CHALMERS
UNIVERSITY OF TECHNOLOGY

Understanding Software Quality Management in Automotive Software Development

A case study

Master's thesis in Quality and Operations Management

Carl-Philip Sjöstrand
Ifeanyi Uzoigwe

DEPARTMENT OF TECHNOLOGY MANAGEMENT AND ECONOMICS
DIVISION OF INNOVATION AND R&D MANAGEMENT

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Understanding Software Quality Management in Automotive Software Development

Carl-Philip Sjöstrand
Ifeanyi Uzoigwe

Department of Technology Management and Economics
Division of Innovation and R&D Management
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Understanding Software Quality Management in Automotive Software Development
Carl-Philip Sjöstrand
Ifeanyi Uzoigwe

Department of Technology Management and Economics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone + 46 (0)31-772 1000

Understanding Software Quality Management in Automotive Software Development

Carl-Philip Sjöstrand
Ifeanyi UzoigweDepartment of Technology Management and Economics
Chalmers University of Technology

Abstract

Managing software quality within the automotive context is of importance as the automotive industry is becoming increasingly software defined. Especially since the recalls of vehicles have increased due to software defects. Furthermore, the theory highlight challenges in defining what software quality is. This study thereby aims to increase the understanding of how software quality is defined within the automotive context, what quality management practices are important, and what factors impact software development and quality. To address this, a qualitative case study was conducted at a global software supplier to the automotive industry. The study primarily focuses on development processes rather than the quality specification of the final software product. The data collection was conducted through observations, documentation reviews, and semi-structured interviews with 28 participants.

The study shows that software quality is perceived as multidimensional and context-dependent, with emphasis on both technical and customer-experience aspects. The study identified important automotive software quality attributes such as safety, security, robustness, and reliability. The study identified important software quality management practices, consisting of reviews, gates, testing, traceability, and continuous improvement. However, the study also indicated ambiguities in the understanding and application of software quality assurance and control. This lack of a shared conceptual understanding creates difficulties in aligning responsibilities, communicating quality expectations and improving processes. Finally, five main factors were identified impacting software development and quality: process, requirements, system complexity & dependencies, time and testing. These factors are related to unclear requirements, fragmented processes, time pressure, and limited testing capabilities that were recurring in the findings.

Keywords: Software quality, Software quality management, Automotive software development, Software Quality assurance, Software Quality control, Software Requirement

Acknowledgements

We would like to express our gratitude to our supervisors and all the colleagues at the case company who welcomed and supported us throughout this thesis. Thank you for participating in the interviews and discussions, your contributions have been valuable for this study.

We would also like to thank our supervisor at Chalmers University of Technology, Hendry Raharjo. We appreciate your continuous support, valuable feedback, and guidance throughout this thesis.

Sincerely,

Carl-Philip Sjöstrand & Ifeanyi Uzoigwe, Gothenburg, June 2026

List of Acronyms

ASQM: Automotive Software Quality Management

ASIL: Automotive Safety Integrity Level

ASPICE: Automotive Software Process Improvement and Capability Determination

EARS: Easy Approach to Requirements Syntax

EM: Engineering Manager

IATF: International Automotive Task Force

ISO: International Organization for Standardization

KPI: Key Performance Indicator

OEM: Original Equipment Manufacturer

PO: Product Owner

QA: Quality assurance

QC: Quality control

QM: Quality Manager

QMS: Quality management system

SDLC: Software Development Life Cycle

SQA: Software Quality Assurance

SQC: Software Quality Control

SQM: Software quality management

SUMS: Software Update Management System

SW: Software

Table of Contents

Table of figures

Table of tables

1 Introduction

As vehicles have become mores software-defined, lines of code have increased, making automotive Software (SW) and its development more complex for OEMs (Ryu & Do, 2023; Wang et al., 2024). The increased complexity of automotive SW has resulted in an increased number of safety accidents and recalls due to errors in the SW (Ryu & Do, 2023). In 2018, in the US alone, 8 million vehicles were recalled due to SW defects (Ayres et al., 2021). Broy et al. (2007) highlighted these challenges early, noting that automotive SW could already consist of tens of millions of. Today, automotive SW can exceed 100 million lines of code more than the Boeing 787 Dreamliner (6.5 million lines of codes) (Khamis & Goswami, 2025). Zhao et al (2025) explain that development methodologies have also undergone changes enabling faster and more frequent release cycles, in addition there is a demand for more advanced functionalities, further increasing SW complexity.

Trovão (2024) emphasizes that safety is a prime concern within the automotive industry, but also that quality is gaining a bigger focus to ensure reliable SW. However, Green et al. (2024) highlight that SW quality is not easily defined. Zhao et al. (2025) also highlights that achieving good SW quality is therefore not merely desirable but essential to avoid the financial and reputational consequences of poor SW quality. SW Quality Assurance (SQA) therefore becomes important in handling increased SW complexity and ensuring high-quality SW (e.g., reliability, robustness, and safety) (Zhao et al., 2025).

Challenges related to increasing SW complexity have also been expressed by the case company, a global SW supplier to the automotive industry, delivering SW products and services. Consequently, there is a need to better understand how SW quality is perceived within the case company. Additionally, it is important to understand which SW Quality Management (SQM) activities are carried out within the case company. This increased understanding of SW quality perceptions and SQM practices is important to support the delivery of high-quality SW to customers and to reduce the risk of costly downstream consequences associated with poor SW quality.

1.1 Purpose

The purpose of this study is to increase the understanding of how SW quality is defined at the case company and identify practices and factors impacting SW quality.

1.2 Research Questions

The following research question will be examined and answered within the automotive SW development context.

1. How is software quality defined within the automotive context?

2. What practices are important for automotive software quality management?
3. What factors impact automotive software development and quality?

1.3 Delimitations

This study will be delimited to investigate the SW development processes and activities at the case company, thereby not looking into the quality specifications of the final product. Due to time constraints, purposively chosen departments within the case company and individuals were picked for interviews based on their expertise, as it would not be feasible to interview all departments and individuals within the case company. Instead, individuals with extensive knowledge about the development activities and experience working with SW were picked for interviews to ensure relevant and valuable information for the study.

1.4 Disposition of Thesis

The thesis's outline is structured in the following way.

Chapter 1 – Introduction; Introduce the reader to the topic, provide the reader with the purpose of the study, research questions, and delimitations.

Chapter 2 – Theory; Provide the reader with relevant theory to give deeper insight into the topic.

Chapter 3 – Methodology; Highlights how the study has been designed and conducted.

Chapter 4 – Results; Provides the reader with empirical findings of the study.

Chapter 5 – Discussion; Compare empirical findings with the literature and identify connections or contradictions with the theory.

Chapter 6 – Conclusion; Will answer the research question, present limitations of the study and discuss future research that may deepen the knowledge in the area.

2 Theoretical framework

In this section, the theoretical background relevant to the research is presented. First, a brief introduction to quality is provided. This is followed by an overview of the foundations of quality management and its primary focus. Third, SW development and development frameworks are discussed. Subsequently, SW quality and SW quality management are addressed. Finally, automotive SW quality management is introduced.

2.1 Quality

Quality is not an easily defined concept, and its underlying meaning can be defined in various ways (De Giovanni, 2023; Solin & Curry, 2022). Garvin's eight dimensions of quality from 1987 have been influential in defining and specifying quality. These dimensions include features, performance, reliability, conformance, durability, serviceability, aesthetics, and perceived quality (De Giovanni, 2023). Two of Garvin's eight dimensions of quality that are relevant to this study are defined as follows (De Giovanni, 2023, p. 311):

- *Reliability: "this signifies the period during which the product offers high-level performance without malfunctioning".*
- *Performance: "This corresponds to the primary operational characteristics of a product and implies the identification of a series of measurable attributes".*
- *Conformance: "this is the ability of a product or service to meet specified standards, considering the number and types of defects found in a product during its life cycle and use".*
- *Perceived quality: "This is the assessment of quality based on a series of indirect measures".*

Perceived quality (PQ) further complicates the definition of quality because it is user (customer)–based rather than based on product characteristics (Solin & Curry, 2022). De Giovanni (2023) similarly highlights a shift in quality perception toward a stronger customer focus, rather than internally defined quality dimension. Quality can therefore be differentiated between "objective quality," which is product-oriented and can be verified by measuring performance against predetermined standards, and "subjective quality," which refers to how users respond to a product when interacting with it (De Giovanni, 2023; Solin & Curry, 2022).

2.2 Quality management

Today's business environment is increasingly competitive, and legislation also influences how companies operate. This places greater pressure on organizations to achieve high product quality. Quality Management aims to improve product quality and enhance processes (Psarommatis & Azamfirei, 2024). Quality management can

be described as a set of practices, procedures, and documentation that helps an organization manage quality, forming the basis for a Quality Management System (QMS). The objective of QMS is to continuously improve an organization's products and processes (Bozola et al., 2023). Quality management typically includes three key procedures: Quality Improvement (QI), Quality Control (QC), and Quality Assurance (QA) (Psarommatis & Azamfirei, 2024)

ISO 9001 is a frequently used foundation for organizations when developing a QMS (Gremyr et al., 2021; Bozola et al., 2023). Gremyr et al. (2021) note that, at the time, more than one million organizations had obtained ISO 9001 certification. Using ISO 9001 as a foundation when developing a QMS may have a positive impact on product quality and process improvement (Gremyr et al., 2021).

While there are no predefined instructions or rules for how to develop and implement a QMS, ISO 9001 may provide guidance (Bozola et al., 2023). In the automotive context, the ISO/TS 16949 standard was updated in 2016 and renamed IATF 16949, which can offer further guidance. It was developed as a complement to ISO 9001 and specifies requirements across the different phases of automotive industry processes (Bozola et al., 2023).

Gremyr et al. (2021) highlights that QMS has been criticized in academia for hindering creativity, being disconnected from actual practices, and failing to provide the intended support for quality improvement (QI). However, they also note that academic research provides evidence that QMS can deliver benefits, such as improved product quality and processes, and can support continuous improvements (Gremyr et al., 2021).

2.2.1 ISO 9000 and ISO 9001

ISO 9000 (2015) establishes the fundamental concepts, principles, and vocabulary of quality management. It aims to improve organizational understanding, thereby increasing the likelihood that a QMS is implemented effectively and efficiently and that its potential value is realized. Quality management is described as management with regard to quality, including establishing quality policies, objectives, and processes to achieve them. It also includes quality assurance, quality control, and quality improvement. The relationship between QMS, QI, QA, and QC is illustrated in *Figure 2.1.*

Figure 2.1 Quality management system build up

According to ISO 9000 (2015), quality assurance, quality control, and quality improvement are defined as follows:

> *"Quality assurance: part of quality management (3.3.4) focused on providing confidence that quality requirements (3.6.5) will be fulfilled"* (Clause 3.3.6, p.14)

> *"Quality control: part of quality management (3.3.4) focused on fulfilling quality requirements (3.6.5)"* (Clause 3.3.7, p.14)

> *"Quality improvement: part of quality management (3.3.4) focused on increasing the ability to fulfil quality requirements (3.6.5)"* (Clause 3.3.8, p.14)

Requirements according to ISO 9000 (2015) is stated in the following way; *"need or expectations that is stated, generally implied or obligatory"* (Clause 3.6.4, p.19). Quality is described as the extent to which a product's inherent characteristics meet requirements. Quality requirements consider requirements in relation to quality.

QMS consists of activities aimed at identifying and establishing the objectives an organization wants to achieve, as well as the processes and resources needed to achieve them. It is important to understand the organization's context, as it may affect these objectives; for example, external factors such as legislation can have an impact. Product quality is defined as an organization's ability to satisfy customers' and stakeholders' needs, while also considering perceived customer value. By establishing a QMS, organizations can identify actions to address both intended and unintended effects of their products (ISO 9000, 2015).

ISO 9000 (2015) outlines seven quality management principles that should be considered. For the purposes of this study, six of these principles are of interest:

customer focus, leadership, engagement of people, process approach, improvement, and evidence-based decision making (ISO 9000, 2015). These six quality management principles, and how they are defined in ISO 9000 (2015), are described below.

Customer focus: As a foundational principle of quality management, organizations need to meet customers' requirements and should aim to exceed their expectations. With this focus, organizations can attract and maintain customers' and other stakeholders' confidence in both their products and the organization. Interactions with external surroundings are therefore important for better understanding current and future needs. This, in turn, influences an organization's ability to sustain success over time.

Engagement of people: To promote engagement within the organization, it is important to empower and involve individuals, as this supports the organization's ability to create and deliver value. This can be achieved through, for example, collaboration, knowledge sharing, and communication about individual contributions. A potential outcome is improved understanding of the quality objectives and greater motivation to achieve them.

Process approach: For organizations to better understand how results are delivered through their processes and what is required to achieve them they need a clear understanding of those processes. By mapping their processes, dependencies, and building a system for how they should be managed; organizations can deliver more predictable and consistent results. A better understanding of the system and how outcomes are produced also enables the identification of improvement opportunities and ways to enhance overall system performance.

Improvement: To maintain success over time, organizations need to focus on improvement by continuously enhancing process performance. In addition to sustaining success, improvement enables organizations to respond to new circumstances arising from internal or external forces by adapting to potential risks and opportunities. This can be achieved, for example, by encouraging improvement objectives at different levels within the organization. In this regard, it is important that personnel have the necessary knowledge and tools to achieve these objectives. Finally, by assessing improvement projects from planning to results through tracking progress and conducting reviews organizations can evaluate the impact of these initiatives.

Evidence-based decision making: Decision making is not always straightforward and can be influenced by personal perspectives. Decisions based on data that have been analyzed and evaluated are more objective and increase the likelihood of achieving the desired result. Data are therefore important; as stated in ISO 9000:2015, organizations should *"determine, measure and monitor key indicators to demonstrate the organization's performance"* (Clause 2.3.6.4, p. 8). Additionally, it is important

for the organization to ensure confidence in the data's reliability and accuracy, as well as its availability to relevant personnel.

The ISO 9000 (2015) standard provides guidance rather than instructions on how to implement a QMS. Each organization's QMS will have its own characteristics because organizational contexts vary; therefore, it needs to be flexible and adaptable to the specific context. An important aspect is to define the organization's processes, the activities within those processes, and how these should be measured, as they form the foundation for how organizations deliver results. Doing so also enables organizations to identify process improvements (ISO 9000, 2015).

ISO 9000 (2015) highlights that the fundamental concepts and principles of quality management can provide valuable insights for organizations on how to develop a QMS. When planning a QMS implementation, organizations can use guidance from ISO 9000 together with the requirements of ISO 9001. In addition, a QMS is not static; it is an iterative process as organizations gain new knowledge and circumstances change (ISO 9000, 2015).

It is therefore important to monitor and measure not only the processes, but also the QMS itself, to assess its performance. Auditing is a tool organizations can use to evaluate the effectiveness of their QMS, identify risks, and determine the extent to which requirements are fulfilled. By collecting data, organizations create a basis for decision making, enabling them to correct the QMS if performance is unsatisfactory or to further improve it (ISO 9000, 2015).

ISO 9001 is built upon the same quality management principles introduced in ISO 9000, but it differs in its use of the term "shall," which indicates a requirement that organizations must fulfill (ISO 9001, 2015). For example, under customer focus, it states: *"Top management shall demonstrate leadership and commitment with respect to customer focus by ensuring that: a) customers and applicable statutory and regulatory requirements are determined, understood and consistently met;"* (ISO 9001, 2015, Clause 5.1.2). Organizations that adhere to these quality management standards, fulfill the requirements in ISO 9001, and undergo auditing may be granted ISO 9001 certification (Cândido, 2024).

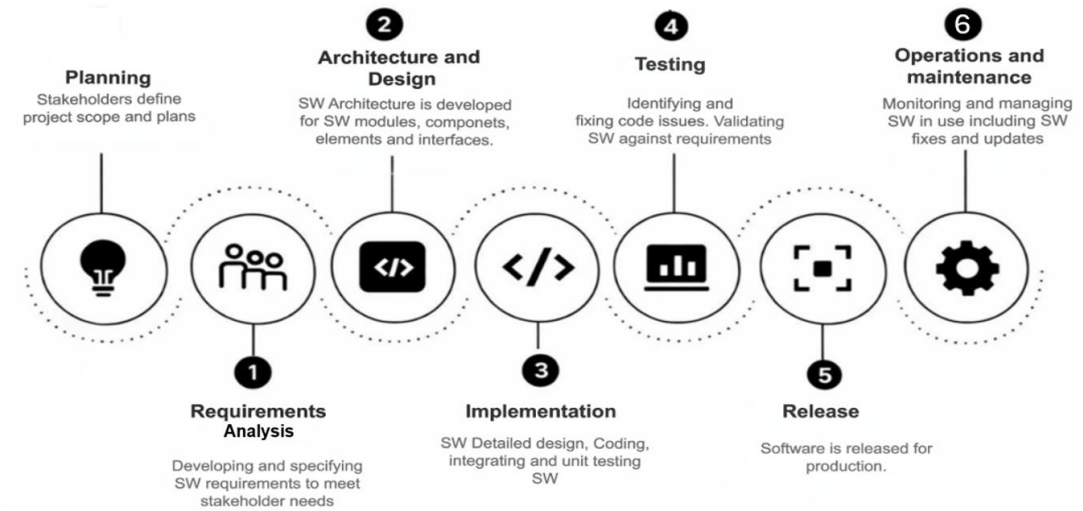
2.3 Software development

SW development is described as organized engineering activities that transform customer and business needs into SW through a set of technical and management processes (ISO/IEC/IEEE 12207:2017). These processes are commonly understood as structured progression from initial SW planning throughout its lifecycle, including post-release maintenance (ISO/IEC/IEEE 12207:2017). Refer to *Figure 2.2* below, SW development is not limited to coding but a sequence of activities including planning, requirements analysis, architecture/design, implementation, integration and testing, deployment and maintenance working together, while also emphasizing

supporting processes such as quality management, configuration management, verification, and validation (ISO/IEC/IEEE 12207:2017).

Within the automotive context, Vogelsang (2020) highlights that automotive SW is decomposed into various vehicle domains. These domains consist of groups of subsystems that are built up by features delivering various SW functions. Previously, organizations tried to keep features independent, but today features are becoming increasingly interconnected to deliver innovative functionalities. As the business landscape becomes more competitive, there is a greater focus on short-term aspects, such as the delivery of features, rather than long-term aspects, such as the maintainability of the SW. The author concludes that current practice still treats features in isolation rather than as interconnected, which is no longer the reality of today's SW-defined vehicles. These dependencies pose challenges for automotive SW development, affecting requirements specification, system integration, and testing (Vogelsang, 2020).

To ensure consistency and avoid ambiguity, this thesis applies a consistent "SW component" prefix to represent all abstraction levels as described by ISO/IEC/IEEE 12207 (2017). Accordingly, the SW hierarchy is represented as **SW system → SW elements → SW components → SW units**. The SW system refers to the complete SW system, SW elements refer to high-level parts of the SW system, SW components refer to architectural or design-level parts, and SW units refer to the lowest-level implementable and testable parts (ISO/IEC/IEEE 12207, 2017).



r

Figure 2.2 Software Development Lifecycle

At the early phase of SW development, typically referred to as requirements stage, customer and other stakeholders' needs are translated into SW technical requirements (ISO/IEC/IEEE 12207:2017). Abrahão et al. (2024) explains the importance of requirements, as an enabler in understanding stakeholders' needs and ensuring quality

attributes. Good SW requirements are expected to be correct, clear, feasible, testable, and traceable (ISO/IEC/IEEE 12207:2017). Palomares et al. (2021) identify two predominant challenges in requirements elicitation; (1) understanding and prioritizing the needs of diverse stakeholders, (2) requirement instability, referring to changes arising from evolving needs and shifting priorities. Atoum et al. (2021) emphasizes the importance of analyzing and validating requirements, as initial requirements are often written down in natural language. Tan et al. (2017) highlights that while not necessarily in the context of SW development, changes to requirements become increasingly costly when performed at later stages of product development. However, the SW context is emphasized, where late bug detection inherent bigger associated costs (Tan et al., 2017). Previous research has identified requirement development as a source of SW failures because of "inconsistent" or "hidden" requirements (Atoum et al., 2021).

Habibullah et al. (2024) highlight that requirement specifications are a significant source of quality risk in complex and safety-critical contexts such as automotive. This is particularly important because downstream verification and assurance activities depend on clear and traceable links between requirements, design, implementation, and tests (Habibullah et al., 2024; Maro et al., 2018). This is further emphasized by Abrahão et al. (2024), who highlight the importance of testing in assessing whether requirements and quality standards are met. While requirements and testing are important, organizations have a hard time realizing this in practice, especially in the context of complex SW systems (Abrahão et al. 2024).

The Architecture and design stage defines the SW structure, interfaces, technical constraints, and SW component responsibilities through which SW requirements will be implemented (ISO/IEC/IEEE 12207:2017). Vogelsang (2020) therefore explains good SW architecture is not only a translation of requirements into SW components and interfaces; it also needs to manage dependencies and balance important system qualities such as modifiability, maintainability, and complexity in ways that support the evolution of automotive SW systems. Furthermore, dependencies possess challenges and must be considered on system level rather than only SW component level, as otherwise the full dependency and interaction within the whole system is not understood (Vogelsang, 2020)

During the implementation, the focus shifts to realizing the requirements, architecture, and design by developing SW components that fulfil part or all the customer needs (ISO/IEC/IEEE 12207:2017). The implementation is carried out through appropriate technical disciplines, specialties, and programming languages. (ISO/IEC/IEEE 12207, 2017) also highlights that these realized SW components which form part of the larger functionality are integrated together according to the architecture design to fulfil the requirements accordingly. During this stage, test activities are also carried out to identify and fix anomalies in the implemented SW component before deployment (ISO/IEC/IEEE 12207, 2017).

The SW development lifecycle combines various roles and responsibilities to achieve the right outcomes of functionality and quality (ISO/IEC/IEEE 12207, 2017). In practice, SW teams include roles that cover requirements, architecture, implementation, integration, testing, project coordination, and quality assurance responsibilities (ISO/IEC/IEEE 12207, 2017). In safety and regulated industries such as automotive, safety and cybersecurity expertise are typically included (Li et al., 2024)

2.3.1 Software Development Life Cycle (SDLC) Models – Waterfall, Spiral, V-model, etc.

SDLC models help structure and organize these SW development lifecycle stages depending on context-specific factors such as requirement stability, technological risks, and regulatory constraints (ISO/IEC/IEEE 12207, 2017; Pavlíčková et al., 2024). Waterfall, Spiral, Iterative, RAD, Z, AZ, and V-model are among the popular model choices over the years (Alam et al., 2022; Pavlíčková et al., 2024).

The V-model is designed to support quality-focused SW development because it pairs development activities with corresponding verification and validation activities (Pavlíčková et al., 2024). The V-model is depicted in *Figure 2.3*, SW requirements are linked to SW acceptance or qualification testing, architectural design to SW integration testing, and implementation to unit testing (Pavlíčková et al., 2024). These direct connections between SW development activities, testing and verification supports traceability and consistency which makes the V-model attractive to regulated and safety-critical contexts (Pavlíčková et al., 2024; Maro et al., 2018).

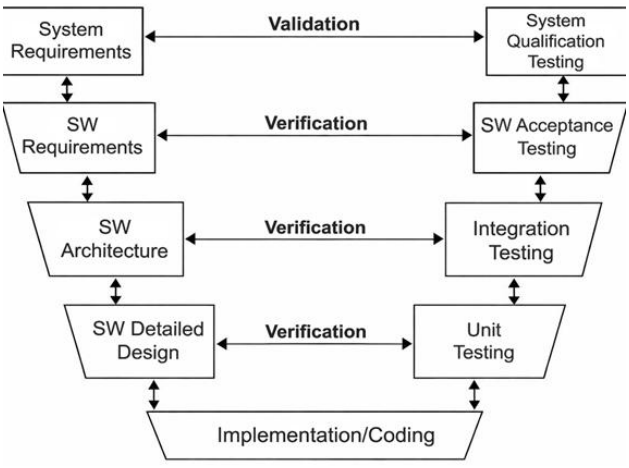


Figure 2.3 V-Model (Adapted from Pavlíčková et al., 2024)

The V-model is relevant in automotive SW development because while the model also supports iterations and adaptation, it focuses on quality and auditable evidence across every stage of the development which is important as it otherwise may lead to

bad consequences (Maro et al., 2018). (Trovão, 2024) strengthens this by stating that bad SW quality can result in consequences such as recalls, reputational damage or safety incidents.

According to Pavlíčková et al. (2024), the evolution of SDLC models reflects the increasing complexity of SW development projects requiring more structured planning, communication, and quality processes. Supporting this understanding, Lee et al. (2021) stated that the choice of SW development processes should be based on the dynamic characteristics of the project including scale, complexity, and delivery conditions. Also, (Pavlíčková et al., 2024) show in their study that models can be used in hybrid forms to improve quality outcomes in regulated environments. Together, these studies highlight that the choice and effectiveness of a SDLC model depends on the factors discussed above with the inclusion of coherence between assumptions and practices.

2.3.2 Software quality

Green et al. (2024) highlight that SW quality is hard to define and may be interpreted differently across roles. For example, managers may view it from a customer perspective, while SW developers may associate it with code quality. Nabot and Al-Qerem (2025) define SW quality as a system's ability to operate as intended and meet customer expectations, reflected in attributes such as usability, reliability, performance, and robustness. ISO 25010 (2023) defines reliability as a SW product's ability to operate under specified conditions for a defined period without failure. Performance refers to its ability to fulfil intended functions within a given time frame and with efficient resource usage. Usability concerns the extent to which users can interact with a product effectively, efficiently, and with satisfaction (ISO 25010, 2023). Robustness is a system's ability to operate correctly under unexpected or stressful conditions, making it important for achieving high system dependability (Shah et al., 2016).

Green et al. (2024) highlight that, due to the challenges of defining SW quality and the differing perceptions of what it entails, measuring SW quality is also difficult, as there is unlikely to be consensus on how it should be measured or improved. They identify four components that affect SW quality: *process quality, code quality, system quality,* and *product quality*, as well as the relationship between these four aspects (Green et al., 2024). These four components and their relationship are depicted in *Figure 2.4*.

