

Nyckelord Kvalitetssäkring, IT-säkerhet, Huld, Mjukvaruutveckling, mjukvarutestning, IEC 62443

Preface

Firstly, I would like to thank Huld Oy for providing the topic for my thesis. I extend my appreciation to my colleagues and advisors at Huld Oy and beyond, whose insights and expertise significantly contributed to shaping this thesis. Writing this thesis has been both a challenging and rewarding journey, one that tested my determination and resilience while also providing countless opportunities for personal and professional growth. Balancing the demands of full-time work and football training sessions and matches with the extensive research and writing required for this thesis was not a small task. However, the experience has only had a positive influence on me.

Lastly, this work would not have been possible without the unwavering support and encouragement of those closest to me. Completing this thesis has been a milestone in my life, and I am incredibly grateful to everyone who made it possible.

Espoo, 13 December 2024

Nico Renlund

Contents

Abstract 3

Abstract (in Swedish) 4

Preface 5

Contents 6

Acronyms and Abbreviations 8

1 Introduction 9

2 Theoretical Framework 11

2.1 Concepts of Software Quality Assurance 11

2.2 Principles of Software Security 13

2.3 The Relationship Between Software Quality Assurance and Security 15

3 Prior Research and Discoveries 16

3.1 Quality Assurance and Security in Software Development 16

3.1.1 Software testing 16

3.1.2 Security compliance with ISA/IEC 62443 series of standards 20

3.1.3 Tools and Technologies 23

3.2 Cost Analysis in Software Projects 28

4 Data Collection and Methodology 31

4.1 Data Collection Methods 31

4.1.1 Interviews 31

4.1.2 Internal channels and data 32

4.2 Research Methodology 33

4.2.1 Qualitative Approach 33

4.2.2 Semi-Structured Interviews 33

4.2.3 Data Analysis 34

5 Interview answers and data collected **35**

5.1 Huld Experts 35

5.1.1 Project Involvement 36

5.1.2 Implementation 37

5.1.3 Costs 38

5.1.4 Benefits and Outcomes 39

5.1.5 Challenges 39

5.1.6 End Product 41

5.2 Product managers and owners 43

5.2.1 Project Overview 43

5.2.2 QA and Security Integration 44

5.2.3 Cost-Benefit Analysis 44

5.2.4 Impact on Project Outcomes 45

5.2.5 Feedback and Iteration 46

5.2.6 Best Practices and Lessons Learned 47

5.3 Internal company channels and data 49

6 Results **51**

7 Discussion **54**

8 Conclusion **57**

References **59**

A Interview questions - Huld employees **67**

B Interview questions - Product Managers and Owners **69**

Acronyms and Abbreviations

ASAT Automatic Static Analysis Tools

ASVS Application Security Verification Standard

CD Continuous Deployment

CI Continuous Integration

COCOMO Constructive Cost Model

CPSQ Cost of Poor Software Quality

DoD Definition of Done

E2E End-to-end

Huld Huld Oy

IEC International Electrotechnical Commission

IACS Industrial Automation and Control Systems

ISA International Society of Automation (ISA)

ISMS Information Security Management Systems

IoT Internet of Things

MORETO Model-based Security Requirement Management Tool

PM Product Manager

PO Product Owner

QA Quality Assurance

SCA Software Composition Analysis

SQA Software Quality Assurance

TA Test Automation

QM Quality Management

QMS Quality Management Systems

1 Introduction

In the near future, new regulations and laws on Quality Assurance (QA) and security are being established, which means that more companies need to invest their time and expertise in their software projects. However, QA and security are not new topics, as they have been around for many years. Technology is evolving every day, and new threats emerge. Therefore, investing time and expertise in QA and security is not only for regulation purposes, but to stay ahead of evolving challenges.

Woodhouse [1, p. 30] defines QA as "*the policies, attitudes, actions, and procedures necessary to ensure that quality is being maintained and enhanced.*" This is an excellent definition of QA. However, it also raises a problem. It is extremely important that all stakeholders involved in the software project have aligned their understanding of quality, ensuring that no misunderstandings occur during the project. Woodhouse also mentions that QA can include one or more approaches to achieve the minimum standard or to assure stakeholders that quality is being achieved. Examples of different approaches that are utilized for QA include Audit, Assessment, and Accreditation.

To achieve quality, it is also crucial to implement the necessary security measures in software projects. According to Assal and Chiasson [2], nearly all software faces increased security risks due to their online connectivity in today's interconnected world. This highlights the importance of using various security measures throughout the software development life cycle. Additionally, it emphasizes the importance of identifying and implementing the most appropriate security measures for a specific software. Developers are often viewed as the weakest link and are often held responsible for security vulnerabilities [2]. By addressing security concerns at every stage of development, from initial planning to end product, companies can safeguard their assets, protect user data, and mitigate the risks of cyber threats and breaches.

When a company invests in QA for a software project, they expect a quality product in return. Operational costs should be reduced, customer satisfaction and revenue from

the software project should increase and result in better profit margins [3]. Therefore, the aim of this thesis is threefold: to discover the costs associated with integrating QA and security measures in a software project, to analyze how QA and security measures can enhance the organization and the end product, and to evaluate the investment required for a software project of varying sizes. To accomplish these aims, a literature review will be conducted to provide a thorough understanding on the existing research on QA and security. In addition to the literature review, interviews will be conducted with key Huld QA and security experts, as well as with product managers and owners of major customer companies to incorporate their perspectives into the results.

The rest of the thesis is divided into seven chapters. Chapter 2 (*Theoretical Framework*) introduces the foundational concepts of QA and software security. Chapter 3 (*Prior Research and Discoveries*) reviews the existing literature and research findings related to QA and security, and their impact on software projects. Chapter 4 (*Data Collection and Methodology*) outlines the methodology used in this thesis and addresses where the data are collected. Chapter 5 (*Interview answers and data collected*) analyzes the company's data and the information gathered from the interviews conducted. Chapter 6 (*Results*) presents the results of the thesis. Chapter 7 (*Discussion*) discusses the results and addresses the observed limitations. Chapter 8 (*Conclusion*) summarizes the key findings and conclusions drawn from the thesis work and suggests directions for future work.

2 Theoretical Framework

In the rapidly evolving field of software development, a strong theoretical foundation is essential to understand and implement effective QA and security measures. This chapter lays the foundation for understanding what QA and security in software development are. The aim of this chapter is to provide a detailed examination of the key concepts and principles that guide effective QA and security practices. Section 2.1 explores the various concepts of QA, QA methodologies, and practices. Section 2.2 discusses the essential principles of software security, including threat modeling, secure coding practices, and the importance of integrating security measures throughout the software development life cycle. Section 2.3 examines the relationship between QA and security, highlighting their interdependence.

2.1 Concepts of Software Quality Assurance

Software Quality Assurance (SQA) differs from quality management in general and should not be mistaken for it. SQA can be summarized as "*firstly SQA has focus on software, secondly its purpose is to make problems visible, and thirdly SQA reports the problems as close as possible to the source for correction.*" [4]. There are two main stages in the life cycle of software development projects: the development life cycle stage and the operation maintenance stage. Several different SQA components enter at different points in the life cycle of software development projects and their use should be planned before the project starts [5]. The main SQA components are the following.

- Reviews
- Software testing
- Software maintenance
- Expert opinions

- Assurance of the quality of the subcontractors' work and the customer supplied parts.

This thesis will place a significant focus on software testing, as it is a crucial element of QA and a primary focus in software development projects at Huld.

Quality cannot be achieved by evaluating software that is already complete. Quality defects and deficiencies must be prevented in the first place. The software needs to be assessable by QA measures already in the development state. Prevention, inspection, internal failure, and external failure are the four components that contribute to the total cost of effective quality management [6]. A considerable amount of people believe that adding quality assurance to a project will add more cost, but it is actually the other way around. Development costs will be reduced in the long run because the sooner one can locate defects and deficiencies and correct them, the less costly it will be [6]. Many developers think it is more important to deliver software on time than to fix problems before deployment. Despite the fact that they know that most software failures or disasters could be avoided by utilizing known software engineering techniques [7].

Quality Management (QM) is a business strategy that has consistently improved company performance and produced positive results over the years. Companies that adopt QM focus on maximizing customer value and improving process efficiency [8]. ISO 9001 [9] is an internationally recognized standard for quality management. It helps organizations of various sizes and industries improve their performance, meet customer expectations, and showcase their dedication to quality. The requirements specify the steps to configure, implement, maintain, and continuously improve Quality Management Systems (QMS). Today, businesses face pressure to obtain ISO 9001 certification, as lacking it makes it difficult to secure a considerable number of contracts. Additional reasons for obtaining certification include improvements in quality, marketing benefits, and customer demands [10].

2.2 Principles of Software Security

The principle of software security is to design software so that it continues to function correctly even during a malicious attack. The first books and academic classes on software security originated at the beginning of this century, in 2001. Software security best practices incorporate good software engineering principles and emphasize considering security early in the software life cycle. This involves understanding common threats, designing with security in mind, and performing thorough risk analyzes and testing of all software artifacts [11].

Most of the security threats that affect software are vulnerabilities that are generated during the software development phase. Nina et al. have conducted a study [12] and identified four steps that should be followed to produce secure software, software requirements security, software design security, software construction security and software testing security. This process aims to improve security requirements, implement threat modeling methodologies during software design, and adhere to best practices for coding, performing code reviews, and performing tests. The process should be continuously updated to provide and meet the latest security requirements.

Software projects are becoming increasingly complex. To ensure that the software has reached an acceptable security level, companies can implement international standards regarding security. There are multiple international standards that impact security. Examples include the ISO 27000 series [13], which provides an overview of information security management systems (ISMS), and ISO 31000 [14], which offers principles and guidelines for risk management. These standards provide a means of attaining the desired security level and fulfilling the expectations of potential customers [15]. Implementing security standards is difficult, and companies need to invest enough time and resources to achieve successful implementation.

Among these security standards, the OWASP Application Security Verification Standard (ASVS) [16] stands out as a practical framework specifically designed for

web applications. ASVS serves as a foundation for testing the technical security controls of web applications and offers developers a comprehensive list of secure development requirements. The main goals of ASVS [17] are to help organizations develop and maintain secure applications while fostering alignment between the requirements and offerings of security service providers, security tools vendors, and consumers. The ASVS defines three levels of security verification, each increasing in thoroughness and scope:

- Level 1: Designed for all software applications, this level focuses on basic security defenses against common vulnerabilities, such as those in the OWASP Top 10. It requires minimal access to source code and is often validated using automated tools or simple manual checks. Achieving Level 1 is the bare minimum that all applications should aim for.
- Level 2: Recommended for most software applications, particularly those that handle sensitive data or business-critical functions. This level ensures that robust security controls are implemented, effective and used within the application.
- Level 3: Tailored for the most critical applications, such as those in military, healthcare, or critical infrastructure. It requires in-depth architectural reviews, coding analysis, and advanced testing to verify the highest level of security assurance.

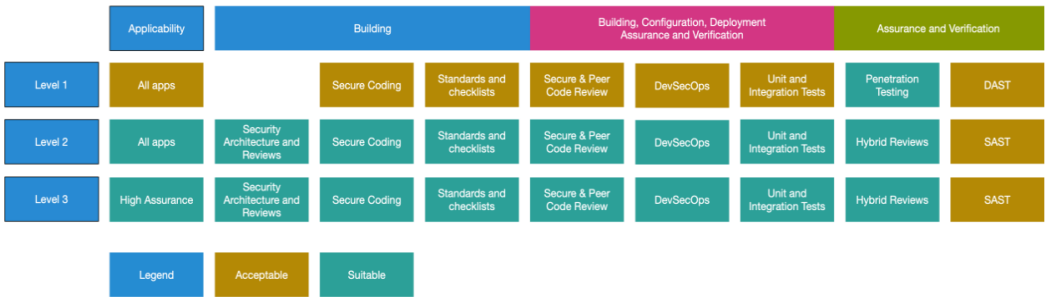


Figure 1: OWASP Application Security Verification Standard 4.0 Levels [17].

2.3 The Relationship Between Software Quality Assurance and Security

There is a strong connection between QA and security, as both are integral to creating reliable and credible software. QA is responsible for verifying the system's functionality and performance, whereas security concentrates on protecting the system from malicious attacks. QA is more concerned with the overall user experience, ensuring that the product is user-friendly and functions as intended [18]. This includes aspects such as usability, performance, and reliability. On the other hand, security focuses on protecting users and their data from unauthorized access or breaches, emphasizing protection and confidentiality [11]. Thus, although both are vital for the success of a product, they focus on different aspects of user experience and safety.

In software projects, there can be conflict when implementing security measures. Specifically, enhancing or adding security measures might have a negative effect on the performance of the software, such as slowing it down or consuming more resources [19]. This suggests a compromise between upholding strong security measures and achieving optimal performance, which developers must consider thoughtfully balancing. Consequently, it is essential for the QA and security teams to work effectively together. The cooperation between these teams ensures the software's reliability and security, which is crucial for delivering high-quality products and safeguarding user data.

Both the QA and security processes in a software project must comply with specific industry standards and regulations. This implies that QA processes and security measures should be aligned with recognized guidelines and legal requirements relevant to the industry. This also ensures that the software meets quality expectations and requirements and is in accordance with security practices that protect data and maintain reliability.

3 Prior Research and Discoveries

Understanding the current situation of QA and security in software projects is crucial to identify the best available practices. This chapter will review the existing literature on QA and security in software development and assess their impact on software projects. The aim of this chapter is to provide a comprehensive understanding of the current state of research that exists and to highlight gaps and limitations in existing studies. Section 3.1 explores the key components of QA and security, examining their impact on software development practices and outcomes. Section 3.2 reviews the cost implications of integrating QA and security in software projects, comparing costs when these measures are considered from the beginning versus being added later in the software development process.

3.1 Quality Assurance and Security in Software Development

3.1.1 Software testing

Ammann and Offutt describe in their book "*Introduction to Software Testing*" [20], how each distinct software development activity in model-driven testing corresponds to a different level of testing. Figure 2, known as the "V model" or the "Verification and Validation model", illustrates a typical scenario for testing levels and their relationship to software development activities by isolating each step. The test should be designed concurrently with the associated software development activity, despite the fact that the software is not executable until the implementation phase.

For testing to be efficient and effective, it should not just be done at one fixed point in the software development process. Majchrzak describes the different levels of testing in his book [21]. The lowest level of testing is unit testing. Unit testing tests individual pieces of code to verify that the piece of code is working as expected. Unit tests are

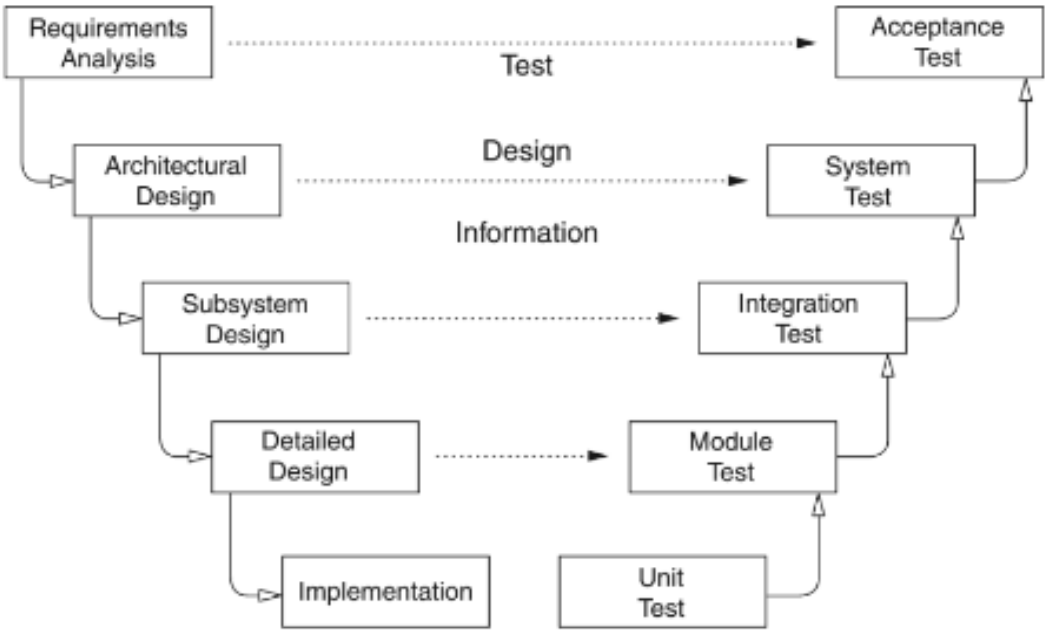


Figure 2: Software development activities and testing levels - the "V Model" [20].

created by the developers that wrote the code that is being tested. Since developers are familiar with their own code, they perform so-called Glass-Box and Gray-Box tests. While unit tests concern individual pieces of code, module tests, and the next level of testing deals with pieces of the system. Module tests are also created by the developers that wrote the modules. Both module tests and unit tests can be categorized under component testing and are worked on in an incremental and iterative way.

The third level of tests performed in the V model is integration tests, also known as product tests. Integration tests are usually done on testing systems and not on developers' own working stations. These tests check if multiple components integrated together work as expected to ensure interoperability. Usually, different developers have worked on the components, and in general the developers are still responsible to create these tests. In case integration related problems are found during the project, interoperability has to be restored, and the integration tests repeated.

The next level following component testing and integration testing is system testing. The system test is a large-scale integration test that tests the whole system and is less

technical focused than previous tests. For system tests, Black-Box tests are used instead of Glass-Box tests. These tests are often performed by dedicated testers, who are not familiar with the source code. They utilize mock objects to simulate the behavior of components in earlier levels. System testing includes extensive documentation and should be finished with a formal approval that ensures the software's quality to be at a level that satisfies the end user.

Lastly, the last level of test is the acceptance test. In this test, the customer, an academic principal, or an internal department tests the almost finished software. The software is tested to ensure that the specification provided for the software is met. This whole testing level can be omitted for certain software development projects. If the development team has followed an agile paradigm, acceptance tests are considered the key test for the entire project. To follow the V model, the company or development team needs to have a sufficient size. Otherwise, there might not exist dedicated testers or reserved systems for testing, and in that case the whole model becomes much simpler.

Stephan Goericke authored a book titled "The Future of Software Quality Assurance" [22], where he discusses the increasing complexity and significance of software testing in today's context. Testing software has become more challenging due to the need to test larger volumes of code within shorter timeframes, driven by the more frequent software releases. To expedite testing processes, companies are investing in test automation (TA) and continuous integration (CI). However, in the case of extensive systems, automated test suites can run for extended periods, ranging from hours to even days. Consequently, the time gap between the appearance of a code error and its detection increases significantly, leading to potentially severe consequences. To address this challenge, innovative solutions are required, one of which is change-driven testing. This approach can pinpoint around 90 percent of errors that the entire test suite might identify, using only 2 percent of the test suite's runtime. In addition, it provides information on untested code changes.

Huld provides Quality Assurance and Testing [23] as one of its services. The primary objective of this service is to ensure reliability, speed to market, cost efficiency, comprehensive testing, consistency, and efficient use of resources. This is achieved by leading the development of testing strategies, plans, and comprehensive documentation essential for validation, the addition of test management, testing, and TA. Moreover, bringing into account security testing alongside safety critical systems. Lastly, implement tools and practices to streamline and automate the integration between the validation and software development processes.

Software testing is one of the keys to ensuring that a software product is reliable and successful. However, software testing is often viewed as uninterested work compared to coding and designing the software product itself. It is important to get developers motivated to test software and utilize a good strategy to make this happen [24]. One of the dominating negative factors about testing is the lack of influence and recognition perceived by the testing staff. Another discovery that Deak et al. [24] points out is that most students prefer a software development position to a software testing position. They see the image of testing work as undesirable status and software development to be more rewarding from a financial and career perspective.

3.1.2 Security compliance with ISA/IEC 62443 series of standards

The ISA99 standards committee was established in 2002 by the International Society of Automation (ISA) [25]. ISA recognized the need to protect the equipment and operations that form U.S. critical infrastructure from cyberattacks, the ISA99 committee has since developed a comprehensive set of standards and technical reports specifically designed to secure automation and control systems. The International Electrotechnical Commission (IEC) [26] was founded in 1906. IEC is the world's leading organization responsible for developing and publishing international standards for electrical, electronic, and related technologies, collectively known as 'electrotechnology'. The IEC publishes around 10 000 IEC International Standards. Together with conformity assessment, these standards form the technical foundation that enables governments to establish national quality infrastructures and allows companies of all sizes to consistently produce and trade safe, reliable products across the globe.

The ISA/IEC 62443 series of standards [27] defines processes and requirements for implementing and maintaining electronically secure industrial automation and control systems (IACS). These standards establish best practices for security and offer a framework for evaluating security performance levels. Their approach to cybersecurity is comprehensive, addressing both operational and information technology aspects, while also linking process safety with cybersecurity efforts. The IEC 62443 standards provide cybersecurity guidelines in all industries that use IACS, including medical devices, building automation, transportation, and electric power generation and distribution. The concept of shared responsibility as an essential building block of cybersecurity automation is a founding principle of the IEC 62443 standard. To ensure the integrity, safety, security, and reliability of a control system, key stakeholder groups must align.