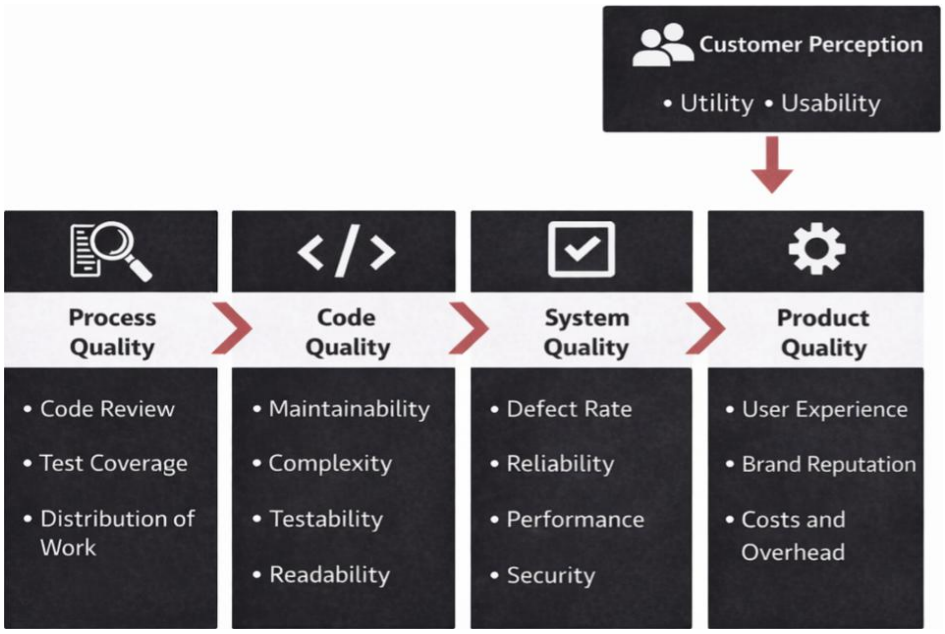


Figure 2.4 Components impacting software quality

Green et al. (2024) highlight that process quality acts as the foundation in achieving good quality outcomes; activities may include established code reviews, comprehensive planning, and organizational consistency. ISO/IEC/IEEE 12207 (2017) further expands the topic by emphasizing consistency and traceability between development stages, reviews, streamlined verification activities being necessary for sustainable code, system and product quality. Good process quality impacts code quality, and there is also an indication that measuring the processes helps in predicting potential product quality issues compared to code quality metrics (Green et al. 2024). This is strengthened by Majumder et al. (2022)'s large-scale analytic study using 722,471 commits, concluding that process metrics are better at predicting defects than product (code) metrics.

Börstler et al. (2023) explain that code quality is a key component of SW development and that several guidelines and frameworks exist for writing "good code." They identify readability, structure, comprehensibility, and documentation as the four most common terms used to define code quality (Börstler et al., 2023). Green et al. (2024) likewise highlight maintainability as a central aspect of code quality, with testability, comprehensibility, complexity, and readability contributing to maintainable code. ISO 25010 (2023) describes maintainability in terms of how effectively and efficiently SW can be modified. Readable and maintainable code enables developers who did not originally write it to understand it and make changes, thereby improving development efficiency and product quality (Green et al., 2024).

Green et al. (2024). Explains that system quality considers both business and operational (development) perspectives. There may be a disconnect between developers and managers in how SW quality is perceived. Developers tend to focus

on process efficiency and code quality, while managers often emphasize overall product quality. As a result, improvement requests for the SW may be interpreted differently. For example, managers may request enhancements to improve product quality and customer satisfaction, whereas developers may interpret these requests primarily from a code- and implementation-focused perspective (Green et al., 2024).

System quality refers to SW quality attributes such as reliability, performance, security, and low defect rates. However, due to the limited availability of data on system-level quality, process quality and code quality can be used as proxies for assessing overall system quality (Green et al., 2024). Similarly, Izurieta et al. (2024) highlight measurable proxies such as cyclomatic complexity for maintainability or failure rate for reliability, helping translate abstract quality attributes into actionable insights. According to Green et al. (2024), using such indicators enables management to make informed corrections to development processes, supporting developers' efficiency and overall SW quality. While system quality is important, impacting user experience, brand reputation and operational cost, it is not guarantee of product quality, as customer perceptions of usability and utility also have an impact (Green et al., 2024).

2.4 Software quality management

Software quality management (SQM) refers to the coordinated organizational, managerial, and technical activities that ensure SW quality is defined, governed, and improved throughout the entire SDLC instead of only testing at the end stages (ISO/IEC/IEEE 12207, 2017). ISO/IEC/IEEE 12207, (2017) presented quality management processes and practices including quality assurance, quality control, verification, validation, configuration management, problem management, reviews, gates, audits among others as SDLC integral processes and practices.

In addition, since SW quality management depends on the coherence between the development processes, process fragmentation could become a major quality risk (Giachetti et al., 2024; Moe et al., 2021). Giachetti et al. (2024) highlight that successful adoption of quality standards and processes is often complex, and requires knowledge that is often missing, poorly documented, and unevenly distributed. Formal quality processes and expectations may exist at high levels of governance, but if operational practices and processes remain fragmented across projects, teams, and roles, it may cause inconsistent quality outcomes (Moe et al., 2021).

Moe et al. (2021) further highlight that in large-scale development organizations, team autonomy without sufficient organizational alignment and process governance may result in uneven process control and additional coordination efforts. Bjarnason et al. (2022) explains SW development as unpredictable, where unexpected issues can surface during various stages of the development, which must be delt with as they occur instead of awaiting management. Especially in circumstances where teams have autonomy over their working practices. Thereby collaboration and communication

between teams comes of importance through both formal and informal channels (Bjarnason et al., 2022).

Boiral (2007) provided a complementary organizational explanation for why this process fragmentation exists even when there are formal and centralized processes. They highlight that employees often have only a "*vague understanding"* of the standards and central processes and are "*seldom involved"* in their implementation, giving a feeling the processes are not theirs, hence contributing to the gap between formal routines and daily practice (Boiral, 2007).

Furthermore, effective SW quality management is also closed tied to planning as empirical research shows that time pressure due to poor scheduling practices systematically shapes developer behavior and quality outcomes (Kuutila et al., 2020). While they argue that time pressure can have positive effects such as increased efficiency in the short term, the negative effects are often highlighted as long-term time pressure reduces quality of outcomes and reduces time for process and SW product improvements (Kuutila et al., 2020). Maintaining balance and applying the right management approach could yield preferred results; however, commercial pressure and operational realities continue to prove difficult (Kuutila et al., 2020).

Therefore, SQM is understood as a governance and coordination capability rather than only a technical one. Beyond only downstream verification, it requires aligning SW development processes and practices to quality standards including organizational and administrative aspects such as planning, governance, and coordination.

2.4.1 Software Quality Assurance (SQA)

Software Quality Assurance (SQA) can be understood as the process-centered and proactive arm of SQM; it is concerned with meeting defined SW quality expectations through streamlined development processes with adequate organizational control, resulting in quality SW outcomes (ISO/IEC/IEEE 12207, 2017). Practically, this includes process definition, standard adoption, reviews, configuration discipline, training, defining quality decision gates and risk criteria, to detect risks early before they become downstream failures (Giachetti et al., 2024; ISO/IEC/IEEE 12207, 2017).

Reviews are understood to be an important SQA practice because they contribute expert judgement to the verification of SW work products (McIntosh et al., 2016). They further highlight that a stronger review coverage and participation have been proven to increase the likelihood of better SW quality outcomes (McIntosh et al., 2016). However, as highlighted by Kudrjavets & Rastogi (2024), the value of review-based SQA depends on responsiveness, since modern code reviews are seen to be time-consuming, and faster response times as well as time-to-merge are considered highly important to SW quality. Therefore, automated gates as presented by Wessel et al. (2022) can provide a complementary control layer in SW development as they

enforce clear quality criteria, making progression decisions faster and more transparent, even though it limits the collaboration reviews provide.

Zhao et al. (2025) Highlight potential benefits of a robust SQA as well, allowing for identification of issues early in the SW development lifecycle rather than at later stages. Being beneficial also from a financial perspective as if firm can rectify the issues earlier, the lower the associated cost is versus later fixes in the SW product lifecycle or even post-release fixes (Zhao et al., 2025).

Furthermore, the importance of SQA becomes even more critical in large-scale development organizations due to team autonomy, which may lead to different interpretations of quality standards, criteria, and processes, thus creating inconsistent compliance and outcomes (Giachetti et al., 2024; Moe et al., 2021). Therefore, SQA in organizations should involve more than just checking if the processes exist; it should also ensure that the process knowledge is shared across all levels, interpretable and aligning across all teams (Giachetti et al., 2024; Moe et al., 2021).

2.4.2 Software Quality Control (SQC)

Software Quality Control (SQC) also exists within SQM, it can be understood as operational and evidence-based techniques through which the SW development process outcomes are checked to determine whether they conform to requirements, standards and quality expectations (Guamán et al., 2020; ISO/IEC/IEEE 12207, 2017). SQC is product-oriented and outcome-oriented, involving objective identification of errors, defects, and faults that may affect SW quality at key points of SDLC (Guamán et al., 2020; ISO/IEC/IEEE 12207, 2017).

ISO/IEC/IEEE 12207, (2017) describes verification activities as a key contributor to SQC as it ensures "*the product is built right*" across the SDLC. This suggests that the implemented SW and associated work products accurately reflect the specified requirements, and they are tested for compliance (ISO/IEC/IEEE 12207, 2017). The standard further specifies that the selection of verification methods depends on the type of SW developed, its objectives, and acceptable risks. The methods include, but not limited to code review, simulation, static and dynamic testing, and they should be coordinated with the relevant stakeholders to ensure they are acceptable to purpose (ISO/IEC/IEEE 12207, 2017).

Additionally, validation is also presented by ISO/IEC/IEEE 12207, (2017) as a key contributor to SQC as it ensures "*the right product is built"*. It provides objective evidence that the SW released satisfies the business and stakeholder requirements when used in its intended operating conditions (ISO/IEC/IEEE 12207, 2017). Similarly, the validation methods depend on the type of SW developed, objectives, acceptable risks, with consideration to legal, regulatory and usability. The methods include, but not limited to demonstration, code reviews, beta testing, operation testing, usability testing, and acceptance testing (ISO/IEC/IEEE 12207, 2017). Zhao et al.

(2025) reflecting on testing, highlights that development methodologies being faster and more iterative, including continuous integration and complex testing, possess a challenge for organizations. Therefore, organizations need to adapt towards automated testing to manage this faster and more iterative approach, as manual testing risk not detecting flaws potentially releasing SW with bugs (Zhao et al., 2025).

In conclusion, SQC complements SQA in delivering a robust SQM. It should not be understood as only "bug-detection" downstream or a final checkpoint of SW quality, but rather as a continuous, integrated function that provides objective evidence throughout the SDLC that the SW development processes are delivering the right outcomes (Guamán et al., 2020; ISO/IEC/IEEE 12207, 2017).

2.5 Automotive Software Quality Management

Automotive Software Quality Management (ASQM) is quality management focused on the automotive industry with high product complexity and even higher safety expectations. As vehicles and safety features become more SW defined, quality failures can have serious consequences, so SW in vehicles must be safe, secure, traceable, and maintainable throughout the SDLC and while in use (Li et al., 2024). This explains why automotive SW engineering cannot be confined to end-stage checks; rather, it must be continuously managed throughout the development, verification, operation, and maintenance (Ferreira et al., 2024). Habibullah et al. (2024) also highlights that automotive system development requires quality management practices that cover not only usual SW artifacts or work products, but also scenarios, operational design, realistic simulations, data quality and statistical performance indicators.

This increase of SW in vehicles comes with increased complexity through the increased connected systems, hence automotive organizations must meet multiple quality obligations, standards, and regulatory frameworks (Li et al., 2024). These quality obligations can be understood as addressing automotive SW quality in three dimensions. The first dimension is product quality – the released SW must satisfy the functional, robustness, safety, and cybersecurity requirements (ISO 26262:2018; ISO/SAE 21434:2021). The second is process or operational quality – the SW development activities must be organized systematically, traceable, assessable, and continually improved (VDA QMC, 2023). The third is post-release quality – incidents, changes and updates to the released SW must be controlled while the vehicle is already in operation (UNECE Regulation No. 156, 2021). Therefore, these quality obligations, standards and frameworks do not only satisfy compliance requirements; they also provide the necessary governance and structure for automotive SW quality. (ISO 26262:2018; ISO/SAE 21434:2021; UNECE Regulation No. 156, 2021, VDA QMC, 2023).

2.5.1 Regulatory and standard frameworks

Regulatory and standard-based frameworks are central to ASQM considering the safety implications of automotive SW. These frameworks include ISO 26262 for functional safety, ISO/SAE 21434 for cybersecurity engineering, and UNECE R156 for SW update governance through a Software Update Management System (SUMS). Combined, these frameworks ensure automotive SW quality is maintained and justified across the lifecycle through documented, traceable processes, and auditable evidence (ISO 26262:2018; ISO/SAE 21434:2021; UNECE R156, 2021). Li et al. (2024) support this by highlighting that these obligations interact within same SW systems and should be treated with a common focus to increase consistency, traceability, and compliance through SW development, release, and operations (Li et al., 2024).

2.5.1.1 ISO 26262

ISO 26262:2018, *Road vehicles — Functional safety*, provides the international framework for the management of functional safety of road vehicle electrical/electronic systems and structures safety activities across the lifecycle. It also supports concept-phase hazard and risk assessment, hardware and SW level product development, configuration management, and change control (ISO 26262:2018).

This is particularly important in this automotive context because it defines quality as more than low defects rates but also as the reduction of risks that come from malfunctioning electrical or electronic systems (ISO 26262:2018). This suggests that safety is integrated into ASQM as SW requirement, SW architecture, SW implementation, verification and validation outcomes must meet safety requirements (ISO 26262:2018). ISO 26262 uses the Automotive Safety Integrity Level (ASIL) to classify risks related to safety-relevant functions according to severity, exposure, and controllability. ASIL has levels from A to D where ASIL D represents the highest safety and integrity requirements (ISO 26262:2018). This is highly important for ASQM because the ASIL of SW functions determines the expected level of diligence in the development and quality assurance activities, higher ASIL means higher expectations for requirements, traceability, consistency, verification depth and evidence or work products quality (ISO 26262:2018; Li et al., 2024).

2.5.1.2 ISO 21434

ISO/SAE 21434:2021, *Road vehicles — Cybersecurity engineering*, extends automotive SW quality by bringing the cybersecurity perspective that defines the standards for managing cybersecurity risks to road vehicle systems throughout development, production and post-production (ISO/SAE 21434:2021). This is important because as vehicles become more SW-defined and connected, they become exposed to attacks that could affect vehicle behavior, safety and data integrity (ISO/SAE 21434:2021; Li et al., 2024). Li et al. (2024) strengthens this by

highlighting that the increasing connectivity, external interfaces, and over-the-air (OTA) functionality of vehicles increases the possibilities for attacks, hence, it is necessary to include standalone requirements for cybersecurity (Li et al., 2024).

While there is justification for separate requirements for functional safety and cybersecurity, Li et al. (2024) highlight limited information sharing, duplication, inconsistency, and limited understanding of system interactions as consequences of fragmented handling these requirements (Li et al., 2024). For ASQM, this suggests that functional safety and cybersecurity must be managed from a common perspective within the SDLC, strengthening the need for traceability between hazards, risks, requirements, architecture interfaces, verification results and evidence or work products quality (Li et al., 2024).

2.5.1.3 SUMS

The United Nations Economic Commission for Europe Regulation No. 156 (UNECE R156, 2021) extends the quality obligations with the SW updates management perspective that mandate automotive manufacturers to maintain a SW Update Management System (SUMS). The UNECE R156 requires manufacturers to maintain documented and assessable processes for identifying SW versions, tracking updates, assessing compatibility and how an update affects type-approved systems, protecting update authenticity and integrity, and informing users about update execution and its outcomes (UNECE R156, 2021). For OTA updates, the regulation also requires safe execution conditions, sufficient power, rollback capability after interrupted or failed updates, and clear user information before and after the updates (UNECE R156, 2021).

For ASQM, this means that SUMS extends traceability, consistency, change management, and release governance beyond the initial SW development to post-production operations (Li et al., 2024). Manufacturers are expected to maintain adequate records of SW updates and related requirements including SW versions, verification and validation data, target vehicles, and compatibility confirmation (UNECE R156, 2021). In practice, this suggests that automotive SW quality must be preserved accordingly due to evolution of technology, user needs or in-service issues beyond the original SW development, and this must be evidenced to approval authorities (UNECE R156, 2021; Li et al., 2024).

2.5.2 Industry frameworks influencing quality assurance - ASPICE

Automotive Software Process Improvement and Capability determination (ASPICE) is described by the VDA Quality Management Center (VDA QMC) as a process reference and assessment model for process capability, covering system and SW engineering, validation, support, management, reuse and process improvement (VDA QMC, 2023). ASPICE is also central to SW supplier governance, it is internationally recognized in the automotive industry because it provides a structured assessment of

process capability and a quality indicator which OEMs often rely on when choosing SW suppliers (VDA QMC, 2023)

While ASPICE does not define SW product quality itself, it is important to ASQM because it evaluates the quality of the development process that creates the product. It provides reference to determine if the processes are systematic, repeatable, and capable of creating consistent and reliable outcomes (VDA QMC, 2023). This process orientation of ASPICE is important to ASQM because automotive SW is developed under increasing conditions of technical complexity and SW is crucial for safety, efficiency and quality of modern vehicles (VDA QMC, 2023). Accordingly, ASPICE influences organizational SQA, ensuring process capability, systematic process monitoring, and continuous improvement are visible and assessable (VDA QMC, 2023).

Furthermore, automotive SW development involves many teams distributed across various interrelated SW functions; hence, traceability is crucial to maintaining coherence between SW requirements, architecture, implementation and tests (Maro et al., 2018). Therefore, ASPICE supports SQM by helping organizations operationalize quality through disciplined SDLC consistency, traceability and evidence preservation (VDA QMC, 2023; Maro et al., 2018).

Furthermore, Maro et al. (2018) highlight that the requirement for traceability in automotive is influenced by ASPICE. They discuss that ASPICE requires both bidirectional traceability between SW development stages and their related verification work-products (Maro et al., 2018). ASPICE remains a dominant automotive industry framework influencing ASQM because it supports process quality, making it assessable, consistent and reliable across projects and suppliers (VDA QMC, 2023; Maro et al., 2018).

3 Method

This chapter covers the methodology chosen for this research to fulfill its purpose and answer the research questions identified. The chapter includes an overview of the research strategy and design, followed by data-collection methods and analysis. Finally, it elaborates on the research quality, reliability, and ethical considerations made during the study.

3.1 Research Strategy

To coherently connect the research questions with the data collection and analysis, a qualitative research strategy was chosen as identified by Bell et al. (2022). This approach is suitable because qualitative research emphasizes the nuanced ways individuals interpret their environment which aligns with the exploratory nature of this study (Bell et al., 2022). This approach is particularly important for the purpose of this study, which is to understand automotive SW quality management and its related activities within the case company. It involves a lot of contact with relevant stakeholders through interviews and observations allowing for depth of understanding of relevant nuances within this context.

Given the exploratory nature of the study and the limited prior knowledge in this area, an abductive research approach is also employed. This allows the study to move iteratively through theoretical concepts and empirical findings, supporting potential theoretical insights, rather than only validating initial theories as a purely deductive study might (Timmermans & Tavory, 2012). During the study, the initial understanding of automotive SW quality management and related processes within the case company was provided by the project sponsors. Followed by literature reviews, then comparing empirical data from qualitative interviews and observations enabled investigation of additional findings rather than only validating theories.

3.2 Research Design

A single-case study design was selected for this research. A case study is highly effective when the research focuses on understanding a phenomenon in its real-life context like one organization and/or one location (Bell et al., 2022). Considering the purpose of the study, this design is suitable as the subjects of ‘Quality’, ‘Quality Management’, ‘Software Quality’, and ‘Software Quality Management’ have different understandings depending on the context. The design choice supports conducting this study within this specific automotive SW development context, as in-depth assessments can be made on how global standards and processes are translated into local ways of working. Refer to *Figure 3.1*, the design choice also includes triangulation of empirical data to increase understanding as described by Bell et al. (2022). Findings from interviews, observations, and documentation reviews and combined and cross-verified to reduce individual bias in the results.

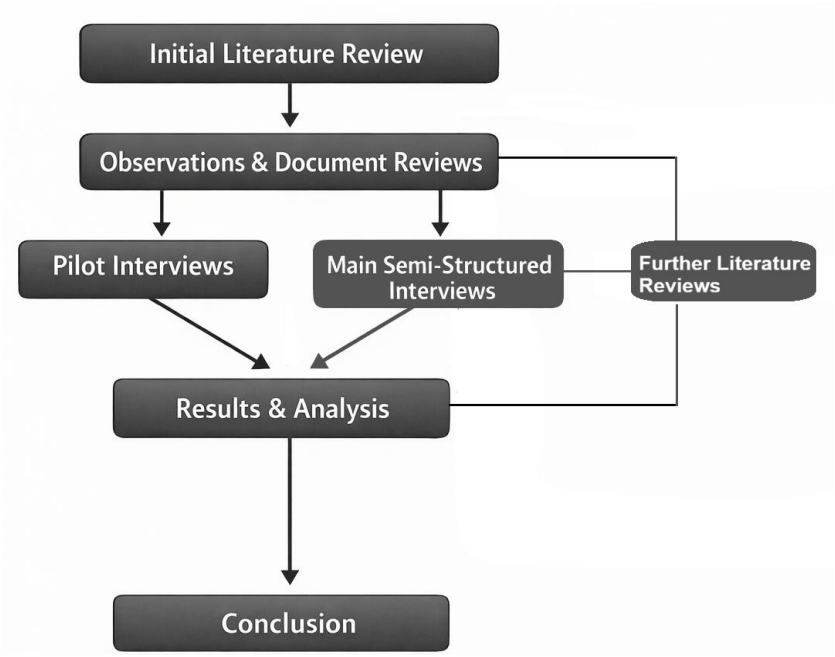


Figure 3.1 Research design

3.3 Sampling

The research participants were selected using a combination of purposive and snowball sampling techniques. Purposive sampling technique is used to target specific roles and participants relevant to the research purpose within the context to capture diverse perspectives (Bell et al., 2022). The participants chosen for this study included people from both leadership and operational roles who are actively involved in the SW development life cycle and support processes. The sample included leadership roles such as Engineering Managers (EM), Quality Managers (QM) and Product Owners (PO), as well as operational roles like Requirements Developers, Architecture Developers, and SW Developers. This categorization was necessary to understand how the phenomenon is interpreted across different organizational layers. Additionally, snowball sampling was used by asking participants to suggest colleagues with specific expertise in certain key areas to be interviewed who might have otherwise been unreachable (Bell et al., 2022).

3.4 Data Collection

The empirical data for this study was gathered through the combination of four channels which included literature reviews, Semi-structured interviews, documentation reviews, and observations, refer to *Figure 3.1*. These channels were chosen to synthesize theory with practical experiences to ensure a comprehensive understanding of SQM within the context.

3.4.1 Literature Review

A systematic review of existing literature was done to establish a theoretical foundation for the research within this automotive SW quality context. According to Bell et al. (2022), a literature review is essential in qualitative research to identify and understand existing knowledge that guides the initial data collection. The initial literature review process of this study is shown in *Figure 3.2.* The review was performed using web of science databases by searching for the following keywords; “automotive software", "vehicle software", "embedded automotive systems", "software-defined vehicles", "automotive architecture", "software updates" OR "ISO 26262", "ISO/SAE 21434". This initial search resulted in 2,750 articles; this was further screened by filtering out papers that were not journal articles in journals with impact factor less than 1, and articles older than 2016 resulting in 711 articles. A second screening was carried out, excluding non-relevant research categories, using the following web of science research categories; “engineering electrical electronic”, “computer science information system” and “computer science software engineering”, resulting in 389 articles. A title and abstract screening were carried out by the authors resulting in 81 articles, and 50 were reviewed fully, resulting in 8 articles being included as the initial literature foundation of this study.

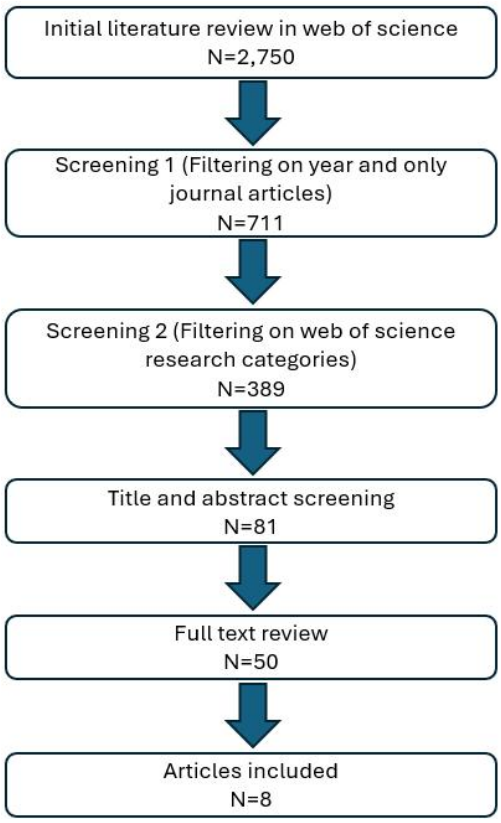


Figure 3.2 The literature review process

This initial review focused on relevant concepts providing the foundations to build the research. However, as the research strategy is abductive, allowing the authors to move between literature and data collection, the initial literature review was not sufficient. Thereby complementary literature reviews were carried out as new insights emerged during the study. Additional articles were found using snowballing of initial articles reference lists and further web of science searches. These additional articles fulfilled similar criteria, such as journal articles from journals with an impact factor greater than one. In total, 48 sources were included in the theory and introduction, consisting of 40 academic articles and 8 standards, regulations, or automotive industry frameworks.

3.4.2 Interviews

Semi-structured interviews were the main source of empirical data. This interview format uses a guiding question framework to get relevant and detailed responses while flexible to explore other interesting topics and perspectives that may emerge (Bell et al., 2022). Participants had the opportunity to share rich and detailed insights which were useful to the research. The guiding questions for the interviews were developed with certain pre-defined topics and targeted roles combined into six separate questionnaires, see section 8 Appendices. A large effort was put into the structure of these questionnaires to ensure a logical structure that made spontaneous follow-up questions and discussions possible.

Bell et al. (2022) emphasize the importance of rigorous data collection and analysis procedures in qualitative research. Therefore, all interviews were conducted via Teams and were recorded and transcribed, after getting consent from the participants, to allow for in-depth analysis. These transcripts were reviewed right after the interviews to ensure correctness before storage. Six pilot interviews were conducted with the project sponsors to test the questionnaires and get feedback for possible improvements. Afterwards, 28 main interviews, all between 30-60 minutes, were conducted with interviewees selected based on the relevance of their roles to the research purpose as described in the sampling section above. *Table 3.1* presents an overview of the interviews conducted, including the interview type, date, and duration of the interviews.
These interviews covered topics such as the respondents' understanding of ASQM and related processes including assurance, control, and verification.