IEC 62443 standard consists of four main groups, each of the groups containing multiple documents [27, 28]:

- IEC 62443-1-X: General, contains documents for defining terminology, concepts, use cases, etc.
  - IEC 62443-1-1: Terminology, concepts, and models [29].
- IEC 62443-2-X: Policies and procedures, contains e.g., secure patch management and security program requirements.
  - IEC 62443-2-1: Establishing an industrial automation and control systems security program [30].
  - IEC 62443-2-3: Patch management in the IACS environment [31].
  - IEC 62443-2-4: Security program requirements for IACS service providers [32].
- IEC 62443-3-X: System security requirements and security levels, system risk assessment, etc.
  - IEC 62443-3-2: Security risk assessment for system design [33].
  - IEC 62443-3-3: System security requirements and security levels [34].
- IEC 62443-4-X: Component security, including component development requirements.
  - IEC 62443-4-1: Secure product development lifecycle requirements [35].
  - IEC 62443-4-2: Technical security requirements for IACS components [36].

Hassani et al. [37] used a qualitative approach, based on IEC 62443-3-2 to evaluate the security risks to which the Industrial Internet of Things (IoT) are exposed. This was done to validate or apply corrective measures before integrating them into the industrial system. They followed a three-step approach that included identification of critical information systems, detailed risk analysis for these information systems, and definition of the security measures applicable to these systems. The result of

the evaluation was a detailed risk assessment method by zone that was successfully implemented. This allowed them to detect the vulnerabilities that caused a high-level risk to the model. Each zone consisted of the likelihood that threats would occur and the degree of impact it would have. Lastly, this allowed them to deduce the level of risk and provide a clear understanding of the most harmful vulnerabilities.

The IEC 62443 series describes a standard approach for developing a secure infrastructure tailored to the security needs of IACS. The study conducted by Shabaan et al. [38] investigates the use of the Model-Based Security Requirement Management Tool (MORETO) to analyze and manage security requirements according to the IEC 62443 security standard. MORETO is a tool designed to analyze, distribute, and handle security requirements through the use of modeling languages like SysML and UML. The AIT Center for Digital Safety and Security is responsible for developing MORETO. In another study, Shabaan et al. [39] use a threat modeling tool known as THREATGET, which is also developed by AIT. THREATGET is utilized to detect current security vulnerabilities that can negatively impact the system in question. Subsequently, the identified threats can be divided into six primary categories using the STRIDE methodology. Both studies [38, 39] introduced tools such as MORETO and THREATGET to support compliance with the IEC 62443 security standard.

Huld provides a wide range of security services to its customers, with IEC 62443 being one of the key offerings [40, 41]. For Huld, the most focused and important part of the IEC 62443 standard is the IEC 62443-4-1: Secure product development lifecycle requirements [35]. Huld has concentrated the majority of its experts within this group. The crucial components and features of this group include the specification of security requirements, secure implementation, security verification and validation testing, and security guidelines.

3.1.3 Tools and Technologies

CI is a fundamental aspect of modern software development. Martin Fowler [42] describes CI as a practice that ensures software stability by continuously merging code changes into a shared repository. A developer begins by checking out a copy of the mainline code, making the necessary changes, and updating or adding automated tests. After successfully building and testing locally, the developer updates their copy with any new changes from the mainline, resolves any conflicts, and commits the changes. An integration built on a separate machine verifies the changes. If conflicts or errors arise, they should be fixed immediately by the developer. This process maintains a stable, functional codebase, reducing time spent on bug fixing and integration issues. Furthermore, CI increases opportunities for rapid feedback, allowing the status of the project to be assessed multiple times a day [43].

One popular open-source CI tool written in Java is Jenkins. Jenkins is used by teams of different sizes and for projects that utilize a wide variety of technologies and languages [44]. One of the main reasons to utilize Jenkins for your CI infrastructure is how easy it is to utilize it. It has an intuitive, simple, and visually appealing user interface. Also, Jenkins is really flexible, can be adapted easily to match the purposes of the project, and has hundreds of open source plugins available with more coming out all the time. Lastly, Jenkins' popularity comes from its community. The community is large, reactive, and dynamic, and the pace of development is fast. New releases are coming out weekly that include new features, plugin updates, and bug fixes [44].

Seth and Khare have written a paper [45] in which they discuss the real-time scenario where the software development teams in software projects are dispersed in different cities or even countries. In these teams, there is a need for CI, TA, and a tool such as Jenkins. They propose implementing automation scripts to build and run tasks, such as the Android Compatibility Test Suite. Utilizing these scripts enhances employee work hour efficiency, as they can be executed, for example, during off-hours. Additionally,

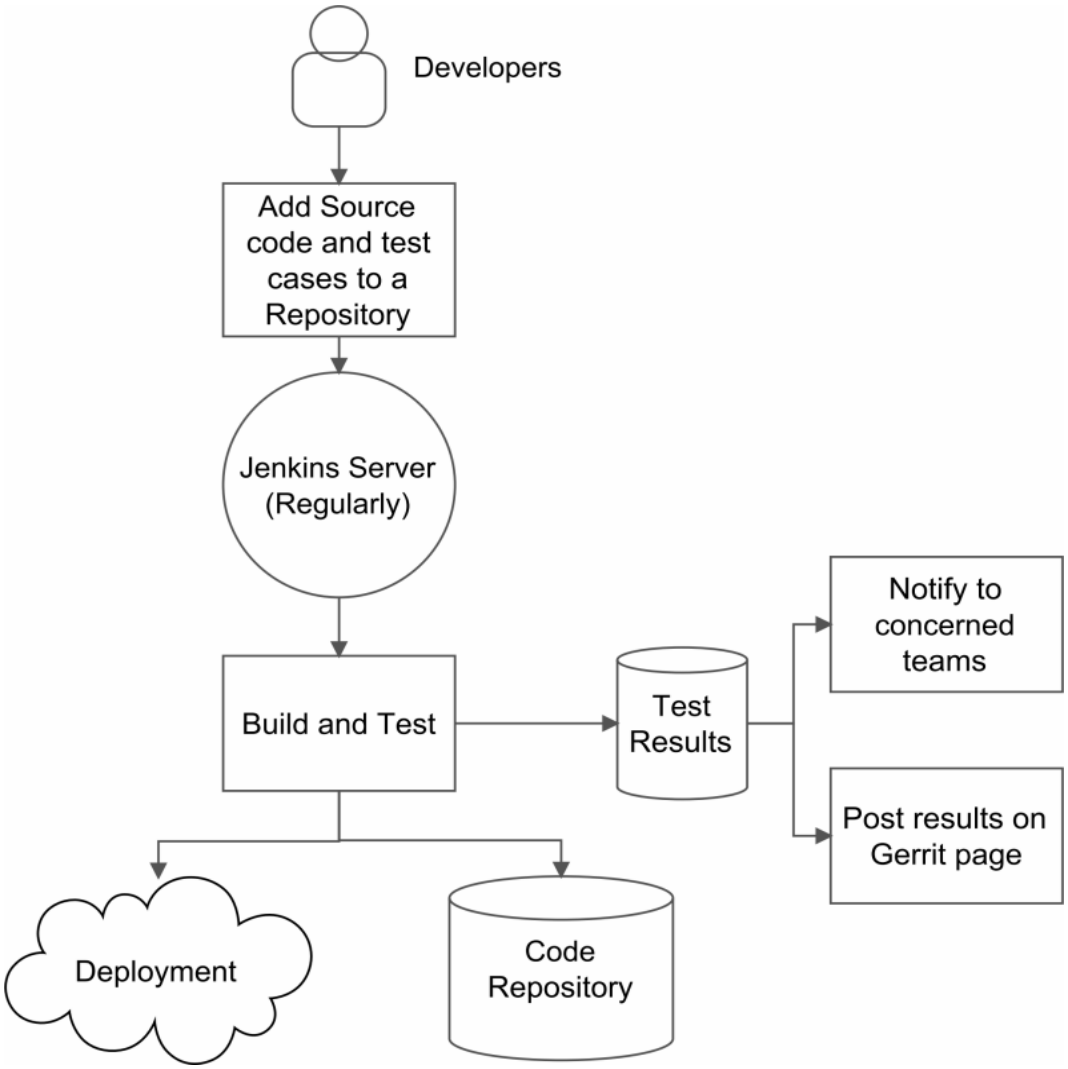


Figure 3: The software development workflow using Jenkins [45].

this solution is cost efficient, as Jenkins is open-source software. Figure 3 illustrates the workflow involved in utilizing Jenkins during the development process. The figure demonstrates how Jenkins automates the integration and deployment tasks, ensuring CI and continuous delivery (CD). By automating repetitive tasks, Jenkins allows developers to focus on more critical aspects of the project, thereby improving overall productivity and efficiency.

Utilizing Automatic Static Analysis Tools (ASATs) is a prominent approach to improving both code quality and security issues in a software project. ASAT allows the user to analyze the quality characteristics of the code without having to execute

the program itself [46]. One well known ASAT and one of the most used ones is SonarQube. SonarQube is an open-source platform with the goal of assessing test coverage, potential security issues, code quality as well as violations of coding rules [47]. SonarQube is utilized by more than 85 000 organizations and supports more than 25 languages. It has its own configurations and rules, but it is possible for the user to add new rules specific to the project [48]. The logic behind SonarQube lies in its source code analyzer component, which performs comprehensive static analysis activities on the code and an application server that displays the results for the user [49]. Integration with CI/CD tools like Jenkins, GitLab CI, or Azure DevOps makes it easy to configure SonarQube as an essential checkpoint, ensuring that code quality and security standards are met with every commit and build.

The issues that SonarQube detects can be divided into three broad kinds. Bug occurs when an issue related to a piece of code is demonstrably incorrect. A Vulnerability occurs when a piece of code can be exploited to compromise the system's security. Finally, a Code Smell occurs when an issue represents instances of poor coding practices that are neither bugs nor vulnerabilities [48]. SonarQube also categorizes the severity of the issue and the possible impact on the system or on the developer's productivity. The issues that negatively affect the system are blocker and critical issues, with blocker issues having a higher probability of negatively affecting the system compared to critical issues. Blocker and critical issues should be fixed as soon as possible when they are found. Additionally, there exist major and minor issues that affect the productivity of a developer. Lastly, information issues encompass all problems that are neither bugs nor quality flaws [48].

Software composition analysis (SCA) tools are utilized during the vulnerability testing of a project. SCA tools are used to detect possible vulnerabilities introduced by open-source software imported as a third-party library [50]. An SCA tool that is commonly used is Black Duck, earlier known as Synopsys [51]. Black Duck provides comprehensive visibility into open source components in a project, helping organizations manage open-source risk by identifying known vulnerabilities, license

compliance issues, and security threats. They maintain a comprehensive database of known security vulnerabilities which is used for cross-referencing the libraries in use. This allows development teams to quickly detect outdated or vulnerable components in their project and allows the team to take action to mitigate potential threats. Black Duck can also be easily integrated into CI/CD pipelines, offering real-time monitoring and automated alerts whenever a new vulnerability is discovered in any third-party library. In addition to detecting vulnerabilities, Black Duck ensures legal compliance by reviewing licensing requirements and flagging potential conflicts, helping to minimize the legal risks associated with using open source software.

Web applications have become an integral part of our lives. However, the number and impact of attacks that exploit web application vulnerabilities are on the rise. Therefore, there are specific tools designed for security testing and OWASP ZAP is one such tool. OWASP ZAP, developed by The Open Web Application Security Project, is a widely used open source web application vulnerability scanner. It is designed for users with varying levels of security experience, making it perfect for developers and functional testers who are new to penetration testing. OWASP ZAP utilizes black box testing, and the user can select which vulnerabilities it should cover [52]. The OWASP Top Ten [53] offers a list of the most major vulnerabilities in web application. This list includes cross-site scripting, broken authentication, SQL injection, and sensitive data exposure. Figure 4 illustrates the complete list of the top ten OWASP vulnerabilities, also highlighting the recent changes, merges, and developments in this ranking over time.

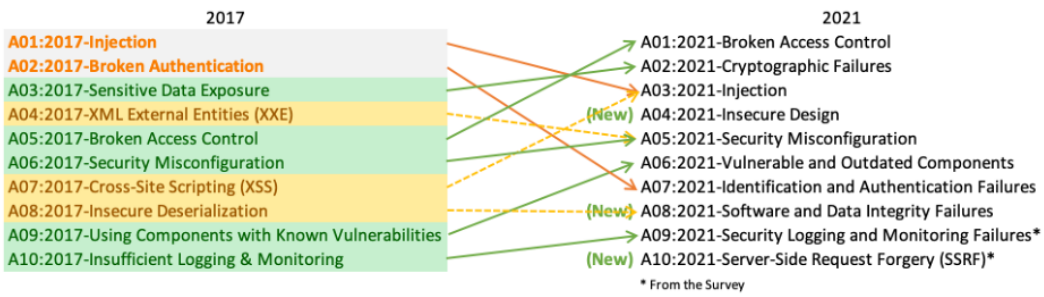


Figure 4: Top 10 Web Application Security Risks[54].

Makino and Klyuev [52] and Mburano and Si [55] have researched and tested the performance of OWASP ZAP, comparing it to other security scanners such as Skipfish and Arachni. OWASP ZAP is able to locate ten times more vulnerabilities than Skipfish, with a false positive percentage near zero. OWASP ZAP also performs better than Arachni in command injection, SQL injection, and cross-site scripting. However, Arachni is able to outperform OWASP ZAP in some specific categories such as LDAP injection. The research concluded that scanners perform differently in different categories. Therefore, no scanner can be considered all-rounder in scanning web application vulnerabilities, and utilizing multiple scanners for different categories is a possibility. Lastly, at this stage, no web application scanner is perfect and able to cover all vulnerabilities.

3.2 Cost Analysis in Software Projects

Software companies invest resources to QA to reduce the costs associated with software development and maintenance, while also increasing revenue and profit margins. To increase net income, an QA organization must evaluate the costs and benefits of the testing tools used to ensure the quality of software components, including code, design, requirements, and specifications [3]. Addressing software bugs and security vulnerabilities in the early stages of development is approximately 50-200 times cheaper than dealing with them after the software has been deployed to production [56]. Early error detection and correction techniques have become increasingly valuable in the context of software requirements, design specification, and verification.

Several cost estimation models are derived from the Constructive Cost Model (COCOMO), which is utilized to predict the effort, expenses, and timetable for software projects. The main motivation of the COCOMO model is to help people understand the financial implications of their choices in the commissioning, creation, and maintenance of a software product [57]. Over time, COCOMO has evolved to incorporate changes in software development methodologies and technological advancements. One such version is COCOMO 2.0, developed to tackle the issues common in modern software engineering, such as swift prototyping, the adoption of component-based development, and the rise of new application areas. COCOMO 2.0 builds on the foundation of the original model while integrating updated cost estimation methods and parameters to represent current development environments. This makes it more adaptable and accurate for today's software projects [58].

The total Cost of Poor Software Quality (CPSQ) exceeded two trillion dollars in the United States during 2020 [59]. However, this figure did not include software technical debt, which is estimated at more than one trillion dollars. Technical debt refers to the costs resulting from severe software defects that need to be addressed and is not included in the CPSQ because it represents expenses that will be incurred

in the future. Figure 5 illustrates the distribution of costs between various areas, including operational failures, unsuccessful IT initiatives, and outdated systems. The

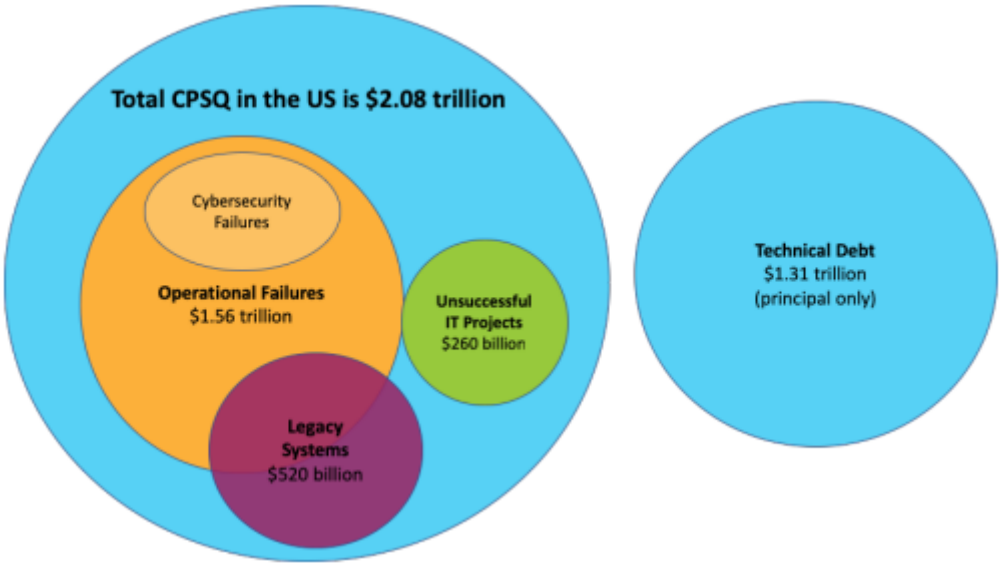


Figure 5: CPSQ in 2020 in the United States [59].

significant numbers associated with CPSQ reveal the substantial financial impact that poor software quality can have on organizations and economies. This emphasizes the vital necessity of implementing strong security protocols, thorough QA methods, and sufficient budget planning in software development projects. Organizations that consider QA and security as essential parts of their software development not only reduce risks, but also ensure the long-term viability of their products. By investing adequate funds and resources in these areas, teams can proactively address possible vulnerabilities, build robust systems, and maintain customer confidence.

Jones and Bonsignour list both the economic value of high software quality and low software quality in their book "*The economics of software quality*" [60]. The results of high software quality from an economical perspective:

- Higher reliability and user satisfaction
- Reduced maintenance costs for released applications

- Reduced customer support costs for released applications
- Reduction in schedule delays, canceled projects and cost overruns
- Rapid deployment of new applications

On the other hand, the results of low software quality from an economical perspective:

- Risk of canceled projects
- Poor customer satisfaction
- High maintenance costs and customer support costs due to poor quality
- Possible litigation from customers suffering business losses
- Possible class-action litigation from disgruntled customers

These opposing outcomes underscore the importance of prioritizing quality at every stage of the software development lifecycle. High-quality software not only enhances user experience and satisfaction, but also results in long-term cost savings and greater business sustainability. In contrast, poor-quality software can cause significant financial and reputational damage, making it clear that investing in quality and security measures from the outset is essential for both short-term success and long-term viability.

4 Data Collection and Methodology

This chapter provides a comprehensive overview of the techniques used for data collection, as well as the detailed methodology utilized throughout this thesis. The primary goal of this chapter is to provide a comprehensive description of the methods used for data collection and analysis to efficiently and thoroughly address the research questions asked. Section 4.1 provides an in-depth discussion on the diverse range of data collection strategies used in the study. In contrast, Section 4.2 offers a comprehensive elaboration on the research methodologies applied, highlighting the steps taken to guarantee the reliability and validity of the findings.

4.1 Data Collection Methods

4.1.1 Interviews

In order to obtain thorough information regarding the integration of QA and security measures within the field of software development, the main data source consisted of conducting interviews with professionals working in the industry. The interviews provided significant qualitative data, primarily through discussions with Huld experts. Furthermore, additional interviews were conducted with product owners and managers associated with Huld's customers to provide a more comprehensive perspective. The interviews conducted with Huld experts primarily provided information on the implementation and effectiveness of QA and security measures. In parallel, discussions with product owners and managers enabled the collection of more detailed and specific data directly relevant to the various software projects under consideration.

4.1.2 Internal channels and data

Another source of data for the thesis came from Huld's internal channels and data. These data consisted of information available to QA and security experts to utilize when implementing QA and security measures within their software projects. This included information on the adherence to security standards, the implementation of best practices, and the results of various measures within different projects. The internal channels and data provided information on the effectiveness of various QA and security protocols. Analysis of these data offers a comprehensive understanding of how Huld helps their QA and security experts integrate QA and security measures into Huld projects and the impact these practices had on project outcomes, including cost, time efficiency, and overall software quality.

4.2 Research Methodology

4.2.1 Qualitative Approach

The thesis utilized a qualitative methodology [61] to obtain an in-depth understanding from industry experts regarding the integration of QA and security measures within software projects. This methodological choice was made because of its ability to provide a more profound understanding of the various factors that influence the integration process. Through qualitative data, the insights obtained from the interviews offered valuable perspectives and personal experiences. In addition, it facilitated a detailed examination of the key themes in the data collected, including the challenges encountered in the implementation of QA and security, the advantages of these practices, and their impact on project outcomes.

4.2.2 Semi-Structured Interviews

Semi-structured interviews were selected as the main method of data collection. These interviews were carried out with industry experts from Huld, in addition to product owners and managers from companies that are customers of Huld. This method was chosen for its flexibility, allowing open questions that can generate detailed and comprehensive responses. Appendix A provides the set of questions asked during interviews with Huld employees, whereas Appendix B features the questions directed at product managers and owners from Huld's customer companies. The interview questions aimed to address key aspects of integrating QA and security within software projects, focusing on implementation strategies, encountered challenges, and the benefits that are perceived. The interviews were carried out in a casual discussion style, encouraging the participants to openly share their experiences and insights. This method guarantees that the information collected is detailed and diverse, offering a more thorough understanding of the topic.