Table 3.1 Interview Overview

Interview type	Role	#	Duration	Date
Pilot	Engineering manager	1	30-60 Mins	Mar 2026
	Product Owner	1		
	Quality Manager	1		
	Requirement Developer	1		
	Software architect	1		
	Software Developer	1		
Main	Engineering manager	7	30-60 Mins	Mar-Apr 2026
	Product Owner	4		
	Quality Manager	6		
	Requirement Developer	2		
	Software architect	4		
	Software Developer	5		

3.4.3 Documentation Reviews and Observations.

In addition to the interviews, additional data was collected through observations and review of relevant documentation. As described by Bell et al. (2022), observations enable a deeper understanding of participants' behavior within their context, without the spotlight of an interview. Therefore, observations were conducted by attending various quality-themed meetings, team demo meetings, and by working on-site at the case company.

Furthermore, an iterative review of internal and external documents relevant to the study was conducted. These documents included; case company ways of working and processes on official sites, relevant to the industry such as ISO 26262 and ISO 21434, and assessment frameworks like ASPICE. As highlighted by Bowen (2009), these document reviews and analyses were used to provide a basis for comparing formal standards with actual practices to evaluate the case company against industry expectations.

3.5 Data Analysis

According to Bell et al. (2022), one of the challenges associated with qualitative research is the large volume and depth of the data collected, which can make it difficult to organize and analyze. To handle this, a thematic analysis approach was applied which follows the guidance of Bell et al. (2022), with the purpose of systematically identifying recurring patterns, categories, and themes within the dataset. In this study, key interview answers were compared with each other as well as observations and documentation reviews to detect potential similarities. Based on the

similarities in the data collected, the most common and relevant themes and codes were identified.

3.6 Research Quality

In qualitative research, trustworthiness is a key aspect for evaluating the quality of a study. According to Bell et al. (2022), trustworthiness consists of four core dimensions: credibility, transferability, dependability, and confirmability.

Credibility refers to the accuracy of the findings as a representation of the participants' experiences (Bell et al., 2022). To improve credibility, a wide range of participants in various relevant roles were interviewed to capture diverse perspectives on the research topic. The interviews were also conducted anonymously to encourage openness and to avoid criticism or blame after the study is published. The use of voice and video recording during the interviews supported the preservation of all details of what each participant has expressed while providing opportunity for several listens during analysis.

Transferability represents the extent to which the findings of the study can be applied to other contexts (Bell et al., 2022). Given the study's focus on a specific organization and non-disclosure agreements preventing sharing of specific details, it may be difficult to apply in another context or setting. However, efforts were made to improve the transferability by giving detailed descriptions about the conditions and organizational environment.

Dependability emphasizes the importance of transparency and consistency in the research process (Bell et al., 2022). To meet this, all steps of the study including; research planning, participant selection, questionnaire creation, interview transcripts, observation notes, documentation review notes, and data analysis were documented in a shared folder.

Confirmability ensures that the findings are shaped by the input of the participants, and not the researchers' personal biases (Bell et al., 2022). To reduce the risk of this, all major discussions and decisions were made by the researchers, and regular meetings were held with both the research supervisor, and the project sponsors to maintain objectivity and alignment with the research purpose. Furthermore, the researchers ensured interview transcriptions represented the right contents as in the voice and video recordings.

3.7 Ethical Considerations

Ethical aspects are important in research to reduce potential harm to participants and ensure the integrity of the study. Bell et al. (2022) highlight four core principles to guide ethical research: avoiding harm, informed consent, respecting privacy, and preventing deception. Following these principles, all participants involved in this study were treated respectfully, and confidentiality was emphasized and upheld. It was important to ensure that all participants are not psychologically or professionally harmed while participating in the study.

Deception was also avoided by clearly describing the purpose of the study, the topics to be discussed, and how the information would be used. Participants were also informed in advance about the use of voice and video recordings, which were only initiated after explicit consent had been given and documented. Transparency was maintained at every stage to avoid deception, including clarifying the researchers' roles and the intended use of the collected data.

The researchers worked with only the computers and technical tools provided by the case company. Also, all communications were made through only the case company’s official channels. These precautions ensured data security and created a confidential environment during the study.

3.8 Use of Generative AI

Artificial Intelligence (AI) was used only as a support tool throughout the study. Microsoft Teams’ built-in AI functionality was used to transcribe recorded interviews, followed by manual verification. In addition, AI was used as language and grammar aids to improve readability and did not contribute to the generation of original content or analytical reasoning. All conclusions, interpretations, and formulations are the result of the authors’ own work.

4 Results

This chapter outlines the findings from observations, case company internal documentation, and semi-structured interviews. The chapter is structured according to the five themes identified and further analyzed by subthemes.

4.1 Software quality definition and perception

Across the interviews, various perspectives on how to define SW quality have been expressed. Quotes below illustrate this, when asked "how they would define SW quality".

> *"[…] That's a good question, I think the answer to this is that we don't have a definition for it […] if you ask anyone in the team, I don't think you will get the same answer from anyone." (Interviewee 8).*

> *"Big, big topic to define […] how good is the quality of what we are delivering out […] when you look at software you have different things of looking at quality when it comes to does it fulfill what was set out to do, does it do what the requirement specifies […] that's one quality […] then you have other quality aspects." (Interviewee 15).*

> *"Software quality yeah there are many aspects to it, of course it's about fulfilling the requirements […] you need to always think about robustness and corner cases […] what should be the safe state of the software? All those things are also quality." (Interviewee 27).*

> *"I think of quality from a vehicle perspective and customer point of view, if they can use the product as without anything unintended happening." (interviewee 17).*

> *"Software quality to me would be to deliver software product that are meeting your own standards that are expected by the customer and external stakeholders." (interviewee 13).*

The quotes above indicate that SW quality is difficult to define and different perspectives of it. While some interviewees emphasize the lack of a shared definition, others describe SW quality as consisting of multiple aspects, and several highlight an external and customer-oriented perspective, that are representative across the interviewees. Based on these findings, SW quality is structured into two subthemes: (1) technical aspects, (2) functionality and customer experience.

4.1.1 Technical

Some interviewees emphasized technical aspects in relation to SW quality, such as requirements, architecture, traceability and code quality.

The requirements and their impact on SW quality were highlighted across interviews, illustrated by the below quotes.

> *"That the requirements are well written, so that you easily can test them". (Interviewee 1)*

> *"Software quality for me […] to verify the requirements [...] and to verify requirements at different levels [...]. The process of defining the requirements at different levels and the traceability and consistency [...] and also to make sure that you have enough test cases. [...] and that you have evidence that you fulfilled all requirements." (Interviewee 7).*

> *"[...] to ensure quality you need to have good test coverage. Because the quality means, at least to me, that the software works as expected according to some requirements, to be able to tell if a piece of software behaving correctly and to see if it has the correct quality you need to assess that via some testing" (Interviewee 14).*

The interviewees highlighted the importance of requirements in relation to how SW should operate, with aspects regarding testability and traceability emphasized. Several interviewees highlighted aspects regarding traceability between requirements and other SW development stages such as architecture and design, as illustrated by below quotes.

> *"So one of the big ones is making sure that all the documentation is in place and all the traceability. So we're talking about maybe the architecture design documentation, we're talking about maybe mapping requirements and test cases […]" (Interviewee 10).*

> *"[…] So they are starting from the beginning, from the design, from the architecture and then connect all the bits and pieces together until the release and the testing and verification […]" (Interviewee 6).*

> *"[…] we know already from an architectural level all the way down to components, what requirements are there. We expect to see those requirements documented, accepted, implemented, verified and validated. We expect to see those evidence" (Interviewee 5).*

The interviewees highlight the importance of traceability from the requirements throughout the SW development stages. However, another interviewee, 25, a SW architecture highlighted challenges in achieving traceability and consistency but nonetheless highlighting their importance to ensure SW quality.

A further perspective is requirements related to safety- critical functionalities, as illustrated by the quote below:

> *"it's safety critical, we have different ASIL requirements that we need to fulfill. [...] it should fulfill all requirements."* (Interviewee 16)

While this perspective was not evident across all interviews, it emerged in interviews where safety- critical aspects impact the SW development.

Two requirement developers, when asked how they would define high quality SW requirements, expressed the following thoughts.

> *" […] they need to be both crisp and super clear on what the function should do […]. The most important thing with the requirements is that it's possible to verify them." (interviewee 27).*

> *"High quality requirement is, uh easily understandable. Preferably also easily testable. […] If you understand what to do and how to test it. Then that's what is really good requirements." (interviewee 17).*

Both interviewees highlighted that requirements need to be clear and understandable, as well as possible to test or verify. Similar to the initial quotes from Interviewee 1, 7 and 14, they highlight testing in relation to the requirements as an important aspect.

SW developers described a perspective that was primarily raised by this role: code design and its impact on SW quality, as illustrated below:

> *"[…] quality term is broader as well because from my perspective as a software developer, I also believe that the code quality and the design quality itself matter as well. So, it's not only about if the code works […], but also the quality lies in how code is structured […] to make it maintainable and so on." (Interviewee 14).*

> *"I would just say that software quality is that the least errors and bug you have the better is the software quality and also the better readability of the code. Let's say someone new is coming to the team or the project and they can understand the code. So clean code" (Interviewee 4).*

The two interviewees highlighted that SW quality is not only about the code working and having as few bugs as possible, but also about code design and structure, emphasizing readability and maintainability as impacting SW quality. Interviewee 14 highlighted that sometimes new code or legacy code can be complex and not easily understandable, making it hard to work with. However, one PO (Interviewee 9) and one QM (Interviewee 21) highlighted the importance of maintainability to correct errors within the SW, stating the following.

> *"[…] I would like to add in that it's also maintainable, so you can maintain development […] to correct bugs. That's also part of the quality that we have a good way of doing that." (Interviewee 21).*

"We must do a system that is easy to change so that when we have detected problems or issues […] it should be easy to fix that. It should not be a system that is hard to update" (Interviewee 9).

4.1.2 Functionality and customer experience.

Each interviewee was asked: *"What are the most important quality attributes for automotive software in your context (e.g., safety, robustness, maintainability, performance, or something else)?"* Across the interviews, seven quality attributes were mentioned: safety, robustness, security, reliability, maintainability, performance, and scalability. The frequency of mention of the quality attributes is shown in *Figure 4.1*.

While these attributes have been expressed from different perspectives by different roles, the case company's internal documentation (2026) further highlights a layer of regulatory obligations tied to some of these attributes like safety and security. The documentation describes "*built-in compliance"* which states; the organization must comply with regulatory requirements, continuously follow defined processes, and provide evidence that its processes and SW meet customer needs and quality standards (Case company internal documentation, 2026).

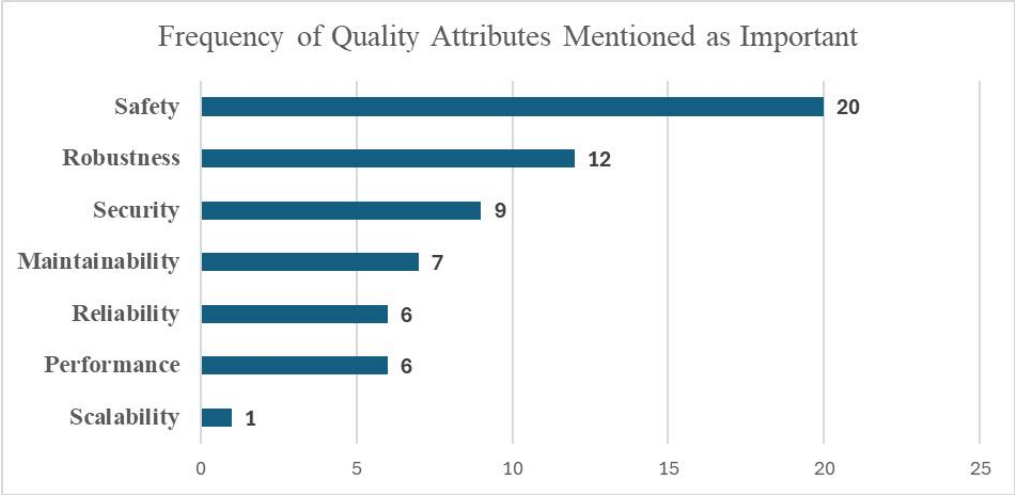


Figure 4.1 Frequency of quality attributes mentioned as important

While some interviewees identified one attribute as most important, many emphasized several. *Figure 4.2* shows how many attributes each interviewee mentioned in response to the question.

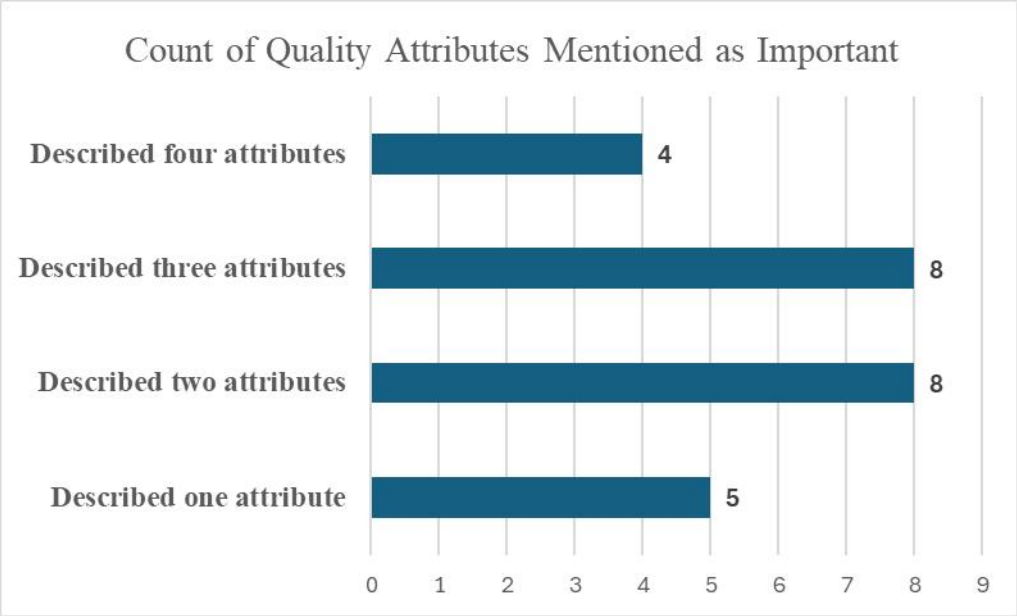


Figure 4.2 Count of quality attributes mentioned as important

Interviewees involved in the development of safety- critical functions emphasized safety and/or security as the most critical quality attributes, as illustrated by the quotes below.

> *"[…] There's a huge focus on system safety, and we obviously have an ISO standard 26262, which we always have to follow." (Interviewee 10).*

> *"From our perspective […] cybersecurity, […] because we have specific cyber security requirements we need to adhere to." (Interviewee 5).*

> *"It needs to be robust and have good quality because we have a safety critical [...] and if it's not safe or good quality in the software can mean very big consequences for the end customer." (Interviewee 16).*

Similarly, interviewees working with safety- critical functionalities and requiring adherence to specific standards or requirements, emphasized safety and security as critical quality attributes. Another interviewee highlighted that multiple quality attributes are important, as illustrated by the quote below.

> *"I think there is no most important, different attributes are important on different stages. […] for example, traceability which is very important on all of the stages. You have safety of course […]. Cybersecurity of course is important. […] So I wouldn't consider some attributes more important than the other. It depends on the market, of course it depends on the product […] some of the attributes of course are important in terms of the legal perspective." (Interviewee 6).*

Interviewee 6 highlighted that the importance of quality attributes may vary depending on circumstances such as product type, market, and legal aspects. At the same time, the interviewee noted that all quality attributes are important and cannot be ranked. This is expressed across the interviews, and *Figure 4.2* indicating that two and three quality attributes were most frequently mentioned. This is illustrated by the quote below.

> *"I would agree with every single one from the list you've mentioned. And it's hard to say that one is you know more important than the other. I think that's only if we have all pieces in place […] then we should say OK, it's a good quality." (Interviewee 14).*

> *"[…] all of them is very important, but the safety, definitely safety and security is definitely one of the top concern that we have to be very, very careful and then the performance and robustness comes for the for the customer." (Interviewee 12).*

> *"But we also need to think about the safety. We need to think about potential situation and how to manager the error or some problem in the code, we need to think about the robustness. We need to think about user experience. So I don't want to say that we focus only on one aspect all of these aspects are really critical from the delivery point of view." (Interviewee 19).*

Interviewees 19 and 12 also highlighted the importance of quality attributes in relation to customer/user experience. Across the interviews, various quality attributes and their relation to customer experience were highlighted. Quotes below illustrate some of these quality attributes and how they relate to customer experience.

> *"For automotive software it's just reliability, I think it is the most important. Pick between three different things […]the third thing is that it will work any day, […] I think that's what the customer want." (Interviewee 2).*

> *"[…] robust system that you can rely on and trust. […] Robust means no obvious issues found constantly. For I look at the end customer perspective in most cases the quality means that a customer should not notice if there are bugs, it will always be bugs. [...] robust system that take care of it and have you known a good way of dealing with the issue." (Interviewee 24).*

> *"Safety I would say as number one and performance for sure as second. The rest are also important. […] when we do some bug fixing is 99% of the cases to improve the performance, to improve the customer experience and of course to be according to legal and according to safety standards." (Interviewee 18).*

Across interviews and roles, robustness was frequently discussed in relation to customer experience. Interviewee 13 further highlighted robustness of the SW as important in case of a cyberattack to withstand it. Reliability was mentioned by six

interviewees according to *Figure 4.1*, three of the interviewees explained it in connection with customer impact, and that the SW should work as intended without it malfunctioning or something unintended happening.

Figure 4.3 visualizes the frequency of quality attributes mentioned per role. Safety is the most frequently mentioned quality attribute across most roles, except for Developers and POs. Developers most frequently mention robustness. As indicated above, robustness and reliability are often mentioned in relation to user or customer aspects. A higher proportion of respondents in hands-on development roles such as architects, developers, and requirement developers, mention of these attributes (73%, 8/11) compared to EM, QM and POs (61%, 11/18).

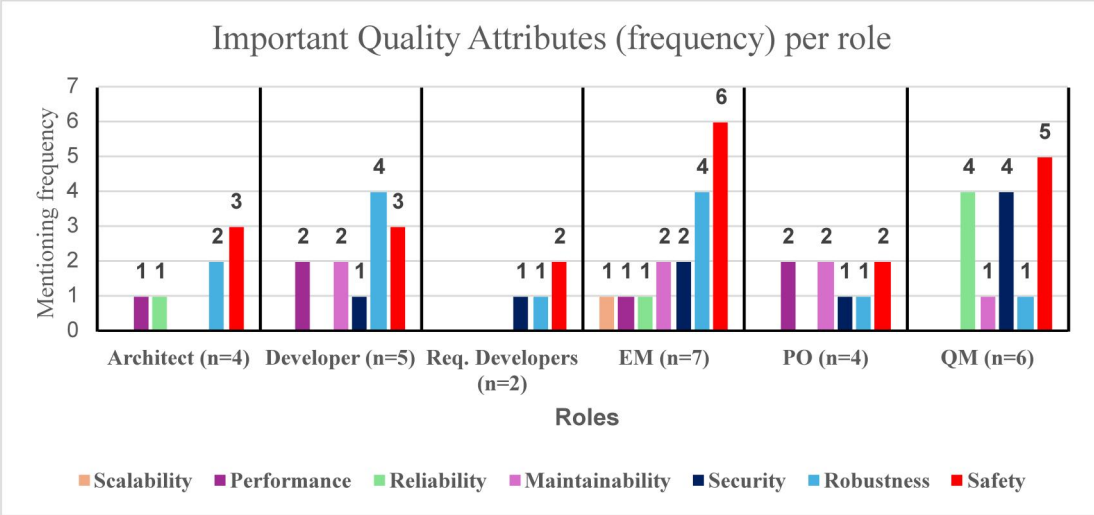


Figure 4.3 Frequency of quality attributes by role

4.2 Quality management in software development

Case company's internal documentation defines quality assurance (QA) and quality control (QC) as parts of the SW quality management. It defines QA and its purpose as *"[...] preventing defects by establishing and following processes and standards.*" This includes process definition, training, awareness, audits, and reviews. It also defines QC and its purpose as "*[...] the actual evaluation of the software to identify defects*." QC activities include testing, defect tracking, and performance monitoring. (Case company internal documentation, 2026)

The empirical findings from the interviews also highlight how QA and QC are understood and described, including their purpose and related practices in the case company. The findings show both similarities and differences in the interpretations and implementation of both quality aspects across the roles interviewed. These findings are further structured into three subthemes: (1) Overlapping and similar descriptions of QA and QC, (2) QA and QC purpose and practices (3) verification of safety/security/quality requirements.

4.2.1 Overlapping and similar descriptions of QA and QC

Across the roles interviewed, many participants described both QA and QC using the same practices, such as checklists, testing, verification, and documentation. Many others also explicitly stated uncertainty about the differences between both quality aspects, expressing uncertainty in the understanding and use of "QA" and "QC" language in the case company.

Some Engineering Managers (EM) expressed this uncertainty of the differences in both concepts, illustrated through the below quotes.

> *"[...] I don't know the specific difference between the two, quality controls and quality assurance." (Interviewee 13).*

> *"What? What is quality control? [...] I don't understand fully what is the difference between quality assurance and quality control. This is my problem." (interviewee 19).*

Both interviewees expressed difficulties in distinguishing between QA and QC, and similar uncertainties were expressed across the interviews.
When asked about QA, Interviewee 21 (QM) described design reviews to ensure correctness, followed by implementation, verification, and checks. Immediately afterwards, the interviewee was asked to define QC and stated they are *"more or less the same."* Another QM also described similarities between QA and QC, emphasizing reviews as a common element. One PO described QA as having a process focus with reviews as central practices. When asked about QC, the interviewee stated the following:

> *"So a quality control, I would more or less connect it to the previous answer about quality assurance. It starts from a set of rules on how to develop like some coding guidelines or model development guidelines[...] The developers themselves are aware of what kind of guidelines they should follow in order to ensure that quality, or at least a level of quality will be there. [...] and then a set of peer reviews" (Interviewee 18).*

As illustrated in the quote above, the PO referred to QA when describing QC rather than distinguishing between the two aspects. The PO emphasized guidelines (process focus) and peer reviews as key activities within QC.

A similar overlap was also expressed by a requirements developer, as illustrated by the quotes below.

> *"I suppose quality assurance is almost the final step where you actually see that we have fulfilled all the required steps during development, actually. So, it's usually for us the checklists we go through to see that we have done [...]*

everything and that they have passed and if they haven't passed have we written a deviation also in our system […].” (Interviewee 27).

When asked about QC, the interviewee stated:

> “Yeah, *back to the checklist again, then I suppose that is the quality control that we see that we can fulfill all the checklist, we have documented test reports, test verdicts, and all those things done” (Interviewee 27).*

Similar to other roles highlighted above, the interviewee linked QC back to QA rather than distinguishing between the two concepts. Further, QA was described as checklist-based checking to confirm that required activities have been completed, occurring in the “final steps” of development.

Across some interviews, as highlighted by quotes above, QA and QC were described using similar terms and mechanisms, and in some cases, interviewees expressed challenges in distinguishing between the two concepts. In several cases, interviewees referred to QA when describing QC.

4.2.2 QA and QC purpose and practices

While overlapping interpretations were evident, some participants expressed clear distinctions between QA and QC. They described differences in both purpose and associated practices. Across the interviews, QA was often framed as process- and prevention-focused, whereas QC was described as checking the operational output, with practices such as testing and gates frequently mentioned. Illustrative examples are provided below.

When asked about QA and its purpose in SW development, Interviewee 10 (EM) stated the following:

> “*Quality assurance is making sure that we have a good roadmap and a plan to make sure that we review our processes and ways of working. We try to align and ensure that we have a standard way of working because if you have a standard way of working then it's easy to measure in terms of metrics and KPIs.” (Interviewee 10).*

The interviewee describes QA as a planning- and process-oriented mechanism, aimed at standardizing and aligning ways of working. Doing so according to the interviewee allows for easier measurability.

Another EM interviewee 12 described QA as involving multiple check across development, highlighted by below quotes

> *“[...] So we have to assure that within the processes that we have all the precautions, all the checks before actually we deliver something and it's*

layered. It is not only one level; I will say it is layered in different levels. So, it's not only one check, but there's multiple checks around it, the KPIs, the checks, the reviews, the tests, verification, validation, all of them as a package." (Interviewee 12)

"[...] we are trained to capture everything [...] everything has to pass through another person's review. So, we don't submit anything without the review basically and we are trained to reduce the risk of getting out something unintended by reviews, checks, KPIs, and then activities around testing and integrations." (Interviewee 12)

The interviewee described QA as a "*layered system*" conducted through multiple checks across the development process rather than a single one-time activity. The interviewee also emphasized review discipline, describing that work products are not submitted without another person's review.

One further EM highlighted QA in relation to a process focus and adherence to the process, stating the following

"Quality assurance is like ensuring that you have all the evidence that you have built your software in the correct way. [...] So quality assurance for me is at least a lot of looking into the evidence of that you have followed your process, and the process is there to make sure that you fulfill like safety standards, security standards and things like that." (Interviewee 15).

The interviewee highlights assessable evidence of process compliance and emphasizes the importance of processes to make sure that safety and security standards are fulfilled.

When asked about QC and its purpose in SW development, Interviewees 10 and 12 described it as follows:

"For Quality control, we have a certain project milestones where we try to make sure we have maturity [...] we do not procced to [field testing] until we've achieved certain quality aspects and there's obviously some control checks that need to be done at certain stages [...] We obviously have our requirements that we manage, and we try to understand if we've implemented them and we've tested them and verified them."(Interviewee 10).

"Quality control, I think it's an activity that is permanent. It is not a one-time activity. It should be daily not only before the release [...]. we have continous activity about the quality monitoring and increasing the quality. [...] Software daily regressions, so that is basically you are ensuring that your software every day is actually in the customer delivery maturity." (Interviewee 12).

Interviewee 10 described QC in relation to project milestones and stage-based control checks, including assessing whether requirements have been implemented, tested, and verified before proceeding to [field testing]. Interviewee 12 described QC as a daily activity involving continuous quality monitoring and daily regressions, emphasizing that it is not only an activity performed prior to release but ongoing to maintain "customer delivery maturity".

When asked how QA, QC, and verification activities are realized in practice within the development process, Interviewee 10 stated the following:

> *"[...] different types of test cases, not just connected to requirements but qualification and verification test cases nightly [...] We run gating what we call smoke test cases every time we try to release or deliver software so that if they fail those test cases, we stop, we do not push any software further up the chain or downstream. [...] We have delivery reviews, so we review the content and documentation before we deliver to the software [...] So we have every single level of testing from unit code level up to the [product] testing [...]" (Interviewee 10)*

This was described as realized through gating mechanisms and recurring tests, where failed smoke tests prevent SW from progressing further, alongside delivery reviews and multi-level testing up to [product] testing. Across the interviews testing is something frequently mentioned either as an important QA/QC practices or in relation to the verification of SW.

Several quality managers also shared distinctions between QA and QC; interviewee 20 described both terms as serving different purposes in SW quality, highlighting some activities involved. Illustrative quotes are shown below;

> *"[...]Quality control are actions which you perform on a daily basis it's usually predefined and quality assurance builds processes which will guarantee results at the end, including management, risk analysis, measurements like metrics and distribution of quality activities across teams and their work products" (Interviewee 20)*

Interviewee 20 emphasized QA as focusing on risk mitigation and prioritizing key risks and deliverables based on factors such as timelines, resources, and safety. The interviewee noted that exhaustive testing is not feasible, describing it as impossible to test every potential "*combination*." The interviewee also described a relationship between QA and QC, as illustrated by the quote below.

> *"Quality control is defined by quality assurance first. So you decompose process, you decompose system, and this decomposition which define by quality assurance and processes. After that, you can define test activity, and quality control happens later. [...] So initially you need to define what to test,*

how to test, which strategy and that is a task of quality assurance and only when it's defined then you are able to do quality control activities." (Interviewee 20)

The interviewee highlights that QC is dependent on QA, which is described as first breaking down the system and processes and establishing the test strategy. QC is then described as occurring later as the execution of control activities based on what QA has defined.

Similarly, other quality managers, Interviewee 5 and 6 shared clear distinctions in purpose of both QA and QC, highlighting both process and outcome perspectives with quotes shown below.

"[...] quality assurance is looking more at that we're following the processes and then the standards that are available while the control itself looks more at the end result that the product that has been delivered is doing, I mean fulfills its purpose." (Interviewee 5).

"Quality assurance I would describe as constant monitoring and providing the full traceability and confidence in your software [...] because if you keep your quality assurance all the way through the development process, then you will be able to deploy your software at any time to your customer with built-in quality." (Interviewee 6).

Together, these participants frame QA with a process-oriented focus, collectively emphasizing adherence to standards, monitoring, and traceability. Interviewee 5 describes QC as outcome-oriented, focusing on whether the delivered product fulfills its purpose. Interviewee 6 similarly describes QC in relation to ASPICE by emphasizing awareness of sub-process interconnections and outcomes, stating *"[...] if you have proper outcomes from every subprocess, then you under control"*. How QA and QC are realized in practice is highlighted by below quote.

"[...] we have a milestone which gives you a go- or no-go decision. So, for specific milestones, you must provide some specific documentations or outcomes [...] So this is both quality control and verification and how these gates are combined to each other gives you the quality assurance overall." (Interviewee 6).

One PO also describes clear distinctions between QA and QC. interviewee 26 described QA as repeatable activities supported by continuous learning and QC as verification activities highlighting a few examples. Illustrative quotes are shown below;

"Quality assurance, some actions or activities that we need to perform to be able to ensure the quality. And that could be using checklists that are maybe created from previous experiences where we have noted some lessons learned.

[...] So, we make sure we fulfill what is relevant for our software." (Interviewee 26)

"Quality control, [...] but I could associate it maybe with the verification [...] so we have some tests that are being automatically run every night, so that's one way of control or maybe we are having gate checks, so you do not merge anything that can break the master, for example" (Interviewee 26)

Another PO, interviewee 9 described the purpose of QA as early detection mechanisms throughout the development chain and described QC as assessment points throughout the chain. Illustrative quotes are shown below;

"I think everything that goes into what we usually call quality assurance should be something that is going throughout the complete chain so that we don't wait until the last steps in the development chain before we detect things. [...]It can be from start to understand that we do the right and when we do the implementation, get feedback as fast as possible." (Interviewee 9)

"For quality control it's more or less that we need to take points throughout the complete chain and we need to do certain tests that are running on the final release, but the sooner we can indicate problems the better." (Interviewee 9).

A requirement developer, interviewee 17, described QA and QC and their purpose as illustrated in the quotes below;

Describing QA - *"It is the Descriptive factor of quality, but then also all the specifications for what it's supposed to do from the start. So all the requirements, all the processes around also relates to quality somehow." (Interviewee 17).*

"Yes, If I would describe quality control, yes. I would mainly think about the process that we apply to try to achieve the specifications, the implementation and the testing of the implementation." (Interviewee 17).

The interviewee described a distinction between QA and QC. QA was framed around specifications and processes from the start of development, whereas QC was framed around implementation and testing in relation to those specifications.

SW architects also describe clear distinctions between QA and QC. interviewee 2 described QA as good habits that support risk reduction and QC as testing and inspections. Illustrative quotes are shown below;

"It's about building good habits I would say to have what you call it a standard procedure that minimize the risk that something that is faulty slips

through. Yeah, I think that would be my definition of quality assurance." (Interviewee 2)

"Quality control. Yeah, that's. I mean, it's really related to testing, but it's also adds a number of other steps, and that is some usability and validity aspects. [...] but I think when it comes to software, it's difficult to just look at something, you need to run it also before you before you can say if it's good or not" (Interviewee 2)

Another SW Architect, interviewee 25 made a distinction between the two aspects, highlighted by below quotes

"Yeah, quality assurance is like very much important in software development because not only the normal software development actually, but in a safety ISO 26262 also like clearly says that actually like you need to have a like a quality management. [...] Usually we think about like mostly on the safety part but then as I mentioned earlier, like a quality also talks about safety or safety also talks about quality. So you need to see the complete life cycle, the quality has to be assured." (Interviewee 25).

"[...] I mean from the software requirement from the customers until actually like a deliverable like we need to have actually like a complete quality control like what changes you are bringing and like what deliverables you are going to make. [...] It's a proper checklist guideline, gates, milestones, gates like actually follow-ups like any deviation and then actually like the deviation and follow like within the timeline." (Interviewee 25).

Interviewee 25 thus frames QA as a lifecycle-oriented perspective in a safety context, emphasizing the need to assure quality across the complete SW lifecycle. In contrast, QC was described through formal checkpoints throughout the lifecycle such as checklists, gates, milestones, and follow-up of deviations, linked to changes, deliverables, and timelines

4.2.3 Verification of safety/security/quality requirements

A recurring pattern seen across the operational roles is that safety, cybersecurity, regulatory compliance and release readiness rely on a few individuals with specialized knowledge or authority holders in some cases.

Requirement Developer (Interviewee 27) described how safety requirements are managed by an architect with specialized knowledge who reviews requirements, documentation and "*safety cases*". Illustrative quote is shown below;

"[...] We have one of our architects specialized in the system safety, all those functional safety and system safety. [...] he is reviewing also all our

requirements and documents from that point of view. So, to see that we actually fulfill everything there [...]" (Interviewee 27)

One SW architect also highlights similar explicit dependency on a key person to ensure compliance with safety and cybersecurity standards. Illustrative quotes are shown below;

"Yeah. we are quite lucky, we have a very good system safety engineer in our ART that is extremely competent with the standards and the process itself. So we in the ART relies heavily on this guy to make sure that we follow the specific safety standard and the security now for the cyber security also. [...] he's also very competent in like ASPICE and so on as well." (Interviewee 17)

Similarly, SW Developer (interviewee 22) highlights a dedicated safety engineer who reviews their work products and documentation, ensuring they comply with standards. Illustrative quotes are shown below;

"We have a lot of SW documents we call SWDD, SWTP, test plans, test requirements, a lot of things. So once after this, will be reviewed by the safety engineer for generating the SACA documentation for safety purposes and all. So that is a higher level thing but In-depth." (Interviewee 22)

Finally, POs (Interviewee 18 and 26), shared similar dependency on a safety manager and system architect for legal guidance and regulatory compliance. Illustrative quotes are shown below;

"Development guidelines we maintain in [locally developed frameworks] and then some legal this we have guidance through our safety manager and we have some local documents like some safety related documents that are local in SharePoint". (Interviewee 18)

"We have in our architects, our system architect, keeping track of the regulatory requirements. [...]we have persons in the team that has a good contact with the homologation team, so they keep track on when do we have the certification, when do we have to be ready. Yeah, so I think more or less persons for making sure that we that we do what we need to do, [...]" (Interviewee 26)

Across the interviews, various participants have highlighted the support of certain specialized colleagues for the review, verification and compliance of safety, security and other necessary regulations within their context.

4.3 Process guidance and ways of working

The case company has several sources that guide SW development, outlining good practices, standards, and regulatory requirements that must be considered. Three

central sources of process guidance were identified: the P- framework, the B- framework, and the New- framework. In addition to these, In addition to these, there are also locally developed frameworks for process guidance and ways of working in some departments. The frequency of mentioned sources for process guidance is visualized in *Figure 4.4*

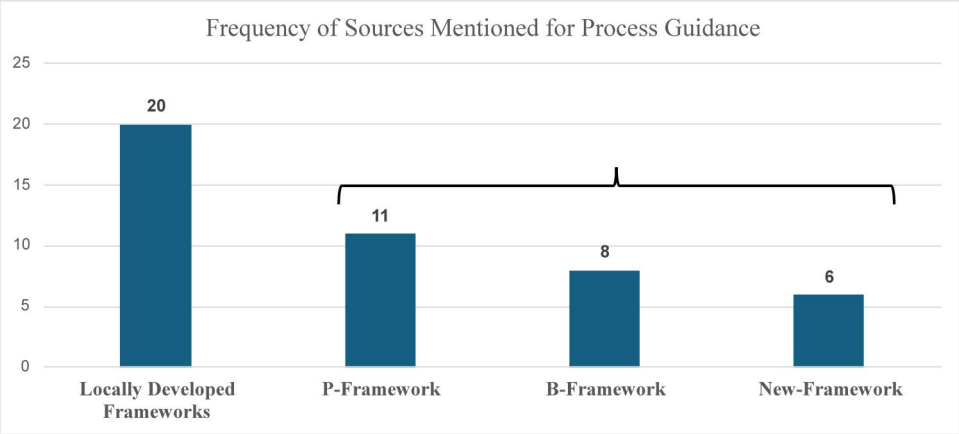


Figure 4.4 Frequency of sources mentioned for process guidance

As *Figure 4.4* indicates, locally developed frameworks are the more referenced source for process guidance regarding how the development work should be performed. However, while some interviewees referenced one source, other interviewees referenced two or more for finding guidance on how development work should be performed. *Figure 4.5* highlights the number of sources used for process guidance.

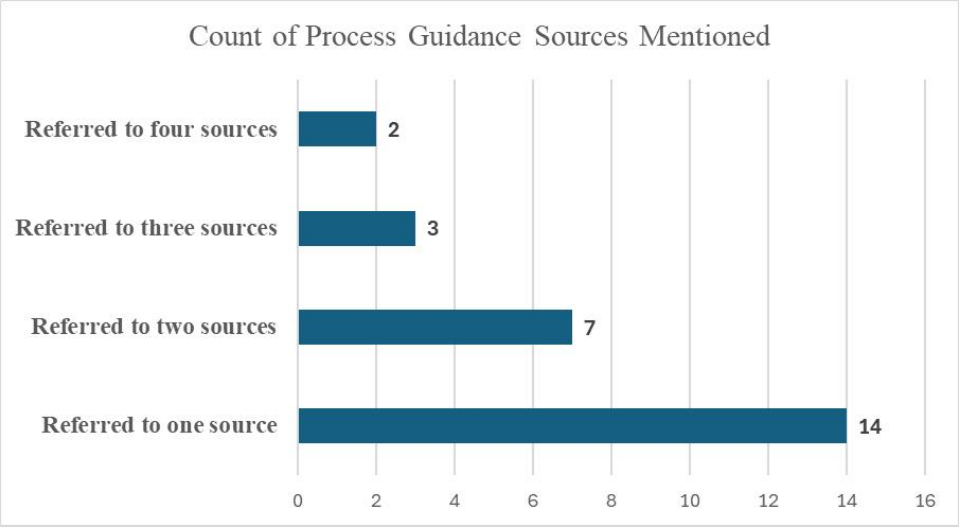


Figure 4.5 Count of process guidance sources mentioned

The reasoning behind referring to one or several sources for process guidance, depends on various aspects such as role and for what purpose discussed further on in the chapter. The results however show that out of the 14 interviewees referencing to one source, of these 10 referenced locally developed frameworks as the source for

process guidance. A breakdown by role and referenced source for process guidance is visualized in *Figure 4.6.*

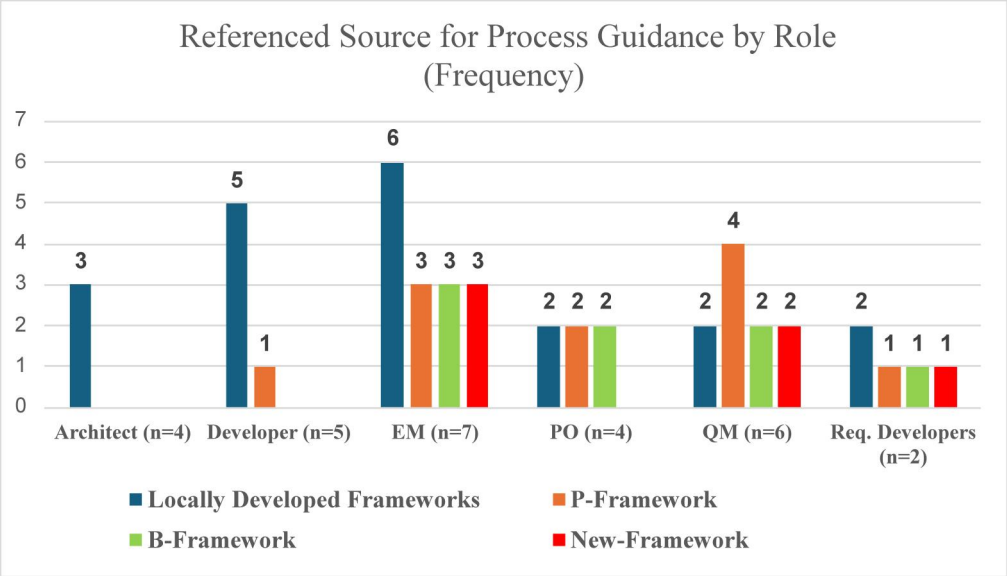


Figure 4.6 Referenced source for process guidance by role (Frequency)

As *Figure 4.6* shows, some roles put more emphasis on specific sources such as developers and architects referring to locally developed frameworks. Meanwhile, other roles such as POs have a more equal distributed reference to different sources for process guidance.

While these three central sources exist, the case company internal documentation also highlights the need for standardized SW engineering processes including methods, tools, and KPIs. It also lists increased productivity, collaboration, and learning as advantages of standardized SW processes (Case company internal documentation, 2026).

The authors asked interviewees where they usually found documented guidance on how SW development should be carried out. Managers were also asked which sources they encouraged their teams to use. Additional questions covered whether the information was easy to find, up to date, reliable, and adequate to support delivery of high-quality SW.

The authors identified three subthemes in the empirical findings: (1) central guidelines and frameworks, (2) locally developed guidelines and frameworks, and (3) locally developed process guidance related to central processes. Within Subthemes 1 and 2, aspects related to ease of finding the information, whether it was up to date, and its reliability were further identified

4.3.1 Global guidelines/frameworks and standards

Some interviewees referred to centrally communicated guidelines and frameworks as sources of information on how SW development should be carried out. This is illustrated by the following quotes.

> *"I try to promote B-Framework and P-Framework." (Interviewee 7).*

> *"I guess P-Framework is more hands on. B-Framework is more like on a higher level. So, for my own sake I review B-Framework and then I would say that the development teams should go to the P-Framework page." (Interviewee 5).*

Both interviewees emphasized B-Framework and P-Framework as the main sources of information for development. Interviewee 5, as a QM, highlighted a distinction by noting that P-Framework should be the primary source for developers. Interviewee 5 also expressed that the information in these sources is thorough enough. In contrast, Interviewee 7 described the information as mixed, where some content is sufficient and some is not, stating *"some information is more or less confusing"*. Overall, interviewees differed in their views: some considered the central guidelines thorough enough, while others saw room for improvement. This is illustrated by the quote below.

> *"[…] I think we can simplify things in the [case company] and we can improve different views of the processes. The higher level like the B-framework and the P-framework they normally describe everything you need to know about requirement management […]. But it's hard for a developer to find what to do as a developer across these processes in order to deliver software". (Interviewee 15).*

Interviewee 15 highlighted that while the B- Framework and P- Framework are useful for requirement management, developers may find it difficult to locate guidance relevant to their work. A similar perspective was provided by a QM (Interviewee 6), who described the P- Framework and B- Framework as *"the big processes"*.

Interviewee 12 (EM) expressed a similar view to Interviewees 5 and 7, describing the B- Framework as a key source of information and stating that the New- Framework is also followed.

> "*We are trying to follow the B-Framework and the New-Framework. […]. So we get the information on these sites and then we actually try to follow them up." (Interviewee 12).*

Although Interviewees 5 and 7 did not refer to the New- Framework as a source of information, both shared views about it, as illustrated by the quotes below.

"New-Framework […] I would say is a pretty recent thing and I don't think its fully communicated and understood by the organization how this should work." (Interviewee 5).

"[…] And then the New-Framework, everybody talking about it. But where is it? What are we doing? Is it official at the [case company] or what? What's happening? And it has been going on for quite some time. So I don't refer to the New-Framework." (Interviewee 7).

Both Interviewees 5 and 7 expressed uncertainties regarding the status of the New- Framework. Although some interviewees described the New- Framework as a source of information during the interviews. The uncertainties were also expressed, as highlighted by Interviewee 8: *"we also have company guidelines from New- Framework but most people in my team is not aware of the content […]".*

Many interviewees expressed both trust and uncertainty regarding the reliability of centrally communicated frameworks and how easy it was to locate information. Interviewees were asked where they found sources of information, whether these sources were easy to locate, and whether the information was up to date and reliable. This is illustrated by the quotes below.

"That's a problem that I think is just getting worse all the time because we add new systems. But we don't close the old ones. So that means that you have to look in several different places […]. You're not 100% sure this is the latest document." (Interviewee 2).

"I would say that I think the biggest problem is that the people don't know where to find everything because it's not easy […]." (Interviewee 1).

"I think it's harder for people to navigate around the intranet." (Interviewee 10).

"I think B-Framework and P-Framework I think they are updated quite often as far I know. Finding information if you are asking me about user friendliness, I will say with latest technologies […] it could be much easier. […] It could have been a little bit more easier, definitely." (Interviewee 12).

"[…] My overall views is that it's a bit messy. When I start it was more clear because there wasn't many places to go. You had B-framework […] so there was one place […] and then over time it has grown and now we have a lot of places for findings different things." (Interviewee 17).

"I Should say B-Framework is often correct, I should say P-Framework yeah they have become better at least. […]. So it's not always reliable what's stated on those pages in P-framework. You have to read them a bit carefully, and when you're unsure you have to ask the people responsible" (Interviewee 27).

As indicated by the quotes above, interviewees expressed differing views on whether the information in the centrally provided frameworks is up to date and reliable. Regarding the ease of finding information, several interviewees indicated that it is not easy and could be improved, an observation that was also raised across the interviews more broadly.

4.3.2 Local guidelines/framework

The case company documentation highlights the need for certain considerations when making domain-level adjustments to processes; "*[...] because we think they suit better to our software development. But there is likely a room for standardizations, however unique we think our software is. Therefore, it's worth to evaluate whether you actually need a unique process, method, or a tool, rather than using the recommended standard [...]*" *(Case company Internal documentation, 2026)*

Several interviewees highlighted department-developed information sources for how development work should be carried out. In the empirical data, most EMs who discussed department-level frameworks or guidelines referred to locally developed frameworks as the main source, as illustrated by the following quotes:

> *"Within our department we have our [locally developed frameworks], and we have a big focus on process methods and tools, so we have clearly defined guidelines and ways of working on the department level. […]" (Interviewee 10).*

> *"And then where the guidelines and where the information is stored is that we our [locally developed frameworks] that actually we publish and update all the time the guidelines and all the time the processes that we have." (Interviewee 12).*

> *"I think we mainly have our own. […] It is local guidelines like [locally developed frameworks] local. [...]." (Interviewee 23).*

The EMs explained that locally developed frameworks are used to find guidelines and descriptions of processes for how development work should be carried out. Interviewees 10 and 23 also provided reasons why these locally developed frameworks are preferred, as illustrated by the quotes below.

> *"[…] I think people always look at our own [locally developed frameworks]. That's probably the only place they would go to […]. I think it's harder for people to navigate around the intranet." (Interviewee 10).*

> *"[…] And I think one reason for that is that those big business systems are overwhelming. It doesn't really, they are so generic, so they don't really apply to the individuals. [..] managers and products mangers need to help break down what is important for respective area." (Interviewee 23).*

The interviewees provide two perspectives: (1) navigating the centrally provided frameworks on the intranet is difficult, and (2) the central frameworks may be too overwhelming and not applicable in all circumstances. The latter point was further highlighted by a QM, who stated the following:

> *"[…] I'm not trying to overwhelm the teams with the big processes because it gives basically non-value for them, but instead focus their intention of the small things that they can do to straight away and make some kind of impact on quality." (Interviewee 6).*

Interviewee 6 echoed Interviewee 23's view that centrally provided process guidance can be overwhelming.

Across the interviews, locally developed frameworks were highlighted across several roles as the main source used to find information. This is illustrated by the quotes below.

> *"Internal documentation mainly. Internal processes we document in our [locally developed frameworks] portal […]." (Interviewee 8).*

> *"B-Framework can be used a little bit as reference, but mostly all of the best practices and ways of working and design guidelines we maintain documentation in [locally developed frameworks]." (Interviewee 18).*

When asked whether the locally developed frameworks were up to date, thorough, and easy to locate, some interviewees expressed the following:

> *"I think it's quite good because it's a live document and it's maintained and updated often. […] I would say that the team has sufficient information to provide good quality results." (Interviewee 18)*

> *"Yes, finding documentation is simple as we can have a clear structure in our [locally developed frameworks] where internal process/guidelines are stored. guidelines are regularly updates." (Interviewee 8)*, further stating; "*I think they are thorough yes. We rely mostly on internal documents and we keep them maintained".*

The two interviewees highlighted that they perceived the information on the locally developed frameworks to be updated regularly and thorough enough to support a high-quality outcome; a pattern also reflected across the interviews. Interviewee 8 (developer working with testing) added that unclear guidelines should be improved and updated, stating:

> "*I usually encourage team members as a first step to read guidelines, if they get stuck on some part, then the guidelines needs to be updated to be more clear.*" *(Interviewee 8)*.

However, one QM highlighted a similar perspective, but with a focus on lack of control, stating the following:

> *"The [locally developed frameworks] is hard to see if they are updated or correct. We don't have that control of the page, so more or less everyone can go in and write […]." (Interviewee 21).*

While Interviewee 8 highlighted the benefit of being able to update guidelines when something is unclear, Interviewee 21 raised concerns about limited control over these pages and the extent to which the information is updated and correct.

4.3.3 Local process guidance in relation to central processes and standards.

While some interviewees highlighted the use of both locally developed guidelines and centrally provided guidelines, as indicated in the previous section (e.g., Interviewee 12 – EM), Interviewee 10 an EM also described utilizing centrally provided guidelines within locally developed guidelines, as illustrated by the quote below.

> *"We have adopted the New-Framework in our [locally developed frameworks], so we have that structure. What we did notice is that it's very much focused on maybe processes, rules, it is maybe a bit hard to navigate, but we have adopted the structure." (Interviewee 10).*

Interviewee 10 (Department 2, Function X) explained that the reason for adopting this structure was to align ways of working with Function X and across the case company overall. This is illustrated by the two quotes below.

> *"[…]. I think that's it's important that my team, we align and collaborate with X at every level just to try to align processes […]." (Interviewee 10.*

> *"[…] We are now setting up our [locally developed frameworks] to make it look like. So, we actually understand we can fulfill New-Framework, we can pull in processes which are aligned across the company, we want to fulfill ASPICE […]." (Interviewee 10).*

Interviewee 15 (EM) also belonging to department 2 expressed similar view,, stating that:

> *"I tell my engineers that of course we have our own tailoring, but they should always refer to the next level of process. […] it should never violate the next level. And for us the next level is X and what processes they have decided. […] So from Department 2 to X, to New-Framework, to P-framework to B-framework." (Interviewee 15).*

Both Interviewees 10 and 15 highlighted that while local frameworks and guidelines are used, the importance of adhering to centrally communicated processes and

frameworks. Interviewee 10 also highlighted ASPICE, a point that was further raised by Interviewee 24 (Developer working with testing), who explained that they maintain their own locally developed frameworks but use ASPICE as a base.

> *"The base we have from us is ASPICE, yeah that is the base we are using at department 2.2. […]. This is how we need to do it. And it rarely contradicts with P-framework processes, in some cases yes it does, but rarely. […] If it doesn't fit us, we do it so it fits us." (Interviewee 24).*

Interviewee 12 (EM), as mentioned, highlighted both centrally provided and locally developed process guidelines and also referred to ASPICE, as illustrated by the quote below.

> *"[…] I will say we have been audited for ASPICE and then we have been in the past working with mapping our own ways of working towards ISO 26262 as well as the ASPICE. So we have this comparison between what is the difference and what's our processes actually fitting where we map them to." (Interviewee 12).*

Interviewee 12 described mapping locally developed process guidelines to the ISO 26262 standard and ASPICE to compare how existing ways of working align with these standards. Interviewee 7, as presented earlier, emphasized centrally communicated process guides such as the B- Framework and P- Framework. During the interview, Interviewee 7 also explained that these frameworks are inspired by, and aligned with, ASPICE and ISO 12297, stating:

> "[…] *you know B-Framework and P-Framework. They are inspired to a large extent by ASPICE and ISO 12297, the life cycle model, so it's not. It fits and it's aligned so." (Interviewee 7).*

The relationship between process guidance and ASPICE and/or ISO standards was raised explicitly by Interviewees 7, 10, 12, and 24 when discussing process guidance and ways of working. ISO/ASPICE were also mentioned across other interviews, but not consistently in connection with process guidance and ways of working.

Aligning with interviewees 7, 10, 12, and 24, the observation of case company ASPICE assessment meeting April 2026, further showed that ASPICE supported the need for a unified way of working. Process alignment was discussed as a major concern; the participants highlighted that process definitions and tools used in all platforms need to be coordinated to ensure consistent implementation across teams. They further discussed that central platforms support process definitions and direction; hence, additional smaller local changes within individual teams could increase fragmentation (Observation, ASPICE assessment meeting, [April, 2026]).