4.2.3 Data Analysis

A thematic analysis methodology was used to perform a systematic and comprehensive examination of the qualitative data that was collected. This approach allowed for a methodical review of interviews, as well as company channels and data, which was fundamental in identifying recurring themes and insights that are essential for QA and security practices. By engaging in the data coding and categorization processes, the analysis brought to light important patterns related to the implementation, benefits, and challenges that arise with the integration of QA and security measures within software projects.

5 Interview answers and data collected

This chapter summarizes the information gathered from the interviews that were conducted with Huld experts, as well as product owners and managers of Huld's customers, and reviews the internal data collected. The aim of this chapter is to get a good understanding of how QA and security measures are implemented and affect software projects from Huld experts perspective. Also, understand how Huld's customers approach and utilize QA and security measures in their projects. Section 5.1 summarizes the interview answers collected from Huld experts based on the questions in Appendix A. Section 5.2 summarizes the interview answers collected from product owners and managers of Huld's customers based on the questions in the Appendix B. Section 5.3 takes a look at the available internal data and how it can help Huld experts.

5.1 Huld Experts

Huld experts from the Digital Secure Services department was chosen as the perfect fit for the interviews. Specifically, experts who work closely with QA and/or security. The chosen interviewees from the Digital Secure Services department were selected for their direct experience in handling QA and security aspects across a range of project types. These interviews aimed to explore the perspectives of each participant on the methodologies, tools, and processes used, as well as the challenges faced in maintaining high standards of security and quality. Through this approach, this section captures a comprehensive view of how Huld experts navigate QA and security challenges in their roles.

5.1.1 Project Involvement

Since Huld is kind of a "design house" that offers its services to other companies, almost every employee is working externally on customer projects. Most of the employees have been part of multiple projects during their time at Huld. Furthermore, some employees want to change projects more often than others to, for example, get an improved skill set and change their working environment. The duration of a project can range from a month to multiple years, and the experts working on a project can also change over time depending on the stage of the project and the needs.

The size of the development team in a project differs from project to project. This is usually based on the complexity of the project and the need of the customer. Additionally, if the project is completely outsourced to Huld, this often means that a larger development team is required compared to a situation where the customer has its own development team that only requires some Huld experts. There could also be other contractors involved in the project who are part of the development team. The size of the Huld development team can amount only to one person in smaller projects, but also up to more than 20 people in larger projects. Usually, the development team consists of a handful of people. Most development teams follow the agile scrum methodology [62], and especially QA employees work in these teams. In larger security teams, there is typically a security test lead whose responsibility is to guide the security testers within the team.

Huld's customers have software projects in different stages in their project development life cycle. This means that Huld participates in ongoing software projects, but also from the start of a project or at the end of a project. Working on a project that is considered "at the end" means that Huld is doing, for example, maintenance for the project, adding some new features for the existing product, or implementing some guidelines so that the project follows new security standards. QA and security measures offered by Huld to a project are affected based on the stage in the project life cycle at

which Huld employees are participating. If the software project is already at the end, then it is difficult to change the QA and security measures created at the beginning of the project and unnecessary work needs to be done, which could have been avoided.

5.1.2 Implementation

In a software project, there exists a significant difference in the manner of implementation between QA and security measures. Furthermore, both QA and security practices are influenced by the complexity of the project, as well as the specific phase within the development life cycle at which the project currently resides. In the process of implementing QA, the majority of employees tend to adopt a similar foundational approach. However, each individual incorporates their own unique perspective and distinctive methodology into this approach. This includes preferred programs and languages to use as well as in which order to implement the tests. The main focus is to apply different types of software test that confirm that requirements are working properly. All developers involved in the implementation of QA practices employ a blend of manual and automated testing methods.

In the majority of projects, most of the team members take part in the creation of tests, and particularly for unit tests, every developer is engaged. For tasks requiring more specialized testing, such as E2E testing, many teams have a designated developer to handle the test development. In addition, many teams implement automated testing pipelines that run regularly. These pipelines are often configured to run during off-hours, such as overnight, to ensure that any new changes made during the day are thoroughly tested before the next workday begins. This approach, typically referred to as nightly builds, helps identify issues early on, providing a fresh report of any test failures for the team to address the following morning. Automated pipelines are particularly beneficial for more complex tests such as E2E tests, which may take longer to run. They allow these lengthy tests to be executed without interrupting the daily workflow of developers.

From a security perspective, developers are utilizing multiple approaches and complying with various standards, placing particular emphasis on IEC 62443 [27], specifically IEC 62443-4-1 [35]. Some developers concentrate on building QMS based on [9] by crafting fundamental components such as policies, guidelines, and templates. These components aim to ensure that QMS can be reviewed and complies with applicable standards and regulations. Upon finishing this phase of development and verifying QMS compliance, it can be rolled out into specific projects and teams within the organization. In addition, security experts can influence the selection of tools utilized for security testing.

It is important to consider the entire life cycle of a product, particularly with an eye toward security and standards compliance. Before starting the design process, the requirements should be clearly defined, including security considerations. There are situations, in which an existing product must comply with a new standard. It is therefore crucial to be familiar with the current product to determine if it already meets the requirements of the new standard. Finally, in terms of security for a product, it is highly beneficial to have at least one member of the development team with an interest or expertise in security. This greatly helps the efforts of security teams and testers.

5.1.3 Costs

QA experts are not involved with, nor do they have insight into, the budget allocated to QA within the software projects they are working on. In addition, many of the experts handling QA responsibilities are also involved in other aspects of development beyond just QA. One cost associated with QA are the dedicated programs that are utilized for testing. These add some extra monthly costs that should be considered in addition to the costs of the QA work itself.

From a security perspective, the experts are more aware of the fees and costs for adding security measures and the costs for conducting security tests, such as vulnerability

testing. The complexity and size of the projects also affect the cost. Assigning additional experts to minimize the workload and the cost might not be an option either, in case there is only one interface that is available for testing purposes.

5.1.4 Benefits and Outcomes

There are many benefits to implementing QA and security measures in a software project, and it significantly affects the outcome of the project. One can argue that the most important benefit is the detection of bugs and security risks in an early stage of the project. Taking QA into account at an early stage allows the development team to discover bugs and resolve them before they become more complex and harder to fix. Furthermore, this reduces both the time and the cost associated with fixing it in a later stage of the project. Implementing QA early on often takes significantly less time than addressing bugs and issues later in the development process.

Typically, the majority of software bugs tend to be identified during the feature testing phase. This phase is considered one of the most critical stages in the testing process due to its primary role in ensuring that each feature functions correctly and meets the required specifications. Software tests are designed not solely for the purpose of identifying bugs and other potential problems within a product. These tests also serve to demonstrate that the product behaves in accordance with its intended specifications and that its features operate as intended. Furthermore, particularly in the context of larger projects, tests serve an important role in allowing developers to understand the modifications implemented by their colleagues.

5.1.5 Challenges

QA and security professionals face many different challenges on a daily basis. This is an element that should be considered more thoroughly during project planning. Among

the challenges that QA experts face are the difficulties of obtaining a clear understanding of the requirements and criteria of a project. People are discussing the requirements and criteria of the product in meetings, but these are not documented well enough. In case the project lacks detailed documentation, it can lead to misinterpretations and inconsistencies in testing, making it difficult for QA teams to ensure that the product meets the intended standards. As a result, QA experts can spend additional time clarifying requirements, leading to delays and potential gaps in test coverage. Without clear documentation, it becomes challenging to create comprehensive test cases that align with the project goals and user expectations.

Once the different software tests are implemented, how often do developers look at the results of the tests, and is there a plan implemented if a test fails? It is important to consistently review the test results to maintain the stability and quality of the product. In the event of a test failure, it is vital to have a predetermined plan in place. Ideally, a rapid response approach should be established, such as designing a developer to quickly analyze and address the issue. A test failure does not necessarily indicate that a bug is present in the product. Newly introduced features and implementations might cause tests to fail, which causes the tests to become outdated. Therefore, it is important to understand how the tests work and what the result actually shows. Furthermore, new features and implementations cause the tests to inaccurately reflect the product's current state. Therefore, maintenance is essential to ensure that tests remain relevant and aligned with ongoing development efforts.

Lastly, it is tricky to handle all the different existing environments. The optimal scenario would be that the test environment would be perfectly similar to the production environment. However, this is rarely the case in practice. Differences in configuration, hardware, software versions, or network conditions can lead to unforeseen problems, leading to code that passes tests in testing environment settings but breaks in the production environment. This may lead to a loss of time and resources as developers hurry to find and resolve problems that were not detected during testing.

To address these challenges, the QA experts is trying to have good communication with both the customer for the project and with the development team. This can be achieved by having additional meetings and having a low threshold for reaching out to colleagues. Lastly, it is important to allocate the time needed to update existing tests as often as possible. If this is neglected, the need to update all tests at once can arise, which is a more demanding and error-prone task.

Security experts encounter challenges that can be grouped into various categories. Technical challenges can be addressed through reviews, implementations, or automated testing. Reviews involve examining systems, codes, or procedures to identify potential security risks or breaches. Implementations involve putting in place security measures, systems, or solutions to address these challenges. Automated testing refers to the use of tools and scripts to automatically scan and test systems for, for example, vulnerabilities, ensuring that any security flaws are identified and mitigated efficiently.

Within the development team, challenges may also arise. The team may have a tight deadline from management and simultaneously need to acquire new skills, learn how new tools work, or adapt to new workflows. In this scenario, management must offer greater support to the team. The organizational culture should embrace the concept that making mistakes is acceptable and everyone makes them. If individuals are too concerned with avoiding errors, it could obstruct their ability to work efficiently and adapt quickly, resulting in a slower development process. By accepting that errors are a natural part of growth, the organization can encourage creativity and maintain momentum in achieving project goals.

5.1.6 End Product

Generally, when a product reaches the release stage or a new version is introduced, no additional QA measures are usually implemented. Instead, it falls under the responsibility of the product owner and/or the product manager, often representing

the customer, to approve the product. This involves confirming that all essential features are included. The software testing carried out during the development phase is regarded as sufficient, with the assumption that it fully addresses and ensures the fulfillment of the product requirements.

Certain projects may involve the execution of specialized tests prior to the launch of a product, to ensure its optimal performance. One prime example of this process is load testing. The load test serves to evaluate how effectively the product operates when subjected to the stress of many, potentially hundreds, of concurrent users. Locust is a well-known tool utilized for conducting load tests within an application setting, facilitating the assessment of the product's behavior under such high-traffic conditions.

In the software project, the project team should have a clearly defined definition of done (DoD). The DoD refers to the point at which all specified conditions and acceptance criteria have been satisfied for a product, indicating its readiness for evaluation by a user, customer, team, or consuming system. In the past, there has been a lot of loose security testing. Today, certain security teams adhere to a checklist template consisting of tests such as vulnerability testing, static testing, and penetration testing that should be performed. The checklist helps to ensure that all necessary aspects are taken into account before finalizing the end product.