4.4 Factors impacting software development and quality

During the interviews, several aspects were identified impacting SW development and quality. Based on the empirical findings, five themes were identified: (1) process focus (2) requirements, (3) system complexity and dependencies, (4) time, and (5) testing.

4.4.1 Process focus

The case company documentation describes built-in SW quality as a strategy to integrate quality into the entire SW development processes and activities ensuring continuous alignment to quality management principles. With early emphasis on the process over end-stage evaluation: *"This is to avoid detecting many defects [...] in later stages of development, where the cost of redesign and corrections is multifold higher." (Case company internal documentation, 2026).*

Several participants also shared this process focus in their definition of SW quality and its related practices. Each interviewee was asked: *"What practices are most important for ensuring high- quality software development (e.g., reviews, audits, assessments, metrics, checklists, gates, or something else)?"*

Across the interviews, various perspectives were emphasized, but the importance of reviews and testing was highlighted in particular, as illustrated by the quotes below.

> *"Something that is really really important is reviews number one. And that means it's reviews of specs, it reviews of requirements and are they good enough and so on. And then of course since I'm working with that down to test, can we assure that if we have reviewed the requirements […] and we see that yeah, it seems good and then you test it." (Interviewee 24).*

> *"I should say reviews almost every part […]. And that include both designers, I mean the reviews of the requirements, the actual design and then the code review […]. So it's kind of a collaboration between developers and testers The testers write the test cases, but we need to be in collaboration all the time to see that they are actually testing what they should be testing. […] So the reviews are the base cornerstone […]." (Interviewee 27).*

> *"[...]if you don't do review of the of the breaking down on the component into the architecture for instance, then you may miss things if you don't make sure that you have consistency in between the software requirements and the software architecture for instance, then you have a link missing and this this needs to be in place." (Interviewee 7)*

> *"I mean reviews, we must already start the reviewing of the requirement. Writing the cases, ideally that should be by the testers but with review from the designers. And then, of course, the actual testing […]." (Interviewee 26).*

Testing was an aspect highlighted by all three interviewees, and it was also evident across the interviews as an important practice. All three interviewees emphasized the importance of reviews, particularly reviews of requirements, an aspect also expressed by several other interviewees. Interviewee 18 (PO) also emphasized gates in relation to reviews, as illustrated by the quote below:

> *"I think the top two I would pick and I cannot really put some of them first or second. It will be gating as well as proper reviewing. […] It's a human factor that is in the review. So, this alone even though something that can be really strong […] this alone is not enough, then you need build upon gating,[…] and some maybe milestones." (Interviewee 18).*

Interviewee 18 highlighted that reviews involve a human factor and described reviews alone as not sufficient, emphasizing gating as complementary. Interviewee 6 (QM) also emphasized gates, describing them as less influenced by human factors.

> *"I think that gates are the most important because gates are not human driven. You cannot get away from it, so, they are mandatory. If you don't do your job, you will not pass the gate. […] It's a pure mechanism that says OK, this is the gate, this is what you have to deliver on this specific level. […]. Otherwise you don't deliver, there is no blame, no offense. It's just a pure mechanism of quality control if you don't meet the requirements, you're out." (Interviewee 6).*

Interviewee 6 described gates as a "pure mechanism" that specifies what must be delivered, rather than being subjective or human-driven. Later in the interview, Interviewee 6 also stated that accompanying tests need to be committed to the SW, and that work cannot proceed otherwise.

Across the interviews, gates were highlighted as an important factor, emphasizing perspectives like those described above. Others emphasized gates in relation to the timeline, specifying when deliverables need to be in place. Interviewee 23 an EM described the software system as large and complex, with several developers working simultaneously, and stated that without gating mechanisms the system could easily break; thus, gating was described as enabling work in such systems.

Other interviewees highlighted that SW quality involves several activities and considerations across development. The two quotes below illustrate this perspective.

> *"To reach a good quality, we need to have a good design, that we have a good design what shall we do and then have a good review of that we have what we need to do in the correct time. So, it's hard to say the gates are what we need or the design or the verification. For me it's the whole chain here that we have the right content at the right time that are doing what it's needed to do." (interviewee 21).*

> *"[…] It's everything because as I tried to say earlier, it's all the activities where you create the requirements, where you review the requirements, where you approve everything related to each level […] if you don't have a process in place, if you don't have consistency in the test cases you will have a problem. […] I wouldn't say that it's one part that is important. It's the whole thing. It's a chain of activities in a way that needs to be in place." (Interviewee 7)*

Both interviewees highlighted several activities as important and described them as interconnected rather than stand- alone. While multiple activities, often in combination, were highlighted across the interviews.

The final perspectives highlighted from the interviews regarding practices important for high-quality SW development concerned processes, ways of working, and standards.

> *"[…] we must have a clear framework or guidelines and processes in place." (Interviewee 5).*

> *"I would say you have to have good standards, and you need to really be up to those standards […]." (Interviewee 4).*

The interviewees highlighted the importance of clear frameworks, standards, and processes for delivering a quality outcome. However, the empirical data also highlighted challenges related to these, as illustrated below.

> *"I think it starts with [case company] that we must have a much clearer process. I think it diverts. We have so many initiatives going on and so on and it's never really been stabilized. So I would like to see like a unified process that everyone follows." (Interviewee 5).*

> *"So it's hard to try to align on ways of working, what I've seen that many departments […] have their processes and ways of working. Many try to align or copy or align each other, but it's quite good if we can align at higher levels […]. (Interviewee 10).*

The interviewees expressed a need for clearer and more unified processes and ways of working across the organization, a view also expressed by other interviewees. In relation to processes, auditing was also highlighted, as illustrated by the quotes below:

> *"I think what we need to do more is look at metrics and auditing and process reviews." (Interviewee 10).*

> *"Audits of processes and ways of working I think it's very relatively in order to do that. […]. To look at it from a control view and the way we work and*

way we do things fulfills things. That's audit and certification perhaps." (Interviewee 15).

Interviewee 10 expressed a need for metrics, which is discussed further in Section 4.5.1. Both interviewees highlighted process auditing as important. This view was also expressed by other interviewees, particularly in relation to ASPICE.

4.4.2 Requirements

The case company documentation provides guidelines to integrate quality in the requirements from elicitation to specification, and to allocation (Case company internal documentation, 2026).

While requirements were described throughout the interviews as an important aspect of SW development as highlighted in Section 4.1.1, requirements were also highlighted across interviews as a challenge in SW development.

Interviewee 1, when asked about common quality issues, explained that requirements were often not good enough and that they could not be tested thoroughly, which the interviewee described as a root cause. This is further emphasized by two EMs, as illustrated by the quotes below.

> *"The biggest problem from my perspective is not clear requirements. […] But the requirements is very crucial […]." (Interviewee 19).*

> *"One thing is that the consistency between the very high level requirement all the way to the units requirements. […] or I will say the requirement quality. So you might have requirement that is not really saying what It should do […]." (Interviewee 12).*

As stated in Section 4.1.1, requirements developers (Interviewees 17 and 27) highlighted that good SW requirements are clear/understandable and testable. Interviewee 27 also noted that writing requirements for everything is challenging and that producing good requirements is not easy, as illustrated by the quote below.

> *"[…] it's not easy to write good requirements, that's for sure. […]." (Interviewee 27).*

During the initial stages of this research, the authors attended introductory sessions where challenges related to translating customer needs into requirements were highlighted (Introductory presentation, February 2026).

Furthermore, according to the case company's internal documentation, requirements should be evaluated for testability and feasibility (Case company internal documentation, 2026). When asked how this is done in practice, Interviewee 17 stated:

> *"So in our teams mainly, I would say that we determine if they are testable and feasible by people's experience […] I don't think we have like a clear guidelines or like rules how requirement should look like […]." (Interviewee 17).*

Interviewee 17 highlighted that assessing requirements often relies on the experience of the people developing them and further explained that guidelines alone are not sufficient for this assessment.
Interviewee 27 also noted that developers who write requirements often struggle with the verification aspects, which makes it important to review requirements together with testers. The internal documentation explains similar ideas, which describes testability and feasibility as potential bottlenecks in later stages and recommends involving relevant roles, for example: *"ask the relevant architects and engineers to provide their estimation of the feasibility"* (Case company internal documentation, 2026).

The case company's internal documentation (2026) further states that SW requirements should be *"[…] readable and understandable from a technical perspective"* and recommends using EARS template. Neither Interviewee 17 nor Interviewee 27 stated that they used EARS when developing requirements. However, both interviewees highlighted the importance of reviews to improve requirements. Interviewee 27 described how developers review requirements and provide feedback, stating the following:

> *"[…] it takes probably two or three weeks or something like that for going back and forth with requirements and questions and us updating the requirements […], refining them and actually making them more clear." (Interviewee 27).*

The case company's internal documentation states that product quality properties, including non-functional qualities, must be considered, measured, and monitored, including *"safety, security, performance, reliability, and usability"* (Case company internal documentation, 2026). It further states that regulatory requirements should appropriately specify safety criticality. As noted in Section 4.2.3, Interviewee 27 described that there is a person within the team with this knowledge who reviews the requirements. Interviewee 17, however, explained that this is primarily addressed through reviews and tagging safety requirements directly in the requirements tool, as illustrated below.

> *"So for the safety standards I think mainly the reviews is actually how we cover this, cause we don't have a formal way of writing them. We have some attributes in the [requirement tool] that we use related to safety […] we tag all the safety requirements." (Interviewee 17).*

Interviewee 26 (PO) highlighted that it was important to be given reasonable time to develop requirements to avoid unnecessary stress. They linked this to quality, noting that doing things right from the beginning is closely tied to requirements. Similar aspects were raised by an EM in relation to time and cost, as illustrated by the quote below.

> *"[…] Time is cost and very often we try to speed up activities. […] We don't spend too much time on the requirements because it's costly." (Interviewee 19).*

Interviewee 26 highlighted the importance of having sufficient time in the requirements phase, while Interviewee 19 explained that this phase is sometimes rushed due to cost considerations. In addition, Interviewee 24 (developer working with testing) described challenges with the requirements process, stating:

> *"The requirement process is not good in [case company] […]. I know there that there is something beautiful called what the [case company] model in requirement setting, but I don't think anyone is following that."* (Interviewee 24).

Interviewee 24 further highlighted challenges related to implementation when requirements are not finalized, stating:

> *"[…] the implementation is done, but the requirements is not really finalized yet. How did you know what to implement then? […] But since the requirement are not there, we can't test it because we don't know what to test against. That has happened quite often." (Interviewee 24).*

A developer confirmed a similar situation from a project they were involved in, stating: "*Actually we are about to finish the implementation, and we don't have any official requirements whatsoever" (Interviewee 14).* Together, these accounts indicate that there are occasions when implementation is close to completion or completed before requirements are finalized. As noted earlier in this section, Interviewee 1 raised testing-related issues, which aligns with Interviewee 24's point that testing becomes difficult when requirements are missing or not finalized.

Overall, the interview data highlights several challenges related to requirements. These challenges were evident across the interviews and were raised by participants in multiple roles.

4.4.3 System complexity and dependencies

Across the interviews, participants described SW and its operating context as highly complex. One interviewee noted that the case company is not delivering a phone application, but automotive SW being complex. The case company internal documentation highlights they are developing a complex SW and thereby the

associated processes are complicated as well. The quotes below further illustrate this complexity

> *"[…] but the system is so complex, so you don't understand the full complexity." (Interviewee 15).*

> *"The amount of complexity is insane and ideally one person should be able to explain everything and to be able to give requirements for everyone. But this is unrealistic of course. So, it's many persons, many experts that need to synchronize." (interviewee 18).*

Together, these interviewees highlight that the SW is complex and difficult to fully grasp the complexity, and that development therefore requires coordination across multiple experts. Another aspect raised when discussing complexity is dependencies between systems, as illustrated below.

> *"[...] if you are not cyber securing your product correctly, then you are jeopardizing all of the network that you are in contact with." (Interviewee 13).*

This highlights that the system being developed must be considered in relation to the wider network it is connected to and the potential impact of those connections. The case company documentation similarly highlights complexity and dependency in its business model, stating that*: "[...] even common functionality is handled in separate branches which creates unwanted variance and complexity[...]"* (Case company internal documentation, 2026).

Several interviewees described dependencies between systems as a challenge in the development of SW, most frequently raised by operational roles. This is illustrated by the quotes below.

> *"The vast majority of quality issues have to do when we interact with other nodes or when we try to integrate software in other nodes." (Interviewee 18).*

> *"And all the other parties of course, not having full understanding of how our neighboring parts in this big software system that the product actually is working. […] they have done some changes in their functions that they haven't realized that also impacts us and vice versa." (Interviewee 27).*

Interviewee 18 emphasized that dependencies are a source of quality issues in development, particularly when integrating with other nodes. The interviewee also noted that internal development does not face the same challenges and suggested that one reason may be a lack of synchronization across teams. Similar observations were raised by other interviewees as well, as illustrated by the quotes below.

> *"If it comes to thing that we control in out team ourselves […] If we own the dependencies in our team, I would say we have no gaps because we can solve*

those issues very quickly. […].I would say the gaps or issues comes if we have dependencies on other teams or even let's say in [case company], if we need to contact other departments to get their help [….]. It can be very time consuming just to get to this collaboration up and running to understand how should we actually work with other teams." (Interviewee 8).

"I think the easiest part is when you work on a unit level […]. This is the easier part to handle, it is very close to the developers to their actual daily work […] and this kind of work doesn't need a lot of collaboration. […] But then the systems grows, grows and grows, it piles up and then the responsibility goes […] you see how bigger it comes from the [function] software. It spreads out throughout the entire organization and there the problem starts to appear because its different priorities, different organization [departments], different decisions and so forth […]. And then you're like stuck, you have a bottleneck." (Interviewee 6).

Together, these interviewees highlighted that when dependencies are owned within the team or department, issues can be resolved more quickly due to limited coordination needs. In contrast, dependencies on other teams or departments make development more challenging because they require collaboration and alignment across organizational departments.
Interviewee 6 also highlighted later in the interview that collaboration and setting this up can sometime be challenging similar to interviewee 8. Interviewee 6 further highlighted that differing priorities and decisions across departments can create bottlenecks for the individual department.

Building on Interviewee 27's earlier point about limited understanding of neighboring parts and how changes in functions may introduce issues, Interviewee 27 also described that such changes can be difficult to identify, stating the following:

"[…] the function have changed slightly, then it's very hard to find that in reviews and so on […]. And it's super hard to find even in testing actually quite not so easy to do that. So that slips through and then it comes to the customer, and then it's discovered in the product and that is not always nice." (Interviewee 27).

According to Interviewee 27, changes in functions can be difficult to detect during reviews and testing, which increases the risk that faults have slipped through to the product and are discovered by customers. Interviewee 26 similarly noted that when faults have slipped through and a fault report is received once the product is in use, it can be difficult to identify what happened due to the overall complexity.

"I think that some faults that we received can be quite complicated ones that we get the reports from the [product]. It's not really obvious what happened. And sometimes it can also be hard to find it […]. Those are the most trickiest

falls when we have a lot of software interacting with each other in a complex environment, the hardest things to find." (Interviewee 26).

4.4.4 Time

The case company internal documentation (2026) highlights time planning instructions that describe how the organization should create logical plans at different levels to have shorter lead times for development, proactive follow-ups, and quality assurance activities. The documentation points out the roles responsible for the plans' creation, follow-up, decisions, and deviations to ensure the plans always reflect operational realities without excessive pressure on the teams (Case company internal documentation, 2026). The documentation also expressed the need to automate repetitive tasks, such as testing, to save time and effort, reduce errors, enhance quality assurance, and follow up activities in the right time and scope.

Time and time pressure were expressed across the interviews in various ways, one EM, when asked what circumstances were critical to ensure a high-quality outcome, highlighted the following:

> *"I think clarity on what is to be achieved that is the main thing […]. then of course there needs to be enough time for quality development […]. So purpose, clear definition of what is needed and time." (Interviewee 23).*

The interviewee emphasized three aspects of importance: purpose, a clear definition, and time. However, the interviewee also highlighted that they often come late into the development life cycle, which impacts how they work:

> *"[…] we have a tendency to favor fast implementation over quality. Also, because maybe or functionality is not safety critical. […]. We are working to shift that mindset to from fix fast or from a quick fix mindset to fixing the root causes. But I would say we are not there yet." (Interviewee 23).*

Thus, while the interviewee highlighted the importance of enough time for high-quality development, they also described that coming late into the development life cycle contributes to an emphasis on fast development. Other interviewees, while not emphasizing the importance of speedy development, highlighted time planning for development efforts, as illustrated by the quotes below by two QM.

> *"Well, what we always want is that we should start the projects earlier so that we have a little bit more time. […]." (Interviewee 1).*

> *"We are not allowed to or don't have time to do faults before we need to deliver. So when we start, it's almost too late to start. We are stating at the deliver day and then we force a lot to just put this through to the production." (Interviewee 21).*

The interviewees highlighted a wish for projects to start earlier to allow for more development time. A related perspective was provided by another interviewee, who described time-related project challenges.

> *"I can see sometime that we are in a hurry, and maybe not always been giving the right prerequisites from start. So, we have sometimes to work a little bit on different levels in parallel where we maybe should have had more things more things cleared out in an earlier phase […]." (Interviewee 26).*

The interviewee indicated that time pressure and not having the right prerequisites in place from the start can lead to parallel work on tasks that should have been completed earlier.

Some interviewees highlighted delivery dates and related challenges. One EM described this in terms of planning and transparency to mitigate risks, stating the following:

> *"[…] The planning is another thing, of course we cannot change always the delivery dates, but we have to be transparent about how to mitigate the risks and that is something about proactivity, we have to foresee what is coming and then if we can deliver one time. And then escalate from the very beginning if we don't have the resources for that delivery." (Interviewee 12).*

Another EM (Interviewee 16) described similar concerns related to delivery pressure. The EM noted that development takes time and that testing is needed to ensure quality. The EM also described occasional pressure from project leaders or line management to deliver on short notice, which can create unease about taking responsibility for quality when testing has not been as thorough as desired. In such cases, the EM described explaining the situation and outlining the risks so that higher management can decide on delivery and the associated risk.

A requirements developer highlighted how fault slip-through reports in relation to delivery dates affect development efforts, stating the following:

> *"Then those things have higher priorities and then we usually need to drop all the development of the new thing and go into the fault reports […]. Then the time plan for the new things kind of since the deadline never moves […]. So I should say that it's really that actually that we have lack of time actually. And then sometimes you need to speed up the reviews a little bit and maybe you don't review exactly as thorough as you should, actually." (Interviewee 27).*

As highlighted in the quote above, when fault slip-through reports are received, they increase the workload and can require teams to pause planned development to address faults. Since deadlines often remain unchanged, this can compress activities such as reviews, which may then be carried out faster and less thoroughly than intended.

When discussing code quality and code maintainability, one developer also highlighted that time pressure impacts the work. The interviewee described encountering code that lacked a good structure and had become complicated and noted that there was not enough time for refactoring.

> *"[…] You need to refactor it as well, just don't add another if statement, of course there is often a pressure regarding time from the outside, from the product owners, from the project […]." (Interviewee 14).*

One developer working with testing further highlighted a circumstance where developers may be under pressure due to short timelines. Within the development process, there is a stage (Z) by which all code from the various departments should have been delivered. By this stage, complete system tests can be performed, and Z is close to the release candidate final. If issues are detected at this point, it can be "mayhem" for developers to resolve the issues. This was identified as an issue, as illustrated by the quote below.

> *"[…] that is actually a problem we have identified that we need to have a big longer how to say? Stabilization period from when we are done until we actually release the code. […]. But we have realized that we see this as a big problem, that the complete system can be tested so late in the process." (Interviewee 24).*

Across the interviews, time and time pressure were expressed as recurring challenges in SW development, with interviewees describing different ways in which they affect development work, team priorities, and the ability to perform quality assurance activities as intended. One project sponsor described *"quality principles"* that the case company must have, setting clear rules for what quality related practices that cannot be skipped irrespective of time pressure in projects. Supporting this, a QM (Interviewee 6) highlighted that cost and time can constrain development efforts and may lead to "cutting corners," and suggested that stricter gates and automated gates could help mitigate this. Similarly, an EM described that delivery pressure can speed up the process and stated that quality assurance activities should not be compromised; however, this pressure has occasionally meant that *"software has been passed or jumped"* (Interviewee 12).

One EM summarized this by stating that there needs to be a balancing act and that it is not possible to achieve everything, as follows:

"[…] you cannot have time, quality and price matching, so you need to make some choices. […] you can always increase quality by increasing price, but you have a limited budget […]. 100% test coverage, yeah you can get there. It will take you and infinite amount of time or at least infinite compared to what we spend. So you need to make trade offs." (Interviewee 13).

4.4.5 Testing

Testing has been described across the interviews as both important and challenging. One recurring aspect was that the feedback loop from testing is not fast enough, as illustrated by the quote from an EM below.

> *"[…] It takes a long time for a developer to get feedback on his or her changes, it can take a couple of hours, including testing and so on […]. That would be the lower levels, and that is something that I believe is actually quite harmful for our operations. The sooner a developer can get feedback on change, the better it is, because then the developer can keep the context fresh in mind." (Interviewee 23).*

A similar perspective was raised by a PO (Interviewee 9), who highlighted that feedback loops are not sufficiently fast and that faster feedback would be beneficial for developers. The aspect Interviewee 23 highlighted, namely that delayed feedback affects a developer's ability to keep the context fresh, was also raised by a developer. The developer emphasized that the CI is not mature enough; while some testing exists, it is still reliant on manual activities.

> *"[…] The real feedback I received a month after I did the commit, […] I had to switch the context, read documentation again, go to requirements again, think what is wrong, understand it, provide a fix and perhaps wait another two weeks. I mean so that is challenging." (Interviewee 14).*

As previously highlighted, the case company encourages automation of repetitive tasks. Across the interviews automated testing was also emphasized as important. This is illustrated by the quotes below.

> *"So, I think the automated tests are really important to enable so many people to work together in a system." (Interviewee 23).*

> *"[…] the automated testing, I think is and really important thing to continue to increase the test coverage, because we adjust, we have a lot of variants also different modules […] And the only way to really be able to test that is to have automation, I think that is key." (Interviewee 26).*

This was also highlighted by a developer working with testing, who emphasized automation as important, stating the following.

> *"[…]. But the main focus for us is actually to automate as much as possible, because manual testing is too time-consuming. Just for your information we talk about thousands of tests executed." (Interviewee 24).*

Thus, while automation was described as a key focus, Interviewee 24 also described manual testing as too time-consuming. At the same time, some interviewees expressed that automated testing is still lacking, as illustrated by the quotes below.

> *"[…]. Even if testing is a bit automated, maybe verdicts are not automated, […]. So it's usually quite hard to test it on the SIL level in many cases is my experience. […] when it comes to testing, that's tricky and that's as we discussed that's a way to assure quality." (Interviewee 14).*

> *"What we are lacking at least on out part, is automated testing, […] because if you think about the amount of code lines that should really needed to be retested for every release, we have to go into automated testing more. We have that, but not for the entire code basis." (Interviewee 27).*

Both engineers working with SW development highlighted that the level of automated testing is not sufficient. Another developer working with testing further highlighted that the SW being developed is highly complex, making it very challenging to test everything. The developer also shared a perspective related to Interviewee 14's point about software-in-the-loop (SIL), stating the following:

> *"[…] what we miss in our team is more test environments […]. If we want to go lower lets say testing earlier phases, lets say we want to test the software in the loop. These are the environments that we don't have really developed. […] The main thing that I would say now that we are missing is to have proper software in the loop environment. If we had that we could have started to test much earlier in the project." (Interviewee 8).*

While Interviewee 8 (a developer working with testing) did not explicitly mention automated testing, the interviewee highlighted a lack of a SIL environment, which was described as an enabler for starting to test earlier in the project.

4.5 Continuous improvement

This theme presents how the case company's internal documentation and study participants described continuous improvement in SW development as implemented in the case company. The empirical findings from the interviews show varying opinions regarding continuous improvement and are further structured into 2 subthemes: (1) Metric and KPIs and (2) Retrospectives and lessons learned.

4.5.1 Metrics and KPIs

The case company describes SW quality KPIs as important to the organization at different levels to continuously measure and meet quality targets and other standards. They also highlight that the KPIs should be applied to the entire SW development processes from requirements through implementation, testing, and release. They also

explicitly describe the responsible personnel for the metrics at every phase of development, as well as suitable presentation templates (Case company Internal documentation, 2026).

Across the roles interviewed, participants described measurement and KPIs as a key mechanism to detecting deviations, but several also highlighted gaps in what is measured and how the metrics are used with concerns that they may not reflect real SW quality.

Engineering Managers from a management perspective highlighted a broad set of metrics that support SW development in their teams. Interviewee 23 and other EMs gave detailed descriptions of the measurements and monitoring being carried out in their teams, as quoted below;

> *"Yeah So we are measuring fault reports. We are measuring fault slip throughs, we are running [static code analysis] and we have been running [open source scanning] and so on for code quality scans and Yeah, so I guess we have plenty of different metrics to try to capture when we are starting to deviate too much." (Interviewee 23).*

> *"The other KPI actually is make making sure that we have 100% mapping of requirements and test cases. We also need to have 100% pass rate of those test cases to the requirements, their key, key metrics. So it's really understanding the documentation connected to that, which is the architecture design, the requirement and test case traceability metrics. [...]and following up continuously throughout the project, not just at the end of the project." (Interviewee 10)*

> *"[...] we have for unit test, we can have code coverage, we can see [static code analysis], that kind of stuff. We have test result from our tests, so you know a lot of such metrics we have [...]" (Interviewee 19)*

Despite the above descriptions and shared measurement focus by the engineering managers, Interviewee 10 also highlighted a drawback stating that - *"[...] we've had different KPIs and metrics, but it's really hard to keep high level metrics which are good on different levels [...] but it's something we need to be probably more structured [...]" (Interviewee 10)*

Another drawback was highlighted by an EM interviewee 19, who described the risk of focusing too much on the metric numbers instead of the actual effectivity of the process as illustrated below;

> *"[...]But what is the crucial point, at least for me, because you know you can prepare 100% coverage of the unit test, [...] But you know what we need to ensure how good this test case is, [...] not just on the metrics. Metrics is for*

the management, to have virtual insurance that we have good quality [...] it doesn't mean that we have good code." (Interviewee 19)

Across the engineering managers, they show a wide range of metrics covering traceability, coverage, and fault monitoring especially tied to release readiness while some expressed concerns over the validity of the metrics used.

From another management perspective, several quality managers described metrics as integrated within their dashboards, gates, and release criteria. QMs (Interviewee 6 and 20) highlighted dashboards they manage and monitor on different levels related to code and release quality as shown below;

"We have of course, yes, we have on different levels we have these dashboards. So for early stages we can have every team can see growing. So when they work, so when it will be released we should have like 100 percentage. And also we can have it on the higher level this metrics and also we have 0 bugs with critical 0 critical bugs when we deliver [...]" (Interviewee 20)

Interviewee 20 also described metric targets related to the test coverage of requirements, highlighting - "*[...] we should have 95% complete requirements fulfillment [...]" (Interviewee 20)*

"We have some dashboards. of course, we're following the progress [...] It's rarely than we're exceeding the expectations. [...]The most difficult thing is to understand what to measure exactly to show the trend and explain to the team what they have to deliver exactly." (Interviewee 6)

Interviewee 6 further discussed that they try to maintain the same working approach that yielded good results and trends, but making necessary changes only when bad.

Furthermore, other QMs (Interviewee 21 and 3) also described similar metric targets related to requirements, release quality, and overall project quality. With some examples illustrated below;

"[...] a metric that at least each requirement needs to have at least one test case, [...]And then also when we talk about code coverage, statement coverage, that's also good measurement to see how much of the software we are testing. Now it's mostly connected to safety that we require that you need to have 100% code coverage [...]" (Interviewee 21)

"[...] we have this FST where you have many faults report, you must do the analysis, and you have to close. So every analysis goes with an improvement activity in our SharePoint or in their project storage [...]" (Interviewee 3)

"The main KPI if you see from quality perspective is the NCR nonconformances. it's the rating which we give to the team when they are not

fulfilling some criteria [...] if you have more than five NCR then your project is at risk [...] So to keep it track so that we improve for that development team [...]" (Interview 3)

Across the QM interviews, they highlighted consistent metrics and targets set within the teams monitored through dashboards and gates. Defects and trends are analyzed to improve the development processes as captured by interviewee 3 stating "*[...] you must do the analysis, and you have to close. So every analysis goes with an improvement activity in our SharePoint or in their project storage [...]"*

Similarly, a PO (Interviewee 18) describes clear metrics and measurements - "[...] *There are a few metrics that we use mainly the requirement coverage, and the test coverage are something that we get every day through a regression, automated regression testing [...]" (Interviewee 18)*

Interviewee 18 also followed up with actions they take when they notice negative trends, stating that - "*[...] just freezing further integration and just focusing more on the fault tracing before this trend escalates to a stage where it is really difficult to either trace back or to fix anymore."*

In contrast, two POs (Interviewee 7 and 9) described limited use of metrics, highlighting gaps in how much is tracked and potential improvements, illustrated in quotes below;

Interviewee 7 - "Not more that we had and we have been focusing on the suppliers progress in relation to the official releases and so we should probably have had more. As I said, the metrics in the quality plan and we should follow up more things than we have done [...] So yeah, we do some, but we should have done more and we should do. We will do more in the future."

Interviewee 9 - "We actually don't use that much metrics [...]There are also some other metrics like things, but in my view it's mostly managers that are interested in metrics to follow up different KPIs and things like that and we provide some data for that [...]"

The POs show contrasting opinions of metrics and KPIs; while some mentioned clear use cases, others describe them as less important and more of a managerial responsibility.

Similar to the POs, the operational roles (Requirement developers and SW developers) also highlighted contrasting opinions of metrics and KPIs. Requirement developers (Interviewee 27 and 17) described metrics as mostly testing-focused with limited indicators for requirement quality with illustrative quotes below;

"Not specifically on the requirements, the KPIs and those things you have are on the test. There is something in [...] called requirement verdict, and of

course there is a KPI for that as well, but I don't think people really know what requirement verdict means, so many testers just set OK to that, or we also do that because we don't really know what is it should check." (Interviewee 27)

"Not really. We keep track of requirements versus testing and testing versus test verdicts but we don't really keep track of requirements per say [...]So the way we check if something is implemented, let's say is by testing, So that's how we close the loop." (Interviewee 17)

While SW developer (Interviewee 14) described monitoring of non-functional attributes such as CPU and memory usage -*"Yes, I mean this I would say monitored continuously like all CPU usage per process, memory usage per process. We have charts and like this is also done in quite good way." (Interviewee 14)*

Interviewee 14 also highlighted the targets related to line coverage and code quality, sharing their reservations to the effectiveness of the tests, noting the need for scenario coverage as well with quotes below.

"And at the least for line coverage at [case company] we aim towards 100% [...] about code quality, some metrics like I mean if there are any compiler warnings, any static analysis warnings. [...]" (Interviewee 14)

"[...] also consider all scenarios already on unit test level. Consider corner cases, [...] I see that in most projects the code coverage is measured and it's like gating factor. So you can't merge code without covering it with unit. But I mean writing test just to like increase coverage is a wrong idea. You need to think about your scenarios." (Interviewee 14)

Interviewee 22, being a developer, highlighted certain metrics used as indicated by; *"So yeah, those are the things. And also we tried to, I mean in the static code analysis, I think there is some complexity metrics out there, cyclomatic complexity [...]"* . Neither interviewee 4 nor 14 also developers, mentioned any metrics for complexity.

Furthermore. SW developers working with testing highlighted no clear or specific KPI, targets or measurement, but they described some kind of monitoring of the SW [*product*] quality. Interviewee 24 and 8 (both developers working with testing) shared the illustrated quotes below.

"*But we don't set any KPI [...]But we try to follow up, especially what is important for us is how many fails have we found and where are they [...] Not really measured, but we have the way of working to secure that the test cases are robust, maybe a good idea by the way, but we should, put some KPIs on that." (Interviewee 24)*

Interviewee 8 -"We run nightly jobs, daily regression every single night [...] we have even our own web infrastructure team where we can monitor the data, we get statistics, trends, patterns, everything that we need [...] The reason of course behind this is that every night we need to run a lot of test cases to get a confidence in the like the daily software."

Across the roles, most metrics and KPIs discussed touch upon requirement quality, code quality and testing with several distinctions between the management and operational roles. However, many quotes also indicate gaps, tensions across various levels, and potential improvements to implementation.

4.5.2 Retrospective meeting & lesson learned

Across the roles, many participants described clear continuous improvement through reflection activities like retrospectives, lesson learned, and defect management.

Several EMs highlight a clear learning culture within their teams; they describe this from both a defect-handling perspective and an end-of-project perspective. Some illustrative quotes and shown below;

"[...]So finding a bug in my world, is a very good thing, it's nothing bad about it and then actually we try to celebrate this. [...]And then every release we are trained to actually first to try to celebrate what we actually achieved. Second, we are trained to do the lessons learned because even the successful release you have something that you can actually improve." (Interviewee 12)

"You know we have at the end of this sprint we have retrospective, we discuss during the stand up the issues we discovered and with other stakeholders to see what we need and what to avoid, yeah, and you know, quality takes time[...]." (Interviewee 19)

"Depends a little bit on what kind of issue, but for all fault reports we are requested to do a fault slip through analysis and that is something that is on the teams and they do that and in that process they need to define what needs to be done to avoid this issue from slipping through in the future. That could mean architectural changes or it could be adding test cases or process changes." (interviewee 23)

In sharp contrast, one EM (Interviewee 10) described a lack of consolidated learning frameworks, highlighting *"a lot of ad hoc work or one-time efforts,"* driven by immediate task forces and *"one-off activities"* that are "*hard to quantify."* They shared that the missing piece was a consistent structure that prevents repeating the same reactive work and helps the team retain knowledge over time. As they put it, *"what we need to do more is regular weekly educations or demos [...] teams may*

change, people may come or go, but we need to keep everyone up to date and educated [...]".

From a quality management perspective, One QM (Interviewee 3) described a clear learning frameworks within their team stating that - "*we have a SharePoint page where we add all the lesson learns and during the review we just go through that so [...] So for every project, yeah before they did release, yeah." (Interviewee 3)*

However, another QM (Interviewee 1) from another department did not share similar learning frameworks rather described a loss of previous learning framework, with illustrative quote shown below

> *"We used to have white books, but have lost that somehow. But then it's for the way you're working, it's the same for. It doesn't matter what SW product you're working with you should do the same thing [...] we're trying to have the company memory, not only from one person." (Interviewee 1)*

A similar lack of structured learning framework and loss of a previous learning framework was highlighted by a SW Architect (Interview 17) stating that "*[...] we had a process before called Lessons Learned. It was a tooling in SharePoint. We filed some things like 10 years ago in there and it's no longer available. So when we would like to use this learnings now again, we couldn't find it [...]" (Interviewee 17)*

Furthermore, the POs who are closer to the development teams described learning and improvement clearly as both trend-driven and defect-based, as well as having structured learning frameworks within the development efforts. One PO (Interviewee 26) shared the quotes below

> *"for issues where we think that this is something that we need to learn from, then we create the so-called FST, fault slip through and that leads to that we create some story in our [backlog tool], so that we can put it in the backlog and bring it up when time and priority allows depending on how important we think that this issue was. So that's more like to keep track on it and not forget it." (Interviewee 26)*

They further highlighted their structured learning framework integrated within their development structure

> *"[...] every end of the [development increment] we have this kind of retrospective action. [...] you will clearly see that this kind of actions for example that we did they really contributed to increase our test coverage, for example, or to increase our quality on our master branch, et cetera [...] the retrospectives are done on a small scale every second week per Sprint, and then it's done on a larger scale where more teams, more teams meet to discuss on a PI scale, so every 12 weeks." (Interviewee 26)*

In another case, learning was described by a PO as even more formal. Interviewee 7 described an "*extensive lessons learned*" after a major product change, where the team invested a lot of time to produce "*a formal report,*" and turning findings into "*recommended actions.*"

However, not all POs felt learning was equally systematized: Interviewee 9 admitted they mostly "*try to act on the current situation*," responding only when trends worsen, but they noted a gap or "*weak point*" in linking their work to any measurable improvement stating that *"maybe we are not that good at relating changes towards actual improvements[…]that's maybe a weak point for the moment*."*(Interviewee 9)*

From a SW developer perspective, learning was often described as connected to implementation and defects. A SW developer (Interviewee 14) also highlight this defect-based learning stating that *"[...] whenever there is a fault report, that was caught too late, maybe by customer, maybe in the [vehicle] when it should have been found earlier, we do fault slip through analysis, so we learn from that [...]"* They also described their structured learning as quoted below;

> *"[...] we have these retrospective meetings when we discuss especially the things that didn't work that well. We introduced some actions based on the outcomes from these meetings. [...]" (Interviewee 14)*

In sharp contrast, another developer from another team described a lack of structured learning, only personal - *"Basically some notes like in one note for myself." (Interviewee 4)*

Another developer working with testing (Interviewee 24) shared similar structured learning process as interviewee 14, describing *"continuous lessons learned"* across sprints - *"at least every second Sprint"* and a bigger reflection point at the "*regular sync,*" where they reflect and discuss "*what could we do better and what should we stop doing." (Interviewee 24)*

Finally, another SW developer working with testing (Interviewee 8) described learning as more of an evolution of processes through continuous improvement of practices over the years with illustrative quotes below;

> *"[...] we are not stuck in our ways. We always try to improve ourselves and find best practices, which is also a case in point like when we started our project, let's say not 5-6 years ago, we work in a completely different way today than we did six years ago, a lot due to new investments and actually finding best practices." (Interviewee 8)*

Across the interviews, continuous improvement was described through structured learning frameworks referred to as retrospectives after each sprint as well as defect-based learnings through the highly mentioned *fault slip-through*. While most participants highlighted these aspects, many also described that learning remains

informal, can be hard to institutionalize, and learning tools can also be lost as attention may move to finding immediate solutions.

5 Discussion

This chapter discusses and analyzes the three research questions, based on the results and previous research. Section 5.1 addresses RQ1 aimed at highlighting how SW quality is defined within the context of automotive SW development. Section 5.2 addresses RQ2 explaining what practices regarding quality management are of importance in delivering quality automotive SW. Lastly, section 5.3 addresses RQ3, highlighting factors that impact SW development and quality.

5.1 Definition of SW quality

This section discusses how SW quality is defined and perceived within the case company, by comparing the empirical findings with previous research and identifying alignments and contradictions with the theoretical framework.

The findings suggest that there is no single shared definition of SW quality within the case company. Instead, interviewees express varying perspectives, where both similarities and differences are evident. This aligns with previous research on quality in general and within the SW context, which highlights that quality is inherently difficult to define (De Giovanni, 2023; Solin & Curry, 2022; Green et al., 2024).

Requirements

The findings indicate that requirements are a central concept when discussing SW quality, particularly in relation to the functionality, realization, and testing/verification of the SW. A possible explanation for the strong emphasis on requirements, is their link to acceptance and qualification testing, as described by Pavlíčková et al. (2024). In this sense, requirements are the basis for evaluating whether the SW fulfills its intended functionality and purpose. Garvin's highlight c*onformance,* defined as a product's ability to meet specified standards, as one dimension of quality (De Giovanni, 2023). Therefore, this is also applicable within the context of automotive SW development.

Secondly, requirements translate customer needs into technical specifications (ISO/IEC/IEEE 12207:2017) and are important in ensuring that the necessary quality attributes are realized in the SW (Abrahão et al., 2024). This may explain the emphasis in the findings on well-developed requirements, as they are perceived as a primary means of ensuring that SW meets both customer expectations and required quality attributes. This interpretation aligns with Nabot and Al-Qerem (2025), who define SW quality as a system's ability to function as intended while meeting customer needs through quality attributes.

As highlighted in the findings, requirements were discussed in relation to functionality, traceability, testing, and verification. Some interviewees also referred to ASIL requirements and the need to fulfill safety and security requirements in the

automotive context. This suggests that requirements are central not only for defining SW functionality and purpose, but also for ensuring that the SW can be developed and assessed in a structured and justifiable way in accordance with specific standards. This aligns with the theory, which emphasizes that automotive SW development is governed by standards and regulatory frameworks that require documented, traceable, and verifiable development processes (ISO 26262, 2018; ISO/SAE 21434, 2021; UNECE R156, 2021).

Quality attributes

As shown in Section 4.2, interviewees typically emphasized multiple quality attributes without clearly ranking them. This suggests that automotive SW quality is inherently multidimensional, where certain attributes such as safety, robustness and security are more prominent, but not sufficient on their own to define overall SW quality. This interpretation aligns with Nabot and Al-Qerem (2025), who conceptualize SW quality as comprising multiple attributes rather than a single defining characteristic.

The emphasis on safety and security was often linked to the need to adhere to regulatory standards such as ISO 26262 (2018) for functional safety and ISO/SAE 21434 (2021) for cybersecurity. Within the context of the case company, compliance with these standards is not optional but a prerequisite for delivering SW, which helps explain why these attributes are prioritized. This aligns with Li et al. (2024), who highlight the necessity of fulfilling quality obligations and regulatory standards in automotive SW development.
Beyond compliance, the prominence of safety and security can also be interpreted in relation to customer and stakeholder trust. ISO 9000 (2015) emphasizes customer focus, including the importance of ensuring confidence from customers and stakeholders in the organization. In this context, considering the potentially severe consequences of SW malfunction, it becomes critical to ensure such trust and confidence among customers and external stakeholders in the delivered SW.

The findings highlight robustness and reliability as important quality attributes in relation to customer experience. Robustness was described as the system's ability to handle errors, bugs, and unexpected situations, aligning with Shah et al. (2016), who define robustness as a system's capability to function under stressful or unforeseen conditions. However, a few interviewees also associated robustness with the ability to withstand cyberattacks. Considering the need for cybersecurity compliance in automotive as defined by ISO 21434 (2021), robustness is thereby important for the SW to handle these unexpected situations. While most of the interviewees discuss robustness in relation to customer experience only, other aspects of robustness of automotive SW should be considered. This further indicates the multidimensional impacts of quality attributes of the SW.

Reliability was commonly described as the expectation that the SW should function without unintended behavior or malfunction. This is partly aligned with ISO 25010

(2023), which defines reliability as a system's capability to operate under specified conditions for a defined period without failure. A similar definition for reliability is provided by Garvin (De Giovanni, 2023). While the findings do not explicitly emphasize these formal aspects, they instead highlight the absence of malfunction as the central concern. The authors argue that, in the automotive context, preventing malfunction is perceived as more critical than defining operational conditions or timeframes, due to the potentially severe consequences of SW failure.

Nabot and Al-Qerem (2025) highlight the role of quality attributes in meeting customer expectations. However, they do not indicate which attributes are most influential in shaping customer experience. The authors argue that the emphasis on reliability and robustness can be understood in relation to fundamental user expectations. In this sense, these attributes become central to customer experience, as they reflect customers' minimum expectations for automotive SW to be trustworthy and dependable.

Nabot and Al-Qerem (2025) and Green et al. (2024), although not within the automotive context, include usability as one quality attribute used to assess overall SW quality and shapes customers' perception of product quality. However, the findings of this study do not identify usability, or similar terminology, when defining SW quality.
While some interviewees referred to user experience or described SW quality in terms of meeting customer expectations and standards, these perspectives were not expressed in line with formal definitions of usability. For example, ISO 25010 (2023) defines usability as the extent to which users can interact with a system effectively, efficiently, and with satisfaction. Such interpretations were not evident in the empirical material.

A possible explanation for this absence is that many of the interviewees develop SW that are not the most customer-facing. While some may contribute to customer-facing SW, their work is often focused on underlying SW functionality that users do not interact with directly, but which enables the overall product. Consequently, greater emphasis is placed on attributes such as reliability and robustness as indirect means of influencing customer experience, rather than on usability itself. However, this does not imply that usability is unimportant in automotive SW development; rather, it indicates that it was not strongly reflected in the perspectives captured in this study.

Role-based perspectives on SW quality

Green et al. (2024) highlight that various roles may hold different interpretations of what SW quality entails. This was evident in the interviews with developers writing code, where code quality and design aspects, such as readability, structure, and maintainability, were central concepts in relation to SW quality. Code quality is a key component in SW development, where readability and structure directly impact code quality (Börstler et al., 2023). This aligns with the developer's perspective of what

code quality entails, while also including maintainability, which Green et al. (2024) highlight as an “overarching” characteristic encompassing aspects such as readability and complexity.
While other roles also discussed code quality, their perspectives were not directly tied to SW quality in the same way, nor did they emphasize similar aspects as developers. The authors argue that this difference can be explained by the roles' responsibilities, where developers, as those who write the code, focus on its internal properties, while for example roles such as EM may adopt a broader project-level perspective.

The findings also highlight a limited focus on maintainability in relation to enabling updates and correcting errors within the SW. This aligns with ISO 25010 (2023), which defines maintainability as the degree to which SW can be effectively and efficiently modified by SW developers. Considering the complexity of the SW described by the interviewees, an increased focus on maintainability may be beneficial for the case company. This is further supported by Green et al. (2024), who highlight that improved readability and maintainability positively influence both developer efficiency and overall product quality.

Summary/Conclusion

To conclude, the findings indicate that there is no single, shared definition of SW quality within the automotive context; it is best understood as a multidimensional and context-dependent concept.

Requirements were identified as a key component in discussions of SW quality. The authors therefore conclude that requirements are central, as they are important for assessing whether the SW fulfills its intended purpose. They also translate customer needs into functionality and serve as a means of achieving quality attributes.

Although some quality attributes may be more prominent than others, the findings indicate that these alone are not sufficient to define overall SW quality. Context also plays an important role in determining which attributes are emphasized. For SW developed for safety-critical applications or exposed to cybersecurity risks, attributes such as safety and security were highlighted. In contrast, attributes such as robustness and reliability were emphasized in relation to customer or user experience.

While a role-based perspective was not strongly evident in the data, some examples did emerge. In particular, developers involved in writing code emphasized maintainability as an important attribute. This further strengthens the conclusion that both the definition of SW quality and the importance of specific quality attributes are context dependent.

5.2 Quality management in SW development

The second RQ investigates what SW quality management practices are important to deliver quality SW products. The empirical findings from the case company shows that quality management in SW development is carried out through the combination of several processes and practices related to QA and QC. The findings revealed practices such as reviews, audits, assessments, testing, gates, traceability, defect management and continuous improvement. These practices reflect the theoretical idea that SQM is a combination of a set of organizational, managerial, and technical activities that ensure SW quality is defined, governed, and improved throughout the SW life cycle (ISO/IEC/IEEE 12207, 2017).

SQA vs SQC

Breaking down SQM, a key finding is that the case company performs several practices that are understood as part of both SQA and SQC. However, both concepts were not always clearly distinguished by the interviewees. This indicates an ambiguity in the quality language used in the case company and implies a misalignment with contemporary quality management theory as SQA and SQC have been described as having different but complementary roles. ISO/IEC/IEEE 12207 (2017) and Guamán et al. (2020) described SQA and SQC respectively, with SQA described as proactive and process-centered while SQC is operational and output-focused, evaluating the SW outcomes.

Despite some of the interviewees not using the terminologies in the right way, they still described important quality practices including reviews, tests, gates, checklists, verification, and documentation. However, some other participants did make clear and correct distinctions between SQA and SQC, highlighting important practices in both aspects. In practice, a review or test may be meaningful and serving its intended purpose even if the employees do not classify it as SQA or SQC, suggesting that SQM is practiced more as a set of activities than as a shared management approach. This contradicts ISO 9000 (2015), which emphasizes that the importance of a unified process approach and shared understanding in a quality management system. This lack of a shared conceptual understanding creates difficulties in aligning responsibilities, communicating quality expectations and improving processes.

While most interviewees either highlighted a distinction or a lack of distinction between both SQA and SQC, one interviewee expressed an interesting aspect by describing SQC as being defined by SQA. They expressed that SQA first decomposes the process and system, defining what should be done, how it should be done, and the strategy of doing it, while SQC performs the control and verification activities on the process and system outcomes. This aligns with Guamán et al. (2020) confirming that SQA and SQC should not be understood as separate practices but as mutually dependent parts of SQM, playing important roles in ensuring high quality outcomes. The author concludes that the theoretical distinction between QA and QC, together

with the ambiguity surrounding what these concepts entail in some cases, may pose a risk to improvement efforts. If SQA is intended to provide confidence that quality requirements will be met, but is instead interpreted as increased control, there is a risk that the necessary proactivity in the processes will not be achieved

Reviews

Reviews appear to be one of the important quality practices expressed in the empirical findings, reviews across the various SW development stages as essential for ensuring high quality. This aligns with ISO/IEC/IEEE 12207 (2017) which emphasizes reviews across all the development stages as an integral part of SW quality. The findings highlight that reviews should already start with requirements and continue through the architecture, implementation and into the test cases. The findings also highlight the reviews of various SW work products to ensure consistency. Reviews were also described as "*base cornerstone*", emphasizing that reviews ensure that the test cases actually test what they should.

As highlighted in Section 4.2 & 4.4.1, reviews serve different purposes for example including supporting collaboration and compliance to regulatory requirements. However, the findings also show that reviews are not sufficient alone to ensure SW quality. While reviews provide necessary human judgement, their effectiveness depends on their responsiveness, review coverage, participation and reviewer expertise as highlighted by McIntosh et al. (2016). Kudrjavets and Rastogi (2024) also describe modern code reviews as time-consuming. This introduces an important tension for SW quality management as the expert evaluations provided by reviews remain necessary in this complex automotive context hence the empirical findings and theory suggest complementing reviews with gates.

Gates

Gates and automated gates were discussed across the findings as important to SW quality. Gates are more objective with clearer go/no-go requirements, requiring specific criteria to be met before SW can proceed to the next stage. Unlike reviews, where judgement is subjective, and "*human driven"*, gates are expressed as mandatory, and difficult to bypass. Therefore, gates function as formalized control points in the development process, reducing ambiguity, and supporting process discipline as highlighted by Wessel et al. (2022). This also corresponds with ISO 9000's (2015) principle of evidence-based decision making, where decisions made based on evaluated data are more objective and increase the likelihood of achieving desired results.

At the same time, the empirical findings and theory do not suggest that gates should replace reviews, they complement each other within the quality management system. The authors argue that reviews are better suited to situations where interpretation and system intent must be evaluated, for example, evaluating whether a requirement is

clearly written and understandable, or whether a test case covers the right system behaviours, both requiring expert human judgement. However, gates are better to check predefined and measurable criteria such as whether coverage targets are reached, or whether a test passes before allowing development to proceed. Therefore, this indicates that SW quality depends on the efficient balance between expert human judgement and automated objectivity.

Testing

Testing was another SQM practice widely discussed in the empirical findings. The interviewees described several forms of testing, for example unit testing, integration testing, SIL testing, and system-level testing. Testing was discussed as necessary to evaluate SW behavior from code level up to product level, to confirm that the requirements are fulfilled, and that changes do not break any existing functionality or introduce new defects. This aligns with the V-model logic, as development activities are connected to corresponding verification and validation activities, SW requirements are linked to qualification testing, architectural design to integration testing, and implementation to unit testing (Pavlíčková et al., 2024). Considering the automotive context, the SW needs to be verified throughout the development stages, as defects in released SW could cause potential safety risks. This aligns with Maro et al. (2018) emphasis on the importance of the V model within automotive.

Therefore, testing is not only a late-stage activity, but it serves as both a control and a feedback mechanism. Several interviewees highlighted the importance of detecting issues earlier. This is connected to the case company's expressed emphasis on built-in quality, where quality should be created and sustained throughout the development process instead of only at the end. This also aligns with Zhao et al. (2025), who highlights the potential cost benefits of a robust SQA as it enables earlier identification of issues. Therefore, the various testing practices in the case company are understood as SQC rather than SQA, since their primary function is to detect defects throughout the SDLC and provide a basis for improving the SW. Aligning with how Guamán et al. (2020) explains the purpose of SQC. However, the authors argue that these practices can also indirectly contribute to SQA, particularly when recurring defects can be traced to specific development stages. This creates opportunities not only to correct faults, but also to identify weaknesses in earlier process activities and target them for improvement.

Traceability

Traceability is another SQM practice frequently discussed in the empirical findings. Several interviewees highlighted the importance of traceability within the SW development, highlighting the need for linking requirements, to architecture, to implementation and testing, highlighting continuous follow-up of requirements and test cases throughout the SDLC. This aligns strongly with Maro et al., (2018) and Habibullah et al., (2024), who emphasize traceability is essential due to system

complexity to maintain coherence in the SDLC. One QM also pointed out a key understanding, defining QA as constant monitoring and providing full traceability and confidence in the SW and related work product. Aligning with V-model and ASPICE requirements that support SDLC consistency and bidirectional traceability, necessary in regulated and safety-critical contexts (Pavlíčková et al., 2024; VDA QMC, 2023). Therefore, traceability is a practical SQM mechanism that supports verification, release readiness, and regulatory compliance

The importance of traceability related to safety, cybersecurity, and automotive standards was also highlighted in empirical findings and supported by Maro et al. (2018), who state that these standards require proper traceability to verify the safety of the resulting SW and systems. While the maturity of traceability in the case company has not been explicitly stated in the results, some interviewees did point out gaps between implementation, testing and requirements which may indicate weak traceability. This is particularly important in the automotive context as strong traceability ensures all development stages are connected, meeting compliance requirements, and justifying SW quality. Otherwise, when weak, uncertainty increases, root cause analysis becomes harder, hindering potential improvements and learning. It also becomes difficult to evaluate the maturity of the SW development process as comparable process outcome evidence cannot be generated as described by ASPICE (VDA QMC, 2023).