5.2 Product managers and owners

The interviews conducted with product owners and managers of Huld's customer companies aimed to uncover the unique approach of each company to QA and security within their software projects. These conversations sought to gain a deeper understanding of how these companies implement QA and security practices, the specific challenges they face, and the objectives they aim to achieve with Huld's support. This section also examines how Huld's collaborative approach aids in addressing customer needs, providing valuable insights into the alignment of Huld's services with the customer companies' QA and security goals. Through this approach, this section captures a comprehensive view of how product owners and managers from Huld's customer companies perceive and address QA and security challenges within their projects.

5.2.1 Project Overview

The software projects extend over various sectors, including industrial automation, renewable energy solutions, and advanced manufacturing. Each customer interviewed was noted to have specialized expertise in these fields. The varying sizes of customer projects have a direct impact on the size of the development team that is needed, as larger projects typically require more experts compared to smaller projects. Development teams may consist entirely of internal developers, entirely of developers outsourced from other companies, or more commonly a mix of in-house and outsourced developers. Recently, with the introduction of new and emerging technologies, the demand for developers to be outsourced has seen a significant increase. Occasionally, there might be budget issues for a project and no funding is available. In cases where funding is delayed or paused, the project may be put on hold and resume only once the budget becomes available. Often, when these projects are restarted, a new development team or new team members may be assigned to continue the work on the project.

5.2.2 QA and Security Integration

QA is being implemented and should be implemented on the first day of the project. When QA work is delayed, it often leads to increased complexity and challenges down the line, as issues can accumulate and become more difficult to address. Generally, there is no specific standard for QA that all projects must meet. The QA standard is typically project-specific, and the level of QA is decided together with the development team. Usually, the development team agrees on certain rules for pull requests, such as a certain percentage of the code must be covered by tests and ensuring that no new issues or security vulnerabilities are introduced.

Security considerations are also addressed early in the project, though active measures typically follow slightly later than QA efforts. The level of security needed are always based on the expectations from the companies, dictated by different standards that should be fulfilled, and based on the specific need from the type of project itself. For example, IEC 62443 is a commonly referenced standard that outlines security requirements across various projects. The IEC 62443 standard, along with other standards, helps to establish clear security expectations and provides a framework to implement the appropriate security measures to the project's demands. Lastly, the only common factor that is expected from the projects is the utilization of QA and security tools that companies use for all of their projects.

5.2.3 Cost-Benefit Analysis

From a cost perspective, QA work is considered to be part of development work, and therefore QA does not have a separate budget. Often, development teams do not have dedicated QA individuals. Instead, everyone on the team is doing the QA work and an individual with more knowledge in QA might have extra responsibilities like end-to-end testing. However, it is more difficult to calculate and forecast the budget needed for security measures when planning a project budget for the next quarters. The

project does not gain anything extra from doing unnecessary security work and can consume resources that would be better allocated elsewhere. It is sufficient for projects to comply with established standards, such as IEC 62443, which already provide a comprehensive framework to ensure security. These standards are robust and detailed and address the critical aspects of secure development. By focusing on meeting these requirements, teams can achieve the desired level of security without spending effort on redundant or excessive measures. Generally, security costs are incurred prior to the release phase, as security measures such as vulnerability testing and penetration testing are conducted at this stage. Therefore, security costs are not an ongoing cost that should be considered every day.

The license fees for tools that are utilized for QA and security work have a separate budget. This includes tools for static code analysis, software composition analysis, and vulnerability testing. QA and security related tools are often on the same budget as other tools with license fees such as Azure DevOps. The cost-benefit of investing in QA and security is not always noticed and appreciated enough. It is often challenging to recognize the benefits while everything is working as expected, but once something breaks, the benefits of good QA and security work will be noticed. QA and security could be compared to an insurance policy that is applied when the product faces unexpected challenges.

5.2.4 Impact on Project Outcomes

The impact of QA and security measures within a project is not always immediate evident or straightforward. Typically, it is only upon the occurrence of an incident, such as one triggered by a software bug or a security breach, that the QA and security protocols implemented for the product come under assessment. Consequently, there have been numerous instances that highlight the negative impact that insufficient QA and security practices can have on projects. On the plus side, such situations present valuable learning opportunities for individuals and forthcoming projects, emphasizing

the importance of prioritizing QA and security from the beginning. One specific security measure introduced to improve the security of a considerable amount of projects is to implement data encryption across the local networks. Within large organizations comprising thousands of employees, a single compromised device has the potential to trigger a security breach that affects numerous projects. This highlights the need for network-wide encryption as a crucial strategy to prevent unauthorized access and efficiently protect sensitive data.

In general, QA work significantly affects project timelines. The time taken to test and ensure that the requirements of the project are achieved and no incidents occur is most likely considerably less, compared to debugging and fixing bugs later on in the project. For the resources allocated, QA should be part of everyone's daily work. Within a team, it is important to have multiple specialists in various fields who can offer guidance to fellow team members in different domains when needed.

5.2.5 Feedback and Iteration

The majority of feedback is collected and is mainly received from different software tools that are used in the project. An additional source of feedback is through the various departments and organizations that conduct security assessments on the product. Feedback is only passed through by the development team for that specific project, which means that other projects are unable to learn and gain insight from the feedback received. Furthermore, upcoming projects lack access to past feedback and findings related to QA and security measures. This limitation hinders their ability to build on previously gathered insights, which are crucial for improving processes and improving security measures.

The information gathered was quite worrying and some improvements could be made. One potential improvement would be to create a centralized feedback repository where the QA and security findings of different projects could be documented,

categorized, and shared between teams. This resource would enable developers and security professionals to identify patterns, recurring issues, and successful mitigation strategies. In addition, integrating these findings into, for example, project planning phases could accelerate the implementation of best practices, particularly for new projects or teams unfamiliar with previous challenges.

Furthermore, regular reviews of this shared repository could provide opportunities for cross-project learning and refinement of QA and security protocols. This approach would encourage a culture of continuous improvement, where teams adapt and enhance their processes proactively based on collective knowledge and shared experiences. By promoting collaboration and knowledge sharing, organizations could reduce incidents and ensure more robust QA and security practices across all projects.

5.2.6 Best Practices and Lessons Learned

There has been a lot of lessons learned through the years and some best practices have been discovered. The most important factor is to start integrating both QA and security directly from the very beginning of a project. It is too late to consider these aspects as the development team approaches the release date. In addition, it is crucial to take into account the capabilities of automated pipelines in the early phases of planning, particularly when designing the architecture of the project. If the architectural design does not account for this consideration, adapting the project to a later phase becomes challenging. Furthermore, prioritizing automation integration at the beginning can help prevent the need for expensive modifications and adjustments in the future should these elements not have been prioritized from the beginning.

Defining acceptance tests and criteria directly within the work items themselves is generally regarded as a beneficial practice. This provides a reliable perspective on the necessary actions that should be implemented and describes how to confirm that all components function according to the specifications. By simply analyzing the

acceptance tests, it provides a clear understanding of how a feature is intended to function. Moreover, upon receiving a bug report from either the testing team or a customer, it becomes possible to revisit the feature in question and examine the acceptance test where the bug was detected. This process allows for the determination of whether the issue actually represents a bug or if it is, in fact, a new feature or a result of user error.

An effective approach for product managers and product owners is to consistently question the assumption that the product is working as expected. The focus should be on understanding how we can verify that it indeed functions as intended. In the absence of a verifiable test, the development team lacks assurance about the reasons behind its proper functioning. Another approach to consider is to rely on the expertise and understanding of the security experts. There have been instances where lone developers have been responsible for creating the security features of a project, believing that these features were sufficiently strong. Eventually, it was discovered that this assumption was incorrect and that potential security breaches were indeed a possibility. Finally, make an effort to develop components that can be reused, such as customized sign-in interfaces specific to the company and pipeline configurations, which can be implemented in multiple projects. The reason for this approach is to avoid multiple implementations of identical components to reduce the likelihood of a security breach or the occurrence of a bug.

5.3 Internal company channels and data

By utilizing internal channels, QA and security experts can collect essential information to support the implementation of QA and security measures within their software projects. In Huld, there are diverse competence teams, each specializing in its own particular area. Huld employees can seek advice and support from other developers who are experts in the field from the broad range of competence teams. There are different competence teams that specialize in various security areas, as well as a QA competence team. Occasionally, certain teams gather for biweekly meetings, providing an opportunity to address various topics. These internal channels can also facilitate real-time collaboration between teams, enabling developers, testers, and security professionals to share insights and address vulnerabilities collaboratively.

In terms of security, different types of documentation are available to help security experts and other interested parties apply security standards and conduct security assessments. These resources serve as essential guides, offering detailed instructions, best practices, and compliance frameworks adapted to meet the varied needs of different project requirements.

Huld has taken a proactive approach to professional development by hosting a range of courses on various topics for its experts to attend. Among these are specialized courses focused on QA and security, which are available free of charge to all employees. These courses serve dual purposes: they enable experts in the respective fields to deepen their knowledge and improve their skills, while also providing opportunities for individuals from other fields to broaden their expertise. Furthermore, courses can possibly spark interest among those with little or no previous experience in a given subject, encouraging them to explore new areas and potentially expand their career focus. By fostering a culture of learning and development, Huld not only strengthens the capabilities of its workforce, but also improves collaboration and knowledge sharing between different teams and departments.

Within an organization, internal company data and communication channels play an important role in supporting developers in their projects. However, these resources are often underutilized or not utilized to their full potential. In many cases, developers may not be fully aware of the information available or how to use it effectively in their day-to-day work. Furthermore, the lack of clear guidelines or promotion of these resources can lead to missed opportunities for collaboration and simplified problem solving.

6 Results

The data collected through the literature review and the insights obtained from the interviews with Huld experts, together with the information acquired from the PO and PM at Huld's customer companies, have produced some notable recognitions. Firstly, quality assurance and security measures significantly impact both software projects and their results. Finding a source that claims the opposite turns out to be quite challenging. Customers tend to refrain from purchasing a product that is associated with numerous security vulnerabilities. These vulnerabilities often arise from insufficient development practices and a lack of attention to security measures throughout the development process. Essential activities like threat modeling and the use of tools like code analyzers have remained unutilized [63]. This principle similarly applies to QA, where customers are generally hesitant to acquire an application that is prone to bugs. This situation often results from the development team's insufficient focus on thorough testing and verification procedures to ensure the product's functionality and reliability.

The interviews underscored the value of automation in QA and security and that the architecture of the project needs to allow this. The interviewees stated that the projects that invested in automated pipelines early in the project lifecycle reported fewer errors and faster iteration cycles. For example, a developer shared: "*Our automated nightly tests caught issues that we would otherwise have missed, saving us days of debugging.*" Automation also freed up team members to focus on more complex tasks, improving overall efficiency within the development team. However, some participants noted challenges in scaling automation for legacy systems, suggesting that additional resources or expertise may be needed in such scenarios. An additional notice from the interviews was that inconsistencies between the testing and production environments emerged as a significant issue in many projects. An expert recalled that, "*We had a system pass all tests in the testing environment, but it crashed in production due to subtle differences in database configurations.*" This highlights the need for standardized environments that use, for example, containerization tools to reduce such

inconsistencies.

One of the most notable trends of the interviewees has been the importance of well-defined requirements for projects. As noted mainly by interviewees, it is difficult to work on a project and implement QA and security if the requirements are not defined so that everyone on the team understands them. Having stakeholders, PO, PM, or some members of the development team understand the requirements is not sufficient. There must be a collaborative effort of the entire project team to define and understand the requirements. One simple way to improve this is to have a low threshold in the team to reach out to fellow team members. Additionally, the threshold for discussing matters with the PO and PM as well as with stakeholders must be low as well.

Another important recommendation is the inclusion of a dedicated security professional or a team member with a strong interest in security within every development team. This individual would serve as a point of focus for integrating security measures throughout the development lifecycle. This would help bridge the gap between development and security, promoting a more proactive approach rather than passively addressing issues after deployment. Moreover, the presence of a security-focused person within the team could improve the team's overall awareness and commitment to secure development practices. By promoting a development culture of shared responsibility for security, this role would help ensure that security considerations are not treated as an afterthought but rather as an integral part of the development process.

Furthermore, having such a security-focused individual within the team would also be highly beneficial for external or specialized security departments responsible for conducting tests on the product. This person could act as a knowledgeable point of contact, ensuring that the testing team has access to the relevant documentation, design details, and understanding of the security measures implemented. This collaboration would facilitate more efficient and effective testing processes, allowing security departments to accurately identify potential vulnerabilities and provide actionable feedback. As a result, this role would streamline communication between

the development and testing teams, contributing to a stronger and more secure final product.

Finally, producing high software quality together with strong security measures affects the cost of the projects in a positive way. Although the cost associated with implementing robust QA and security measures is undeniable, it is frequently considerably lower than the costs that would be incurred by entirely ignoring QA and security oversight throughout the project's development lifecycle. In fact, investing in QA and security early in the project can significantly reduce the likelihood of expensive post-deployment fixes, security breaches, and costly rework. Additionally, high-quality software can result in fewer customer complaints, reducing support costs, and enhancing customer loyalty.

7 Discussion

Considering that the thesis is based on qualitative rather than quantitative data, there are natural limitations regarding statistical validation and the ability to generalize the results. Although qualitative data provide in-depth understanding and context, it naturally remains subjective, representing the individual experiences and perspectives of the participants, in this case the interviewees. As a result, this limitation suggests that the findings may not be universally applicable to every organization or project, particularly those with significantly different practices, resources, or industry-specific needs. To address this limitation, it is crucial to interpret the results as insights rather than as definitive conclusions. Due to the subjective aspect of the data, the experiences that the interviewees share may be shaped by their distinct roles, the scope of their projects, and the cultures of their companies.

Expanding on this, future studies could benefit from incorporating a hybrid approach, combining qualitative insights with quantitative data, such as performance metrics, deeper cost analysis, and bug incidences, to provide a more comprehensive and statistically robust understanding of the impact of QA and security measures. This would help validate the findings, allowing more broadly applicable recommendations while preserving the depth of contextual insights provided by qualitative research.

Another limitation is that Huld is involved in a wide variety of projects, each of which tends to be distinct and is not comparable to other software projects. That is, these projects are either created for or in collaboration with unique customers, involve several different programming languages and combinations of programming languages, and comply to multiple different standards based on the project needs. In addition, various tools might need to be utilized that are used specifically in different programming languages or by specific customers. This makes it rather difficult to standardize a certain way of employing QA and security measures in a project. This implies that a universal approach to QA and security is not achievable within Huld's projects.

Instead, a customized approach must be adopted for each project, taking into account the unique requirements, technologies, and standards associated with the customer and the project itself. This lack of standardization can create challenges in terms of knowledge transfer, consistency, and efficiency, as teams may need to invest additional time and resources to adapt practices to fit each specific context.

Furthermore, the diversity of programming languages and tools means that maintaining expertise across all relevant technologies becomes a considerable challenge for QA and security experts. This fragmentation can also complicate the implementation of automated processes or pipelines, as they may need to be customized for each environment. As a result, the ability to establish and maintain best practices that are universally applicable across projects is inherently limited, potentially leading to variability in the quality and security outcomes achieved. Addressing this limitation may require greater investment in knowledge sharing between projects, standardized guidelines for flexible QA and security practices, and the development of modular tools or frameworks that can adapt to different contexts while maintaining a baseline of quality and security.

Lastly, maintaining automated tests has emerged as a recurring challenge in software projects, particularly in dynamic environments with frequent updates. One of the key issues identified was the tendency for tests to become outdated as the codebase evolves. This often leads to tests beginning to fail due to changes in functionality or interface rather than actual defects. This problem is increased in teams with tight deadlines, where the priority is often on delivering features rather than keeping the test suite up to date. The eventual build-up of unsuccessful tests may undermine trust in the testing framework, prompting developers to either ignore or dismiss the outcomes.

Addressing this challenge requires a proactive approach. Regularly scheduled reviews of the test suite, combined with refactoring outdated tests, can prevent failures from building up. Some interviewees also highlighted the importance of fostering a testing culture within the development team, where maintaining tests is treated as an integral

part of the development process rather than an afterthought. By prioritizing test maintenance, teams can preserve the reliability of their testing frameworks and ensure that automated tests continue to provide value throughout the software lifecycle.

8 Conclusion

To conclude, this thesis demonstrates that carrying out a software project is impractical without thoroughly considering QA and security measures. As discussed and noticed, both QA and security play an important role for all software projects regardless of size, and the impact is irreplaceable. From both a customer and user perspective, investing in or utilizing a product filled with bugs and security vulnerabilities is not a viable option. Beyond ensuring functionality and safety, robust QA and security practices contribute to building trust and reliability, which are essential for the long-term success of software products. Neglecting these measures not only risks project failure, but can also damage a company's reputation and financial stability. Hence, it is advisable for Huld to continue investing in and providing QA and security solutions to its customers. QA and security will not become less important in the future; on the contrary, they will become increasingly important as technology evolves, security threats grow, and new regulations are implemented.

One key takeaway from this thesis is the critical role of early integration of QA and security measures. By embedding these practices into the development lifecycle from the beginning, teams can identify and address potential problems before they become costly problems. Early integration not only ensures better alignment with project goals, but also facilitates smoother collaboration between stakeholders. This proactive approach promotes QA and security, reducing the likelihood of post-deployment vulnerabilities and enhancing the overall reliability of the product. In addition, investing in QA and security results in long-term benefits that extend beyond individual projects. Products that consistently deliver high quality and robust security build a positive reputation for the organization, increasing customer trust and retention. Over time, these benefits compound, positioning companies that prioritize QA and security as leaders in the competitive software industry.

Selecting the "correct" methods, tools, and level of QA and security can be quite

a challenge, especially from a QA perspective. Since most software projects differ greatly in scope, requirements, and context, it is difficult to recommend a standardized approach to the implementation of QA and security. All developers must adapt to the specific needs of their projects, learning to balance priorities while ensuring quality and safety. However, it is equally critical to establish a solid foundation on which all projects can be based. The foundation should ideally consist of a suitable development team, which, if possible, includes a security-focused developer or one with a keen interest in security. Furthermore, it is essential that all members of the development team are aligned in incorporating the QA and security measures into the software project.

In addition, a clear and mutual understanding of the project requirements between all stakeholders is essential for project success. Misaligned expectations or poorly defined requirements can lead to inefficiencies, rework, and vulnerabilities in the final product. Therefore, it is crucial to create low barriers for communication between stakeholders, such as product owners, managers, and developers. Regularly revisiting and refining requirements as the project evolves can further ensure alignment and maintain focus on quality objectives.

References

[1] David Woodhouse. Quality and quality assurance. *Quality and internationalisation in higher education*, pages 29–44, 1999.

[2] Hala Assal and Sonia Chiasson. Security in the software development lifecycle. In *Fourteenth symposium on usable privacy and security (SOUPS 2018)*, pages 281–296, 2018.

[3] Borislav Nikolik. Software quality assurance economics. *Information and Software Technology*, 54(11):1229–1238, 2012.

[4] Per Runeson and Peter Isacsson. Software quality assurance-concepts and misconceptions. In *Proceedings. 24th EUROMICRO Conference (Cat. No. 98EX204)*, volume 2, pages 853–859. IEEE, 1998.

[5] Daniel Galin. *Software quality assurance: from theory to implementation*. Pearson education, 2004.

[6] Maneela Tuteja, Gaurav Dubey, et al. A research study on importance of testing and quality assurance in software development life cycle (sdlc) models. *International Journal of Soft Computing and Engineering (IJSCE)*, 2(3):251–257, 2012.

[7] Bruce R Maxim and Marouane Kessentini. An introduction to modern software quality assurance. In *Software quality assurance*, pages 19–46. Elsevier, 2016.

[8] Juan José Tarí, José Francisco Molina-Azorín, and Iñaki Heras. Benefits of the iso 9001 and iso 14001 standards: A literature review. *Journal of Industrial Engineering and Management (JIEM)*, 5(2):297–322, 2012.

[9] ISO 9001:2015 — iso.org. https://www.iso.org/standard/62085.html.

[10] Paulo Sampaio, Pedro Saraiva, and António Guimarães Rodrigues. Iso 9001 certification research: questions, answers and approaches. *International journal of quality & reliability management*, 26(1):38–58, 2009.

[11] Gary McGraw. Software security. *IEEE Security & Privacy*, 2(2):80–83, 2004.

[12] Hernan Nina, José Antonio Pow-Sang, and Mónica Villavicencio. Systematic mapping of the literature on secure software development. *IEEE Access*, 9:36852–36867, 2021.

[13] International Organization for Standardization. *ISO/IEC 27000:2018(en) Information Technology - Security Techniques - Information Security Management Systems - Overview and vocabulary*. ISO, 2018.

[14] International Organization for Standardization. Iso 31000:2018(en) risk management – guidelines. 2018.

[15] Kristian Beckers, Isabelle Côté, Stefan Fenz, Denis Hatebur, and Maritta Heisel. A structured comparison of security standards. *Engineering secure future internet services and systems: Current research*, pages 1–34, 2014.

[16] OWASP Application Security Verification Standard (ASVS) | OWASP Foundation — owasp.org. https://owasp.org/www-project-application-security-verification-standard/.

[17] A Stock, D Cuthbert, J Manico, J Grossman, and M Burnett. Owasp application security verification standard. *Std. Rev*, 4(1):70, 2019.

[18] Kshirasagar Naik and Priyadarshi Tripathy. *Software testing and quality assurance: theory and practice*. John Wiley & Sons, 2011.

[19] John Babatunde. *Evaluating the Impact of Security Measures on Performance of Secure Web Applications Hosted on Virtualised Platforms*. PhD thesis, University of East London, 2015.

[20] Paul Ammann and Jeff Offutt. *Introduction to software testing*. Cambridge University Press, 2016.

[21] Tim A Majchrzak. *Improving software testing: technical and organizational developments*. Springer Science & Business Media, 2012.

[22] Stephan Goericke. *The future of software quality assurance*. Springer Nature, 2020.

[23] Quality Assurance & Testing - Huld — huld.io. https://huld.io/what-we-do/digital-services-and-software/quality-assurance-testing/.

[24] Anca Deak, Tor Stålhane, and Guttorm Sindre. Challenges and strategies for motivating software testing personnel. *Information and software Technology*, 73:1–15, 2016.

[25] International Society of Automation (ISA) — isa.org. https://www.isa.org/.

[26] International Electrotechnical Commission — iec.ch. https://iec.ch/.

[27] ISA/IEC 62443 Series of Standards - ISA — isa.org. https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards.

[28] Björn Leander, Aida Čaušević, and Hans Hansson. Applicability of the iec 62443 standard in industry 4.0/iiot. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, pages 1–8, 2019.

[29] ISA-62443-1-1-2007 Security for Industrial Automation and Control Systems Part 1-1: Terminology, Concepts, and Models — isa.org. https://www.isa.org/products/isa-62443-1-1-2007-security-for-industrial-automat.

[30] ISA-62443-2-1-2009, Security for Industrial Automation and Control Systems Part 2-1: Establishing an Industrial Automation and Control Systems Security

Program — isa.org. https://www.isa.org/products/isa-62443-2-1-2009-security-for-industrial-automat.

[31] ISA-TR62443-2-3-2015, Security for industrial automation and control systems Part 2-3: Patch management in the IACS environment — isa.org. https://www.isa.org/products/isa-tr62443-2-3-2015-security-for-industrial.

[32] ANSI/ISA-62443-2-4-2018 / IEC 62443-2-4:2015+AMD1:2017 CSV, Security for industrial automation and control systems, Part 2-4: Security program requirements for IACS service providers (IEC 62443-2-4:20 — isa.org. https://www.isa.org/products/ansi-isa-62443-2-4-2018-iec-62443-2-4-2015-amd1-20.

[33] ANSI/ISA-62443-3-2-2020, Security for industrial automation and control systems, Part 3-2: Security risk assessment for system design — isa.org. https://www.isa.org/products/ansi-isa-62443-3-2-2020-security-for-industrial-a.

[34] ANSI/ISA-62443-3-3-2013, Security for industrial automation and control systems Part 3-3: System security requirements and security levels — isa.org. https://www.isa.org/products/ansi-isa-62443-3-3-2013-security-for-industrial-au.

[35] ANSI/ISA-62443-4-1-2018, Security for industrial automation and control systems - Part 4-1: Secure product development lifecycle requirements (formerly Part 4-1: Product security development life-cycl — isa.org. https://www.isa.org/products/ansi-isa-62443-4-1-2018-security-for-industrial-au.

[36] ANSI/ISA-62443-4-2-2018, Security for industrial automation and control systems, Part 4-2: Technical security requirements for IACS components, 2nd Printing — isa.org. https://www.isa.org/products/ansi-isa-62443-4-2-2018-security-for-industrial-au.

[37] Hicham Lalaoui Hassani, Ayoub Bahnasse, Eric Martin, Christian Roland, Omar Bouattane, and Mohammed El Mehdi Diouri. Vulnerability and security risk assessment in a iiot environment in compliance with standard iec 62443. *Procedia Computer Science*, 191:33–40, 2021.

[38] Abdelkader Magdy Shaaban, Erwin Kristen, and Christoph Schmittner. Application of iec 62443 for iot components. In *Computer Safety, Reliability, and Security: SAFECOMP 2018 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Västerås, Sweden, September 18, 2018, Proceedings 37*, pages 214–223. Springer, 2018.

[39] Abdelkader Magdy Shaaban, Sebastian Chlup, Nahla El-Araby, and Christoph Schmittner. Towards optimized security attributes for iot devices in smart agriculture based on the iec 62443 security standard. *Applied Sciences*, 12(11):5653, 2022.

[40] Industrial & Software Security - Huld — huld.io. https://huld.io/what-we-do/safety-and-security/industrial-software-security/.

[41] Security Management - Huld — huld.io. https://huld.io/what-we-do/safety-and-security/security-management/.

[42] Martin Fowler and Matthew Foemmel. Continuous integration, 2006.

[43] Paul M Duvall, Steve Matyas, and Andrew Glover. *Continuous integration: improving software quality and reducing risk*. Pearson Education, 2007.

[44] John Ferguson Smart. *Jenkins: The Definitive Guide: Continuous Integration for the Masses*. " O'Reilly Media, Inc.", 2011.

[45] Nikita Seth and Rishi Khare. Aci (automated continuous integration) using jenkins: Key for successful embedded software development. In *2015 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS)*, pages 1–6. IEEE, 2015.

[46] Carmine Vassallo, Sebastiano Panichella, Fabio Palomba, Sebastian Proksch, Harald C Gall, and Andy Zaidman. How developers engage with static analysis tools in different contexts. *Empirical Software Engineering*, 25:1419–1457, 2020.

[47] Maria Teresa Baldassarre, Valentina Lenarduzzi, Simone Romano, and Nyyti Saarimäki. On the diffuseness of technical debt items and accuracy of remediation time when using sonarqube. *Information and Software Technology*, 128:106377, 2020.

[48] Diego Marcilio, Rodrigo Bonifácio, Eduardo Monteiro, Edna Canedo, Welder Luz, and Gustavo Pinto. Are static analysis violations really fixed? a closer look at realistic usage of sonarqube. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, pages 209–219. IEEE, 2019.

[49] Javier García-Munoz, Marisol García-Valls, and Julio Escribano-Barreno. Improved metrics handling in sonarqube for software quality monitoring. In *Distributed Computing and Artificial Intelligence, 13th International Conference*, pages 463–470. Springer, 2016.

[50] Lida Zhao, Sen Chen, Zhengzi Xu, Chengwei Liu, Lyuye Zhang, Jiahui Wu, Jun Sun, and Yang Liu. Software composition analysis for vulnerability detection: An empirical study on java projects. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 960–972, 2023.

[51] Software Composition Analysis: SCA Solutions | Black Duck — blackduck.com. https://www.blackduck.com/software-composition-analysis-tools.html#2.

[52] Yuma Makino and Vitaly Klyuev. Evaluation of web vulnerability scanners. In *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, volume 1, pages 399–402. IEEE, 2015.

[53] Dave Wichers and Jeff Williams. Owasp top-10 2017. *OWASP Foundation*, 3:4, 2017.

[54] OWASP Top Ten | OWASP Foundation — owasp.org. https://owasp.org/www-project-top-ten/.

[55] Balume Mburano and Weisheng Si. Evaluation of web vulnerability scanners based on owasp benchmark. In *2018 26th International Conference on Systems Engineering (ICSEng)*, pages 1–6. IEEE, 2018.

[56] Barry W Boehm and Philip N. Papaccio. Understanding and controlling software costs. *IEEE transactions on software engineering*, 14(10):1462–1477, 1988.

[57] Barry W Boehm. *Software engineering economics*. Springer, 2002.

[58] Barry Boehm, Bradford Clark, Ellis Horowitz, Chris Westland, Ray Madachy, and Richard Selby. Cost models for future software life cycle processes: Cocomo 2.0. *Annals of software engineering*, 1:57–94, 1995.

[59] Herb Krasner. The cost of poor software quality in the us: A 2020 report. *Proc. Consortium Inf. Softw. QualityTM (CISQTM)*, 2, 2021.

[60] Capers Jones and Olivier Bonsignour. *The economics of software quality*. Addison-Wesley, 2012.

[61] Renate M Kahlke. Generic qualitative approaches: Pitfalls and benefits of methodological mixology. *International journal of qualitative methods*, 13(1):37–52, 2014.

[62] Apoorva Srivastava, Sukriti Bhardwaj, and Shipra Saraswat. Scrum model for agile methodology. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 864–869. IEEE, 2017.

[63] Andreas Poller, Laura Kocksch, Sven Türpe, Felix Anand Epp, and Katharina Kinder-Kurlanda. Can security become a routine? a study of organizational change in an agile software development group. In *Proceedings of the 2017*

ACM conference on computer supported cooperative work and social computing, pages 2489–2503, 2017.

A Interview questions - Huld employees

1. Project Involvement
    - Describe the types of software project you typically work on (internal or external)?
    - What is the size of the development teams you have worked with?
    - At what stages of the project life cycle are you usually involved (start, ongoing, or end)?
2. Implementation
    - How do you integrate QA and security measures into the project?
3. Costs
    - How do you estimate the costs associated with implementing QA and security measures in a project?
    - How do these costs scale with the size and complexity of the project?
4. Benefits and Outcomes
    - What noticeable benefits have you observed from incorporating QA and security measures in projects?
    - Can you provide examples of how early integration of QA and security has benefited a project?
    - Can you share anything or case studies that highlights the impact of these measures on project success?
5. Challenges
    - What are the common challenges you face when implementing QA and security measures?
    - How do you address these challenges to ensure project success?

6. End Product

    - How do you assess the quality and security of the end product?
    - What criteria and tools do you use for this assessment?

B Interview questions - Product Managers and Owners

1. Project Overview
    - Can you describe the nature and scope of the software projects that you oversee?
    - Are these projects typically developed in-house or outsourced?
2. QA and Security Integration
    - At what point in your projects do you start integrating QA and security measures?
    - How do you decide on the appropriate level and extent of QA and security for a project?
3. Cost-Benefit Analysis
    - How do you budget for QA and security measures in your projects?
    - What are the perceived and actual benefits of investing in these areas?
4. Impact on Project Outcomes
    - Can you provide examples in which QA and security measures have significantly influenced the project outcome?
    - How do these measures affect the project timeline and resource allocation?
5. Feedback and Iteration
    - How do you gather feedback on QA and security from your teams and stakeholders?
    - How is this feedback used to improve future projects?
6. Best Practices and Lessons Learned

- What best practices have you developed for integrating QA and security into your projects?
- Can you share any lessons learned from previous projects in regard to QA and security?