Continuous Improvement

Continuous improvement practices also form an important part of SQM. Across the interviews, various aspects of continuous improvement were highlighted such as metrics, KPIs, audits, assessments, defect management, retrospectives, and lessons learned. This corresponds with the improvement principles in ISO 9000 (2015), which emphasizes that organizations should continually improve processes to respond to risks and opportunities. The findings also show that case companies try to learn from defects instead of only detecting them. For example, the fault slip-through analysis was frequently mentioned across the interviews as important; it investigates why a fault or defect was not detected earlier and what should be improved to prevent it from reoccurring. However, this appears to be more reactive, rather than trying to continuously improve the processes.

Nevertheless, the findings also show that continuous improvement is not equally institutionalized across the case company. While some interviewees described structured retrospective and formal lessons learned frameworks, others described more informal learning, personal notes, one-time efforts, and even lost learning systems. This suggests that learning and improvement activities exist, but not commonly shared, and organizational retention is inconsistent, thereby indicating a misalignment with the improvement principles in ISO 9000 (2015). This is particularly important from an SQM perspective, because continuous improvement depends not only on collecting data and identifying lessons, but on making sure those

lessons are captured, shared, and implemented to improve future projects. If learning is not institutionalized, departments may repeatedly solve similar problems locally without contributing knowledge to a shared organizational memory.

Metrics and KPIs

Metrics and KPIs were described as important tools for managing SW quality. The findings discuss metrics used to monitor requirement quality, code quality, testing and release readiness. With indicators mentioned such as requirement coverage, test coverage, code coverage, test pass rates, fault slip-through, non-conformance rates, CPU/memory usage, and regression results, etc. These metrics and indicators align with ISO 9000's (2015), who highlight evidence-based decision making which helps organizations improve development processes and SW quality.

However, the findings also reveal critical views on metrics. Some interviewees highlighted that metrics could give a misleading picture of SW quality. One developer pointed out that high code coverage does not necessarily mean the tests cover important scenarios and corner cases of the SW function. Furthermore, challenges were highlighted regarding the use of KPIs/metrics applicable at different organizational levels. In line with the theory presented by Izurieta et al. (2024), the authors identify the value of measurement proxies in making KPIs/metrics useful across these different levels of the organization. In addition, one developer highlighted the use of cyclomatic complexity, while others did not, which indicates inconsistencies in the metrics being used. This indicates that several metrics are monitored in the case company, but it is not clear if the data represents meaningful aspects of SW quality. Also, this implies that if metrics only measure activity rather than effectiveness of the activity, they may create false confidence without necessarily improving SW quality.

Audits and assessments

Audits and assessments also appear as SQM practices related to process maturity, compliance, and continuous improvement. Several interviewees mentioned audits, process reviews, and ASPICE assessments. These activities correspond to the evaluation functions of a quality management system; objectively evaluating processes, identifying nonconformities, and recommending improvements (ISO 9000, 2015). ASPICE, which is an automotive industry recognized assessment framework, was also highlighted across the interviews and was also observed to be currently implemented across the case company. This indicates that audits and assessment provide an external and objective view on whether processes are followed, managed, and capable of achieving their intended outcomes with applicable evidence.

However, the empirical findings suggest that audits and assessments are not experienced equally across the case company. Leadership and process governance roles like quality managers appeared to be more aware of audits, assessments, and

related nonconformances, while operational roles like developers emphasized reviews, testing, and gates related to their daily work. This indicates that the value of these formal practices to everyday work should be expressed clearly, especially to the operational roles, to encourage participation and implementation of potential improvements.

Summary/Conclusion

Overall, the empirical findings show that SQM at the case company relies on key practices including reviews, gates, testing, traceability, continuous improvement, and its related activities. The findings also highlight several challenges related to these practices, such as ambiguity between SQA and SQC, potential gaps in traceability, limitations in the metrics, and uneven continuous improvement. While these key practices largely align with SQM theory, and automotive industry standards, SW quality requires more than only having them formally in place. Their effectiveness depends on how they are consistently understood, integrated, and efficiently applied across the SDLC. When practices are effectively connected, they provide confidence that the SW has been developed, verified, controlled, and improved in a systematic way. Therefore, the synthesis in this section hinges on the understanding that SQM is the efficient interaction of a whole chain of people, processes, tools, and practices towards the shared purpose of ensuring SW quality. This aligns with Green et al.'s (2024) description of the relationship between process, code and system in ensuring SW quality.

5.3 Factors that affect SW development and quality

This section highlights how factors impact SW development and quality. Comparing the findings with previous research to identify alignments and contradictions with the theory.

5.3.1 Process focus

The findings indicate that a strong process focus is perceived as essential for ensuring SW quality. Interviewees emphasized the need for clear frameworks and standards to guide development work. Process-related activities such as reviews, gates, and testing were highlighted across interviews as critical practices, as discussed in Section 5.2. Two interviewees (7 and 21) highlighted in Section 4.4.1 a chain perspective where, processes are interconnected and important to achieve SW quality. The authors conclude that while most have highlighted specific process activities, these alone aren't sufficient to achieve SW quality, but rather this chain perspective comes of importance as SW development needs consideration of several activities throughout the development life cycle. This aligns with the case company's internal documentation, which promotes a "built-in quality" strategy that prioritizes process-oriented approaches over end-stage evaluation. Similar arguments are presented by Green et al. (2024), who identify process quality as a key factor in achieving SW

quality. Based on this, the authors conclude that well-defined and effectively implemented processes are fundamental to increasing the likelihood of achieving high SW quality.

However, the findings also highlight challenges related to processes and their alignment. Interviewees expressed that aligning processes across the case company is difficult. As highlighted in Section 4.3, four main sources of process guidance exist, with the majority referring to locally developed frameworks. Furthermore, most interviewees rely on a single primary source for process guidance.
The authors identify the use of multiple sources for process guidance as a risk that may lead to fragmented practices and ways of working. Giachetti et al. (2024) and Moe et al. (2021) similarly highlight process fragmentation as a quality risk, emphasizing that SQM depends on coherence across development processes. As indicated in *Figure 4.6,* some roles rely heavily on locally developed frameworks for guidance. One interviewee (8) noted that when internal guidelines are unclear they are updated. While this flexibility may improve clarity in the short term, it also introduces risks, as local process changes may not align with centralized practices, thereby increasing fragmentation. This aligns with Moe et al. (2021), who argue that excessive team autonomy, without sufficient alignment with central governance, can result in inconsistent processes and increased coordination efforts. This concern is further reflected by a quality manager, who highlighted the lack of control over locally developed frameworks as problematic, making it difficult to ensure the correctness and reliability of the information.

A potential explanation for this fragmentation, particularly regarding the use of different sources for process guidance, emerges from the findings. The centrally provided frameworks, often perceived as “higher-level,” are sometimes too generic and not sufficiently useful for operational roles. This appears to drive reliance on locally developed frameworks. This can be interpreted through Boiral (2007), who explains that process fragmentation may occur when centrally defined processes are not fully understood and when employees are not actively involved in their implementation.

In contrast, locally developed frameworks are often tailored to the specific context of the teams, making them more relevant and practical for daily work. However, the case company's internal documentation highlighted a need for consideration of the need for unique local processes, and that there is potential for standardization. In addition, the data also highlights uncertainty regarding how updated and reliable the central process guidance is, as well as difficulties in locating information. These frameworks were described as “messy” and “confusing,” with new sources of information/frameworks introduced without closing old ones. This may further reinforce trust in, and reliance on, locally developed process guidance.

Despite this, some interviewees still expressed the use of central process guidelines either exclusively or in combination with locally developed guidance. They perceived

the central process guidelines as reliable and up to date, although limitations in user-friendliness were still highlighted. Additionally, some other interviewees also emphasized the benefits of central process guidance in relation to compliance with standards such as ISO and ASPICE. This may contribute to a sense of confidence among employees, as adherence to central process guidance can be perceived as ensuring that necessary standards and requirements are fulfilled.

Lastly, although not evident across all interviewees, two engineering managers (Interviewees 10 and 15) emphasized the importance of aligning locally developed process guidance with central process guidance. Similarly, Interviewees 10 and 12 highlighted the need for locally developed processes to adhere to standards such as ISO and ASPICE. While there is an emphasis on the alignment of locally developed processes with standards and the central process guidance, this was not evident across the interviews. The authors identify a key strength in aligning the way of working in the organization, which can enhance consistency across the development process. This aligns with the ISO 9000 (2015) process approach, which emphasizes the importance of understanding processes and their interactions to achieve consistent and predictable results. Such an approach also facilitates the identification of improvement opportunities and supports the enhancement of overall system performance.

Conclusion/summary

The authors conclude that the reliance on locally developed frameworks represents a potential source of process fragmentation. Due to employees referencing different sources for process guidance across the organization, potentially working in a less streamlined or standardized way. As highlighted in the literature, such fragmentation constitutes a significant quality risk, as SQM depends on coherence across development processes (Moe et al., 2021; Giachetti et al., 2024). This risk is further reinforced by the limited clarity in the findings regarding the extent to which locally developed frameworks must adhere to central process frameworks. The identified flexibility to adjust processes based on unclear or incomplete central guidance may improve local usability, but it also risks reducing process standardization across the organization. Furthermore, personnel may reference different sources thereby potentially not aligning in how development should be performed.

While ISO 9000 (2015) highlights that context-specific aspects influence QMS, it also emphasizes the importance of clearly defining processes and activities to enable measurement. This may be difficult to achieve if processes are not aligned across the organization, which is also reflected in the challenges related to metrics and KPIs (see Section 5.2). The authors acknowledge that there may be a need for local adaptation, as teams do not necessarily develop similar SW. However, concluded that it is of importance that such adaptations adhere to centrally defined case company standards, particularly considering the impact of process quality on overall SW quality (Green et al., 2024).

5.3.2 Requirements.

As discussed in Section 5.1, the findings highlight the importance of SW requirements, particularly in relation to their impact on SW quality. However, several challenges were also identified.

Two requirement developers emphasized that requirements should be clear and understandable, which is also highlighted by ISO 12207 (2017) as a characteristic of good SW requirements. However, the findings indicate challenges related to requirement quality, including lack of clarity and difficulties in testing. One requirements developer further emphasized that writing good requirements is challenging in practice. The theory provides potential explanations for these issues. Palomares et al. (2021) identify challenges in requirements elicitation related to understanding and prioritizing stakeholders' needs and requirements instability due to changing priorities. Considering the context of the case company, this appears highly relevant given the need to address diverse stakeholders, such as the OEMs, the OEM's customers, as well as regulatory standards. In addition, Abrahão et al. (2024) highlight challenges of requirements specifically in the context of complex SW systems, which aligns with the case company's development of SW for the automotive industry.

The case company documentation recommends a method called EARS when specifying requirements; none of the interviewed requirements developers mentioned using it in practice. This highlights a fragmentation between actual practices and centrally defined process guidance, as discussed in Section 5.1. However, the case company's internal documentation also highlights the importance of reviews for assessing feasibility and testability, which both interviewees described as a means of evaluating requirements in an iterative manner to improve clarity.

Given that requirement specification is identified as a key source of quality risk (Habibullah et al., 2024), a more standardized and streamlined requirements process across the organization is thereby preferred. According to ISO 9000 (2015), this process approach enhances understanding of how results are produced, thereby facilitating the identification of improvements.
Green et al. (2024) and Majumder et al. (2022) highlight the value of process metrics in predicting quality issues, primarily in relation to code quality, the same logic may also be applied for requirements activities. A well-defined and consistently applied process would increase transparency and measurability of requirement-related work, allowing the organization to collect relevant data on process performance. This supports evidence-based decision-making, as emphasized in ISO 9000 (2015), where decisions based on data increase the likelihood of achieving the desired result.

A final challenge concerns requirements in relation to time and planning. One interviewee highlighted that requirements are sometimes rushed due to time and cost considerations. Another described the importance of having sufficient time in the

requirements phase, linking this to SW quality by emphasizing the importance of doing the right things from the beginning. Considering that time pressure affects quality outcomes (Kuutila et al., 2020) and that requirements elicitation is already challenging due to changing priorities and stakeholder complexity (Palomares et al., 2021), sufficient time in the requirements stage appears to be important. This is particularly relevant given that requirements are developed early in the process and influence later development stages. High-quality requirements in terms of clarity, understandability, and fulfilment of stakeholder needs are therefore important, especially since later adjustments to requirements tend to be more costly (Tan et al., 2017).

A further concern relates to requirements in relation to implementation, where two interviewees described situations in which implementation was completed or near completion while requirements were not yet finalized. Although this was not evident across all interviews, it represents a critical issue. As highlighted in the literature, requirements serve to translate stakeholders' needs into technical specifications (Abrahão et al., 2024). When requirements are incomplete or not fully defined, there is therefore a risk that the developed SW does not reflect stakeholder needs.
In addition, downstream verification depends on clear and traceable links between requirements, implementation, and testing (Habibullah et al., 2024; Maro et al., 2018). Unfinished or missing requirements therefore make verification difficult, or in some cases not feasible, which aligns with the challenges related to testing expressed by interviewees. Furthermore, this undermines the importance of achieving the necessary traceability, which is critical in the automotive context, where SW development must adhere to strict quality obligations, standards, and regulatory frameworks (Li et al., 2024).

5.3.3 System complexity and dependencies

It is evident across the interviews that the SW systems being developed are inherently complex. This aligns with the literature highlighting that automotive SW has become increasingly complex as vehicles have become more SW-defined (Li et al., 2024; Ryu & Do, 2023; Wang et al., 2024). More broadly, in line with technological advancement, it is also likely that the automotive industry will continue to become increasingly SW-defined, bringing further complexity. While the SW itself is characterized by complexity, the findings suggest that dependencies make SW development further complex and a risk for quality issues. As Vogelsang (2020) highlights, features are no longer independent, which complicates development because dependencies must be managed from a broader system perspective.

The findings highlight that dependencies within the SW constitute a challenge, which can be understood from two perspectives: (1) dependencies between features and (2) dependencies between teams. Regarding the first perspective, Interviewee 13 highlighted the importance of cyber securing developed features even when they are not considered a cyber security risk, since they are part of a bigger network that may

be affected if not handled properly. This aligns with Li et al. (2024), who emphasize that increased connectivity places greater demands on development to include standalone cybersecurity requirements to withstand attacks. The findings therefore suggest that SW need to be understood and evaluated from a broader system perspective because their interactions with other parts of the system may create quality-related consequences. Thereby highlighting the need for dependencies management within the automotive context as Vogelsang (2020) highlights.

Some interviewees expressed challenges related to dependencies between teams, for example when integrating SW on other teams' nodes or when lacking understanding of how changes of one team's SW may affect other teams' SW. This aligns with Vogelsang (2020), who highlights that dependencies create integration challenges and therefore need to be considered from a system-level perspective. The findings suggest that achieving such a perspective is difficult in practice, as teams do not always have full visibility of how their own changes interact with neighboring parts of the whole system. This, in turn, creates challenges in development and increases the risk that dependency-related issues are not identified until later stages.

Challenges regarding dependencies between teams were further highlighted as being easier to handle when managing the SW feature within the team. However, when dependency exists between teams, collaboration is challenging due to different priorities and risk becoming a potential bottle neck. In line with Bjarnason et al. (2022), who highlights the importance of communication and collaboration when handling issues, particularly with autonomous teams. The findings suggest that this collaboration is perceived as hard to set up and time consuming, risking slowing down the development. A possible explanation for this can also be linked to the process fragmentation discussed in Section 5.3.1. If teams operate with a high degree of autonomy, there is a risk that they do not fully see how their local process guidance relates to the broader system, which in turn may make collaboration more difficult when dependencies across teams need to be managed.

To conclude, the findings suggest that SW system complexity impacts development and potentially also SW quality, not only through the technical and feature dependencies themselves, but also through dependencies across teams. The authors therefore conclude that SW quality within automotive SW development requires system-level visibility, coordination, and alignment across teams.

5.3.4 Time

Time also appeared as a consistent factor affecting SW development and potentially the SW quality in the case company. Several interviewees highlighted that quality outcomes depend on having sufficient time and realistic planning from the beginning of development. However, the empirical findings show that this is not always the case in practice, for example unrealistic timelines, requirements development, code refactoring, code adjustment in relation to release candidate (stage Z). This shows a

wide coverage of time related issues expressed in the case company indicating a misalignment with quality management planning and proactive process principles, where quality depends on establishing the right conditions needed to achieve intended results (ISO 9000, 2015).

The findings also show that time pressure can lead to situations where SQA activities are postponed or performed with less attention. One project sponsor highlighted the need for clear rules for quality related activities that cannot be skipped even with time pressure. However, the findings show a different reality, as one EM highlighted pressure to deliver on short notices creates uncertainty in the quality of the SW especially when testing was not as thorough as expected. Another requirement developer also expressed a lack of time to perform thorough requirement reviews especially when dealing with fault slip-through reports as they increase workload, and time plans do not account for this. These consequences of time pressure were visible across several roles in the case company, aligning with Kuutila et al. (2020), who argue that sustained time pressure can reduce SW quality and encourage shortcuts in development. It is clear from the empirical findings that several SQM practices have been rushed due to time pressure. Therefore, time affects the amount of development that can be done, but by also influencing how thoroughly the development process and quality practices can be performed, it potentially affects the SW quality. Preventive quality practices such as requirement reviews, code refactoring, work product reviews, and early testing risk becoming weak, increasing the reliance of late-stage quality control and defect correction. Therefore, time pressure should be understood as a quality risk rather than only project-planning issue.

5.3.5 Testing

Testing is another important factor affecting SW quality (see Section 5.2); it evaluates SW behavior and detects defects across the development stages. However, the findings also show that the value of testing depends on the speed and effectivity of the tests. Refer to section 4.4.5; several interviewees highlighted testing-related challenges affecting both SW quality and developer efficiency. Slow feedback loops were described as particularly important for developers to maintain the context fresh in mind as it affects operational efficiency. Green et al. (2024) highlight developers emphasizing process efficiency as important, furthermore that developers' efficiency impact product quality. Therefore, the authors concluded that it is important to improve feedback loops to enhance developer efficiency. While slower or late test results may still support SQC by showing whether the SW fulfills certain criteria, it contributes less to the prevention focus and developer efficiency.

Automated testing is also viewed as critical at the case company due to the large-scale development and need for faster outputs and precision as manual testing is insufficient. This aligns with Zhao et al. (2025) who argue that modern automotive SW development requires increased automation due to the complexity of the automotive

context and the risk of failing to detect flaws before SW release. Additionally, the SIL test environments were highlighted as not being well developed, affecting the ability to test the SW earlier in the project. This operational reality contradicts the V-model logic, where verification activities should be linked to the corresponding development stages rather than left until later stages (Pavlíčková et al., 2024).

These findings indicate that while testing is recognized as important, the organization's ability to test efficiently is not mature enough. They also suggest that testing goes beyond the number of tests performed, but involves ensuring test effectiveness, speed of feedback, and the availability of adequate test environments. When these are not in place, testing becomes more reactive than preventive and increases the reliance on costly late-stage defects detection rather than a built-in quality focus.

5.3.6 Summary

Taken together, Section 5.3 shows that SW quality is affected by several interdependent factors rather than standalone factors. This is illustrated in *Figure 5.1.* The minus in the figure indicates a negative impact and plus indicates a positive impact. Dashed arrows are the author's conclusions based on the findings in comparison to the literature and solid arrows are where theory and the findings coincide.

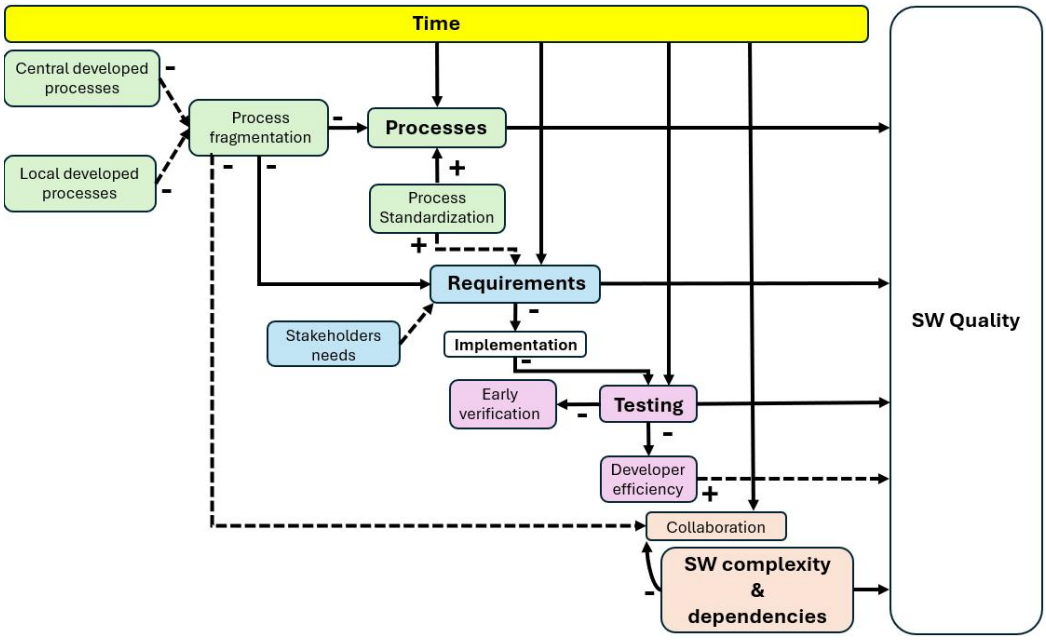


Figure 5.1 Factors impacting SW quality

A clear process focus provides the structure for quality assurance and control, but its effectiveness depends on whether the processes are unified, consistently understood and applied across the organization. Requirement quality has been seen to form an important foundation connecting everything else. Unclear requirements or insufficiently testable requirements create downstream issues for implementation, verification, and testing. These issues are further intensified due to the dependencies

and complex nature of automotive SW systems, making collaboration challenging. Additionally, testing is highlighted as central to SW quality even though it is limited at the case company. Finally, time pressure and limited planning act as an amplifier to all the highlighted factors. Therefore, improving SW quality involves more than individual process improvements, it needs to address how these highlighted factors interact across the development processes.

6 Conclusion

This chapter concludes the research by answering the research questions, presenting limitations, and suggestions for future research. The purpose of this study is to increase the understanding of how SW quality is defined within an automotive SW development context, and to identify practices and factors that impact SW quality. To address this, relevant literature has been reviewed to form a basis for the research and empirical data have been collected through semi-structured interviews, observations, and documentation reviews.

6.1 Answering the Research Questions

RQ1: How is software quality defined within the automotive context?

The research shows that SW quality is difficult to define within the automotive SW development context. Instead, SW quality is understood as a concept with many dimensions, shaped by role and development context. There are indications that the development context determines which SW quality attributes are being prioritized. Furthermore, the role perspective seems to also determine the emphasis on specific SW quality attributes. However, two perspectives are heavily emphasized. First, there is a strong focus on requirements when defining the SW quality within automotive, either in adherence to regulatory aspects, verification, or fulfilling customer needs. Secondly, there is a strong emphasis on safety of the automotive SW, considering potential consequences of SW failure in the automotive context.

RQ2: What practices are important for software quality management?

The research identifies several practices that are important for SW quality management. The important practices identified include reviews, gates, testing, traceability, metrics and KPIs, audits and assessments, retrospectives and lessons learned, and continuous improvement. Therefore, the authors conclude that while all the practices are necessary; reviews, gates and testing should have more emphasis as they provide expert judgement, objective control, and evaluation of SW behavior respectively across all the development stages.

RQ3: What factors impact automotive software development and quality?

The research identifies five factors that impact SW development and quality such as process focus, requirements, system complexity and dependencies, time, and testing, as discussed in Section 5.3. The research shows that SW development and SW quality are influenced by a combination of technical, organizational, and process-related factors that interact with each other throughout the SDLC. Additionally, the authors identify that the process focus is a critical factor in automotive SW development, as

they significantly affect both the development process itself and the resulting SW quality.

6.2 Limitations

This research aims to contribute to an increased understanding of SW quality management in the automotive context. However, certain limitations are acknowledged. Firstly, the research is based on a single-case study, connected to a specific organizational context. Therefore, the findings may not be directly generalizable and transferable to other companies, or SW development contexts. Secondly, managerial roles were overrepresented compared to operational roles. This may have influenced the findings giving less representation to the roles related to actual SW implementation. Finally, the research had limited access to interviewees from the departments developing the most customer-facing technology. The absence of these departments may have limited the understanding of customer perspective in SW quality.

6.3 Future Research

Future research could build on this study by including more operational roles, particularly requirement developers, SW developers, and testers. This could provide deeper insights into the application of SQM practices by the SW implementation practitioners. Additionally, including more customer-facing development practitioners, could strengthen the understanding of customer perspectives related to automotive SW quality. Finally, future research could complement the findings of this study with a quantitative approach to investigate whether the identified factors are experienced by a wider sample across departments, and even other organizations within the automotive context.

References

Abrahão, S., Staron, M., Gay, G., Penzenstadler, B., & Honnenahalli, C. (2024). Emerging trends in requirements engineering and testing. *IEEE s*. https://doi.org/10.1109/MS.2024.3439092

Alam, I., Sarwar, N., & Noreen, I. (2022). Statistical analysis of software development models by six-pointed star framework. *PLOS ONE, 17*(4), e0264420. https://doi.org/10.1371/journal.pone.0264420

Atoum, I., Baklizi, M. K., Alsmadi, I., Otoom, A. A., Alhersh, T., Ababneh, J., Almalki, J., & Alshahrani, S. M. (2021). Challenges of software requirements quality assurance and validation: A systematic literature review. *IEEE Access, 9*, 137613–137634. https://doi.org/10.1109/ACCESS.2021.3117989

Ayres, N., Deka, L., & Paluszczyszyn, D. (2021). Continuous automotive software updates through container image layers. *Electronics, 10(6), 739*. https://doi.org/10.3390/electronics10060739

Bell, E., Bryman, A., & Harley, B. (2022). *Business research methods* (6th ed.). Oxford University Press.

Bjarnason, E., Gislason Bern, B., & Svedberg, L. (2022). Inter-team communication in large-scale co-located software engineering: A case study. *Empirical Software Engineering, 27*, Article 36. https://doi.org/10.1007/s10664-021-10027-z

Broy, M., Krüger, I. H., Pretschner, A., & Salzmann, C. (2007). Engineering automotive software. *IEEE, 95**(2), 356–373. https://doi.org/10.1109/JPROC.2006.888386

Boiral, O. (2007). Corporate greening through ISO 14001: A rational myth? *Organization Science, 18*(1), 127–146. https://doi.org/10.1287/orsc.1060.0224

Bowen, G. A. (2009). Document analysis as a qualitative research method. *Qualitative Research Journal, 9*(2), 27–40. https://doi.org/10.3316/QRJ0902027

Bozola, P. M., Nunhes, T. V., Barbosa, L. C. F. M., Machado, M. C., & Oliveira, O. J. (2023). Overcoming the challenges of moving from ISO/TS 16949 to IATF 16949: recommendations for implementing a quality management system in automotive companies. *Benchmarking: An International Journal, 30*(9), 3699–3724. https://doi.org/10.1108/BIJ-04-2022-0215

Börstler, J., Bennin, K. E., Hooshangi, S., Jeuring, J., Keuning, H., Kleiner, C., MacKellar, B., Duran, R., Störrle, H., Toll, D., & van Assema, J. (2023). Developers

talking about code quality. *Empirical Software Engineering, 28*, Article 128. https://doi.org/10.1007/s10664-023-10381-0

Cândido, C. J. F. (2024). Strategies for the ISO 9001 certification life cycle (StrategISO). *International Journal of Productivity and Performance Management, 73*(6), 1856–1884. https://doi.org/10.1108/IJPPM-05-2023-0224

De Giovanni, P. (2023). The modern meaning of "quality": Analysis, evolution and strategies. *The TQM Journal, 36*(9), 309–327. https://doi.org/10.1108/TQM-12-2023-0413

Ferreira, A. M., Brito, M. A., & de Lima, J. (2024). Software quality in an automotive project: Continuous inspection. *International Journal of Automotive Technology*, 1–10. https://doi.org/10.1007/s12239-024-00132-5

Giachetti, G., de la Vara, J. L., & Marín, B. (2024). Model-driven gap analysis for the fulfillment of quality standards in software development processes. *Software Quality Journal, 32*(1), 255–282. https://doi.org/10.1007/s11219-023-09649-x

Green, C., Jaspan, C., Hodges, M., & Lin, J. (2024). Developer productivity for humans, part 7: Software quality. *IEEE Software, 41*(1), 25–30. https://doi.org/10.1109/MS.2023.3324830

Gremyr, I., Lenning, J., Elg, M., & Martin, J. (2021). Increasing the value of quality management systems. *International Journal of Quality and Service Sciences, 13*(3), 381–394. https://doi.org/10.1108/IJQSS-10-2020-0170

Guamán, D. S., Del Alamo, J. M., & Caiza, J. C. (2020). A systematic mapping study on software quality control techniques for assessing privacy in information systems. *IEEE Access, 8*, 74808–74833. https://doi.org/10.1109/ACCESS.2020.2988408

Habibullah, K. M., Heyn, H.-M., Gay, G., Horkoff, J., Knauss, E., Borg, M., Knauss, A., Sivencrona, H., & Li, P. J. (2024). Requirements and software engineering for automotive perception systems: An interview study. *Requirements Engineering, 29*(1), 25–48. https://doi.org/10.1007/s00766-023-00410-1

International Organization for Standardization. (2015). *ISO 9000:2015, quality management systems—Fundamentals and vocabulary*. https://www.iso.org/standard/45481.html

International Organization for Standardization. (2015). *ISO 9001:2015, quality management systems—Requirements*. https://www.iso.org/standard/62085.html

International Organization for Standardization. (2018). *ISO 26262 road vehicles functional safety*. https://www.iso.org/publication/PUB200262.html

International Organization for Standardization, & International Electrotechnical Commission. (2023). *ISO/IEC 25010:2023, systems and software engineering—Systems and software Quality Requirements and Evaluation (SQuaRE)—Product quality model*. https://www.iso.org/standard/78176.html

International Organization for Standardization, International Electrotechnical Commission, & Institute of Electrical and Electronics Engineers. (2017). *ISO/IEC/IEEE 12207:2017 systems and software engineering — Software life cycle processes*. https://www.iso.org/standard/63712.html

International Organization for Standardization, & SAE International. (2021). *ISO/SAE 21434:2021 road vehicles — Cybersecurity engineering*. https://www.iso.org/standard/70918.html

Izurieta, C., Reimanis, D., O'Donoghue, E., Liyanage, K., Muneza, A. R. M., Whitaker, B., & Reinhold, A. M. (2024). A generalized approach to the operationalization of software quality models. *PeerJ Computer Science, 10*, e2357. https://doi.org/10.7717/peerj-cs.2357

Khamis, A., & Goswami, P. (2025). Rethinking vehicle architecture through softwarization and servitization. *IEEE Access*, 13, 126213–126226. https://doi.org/10.1109/ACCESS.2025.3588432

Kudrjavets, G., & Rastogi, A. (2024). Does code review speed matter for practitioners? *Empirical Software Engineering, 29*, Article 7. https://doi.org/10.1007/s10664-023-10401-z

Kuutila, M., Mäntylä, M., Farooq, U., & Claes, M. (2020). Time pressure in software engineering: A systematic review. *Information and Software Technology, 121*, 106257. https://doi.org/10.1016/j.infsof.2020.106257

Lee, J.-C., Chou, I.-C., & Chen, C.-Y. (2021). The effect of process tailoring on software project performance: The role of team absorptive capacity and its knowledge-based enablers. *Information Systems Journal, 31*(1), 120–147. https://doi.org/10.1111/isj.12303

Li, Y., Liu, W., Liu, Q., Zheng, X., Sun, K., & Huang, C. (2024). Complying with ISO 26262 and ISO/SAE 21434: A safety and security co-analysis method for intelligent connected vehicle. *Sensors, 24*, Article 1848. https://doi.org/10.3390/s24061848

Majumder, S., Mody, P., & Menzies, T. (2022). Revisiting process versus product metrics: A large scale analysis. *Empirical Software Engineering, 27*, Article 60. https://doi.org/10.1007/s10664-021-10068-4

Maro, S., Steghöfer, J.-P., & Staron, M. (2018). Software traceability in the automotive domain: Challenges and solutions. *Journal of Systems and Software, 141*, 85–110. https://doi.org/10.1016/j.jss.2018.03.060

McIntosh, S., Kamei, Y., Adams, B., & Hassan, A. E. (2016). An empirical study of the impact of modern code review practices on software quality. *Empirical Software Engineering, 21*(5), 2146–2189. https://doi.org/10.1007/s10664-015-9381-9

Moe, N. B., Šmite, D., Paasivaara, M., & Lassenius, C. (2021). Finding the sweet spot for organizational control and team autonomy in large-scale agile software development. *Empirical Software Engineering, 26*, Article 101. https://doi.org/10.1007/s10664-021-09967-3

Nabot, A., & Al-Qerem, A. (2025). Impact of software quality on organizational performance. *Array, 27*, 100476. https://doi.org/10.1016/j.array.2025.100476

Palomares, C., Franch, X., Quer, C., Chatzipetrou, P., López, L., & Gorschek, T. (2021). The state-of-practice in requirements elicitation: An extended interview study at 12 companies. *Requirements Engineering, 26*(2), 273–299. https://doi.org/10.1007/s00766-020-00345-x

Pavlíčková, M., Mojžišová, A., Bodíková, Z., Szeplaki, R., & Laciak, M. (2024). Integration and implementation of scaled agile framework and V-model in the healthcare sector organization. *Electronics, 13*(11), Article 2051. https://doi.org/10.3390/electronics13112051

Psarommatis, F., & Azamfirei, V. (2024). Zero defect manufacturing: A complete guide for advanced and sustainable quality management. *Journal of Manufacturing Systems, 77*, 764–779. https://doi.org/10.1016/j.jmsy.2024.10.022

Ryu, C., & Do, S. (2023). A Method for Managing Software Assets in the Automotive Industry (Focusing on the Case of Hyundai Motor Company and Parts Makers). *Applied Sciences, 13*(7), 4174. https://doi.org/10.3390/app13074174

Shah, S. M. A., Sundmark, D., Lindström, B., & Andler, S. F. (2016). Robustness testing of embedded software systems: An industrial interview study. *IEEE Access, 4*, 1859–1875. https://doi.org/10.1109/ACCESS.2016.2544951

Solin, A., & Curry, A. (2022). Perceived quality: In search of a definition. *The TQM Journal, 35*(3), 778–795. https://doi.org/10.1108/TQM-09-2021-0280

Tan, J. J. Y., Otto, K. N., & Wood, K. L. (2017). Relative impact of early versus late design decisions in systems development. *Design Science, 3*, e12. https://doi.org/10.1017/dsj.2017.13

Timmermans, S., & Tavory, I. (2012). Theory construction in qualitative research: From grounded theory to abductive analysis. *Sociological Theory, 30*(3), 167–186. https://doi/10.1177/0735275112457914

Trovão, J. P. (2024). The evolution of automotive software: From safety to quality and security. *IEEE Vehicular Technology*. https://doi.org/10.1109/mvt.2024.3468128

United Nations Economic Commission for Europe. (2021). *UN Regulation No. 156: Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system*. https://unece.org/transport/documents/2021/03/standards/un-regulation-no-156-software-update-and-software-update

VDA Quality Management Center. (2023). *Automotive SPICE® guidelines: Process assessment using Automotive SPICE in the development of software-based systems* (2nd revised ed.). https://webshop.vda.de/QMC/en/automotive-spice-guideline-edition_2023

Vogelsang, A. (2020). Feature dependencies in automotive software systems: Extent, awareness, and refactoring. *Journal of Systems and Software, 160*, Article 110458. https://doi.org/10.1016/j.jss.2019.110458

Wang, W., Guo, K., Cao, W., Zhu, H., Nan, J., & Yu, L. (2024). Review of electrical and electronic architectures for autonomous vehicles: Topologies, networking and simulators. *Automotive Innovation, 7*, 82–101. https://doi.org/10.1007/s42154-023-00266-9

Wessel, M., Serebrenik, A., Wiese, I. S., Steinmacher, I., & Gerosa, M. A. (2022). Quality gatekeepers: Investigating the effects of code review bots on pull request activities. *Empirical Software Engineering, 27*(5), Article 108. https://doi.org/10.1007/s10664-022-10130-9

Zhao, X., Wang, H., Ding, J. Q., Hu, Z., Tian, Q., & Wang, Y. (2025). Augmenting software quality assurance with AI and automation using PyTest-BDD. *Automated Software Engineering, 33*, Article 23. https://doi.org/10.1007/s10515-025-00566-w

Appendices

Appendix A Questionnaire for Quality Managers

General questions:

- Please introduce yourself and which line organization you belong to and your role?
- How long have you been working at the [case company] and with SW development in total?

Quality specific questions:

- How do you define “software quality” and “built-in quality”?
- What are the most important quality attributes for automotive software in your context?
- How would you describe quality assurance and its purpose in SW development?
- How would you describe quality control and its purpose in SW development?
- How are quality assurance, quality control, and verification activities applied in your development process in practice?
- What practices are most important for ensuring high-quality software development?

Frameworks, Standards & “Ways of Working”:

- Where do you typically find documented guidance on how software development work should be performed, and which source do you encourage teams to follow?
    - Do you perceive the information to be thorough and adequate for your team members to be able to deliver a quality SW?
- How do you communicate these quality practices/activities and ways of working within the team?
    - How do you ensure that communicated quality processes are understood and followed by development teams?

Collaboration:

- What kind of collaborative efforts are you involved in to increase quality of the development processes?
    - What kind of collaborative efforts are carried out with other teams within [Case Company] to improve quality?
- What would make these collaborative efforts smoother or more effective?

Quality expectations in relation to development efforts.

- What kind of circumstances do you perceive to be critical for teams to be able to deliver quality outcomes?
- Which quality-related practices or activities are considered non-negotiable, even when the project is under time or cost pressure?
- How do you ensure consistency in quality practices across all the development stages?

Defects handling and Continuous Improvement

- What metrics do you use to measure the quality of your development efforts?
    - How do you handle trends (good & bad) within your team/organization, based on metrics?
- What typical quality issues occur and at what stages in the development efforts?
    - How do you capture and apply lessons learned to ensure continuous improvement of quality in the development processes?

Final Reflection

- What would you say are currently working well and what is missing regarding the development efforts to be able to ensure a quality outcome?

Appendix B questionnaire for engineering managers

General questions:

- Please introduce yourself and which line organization you belong to and your role?
- How long have you been working at [Case Company] and with SW development in total?

Quality specific questions:

- How do you define “software quality” and “built-in quality”?
- What are the most important quality attributes for automotive software in your context?
- How would you describe quality assurance and its purpose in SW development?
- How would you describe quality control and its purpose in SW development?
- How are quality assurance, quality control, and verification activities applied in your development process in practice?
- What practices are most important for ensuring high-quality software development?

Quality Ownership and Responsibilities

- Where do you believe quality responsibility primarily sits, and what are the roles responsible?
- What kind of circumstances do you perceive to be critical for teams to be able to deliver quality outcomes?

Processes, Culture & Ways of Working

- How would you describe the culture of quality within your unit/group?
    - Where do you see gaps in the quality mindset within your unit/group?
- Where do you typically find documented guidance on how software development work should be performed, and which source do you encourage teams to follow?
    - Is it always easy to locate these guides, and do you perceive the information to be updated and reliable?
    - Do you perceive the information to be thorough and adequate for your team members to be able to deliver a quality SW?
- From your point of view what practices are lacking currently to ensure quality throughout the development efforts?

Measurement & Continuous improvements

- What metrics do you use to measure the quality of your development efforts?
    - How do you handle trends (good & bad) within your team, based on metrics?

- What typical quality issues occur and at what stages in the development efforts?
- How do you capture and apply lessons learned to ensure continuous improvement of quality in the development processes?

Quality expectations in relation to development efforts.

- How do you handle situations where the quality assurance related activities risk being deprioritized due to time or cost pressures?
    - Which quality-related practices or activities are considered non-negotiable, even when the project is under time or cost pressure?
- How do you ensure consistency in quality practices across the development effort?

Final Reflection

- What would you say are currently working well and what is missing regarding the development efforts to be able to ensure a quality outcome?

Appendix C questionnaire Product owners

General questions:

- Please introduce yourself and which line of organization you belong to and your role?
- How long have you been working at [Case Company] and with SW development in total?

Quality specific questions:

- How do you define “software quality” and “built-in quality”?
- What are the most important quality attributes for automotive software in your context?
- How would you describe quality assurance and its purpose in SW development?
- How would you describe quality control and its purpose in SW development?
- How are quality assurance, quality control, and verification activities applied in your development process in practice?
- What practices are most important for ensuring high-quality software development?

Responsibilities and quality aspects

- What kind of circumstances do you perceive to be critical for teams to be able to deliver quality outcomes?
- What quality-related activities would you say are most critical, and how do you ensure these are being realized within the team?
    - Are these quality-related activities adequate, and where do you see gaps?
- How do you handle situations where quality related activities in the development efforts risk being deprioritized due to time and cost pressures?
    - Which quality-related practices or activities are considered non-negotiable, even when the project is under time or cost pressure?
- How do you ensure regulative automotive requirements and certifications have been fulfilled?
- How would you describe the responsibilities regarding quality within the team?

Ways of working

- Where do you typically find documented guidance on how software development work should be performed, and which source do you encourage teams to follow?
    - Is it always easy to locate these guides, and do you perceive the information to be updated and reliable?

    - Do you perceive the information to be thorough and adequate for your team members to be able to deliver a quality SW?
- Are there any “local” adaptations to this recommended documented guidance?

Metrics, Continuous improvement & Knowledge Sharing

- What metrics do you use to measure the quality of your development efforts?
    - How do you handle trends (good & bad) within your team, based on metrics?
- What typical quality issues occur and at what stages in the development efforts?
    - How do you capture and apply lessons learned to ensure continuous improvement of quality in the development processes?
- Do you have any knowledge sharing outside your team for potential adoption of other team ways of working or sharing own process developments?

Final Reflection

- What would you say are currently working well and what is missing regarding the development efforts to be able to ensure a quality outcome?

Appendix D questionnaire for requirement developers

General questions:

- Please introduce yourself and which line organization you belong to and your role?
- How long have you been working at [Case Company] and with SW development in total?

Quality specific questions:

- How do you define “software quality” and “built-in quality”?
- What are the most important quality attributes for automotive software in your context?
- How would you describe quality assurance and its purpose in SW development?
- How would you describe quality control and its purpose in SW development?
- How are quality assurance, quality control, and verification activities applied in your development process in practice?
- What practices are most important for ensuring high-quality software development?

Requirement understanding & quality aspects

- How would you define high-quality SW requirements?
    - How do you determine whether requirements are feasible and testable?
- How do you ensure that SW requirements reflect customers' and vehicle-levels expectations?
    - How do you evaluate the trade-off between potential customer benefits that specific SW requirements may introduce versus the complexity it may add to the software?

Requirements specifying:

- What practices ensure that SW requirements when specified are clear and complete?
- How do you ensure that various ISO standards and any other safety or quality standards are considered when specifying requirements?
- Are there rooms for interpretations of SW requirements and how are ambiguities detected and handled before development starts?
- Do you within the team or externally have systematic reviews of the SW requirements, who is involved, and what is the purpose and focus?
    - How are system properties represented in your software requirements?

- When defects are discovered in later stages of the development efforts, how often are they found to originate from SW requirements?
    - From your perspective, what would you say are the frequent causes of these defects?
- How are changes to SW requirements handled once implementation has started?
    - How is backward compatibility addressed at SW requirement level?

Ways of working

- Where do you typically find documented guidance on how SW requirements should be developed ?
    - Is it always easy to locate these guides, and do you perceive the information to be updated and reliable?
    - Do you perceive the information to be thorough and adequate for you to be able to deliver quality SW requirements?
- What works well and what are challenges with your current SW requirement development tool?
- How do you measure the quality of the requirements? Are there any specific KPIs that you are tracking?
    - How do you handle trends (good & bad) within your team, based on metrics?

Final Reflection

- What would you say are currently working well and what is missing regarding the development efforts to be able to ensure a quality outcome?

Appendic E questionnaire for software architects

General questions:

- Please introduce yourself and which line organization you belong to and your role?
- How long have you been working at [Case Company] and with SW development in total?

Quality specific questions:

- How do you define “software quality” and “built-in quality”?
- What are the most important quality attributes for automotive software in your context?
- How would you describe quality assurance and its purpose in SW development?
- How would you describe quality control and its purpose in SW development?
- How are quality assurance, quality control, and verification activities applied in your development process in practice?
- What practices are most important for ensuring high-quality software development?

Understanding & Interpreting Requirements

- How do you interpret software requirements when starting architectural design, and how do you ensure that important quality attributes are properly understood?
- When requirements are ambiguous or conflicting, what is your process for resolving these issues before making architectural decisions?

Allocation & Traceability

- What approach/tools do you use to allocate software requirements to architectural elements?
    - What is working good and bad with your current tool to develop SW architecture?
- What is your approach for ensuring bidirectional traceability and consistency between software requirements and software architecture?
    - If not touched upon ask: In your opinion, what is the main purpose and value for achieving traceability?
- How do you ensure that your architectural design remains aligned with updated or evolving SW requirements?

Ways of Working -

- Where do you typically find documented guidance on how SW architecture should be developed.

    - Is it always easy to locate these guides, and do you perceive the information to be updated and reliable?
    - Do you perceive the information to be thorough and adequate for you to be able to deliver quality SW architecture?

Trade-Off Analysis & Quality attributes –

- How do you approach evaluating trade-offs and prioritize competing design characteristics versus the complexity they may add to the software?
- How do you ensure that various ISO standards and any other safety or quality standards are considered when designing SW architecture?
- How do you avoid unnecessary complexity in your architecture to ensure easy maintainability and unit testing?
- What characterizes a high quality SWAD (software architecture design description) and what risk does it mitigate in later development stages?
    - Which quality attributes are hardest to verify at the architectural level, and why?

Collaboration, Review Process & Defect Handling

- Are architecture designs reviewed systematically and what's the purpose?
    - Who participates in these reviews, and what is their focus?
- How are architectural risks or quality issues identified in these reviews and how are they escalated/addressed?
- What works well in the current architecture review process, and where do you see improvement opportunities?
- After architectural changes, how are updates communicated to downstream stakeholders?
- At what stages of development are most architectural defects discovered?
    - In your experience, what are the most common causes of these defects?

Final Reflection

- What would you say are currently working well and what is missing regarding the development efforts to be able to ensure a quality outcome?

Appendix F questionnaire for software developers

General questions:

- Please introduce yourself and which line of organization you belong to and your role?
- How long have you been working at [Case Company] and with SW development in total?

Quality specific questions:

- How do you define “software quality” and “built-in quality”?
- What are the most important quality attributes for automotive software in your context?
- How would you describe quality assurance and its purpose in SW development?
- How would you describe quality control and its purpose in SW development?
- How are quality assurance, quality control, and verification activities applied in your development process in practice?
- What practices are most important for ensuring high-quality software development?

Questionnaire (F1) - Detailed Design

SWDD development and quality

- How would you define high-quality SWDD?
    - What stage of development should SWDD be written?
    - What typical defects do you see in SWDD?
- Before you develop a SWDD, what do you usually do to make sure you understand and address both functional and non-functional requirements in the design*?*
    - How are ambiguities detected and resolved before design implementation begins?
- How do various ISO standards and any other safety or quality standards influence your design decisions?
- How do you ensure bidirectional traceability between e.g. requirements, architecture, and detailed design?
    - What quality risks do you see when traceability between these elements is incomplete or missing?
- When modifying an existing function or module, how do you identify and manage dependencies that might be affected?
    - How are these impacts communicated to other teams or stakeholders?
- How do you avoid unnecessary complexity in your designs to ensure easy maintainability and unit testing?

Frameworks & Tools

- Are there any tools, templates or frameworks you utilize when developing SWDD, what are working good and bad with your current tools?
- Do you perceive the tools/templates/frameworks to be sufficient to develop quality SWDD or are there any gaps?

Reviews

- Are detailed design documents (SWDD) reviewed systematically and what's the purpose?
    - Who participates in these reviews, and what is their focus?
    - How do design reviews confirm that the detailed design supports efficient and complete verification activities, such as unit testing?
- What works well in the current detailed design review process, and where do you see improvement opportunities?

Questionnaire (F2) - Implementation (Core Programming)

Before Coding - Clarity of requirements, architecture & Design:

- How do you make sure you fully understand the requirements, architecture, and detailed designs before starting implementation?
    - How are ambiguities detected and resolved before implementation begins, and what impact does it have on quality?
- Are you aware of the Definition-of-Ready (DoR) criteria?... *If yes follow up with questions below;*
    - From your perspective, what kind of gaps in DoR are common and how does it affect the quality of your implementation?

Coding "guidelines" & Ways of Working

- Where do you typically find documented guidance for developing SW.
    - Is it always easy to locate these guides, and do you perceive the information to be updated and reliable?
    - Do you perceive the information to be thorough and adequate for you to be able to deliver quality SW?
- What practices help you keep your code readable, maintainable, and easy for others to understand?
- How are code reviews carried out in your team and who is typically involved?
    - What works well in the current review process, and where do you see improvement opportunities?
- How do you typically approach unit testing when implementing new functionality?

    - In your opinion, what makes unit testing sufficiently complete for a software unit?
- What level of code coverage (for example line coverage or branch coverage) do you consider appropriate, and why?
    - Are there any coverage targets or thresholds communicated in your organization? *If yes, how are they used in practice?*
- Are there situations where the expected coverage levels are not reached? What are the typical reasons?
    - How do you determine whether a unit test is meaningful and valuable, rather than just increasing coverage numbers?
- When implementing new functionality or modifying existing code, how do you ensure that your changes do not introduce unintended side effects?
- What works well and what are challenges with your current development and testing tools?

Code Storage & Traceability

- How do you manage changes to your code?
    - How do you ensure that all code changes are traceable back to specific requirements, tasks, issues, or change requests?
    - How do you manage versions of your code together with versions of requirements and other data sources with separate version handling?
- Have you experienced issues caused by inadequate version control practices and how does this affect quality?
- When investigating bugs, how often does the root cause turn out to be outside the implemented code (e.g., architecture, requirements, test setup)?
- How do you capture learnings from your early testing activities?

Final Reflection –

- What would you say are currently working well and what is missing regarding the development efforts to be able to ensure a quality outcome?

Questionnaire (F3) – Integration & Testing

Pre-integration readiness

- Before integrating your unit/component, what conditions do you personally consider “must-have” to ensure high-quality integration at component and SW level?
    - What is often missing in practice?
- How do you ensure that requirements, interface definitions, and architectural constraints are clear and traceable before starting integration?
    - What kinds of gaps or misunderstandings do you typically see that can jeopardize integration quality?

- How is the integration strategy (order of integration, dependencies, test approach) decided and documented?

Workflow and integration quality:

- What works well and where does you see gaps in the coordination of integration activities within your team and across other teams?
- How do you verify interactions between units and components, especially when they are developed by different teams?
- How are changes to interfaces or software behavior communicated and managed to ensure compatibility during integration?
- What types of integration tests do you perform at the full software level, and where do you see gaps?
- How do you monitor regression risks during ongoing integrations?

Post Integration, traceability & defects -

- When integration reveals a problem, how easy is it to trace the root cause back to requirements, architecture, design, or implementation?
    - Where do most defects originate from your experience?
- How do you handle deviations or defects discovered during integration, and how do you assess their impact, risk, and severity?
    - Are there any deviations that can't be overlooked and if so, what kind?
- What quality indicators do you consider before declaring an integration successful?

Verification against software requirements

- Once the software is integrated, how do you verify that the complete software behavior satisfies all the defined software requirements?

Software-level testing practices

- What types of software-level tests do you typically perform before the software is considered ready for system testing?
    - Where do you see gaps or challenges in software qualification testing today?
- How do you verify non-functional attributes such as timing behavior, resource usage, robustness, or error handling at the software level?
- How do you ensure that the software behaves correctly under realistic operating conditions or system interactions?

Final Reflection

- What would you say are currently working well and what is missing regarding the development efforts to be able to ensure a quality outcome?

DEPARTMENT OF TECHNOLOGY MANAGEMENT AND ECONOMICS
DIVISION OF INNOVATION AND R&D MANAGEMENT
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY