Automated Data Collection and Dashboard Development : Enhancing Quality Monitoring and Analysis

University of Oulu
Information Processing Science
Master's Thesis
Amine Zeggaf
2022-2023

Abstract

This master thesis focuses on optimizing software quality analysis at target projects in Elektrobit, a key player in the automotive software industry. In the light of continuous technological advancements in the industry, maintaining high-quality software is imperative. Current verification procedures, though effective, are time-consuming and often entail a significant amount of manual work.

Additionally, the absence of efficient visualization tools leads to missed opportunities for improvement. I proposed an automated quality data collection process and a user-friendly visualization system, designed to streamline the process of software quality analysis. The solution is delivered in the form of an interactive dashboard, offering a centralized view of all pertinent quality metrics data, thereby saving time, reducing manual work, and improving user experience.

The study employs a design science research (DSR) methodology, focusing on iterative design, implementation, and evaluation. The findings from the evaluation confirm the dashboard's efficacy and relevance, illustrating its value in enhancing efficiency, productivity, and team collaboration. The user-centered design approach and regular user engagement during development proved to be instrumental in achieving these results.

The research acknowledges certain limitations, such as the proof-of-concept stage of the dashboard and the relatively small user feedback group and suggests potential areas for future work. In conclusion, this research serves as a testament to the transformative potential of automated data collection and effective visualization systems in the software quality analysis domain, contributing to both theory and practice.

Keywords
Software Quality, Analytics Dashboard, Data Collection, Web Development, User-Centric Design, DSR, API

Supervisor
Professor Burak Turhan

Foreword

It is with great pleasure and pride that I introduce this master's thesis, titled "Automated Data Collection and Dashboard Development: Enhancing Quality Monitoring and Analysis." As the sole developer of this project, I have poured my passion and dedication into creating an innovative solution to optimize software quality analysis at certain projects in Elektrobit.

I have always sought projects that challenge me and allow me to make a meaningful contribution to the field. When I was offered the opportunity to work on the project, I was overwhelmed with excitement. The realization of the project's importance and its potential impact further heightened my enthusiasm, as I recognized the immense value in developing a solution that could streamline processes, reduce manual work, and enhance the overall user experience.

Elektrobit holds a prominent position in the automotive software field. Its software operates in over five billion devices across more than 600 million vehicles, offering adaptable solutions for car infrastructure software, connectivity and security, self-driving technology, and user experience. Delivering top-quality software is paramount to maintaining the trust of its customers, driving the need for robust quality assurance processes.

I have proposed an innovative solution in form of a centralized, user-friendly dashboard coupled with an automated quality data collection process. The interactive dashboard provides a comprehensive view of essential quality metrics data, saving time, reducing manual work, and improving the user experience. The dashboard's modern and intuitive interface empowers project members to gain valuable insights into the quality level of their modules, enabling them to address quality issues promptly and efficiently.

Throughout the development process, I embraced a user-centered design approach, continuously engaging with stakeholders and incorporating their feedback to refine and enhance the dashboard's functionalities. The research employed a design science research (DSR) methodology, emphasizing iterative design, implementation, and evaluation. This problem-solving approach has allowed me to create a solution that meets specific user needs while contributing to the advancement of software quality data management practices.

This master's thesis represents the culmination of my efforts and a testament to my commitment to excellence in software development, and whether you're an industry professional, academic, student, or simply a curious reader, I hope that this work offers you valuable insights and provokes thoughtful conversations on the future of software quality assurance in the digitized automotive industry. I extend my deepest gratitude to my family, supervisors, colleagues at Elektrobit, and everyone else who supported me during this journey. Without their invaluable input and unwavering support, this thesis would not have been possible.

Contents

1. Introduction

Economies are experiencing major changes caused by growth in emerging markets, the swift evolution of technology, sustainability practices, and evolving consumer views on ownership. The digitization process, greater automation, and innovative business models have already transformed other industries and the automotive sector will not be an exception. These shifts are bringing forth four major, technology-driven trends that are heavily impacting the automotive industry: a diverse range of mobility options, self-driving vehicles, electrification, and interconnectedness, and Elektrobit is no stranger to this field, it holds a prominent position in the automotive software industry, boasting over 35 years of experience serving the field. Its software operates in over five billion devices across more than 600 million vehicles and provides adaptable, cutting-edge solutions for car infrastructure software, connectivity and security, self-driving technology, and associated tools, as well as user experience.

Therefore, delivering top-quality software is of the utmost importance, as it is crucial to maintain the trust of its customers. In a company with numerous projects, each containing multiple modules and software engineers continuously pushing code, it is imperative to have a robust CI/CD workflow that consistently verifies software quality. Adherence to industry standards such as ISO-9001 and A-SPICE is also a must to earn and retain the trust of the customers. By being aware of quality issues as soon as possible, it enables the teams to promptly address them and prevent future problems from arising.

Due to the importance of delivering high-quality software, Elektrobit has implemented rigorous verification methods and processes aimed at ensuring the desired level of quality. However, these verification procedures, while effective in achieving the quality objective, can result in a time-consuming task that often involves a significant amount of manual work. Additionally, without proper visualization tools, it can be difficult to spot trends and patterns in the data, leading to missed opportunities for improvement. Therefore, there is a need to explore ways to improve the data collection process, reduce manual work, and develop effective visualization methods to help identify and analyse important trends and patterns. This master’s thesis aims to investigate and propose a solution to these challenges, ultimately contributing to more efficient and effective data collection processes in various general. Consequently, I will be using design science research to resolve the issue at hand where will be trying to answer the following research question:

"How can we design and evaluate an automated quality data collection process and a visualization system that enables the organization to process and analyse quality data more effectively?"

We are proposing as solution to this problem, a centralized way to access the quality data, instead of having to browse through multiple tools and sources, the user would need to only visit the main website which will be presented in a form of a dashboard that keeps track of all import quality metrics data, while focusing also on making this dashboard as easy as possible to use and to browse through with a friendly modern user interface, the main idea is that a project member could get a clear idea on the quality level of the module that they are working on, while also getting relevant information that will help them fix quality issues in case they exist from the same place.

In the following sections, I will begin by presenting a Literature Review, which will provide an overview of the relevant research conducted so far and how it relates to the project at hand. This section will be a critical evaluation of the existing body of knowledge. Next, I will delve into the research methods where I will state what research methods were chosen and why they were selected, then, I will clearly define the objectives and state the evaluation method. Afterwards, I will explain how the overall solution was designed and developed, review the development process, discuss the obtained development and evaluation results, limitations of the solution and finally finish with a conclusion where I will identify the implications of the results for future work, and provide a synthesized evaluation of the project, highlighting its strengths and limitations.

2. Related Work

In the software development industry, delivering high-quality software has become a critical factor in ensuring customer satisfaction, reducing costs, and staying competitive in the market. To achieve this objective, Elektrobit has implemented rigorous verification methods and processes aimed at ensuring the desired level of quality. However, these verification procedures can be time-consuming, requiring a significant amount of manual work. Additionally, without proper visualization tools, it can be challenging to spot trends and patterns in the data, leading to missed opportunities for improvement.

Therefore, there is a need to explore ways to improve the data collection process, reduce manual work, and develop effective visualization methods to help identify and analyse important trends and patterns. This literature review aims to investigate the different approaches and solutions that have been proposed to address the challenges of software quality, data collection, and analytics dashboards, ultimately contributing to more efficient and effective data collection processes.

To achieve this goal, this literature review will be structured as follows. First, I will provide an overview of software quality, its definition, importance, models, and frameworks, metrics, and measurements, as well as the challenges in ensuring software quality. This will provide a foundation for understanding the importance of collecting data from information systems and the challenges involved.

Second, I will delve into the topic of data collection from information systems. This will include an overview of data collection techniques, types of data collected, the quality of data collected, and the challenges involved in collecting data from information systems.

Finally, I will explore the topic of analytics dashboards. I will define and discuss the importance of analytics dashboards, the types of analytics dashboards, the components of analytics dashboards, the design and development of analytics dashboards, and the challenges involved in designing and developing analytics dashboards.

Overall, this literature review aims to provide a comprehensive understanding of the different approaches and solutions that have been proposed to address the challenges of software quality, data collection, and analytics dashboards. The insights gained from this literature review will be used to inform the design and evaluation of an automated quality data collection process and a visualization system that enables the organization to process and analyse quality data more effectively, as part of the design science research methodology employed in the master thesis.

2.1 Software Quality

An in-depth review of literature on software quality plays a significant role in the project's success. This research aids in grasping the key concepts, theories, and frameworks integral to software quality. Examining such relevant research offers insight into the definition of software quality and its relevance to the project goals. Established models and structures for software quality, known to ensure software's efficacy, are also investigated. Other areas of focus include various methods to assess software quality, which are routinely utilized to monitor software performance. Lastly, an exploration is made into the challenges and hurdles faced by companies in the quest for achieving and maintaining high software quality. This methodical approach to research ensures that the

development of the automated data collection and dashboard solution is knowledgeable, keeping pace with the latest industry insights and best practices in software quality.

2.1.1 Definition of software quality

Over the years, the concept of software quality has undergone significant development, with numerous academics and industry professionals putting forth various viewpoints and structures. The section will present and discuss different definitions of software quality offered by researchers in the field.

One can trace the concept of software quality to the early work of McCall et al. (1977), who suggested a model that consisted of 11 essential factors for evaluating software products. These factors include reliability, maintainability, usability, and efficiency. This model was a critical milestone in comprehending and defining software quality, supplying a basis for additional research.

During the 1990s, the ISO/IEC 9126 standard materialized as an internationally acknowledged definition of software quality. The standard divided software quality into six principal characteristics: functionality, reliability, usability, efficiency, maintainability, and portability (ISO/IEC 9126, 1991). This standard furnished a comprehensive framework for appraising software products and is still employed as a reference in numerous research studies.

The Capability Maturity Model (CMM) by Paulk et al. (1993) further underscored the significance of software quality by delineating five maturity levels for software development processes. The CMM was later succeeded by the Capability Maturity Model Integration (CMMI), which broadened the scope to encompass multiple disciplines and aimed to enhance software quality through process improvement (Chrissis et al., 2003).

Kitchenham and Pfleeger (1996) presented an all-encompassing perspective on software quality by delving into its intricacies and obstacles. they examined an array of quality models, metrics, and evaluation techniques, emphasizing that a single model or metric is insufficient for capturing the true nature of software quality. Their work emphasized the significance of taking multiple factors into account when assessing software quality, including the attributes of the software product, the development processes utilized, and user satisfaction levels.

In conclusion, the definition of software quality has progressed over the years, with various models and frameworks proposed by researchers. From the initial work of McCall et al. (1977) to the ISO/IEC 9126 standard and Kitchenham and Pfleeger (1996) challenging the previously proposed models, the field has made significant strides. This section has delved into the primary definitions and perspectives on software quality, offering a foundation for an all-encompassing understanding of the concept.

2.1.2 Importance of software quality

In an increasingly digital age, the need for software quality is crucial. It is crucial to assure the durability, reliability, and security of software systems given the rising reliance on them. The significance of software quality, its components, and the effects of poor quality will all be covered in this section.

One of the software quality characteristics that we mentioned earlier was reliability, described as the likelihood of a system performing its intended function without failure over a specified period (Lyu, 1996). Highly reliable software systems not only decrease the chances of failures but also reduce the associated expenses, which can be significant (Jones, 2008). Furthermore, the importance of software reliability is heightened in critical systems such as healthcare, aviation, and nuclear power plants, where failures can result in disastrous outcomes (Avižienis et al., 2004).

Another important characteristic of software quality is usability, as it directly affects user satisfaction and productivity where poor levels of usability can lead to increased user errors, decreased productivity, and a higher likelihood of software rejection (Nielsen, 1994). Effective usability encompasses ease of use, learnability, and the overall user experience (Shneiderman et al., 2016).

Software maintainability, how easily software can be modified to fix defects, enhance performance, or adapt to new requirements, is also essential for long-term success (Pigoski, 1996). Glass (2001) states that maintenance activities account for approximately 40-80% of software development costs, highlighting the importance of considering maintenance in software development projects. Therefore, emphasizing maintainability in software development can lead to more manageable software evolution, which ultimately contributes to cost savings and extended software life (Lehman, 1980).

Security is a vital aspect of software quality, as it directly influences user trust and can have severe financial and reputational consequences for organizations (Howard & Lipner, 2006). The rapidly evolving landscape of cybersecurity threats highlights the need for robust and secure software systems to address these challenges. Research demonstrates that incorporating security practices throughout the software development lifecycle, such as threat modeling and secure coding, can significantly improve software security (McGraw, 2006).

In summary, the significant impact of software quality on reliability, usability, maintainability, and security is evident. Achieving high-quality software systems is crucial for user satisfaction as well as for business growth, safety, and reputation. As technology develops, placing a strong focus on software quality will continue to be an essential part of efficient software development.

2.1.3 Software quality models and frameworks

Software quality models and frameworks play a crucial role in assisting project managers, software developers, and other stakeholders in comprehending, assessing, and improving software quality. Many models and frameworks have been put forth to create a systematic method of assessing and ensuring software quality. The most well-known and frequently used software quality models and frameworks are the topics of this section.

The McCall model, created by McCall et al. (1977), is one of the first and most important software quality models. The 11 quality elements introduced by this approach each stand for a distinct aspect of software quality, such as reliability, maintainability, and usability. As one of the earliest attempts to provide a thorough framework for evaluating software quality, the McCall model served as the foundation for many subsequent software quality models.

Another significant model is the ISO/IEC 9126, which was later replaced by ISO/IEC 25010:2011, known as the System and Software Quality Models (ISO/IEC, 2011). These models propose a hierarchical structure that consists of quality characteristics and their related sub-characteristics. The ISO/IEC 25010 model expands on its predecessor by considering eight main quality characteristics: functional suitability, reliability, performance efficiency, usability, security, compatibility, maintainability, and portability (ISO/IEC, 2011). These models have gained widespread acceptance and usage, as they provide a standardized and comprehensive approach to evaluating software quality.

Another important framework for managing software quality is the Capability Maturity Model Integration (CMMI)., it was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University, and it provides organizations with guidelines to enhance their processes and ensure the development of high-quality software (Chrissis et al., 2011). The CMMI model identifies five maturity levels, each featuring a set of process areas that organizations should address to improve their software development processes.

Besides these widely recognized models and frameworks, researchers have also suggested various domain-specific quality models. For instance, Kitchenham and Pfleeger (1996) introduced the Quality in Use Model (QUIM) to assess software quality from an end-user perspective, taking into account factors such as effectiveness, productivity, safety, and satisfaction. Similarly, Olsina et al. (2001) proposed the Web Quality Model (WQM) to address the unique quality requirements of web applications, incorporating elements like content, usability, navigation, and presentation.

Elektrobit is currently following ISO 9001 which is a global standard for quality management systems (QMS) established by the International Organization for Standardization (ISO). This standard outlines criteria for a QMS, focusing on principles such as customer-centricity, leadership engagement, process orientation, continuous improvement, and evidence-based decision-making (International Organization for Standardization, 2015). Other quality standards that Elektorbit is following are ISO 14001, TISAX, ISO 26262, 61508, ISO/SAE 21434:2021 and Automotive SPICE.

Overall, software quality models and frameworks are essential for ensuring the development of high-quality software. Models such as McCall, ISO/IEC 25010, and CMMI have laid a solid foundation for software quality evaluation. Nevertheless, as software development practices continue to progress, ongoing research and development of new or adapted quality models are necessary to address the unique challenges and requirements of emerging software development paradigms.

2.1.4 Software quality metrics and measurements

Software quality metrics and measurements serve a vital role in assessing the effectiveness and efficiency of software systems. The body of literature surrounding this topic is extensive and has experienced significant evolution over the past few decades, as researchers have identified a plethora of metrics and measurements that aid developers and managers in ensuring their software products meet or surpass the necessary quality standards (Kitchenham et al., 2002). This section aims to offer a comprehensive understanding of the most relevant and current software quality metrics and measurements.

Fenton and Pfleeger (2014) explain that software quality metrics represent quantifiable attributes that allow for the assessment and comparison of software products or processes.

Software quality measurements, on the other hand, are the values or outcomes obtained through applying these metrics.

Due to the complicated nature of software systems, numerous metrics have been developed to concentrate on various aspects of software quality. McCabe's (1976) cyclomatic complexity metric is a widely acknowledged metric that evaluates the structural complexity of a software module. In more recent years, Halstead's (1977) software science metrics, such as volume, effort, and programming level, have been utilized to assess the complexity of software products.

Additionally, code-level metrics, such as the Chidamber and Kemerer (1994) suite of object-oriented metrics, encompass the depth of inheritance tree, coupling between objects, and lack of cohesion in methods, and have been extensively researched and applied in the realm of software quality. The development of the Goal-Question-Metric (GQM) approach by Basili et al. (1994) has proved instrumental in defining and selecting software quality metrics based on specific goals and questions that need addressing in a particular software project.

Elektrobit tracks a variety of quality metrics that differs from one project to another, these metrics belong to different categories such as test coverage, code analysis, process, requirement tracing and documentation metrics, where each category contains multiple metrics that are tracked and dealt with daily to ensure the maintainability of high quality of the projects across the whole organization.

In conclusion, the importance of software quality metrics and measurements in assessing and improving software products' quality cannot be overstated. Numerous metrics have been developed to tackle various facets of software quality, including complexity, size, and reliability. By effectively employing these metrics and measurements, combined with the use of automated tools and adherence to best practices, software systems' quality can be substantially enhanced.

2.1.5 Challenges in ensuring software quality.

In the field of software development, maintaining high-quality output is of paramount importance. However, the task of achieving and ensuring this quality presents its own set of hurdles, as is evident from the body of research on this subject. The literature points out an array of issues, from evolving project specifications to insufficient validation procedures, as well as documentation shortcomings, communication breakdowns within teams, and ineffective application of available quality assurance resources.

Jones (2010) illustrates the obstacles posed by altering requirements throughout the software creation lifecycle. According to his findings, these shifting specifications frequently result in overruns in both cost and time, detrimental factors that can erode the quality of the resultant software. The author also puts forth the argument that a lack of clear, comprehensive documentation can lead to misunderstandings about software functionality. Such misconceptions, in turn, can give rise to bugs, significantly impacting the quality of the software.

The process of validating software through testing is another challenge, as brought to light by Kitchenham & Pfleeger (1996). These authors drive home the point that not just the presence, but the comprehensiveness of these tests play a vital role in ensuring software

quality. They argue that inadequate testing or poorly constructed test cases may allow faults to slip into the final product, thus negatively impacting quality.

Lastly, Whittaker (2000) highlights the importance of robust communication within development teams. He underscores the point that ineffective communication can result in misinterpretation of requirements, which can subsequently compromise software quality, and he also highlights the problem of inefficient usage of quality assurance tools and techniques available, and its negative effect on software quality.

To summarize, the body of literature identifies a complex, interconnected network of challenges in achieving software quality. Future investigations should concentrate on devising strategies to navigate these issues, with particular attention given to the enhancement of requirements management, testing, documentation, intra-team communication, and the competent usage of software quality assurance tools.

2.2 Data collection from information systems

Undertaking an exhaustive literature review on data collection from information systems is necessary for the project's successful realization. This study reveals the variety of methods used to gather data from these systems, spanning both time-tested and innovative strategies. By exploring the related academic sources, an enhanced understanding of the different types of data typically extracted from information systems can be achieved. This exploration further unravels the importance of data quality. The literature also brings to light the inherent difficulties in obtaining data from these information systems. With such comprehensive knowledge, an automated data collection procedure that effectively addresses these challenges can be designed, optimizing the collection of quality data from the respective information systems.

2.2.1 Data collection techniques from information systems

The field of Information Systems incorporates several data collection techniques, each offering unique benefits and challenges. The specific data collection technique applied heavily depends on the data source and the desired form of the data.

APIs, as elaborated by Perera et al. (2013), act as gateways to extract data from online platforms and services in a structured and controlled manner. However, the data retrieved via APIs is typically bounded by the limits set by the API provider. The flexibility of SDKs, as discussed by Shull et al. (2007), can alleviate these constraints. SDKs, with their pre-written code and libraries, offer a means to develop custom software that can manipulate and process data more effectively.

Web scraping, another key data collection method, becomes crucial when API or SDK access is insufficient or unavailable. Heer and Bostock (2010) noted the use of interactive visualization as a potential way to understand and manage data collected through web scraping. While this method may be more labor-intensive, it offers an alternative path to data acquisition.

Direct database access, particularly from NoSQL databases, is a common approach for handling large, structured data repositories (Sadalage & Fowler, 2013). However, access can often be restricted due to privacy and security considerations. Finally, file importing allows us to handle offline datasets, such as spreadsheets or text files, as discussed by

Teece (1986). Although this method may appear less technologically advanced compared to others, it is essential for accessing long-term, rich datasets not readily available online.

Chen et al. (2012) emphasized the significant role that these data collection methods play in Business Intelligence and Analytics. By turning collected data into actionable insights, these techniques enable the conversion of 'big data' to a 'big impact'. Nevertheless, the reliability of the data sourced through these techniques' hinges on their proper implementation and validation. Halevi et al. (2017) also highlighted the importance of the careful evaluation of the sources, emphasizing Google Scholar as a viable source of scientific information.

To conclude, the selection of an appropriate data collection technique is of prime importance for delivering high quality software. Depending on the data source and the nature of the study, engineers might use one or more of these methods to gather data most efficiently.

2.2.2 Types of data collected from information systems.

There is an extensive body of literature that delves into the different types of data collected from information systems (IS). Central to this field is transactional data, primarily generated by the operations of an organization (Chen et al., 2012). As noted by Chen et al. (2012), transactional data, which ranges from sales receipts to order histories, is the most prevalent type of data collected by IS. It provides invaluable insights that facilitate trend identification, anomaly detection, and improved decision-making processes.

Another important form of data is demographic data. Bajari et al. (2019) argue that demographic data, detailing aspects such as age, gender, and ethnicity, is vital in shaping marketing strategies. Their research on the impact of big data on firm performance found a positive correlation between demographic data utilization and market segmentation, enhancing a firm's competitive positioning.

Kallinikos (2013) highlights the significance of behavioural data, which encapsulates the patterns of user interactions with IS. Behavioral data includes metrics like click-stream data, browsing history, and user engagement levels. Kallinikos (2013) posits that behavioral data offers profound insights into consumer behavior, which can significantly influence personalized service delivery.

Another data type emerging from information systems is web data, which covers various data forms gleaned from the internet, such as search queries, website traffic, and social media interactions (Bughin et al., 2010). The authors argue that businesses can leverage web data to better understand consumer preferences and trends, thereby informing product development and positioning strategies.

Finally, the advent of the Internet of Things (IoT) has spurred an increased focus on sensor data, captured from numerous sources like mobile devices, industrial machines, weather stations, and more (Hashem et al., 2016). Hashem et al. (2016) underscored the role of sensor data in real-time monitoring, predictive maintenance, and overall process optimization within organizations.

In conclusion, the literature recognizes transactional, demographic, behavioural, web, and sensor data as the key types of data collected from information systems. Each type

presents unique characteristics and applications, and their combined use enhances the functional efficiency of modern organizations.

2.2.3 The quality of data collected from information systems.

Data quality extracted from information systems has been a subject of immense research significance, given the crucial role of data quality in shaping decision-making, insights discovery, and policy development. An early influential study by Wang and Strong (1996) put forth a multifaceted understanding of data quality, encompassing elements such as completeness, accuracy, timeliness, and consistency, thus laying the foundation for future studies on data quality in information systems.

In a progressive stride, Pipino, Lee, and Wang (2002) proposed a quantifiable approach to data quality, building a framework to measure the same, thus substantiating the work of Wang and Strong. The development of this framework became a turning point in the perception and measurement of data quality.

Batini and Scannapieco (2006) continued this journey by diving deeper into the methodologies employed for data quality evaluation in information systems. They underscored the importance of comprehensive data quality assessment and brought to light the fact that the quality of data is as much about the data itself as it is about the processes involved in its production, maintenance, and application.

DeLone and McLean (2003) enriched the discourse further by exploring the impact of data quality on the efficacy of information systems. Their findings echoed those of previous studies (Madnick, Wang, Lee, & Zhu, 2009), reinforcing the belief that data quality could indeed affect the performance of information systems and thereby the organizations that depend on them.

Cappiello et al. (2004) took this discussion further by identifying a crucial challenge in this field. They noted that the heterogeneous nature of data sources often leads to data quality issues in information systems, with the handling of data quality in such scenarios involving complex procedures and necessitating the creation of superior data integration tools.

In a nutshell, the quality of data gathered from information systems is undeniably central to the operation and success of any organization. As such, it's crucial that we don't just look at the data itself but also at the processes of data creation, integration, and usage. Future research efforts would do well to focus on devising advanced methodologies and tools to secure high-quality data from diverse data sources.

2.2.4 Challenges in collecting data from information systems.

Collecting data from information systems can be faced with numerous challenges. This section aims to explore some of these notable challenges as they have been examined in various scholarly works.

Issues relating to data quality present one of the foremost challenges. As highlighted by Wand and Wang (1996) in their crucial work on the ontology of data quality dimensions, flawed data quality management can lead to sub-optimal decision-making within organizations. Their research underscores the need to understand and manage data

quality, particularly when drawing data from diverse information systems, where the quality may vary significantly.

Another fundamental difficulty lies in data integration. Many organizations operate multiple information systems, each with unique data formats and structures. In their study, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," Sheth and Larson (1990) spotlight the struggles of harmonizing data from disparate systems. Tasks such as schema matching and instance matching, integral to integration, represent persistent obstacles when collecting data from various information systems.

Moreover, data privacy and security have become key concerns, particularly following the introduction of rigorous privacy laws like the GDPR. Solove's 2006 work "A Taxonomy of Privacy" offers a comprehensive analysis of potential privacy issues arising during data collection from information systems. Solove argues that a lack of proper data governance and control over access could lead to a violation of privacy laws, further complicating data collection.

Additionally, the rise of Big Data has seen organizations grappling with the sheer volume of data. Jacobs (2009), in his piece "The Pathologies of Big Data," sheds light on how the immense volume of data can create information overload, adding to the data collection challenges.

Lastly, the lack of a universally accepted methodology for data collection from information systems adds to the complexity. Moody and Walsh (1999) highlight this issue in their work "Measuring the Value of Information: An Asset Valuation Approach," emphasizing the need for standardized methods in data collection, the lack of which can lead to inconsistencies and further obstacles.

In summary, several challenges persist in the realm of data collection from information systems, including problems with data quality, integration, privacy and security, volume, and the absence of a uniform method. These issues provide a platform for future research to devise practical solutions for more efficient data collection practices.

2.3 Analytics dashboard

An in-depth study of academic materials related to analytics dashboards is necessary for the successful completion of this project. This research provides essential knowledge about the role and importance of analytics dashboards in making data analysis simple and supporting decision-making. A detailed examination of the related scholarly writings allows for an understanding of the wide variety of analytics dashboards available. Furthermore, this study looks into the intricate process of building and improving analytics dashboards, discussing important factors, established guidelines, and effective visualization methods. This academic investigation also highlights potential challenges in the development of analytics dashboards. With this comprehensive understanding, it becomes feasible to create an intuitive, user-focused dashboard solution that efficiently tackles the challenges commonly faced in analytics dashboard development and improvement.

2.3.1 Definition and importance of analytics dashboard

In the world of business, the rise of analytics dashboards has been nothing short of revolutionary. These tools, which aggregate and visualize key data in an efficient, easily digestible manner, have found wide-ranging applications in various sectors (Few, 2006). To maximize the benefits they offer, one must delve into their precise definition and why they are integral to modern business strategies.

Furthermore, (Few, 2006), eloquently defines dashboards as a "visual display of the most important information needed to achieve one or more objectives, consolidated and arranged on a single screen so the information can be monitored at a glance." This definition highlights their inherent versatility, as they can be moulded to serve diverse business requirements (Rasmussen, Chen, & Bansal, 2009).

Their significance lies in their capacity to transform a multitude of raw data into visually appealing and interpretable formats. This transformation unveils patterns and trends that could have easily gone unnoticed in the raw data, facilitating informed decision-making processes (Few, 2006).

Moreover, dashboards are a driving force behind creating a data-driven work culture (Eckerson, 2010). When complex data is presented in an understandable format, employees across various levels of the organization can actively participate in data analytics, thereby raising the organization's overall data literacy (LaValle, Lesser, Shockley, Hopkins, & Kruschwitz, 2010).

The value of dashboards is further heightened by their ability to provide updates in real-time, a crucial feature in the fast-paced business world where decisions must be made quickly (Watson & Wixom, 2007). Additionally, dashboards give users the freedom to customize data visualization as per their needs, a feature Shneiderman (1996) recognizes as a key strength.

Various studies validate the significance of dashboards in decision-making processes (Eckerson, 2010; LaValle et al., 2010; Rasmussen, Chen, & Bansal, 2009; Watson & Wixom, 2007). However, it is worth noting that the effectiveness of a dashboard is closely tied to the quality and relevance of the data it displays (Few, 2006), underscoring the need for robust data management practices within organizations.

To sum up, in the data-driven business landscape of today, analytics dashboards are indispensable. Their relevance extends across industries and roles, aiding in efficient decision-making and fostering a culture that values data literacy.

2.3.2 Types of analytics dashboards

The advent and increasing prevalence of analytics dashboards as integral components of modern businesses and organizations has been well-documented in the existing literature. As explained by Few (2006), these dashboards consolidate crucial business metrics and data analytics into a singular screen, thereby permitting leaders to swiftly evaluate their organization's condition. Nevertheless, dashboards aren't a universal solution and they come in varied forms, each catering to unique requirements and proffering distinct functions.

For instance, operational dashboards are marked by their real-time updates, delivering swift snapshots of ongoing operations (Eckerson, 2010). These dashboards are equipped with alerts to facilitate immediate actions when necessary. Designed to closely monitor ever-changing activities or tasks demanding constant attention, they have proven valuable in fields like transportation, hospitality, and healthcare, where real-time information is indispensable (Wexler, Shaffer, and Cotgreave, 2017).

In contrast, strategic dashboards offer a macroscopic view of key performance indicators (KPIs), assisting management in the formulation of strategic decisions (Few, 2006). Unlike operational dashboards, these are updated less frequently - perhaps daily, weekly, or monthly, and provide a bird's eye view of organizational performance. Rasmussen, Chen, and Bansal (2009) contend that strategic dashboards play a crucial role in aligning with and maintaining business goals and strategies.

Further, analytical dashboards are versatile tools for data exploration and comprehensive analysis, with advanced features for filtering, drilling down, and visualizing intricate data (Few, 2006; Eckerson, 2010). These dashboards are primarily intended for data analysts and users with a deep understanding of data. They help reveal hidden insights and trends. Yigitbasioglu and Velcu (2012) underscore the role of analytical dashboards in promoting data-driven decision-making. There also exist niche types of dashboards, such as diagnostic dashboards, which offer detailed information about specific metrics (Eckerson, 2010).

The body of literature is clear that the type of dashboard chosen should be well-suited to the specific needs and capabilities of an organization. Additionally, irrespective of the type, an effective dashboard ought to be user-friendly, aesthetically pleasing, and capable of representing data in a coherent and interpretable format (Few, 2006; Yigitbasioglu and Velcu, 2012).

2.3.3 Design and development of analytics dashboards

The craft of designing and developing analytics dashboards is a key area of focus within contemporary data science and IT fields. Surveying the current body of literature reveals numerous considerations and strategies related to this undertaking.

The visualization component of dashboard design plays a significant role, as expounded by Few (2006). He contends that thoughtfully constructed visual elements enhance comprehension speed and promote prompt decision-making processes, which underscores the vital role of effective data representation. Additionally, user-friendly designs are not just about ease of use, they are also central to effective data communication within the dashboard interface (Yigitbasioglu and Velcu, 2012). Yigitbasioglu and Velcu (2012) stress the importance of real-time data, asserting that the chief value of dashboards comes from delivering timely and relevant information.

From a developmental perspective, the body of literature heavily endorses the use of agile methodologies (Few, 2006). These iterative techniques, often associated with software development, allow for continual refinement in response to user needs throughout the development process. Further, agile methodologies encourage collaboration between developers and users, ensuring that the resultant dashboards meet user requirements effectively (Few, 2006).

Existing literature also acknowledges the challenges associated with the design and development of dashboards. Among the most frequently reported issues is the complexity

of the information to be displayed, diverse user needs, and various technical constraints (Few, 2006; Yigitbasioglu and Velcu, 2012). Additionally, with the pace of technology constantly accelerating, it is essential to ensure that dashboards are scalable and adaptable to future changes (Gotz, Zhou, and Wen, 2006).

In conclusion, the process of designing and developing analytics dashboards necessitates an intimate understanding of visualization principles and user requirements, as well as the implementation of agile development methodologies. Even with the challenges involved, dashboards that are well-designed and well-developed play a crucial role in supporting data-driven decision-making processes.

2.3.4 Challenges in designing and developing analytics dashboards.

A myriad of hurdles exists when it comes to the creation and refinement of analytics dashboards, a topic many scholars have sought to dissect. Among these hurdles, one highlighted by Sacha and his associates (2015) is the management and interpretation of vast data sets. In order to distill meaningful insights, they argue that visuals must be effectively employed.

Parallel to this, Brehmer and Munzner (2013) explore the notion of 'information overload', or the overwhelming sensation that comes from an excess of data. They posit that one of the main challenges in the use of analytic dashboards is to deconstruct this complex data into a digestible format. By simplifying the presentation, users can avoid feeling swamped.

Another hurdle to consider is the broad spectrum of users that dashboards must cater to. This can range from data science veterans to those less technologically adept. Gotz and Wen (2009) put forth the argument that dashboards need to cater to this user diversity, offering variable detail levels according to the user's grasp of the subject matter. Reinforcing this, Yi et al. (2007) stress the importance of interactivity within analytics dashboards. They contend that user interactivity plays a crucial part in the data exploration process, adding an extra layer of complexity to design and development.

The capacity for adaptation is also a concern in dashboard design. As Rind et al. (2013) explain, the evolution of businesses may instigate changes in data sources and analysis needs, requiring dashboards to evolve in kind. This means that dashboards must be designed with flexibility and adaptability in mind. Alongside this, Wang and his colleagues (2008) raise the issue of real-time data processing and representation in analytics dashboards. They suggest that traditional dashboards often struggle to handle large volumes of real-time data, leading to inefficiencies and inaccuracies.

A significant concern, particularly in our digital age, is data privacy and security in dashboard design and development. As (Burrus, 2017) note, the integration of multiple data sources into one dashboard presents a risk of data breaches. They recommend the implementation of strong security measures during both the design and development phases to protect the privacy and integrity of the data presented.

Finally, user engagement must not be ignored. Tufte (2001) offers an approach centered on user experience, arguing that a well-crafted design should avoid creating confusion and should clearly present information, no matter how complex the data may be. This

user-focused design approach, while potentially adding to development complexity and timeline, is widely acknowledged as a key component of successful dashboard creation.

3. Research methods

The research methodology section of this master's thesis report will outline the approach and methods used to investigate and propose a solution to the challenges of improving data collection processes and developing effective visualization tools to analyze trends and patterns in quality data. In this study, the design science research approach will be employed to address the research question of how to design and evaluate an automated quality data collection process and a visualization system that enables the organization to process and analyze quality data more effectively.

This method involves a problem-solving approach that focuses on designing, implementing, and evaluating a solution to a real-world problem. The research methodology section will describe the research design, data collection methods, data analysis techniques, and evaluation methods used to develop the proposed solution. The goal of this thesis is to contribute to the development of more efficient and effective quality data collection processes and visualization systems in a real-world environment.

The objective of this master thesis report is to propose a software quality analysis solution that achieves three main goals: saving time, reducing manual work, and providing a smooth user experience with easy usage. By automating the data collection process and developing a visualization system, the proposed solution aims to minimize the time required for quality checks, freeing up resources for other development tasks. Additionally, reducing manual work will reduce the risk of human error and increase the accuracy of the analysis. Finally, by providing a user-friendly interface and intuitive features, the solution will ensure a smooth user experience, enabling users to easily understand and interact with the software quality analysis tool.

3.1 Selected Research method

Design science research (DSR) was deemed as an appropriate methodology for this master thesis, this is because DSR emphasizes the iterative process of design, implementation, and evaluation, allowing researchers to refine and improve their solutions over time based on feedback from stakeholders and users (Hevner et al., 2008). The research question involves the creation of a novel artifact, namely an automated quality data collection process and a visualization system, to address a specific problem, which is the need for the organization to process and analyze quality data more efficiently. The DSR approach allowed me to design and develop the dashboard in an iterative incremental manner which made it possible to provide value very early on, while also ensuring that the resulting solution is tested and validated through empirical evaluation. Through this methodology, this master thesis can contribute to the development of new knowledge related to the design and evaluation of automated quality data collection processes, integrating it to a visualization system and provide practical recommendations to organizations seeking to improve their software quality data management processes.

3.2. Evaluation

A well-designed evaluation plan is essential in DSR to ensure the quality and validity of the research. In this master thesis, the evaluation plan consists of both formative and summative evaluations.

DSR formative evaluation is an important part of any project that aims to create a user-centered design. The evaluation was conducted in the form of weekly meetings, the team focused on evaluating the features that had been implemented so far from the perspectives of usability, usefulness, and performance. The technical aspect of the implemented features was also reviewed to ensure that they were functioning as intended and to identify any potential issues. Additionally, the team discussed what features should be implemented next, considering feedback from users and any changes in project goals. This approach helped to ensure that the project was on track and that the artifact would meet the needs and expectations of its users.

The summative evaluation, on the other hand, was conducted after it was decided that the status of the artifact will be changing from a proof of concept to an official organization wide used quality metric analysis tool. The evaluation aimed at measuring the effectiveness and impact of the developed system and comparing it with the previous system. The evaluation includes an interview to collect data from relevant stakeholders, such as quality experts, project managers, and dashboard users. The interview questions are designed to measure the system's performance, usability, efficiency, and effectiveness. The collected data will be analyzed and used to evaluate the system's success in meeting the research objectives and answering the research question.

To ensure the validity and reliability of the evaluation results, the evaluation plan is carefully designed, and the evaluation data is collected and analyzed using rigorous research methods. The interview questions are tested to ensure they are clear and understandable to the participants. The participants are selected based on their expertise and experience around quality data collection and analysis to ensure the quality and relevance of the collected data.

3.2.1 What is a structured interview.

Structured interviews can be a useful research method for evaluating software, as they provide a systematic and objective approach to collecting data from users (Creswell & Plano Clark, 2018). One of the primary advantages of structured interviews for software evaluation is their ability to provide detailed and specific feedback from users (Kujala, 2003). By using a standardized set of questions, researchers can ensure that all users are asked the same questions, which can help to identify common themes and issues in the data. Additionally, structured interviews can be designed to capture both quantitative and qualitative data, which can provide a comprehensive picture of user experiences with the software (Creswell & Plano Clark, 2017).

3.2.2 Why was it selected as an evaluation method?

Structured interview was selected as the summative evaluation method for the project for several reasons. Firstly, as mentioned above, a structured interview allowed me to ask the same set of questions to all participants in the same order, which ensures consistency in

the data collected. This consistency is particularly important when comparing responses across different participants or groups.

Secondly, I wanted to measure specific variables, such as time efficiency, minimizing manual tasks, and improving ease of use. A structured interview enabled me to design questions that directly address these variables, ensuring that relevant data is collected and can help in evaluating the effectiveness of the proposed solution.

Finally, a structured interview is a reliable way to ensure that the results obtained are representative of the population being studied. By using the same set of questions and following a standardized procedure, the data collected is ensured to be consistent and accurately reflecting the experiences and opinions of the participants.

3.2.3 Interview questions

When selecting questions for the interview, I carefully curated questions that would test the application's usefulness, usability, and efficiency, as well as the alignment of the application with the initial problem statement and solution objectives, I also included questions about any challenges faced during the development process and how they were overcome.

Q1. How often do you use the dashboard?

This question helps assess the frequency of usage, which is indicative of the dashboard's utility and value to the user. If a tool is frequently used, it generally implies that it provides value to the user's workflow.

Q2. Can you describe the impact of the dashboard on your work and productivity?

Our purpose is to increase productivity and improve work efficiency. This question directly addresses this point and allows the user to express the extent to which the dashboard has achieved its goal.

Q3. What are your thoughts on the usability of the dashboard?

Developing an easy-to-use application was one of main objectives, understanding a user's experience with the dashboard's usability can guide future improvements.

Q4. What was the impact of the dashboard on the collaboration and communication among team members?

Dashboards are often used as a communication tool to ensure everyone is aligned and has access to the same data. This question explores how well the dashboard is facilitating team interactions and collaborations.

Q5. What is your opinion on the overall design and layout of the application?

Design and layout greatly affect the user experience and overall acceptance of the tool. This question is designed to get feedback about these aspects, including aesthetics and organization.

Q6. Are there any existing features that you think we should remove?

This question can help identify unnecessary or confusing features that are not adding value to the user experience. Eliminating these can streamline the tool and make it more effective.

Q7. Have you encountered any difficulties when using the dashboard? If so, what were they?

This question is designed to uncover any technical issues, usability problems, or areas of confusion that need addressing. It can guide the further development and troubleshooting of the tool.

Q8. Overall, are you satisfied with the developed tool?

This general question is aimed at understanding the overall user satisfaction with the tool. Satisfaction is a key indicator of the tool's success and acceptance by its users.

3.2.4 Details about the interview

All interviews were conducted within a two-week period with 10 participants, where each one of them was contacted individually to schedule the questioning session, their roles varied from one to another (project managers, developers, test engineers, scrum masters,). The interviews were held remotely by using Microsoft teams as the go to tool for all the participants and each call lasted from 15 to 25 minutes.

3.2.5 Interview results analysis

After conducting the structured interviews as part of the summative evaluation, I will analyze the results using several steps. Firstly, I will transcribe the interviews to ensure that I have an accurate record of the data collected. Next, I will clean the data and remove non relevant words (stop words, off topic, …), then for Q2 to Q5 I will choose themes based on the questions asked, the responses given, and the variables being measured. These themes will guide my analysis and help me to identify patterns and trends in the data, for Q6 and Q7 I will list all the answers of the participants, for Q1, I will categorize the answers into three categories (daily, weekly, monthly) and for Q8, I will just share the direct answers of the participants, as it is a yes or no question.

Once I have identified the themes, I will group the words and phrases from the interview transcripts into these themes, making it easier to identify patterns and trends in the data. I will then use frequency analysis to identify common themes and keywords, helping the team to understand which issues most discussed and which aspects of the quality metrics analysis dashboard were most important to the participants.

Finally, I will present the results of my analysis in tables and graphs, highlighting the key themes, common keywords, overall attitude toward the application, and any other notable findings. By following this process, I can ensure that I have a comprehensive understanding of the data collected through structured interviews and can use this information to evaluate the effectiveness of the quality metrics analysis dashboard.

Overall, the evaluation plan in this master thesis aims to ensure the quality and effectiveness of the developed system in processing and analyzing quality data within an organization. The formative evaluation enables the research to identify and address any issues that arise during the development process, while the summative evaluation enables the research to measure the system's success in meeting the research objectives and answering the research question.

3.2.6 Ethical considerations

In conducting this research, several ethical considerations were considered. First, all participants were informed of the purpose of the study, and their rights. Informed consent was obtained from all participants prior to their participation in the study. Second, all data collected was kept confidential and anonymous to ensure the privacy and confidentiality of the participants. Third, the data collected was used solely for the purpose of the research and will not be shared or used for any other purposes. Finally, the results of this research are intended to contribute to the advancement of knowledge and not to harm any individual or group. Overall, this study was conducted in a manner that is consistent with ethical guidelines for research involving human participants.

4. Design and Development

The design and development section of this project encompasses several key aspects, including the development methodology, development process, high-level architecture, and design considerations for different components. This section focuses on outlining the approach taken to design and develop the automated data collection and dashboard solution. Together, these subsections provide a comprehensive overview of the design and development process, guiding the implementation of the automated data collection and dashboard solution.

4.1 Development methodology

Agile was selected as the development methodology for this project, and this selection was primarily motivated by Agile's emphasis on adaptability, iterative development, and customer collaboration (Laoyan, 2022). Agile methodology aligns seamlessly with the design science research approach, where the process is cyclic, and artifacts are continuously designed, developed, tested, and refined based on empirical data. The feedback loop central to Agile is symbiotic with the iterative nature of design science, where each iteration of the dashboard development informs and improves the next. Furthermore, Agile's inherent flexibility allows for necessary adjustments to the dashboard based on evolving research findings and user feedback.

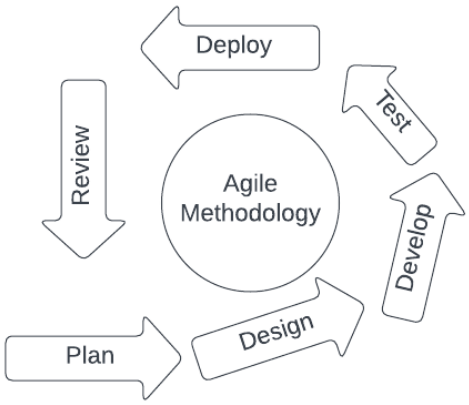

Figure 1. Agile methodology process diagram

Within the Agile spectrum, the choice of Kanban as the guiding framework facilitates clear visualization of the workflow, ensuring transparency and efficient handling of tasks during dashboard development. Kanban's principle of limiting work in progress helps manage the complexity of the project by encouraging focus on a limited number of tasks at a time, thereby reducing the possibility of errors and promoting quality (Martins, 2022). Moreover, Kanban's continuous flow model dovetails well with the cyclic nature of design science research. Together, Agile and Kanban foster an optimal environment for the design and development of a research-driven, user-centric dashboard.

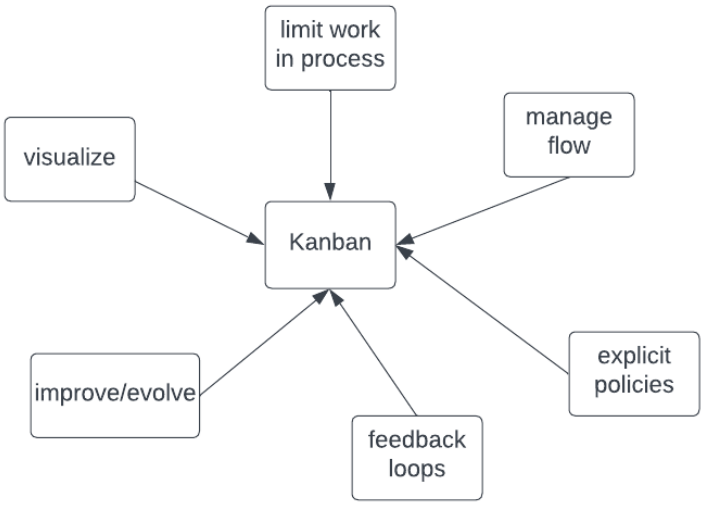

Figure 2. Kanban main principles donut chart

4.2 Development process

As mentioned before, since I am using design science research as the main research method and agile as the development methodology, it is to be expected that the development process will be composed of multiple development iterations, and since the project is still in proof of concept state, I wanted to deliver value to the users as fast as possible, and figure 3 explains perfectly the simple project development process, where it started with a kick off meeting where the customers explain their problem, followed by that, I started with short one week development iterations, where each week I have an hour meeting where the status of the project is discussed, evaluated and next iteration goals are defined.

When new features requests and updates stopped coming in, I decided to conduct a summative evaluation which was in the form of a structured interview that aimed to verify if I basically achieved my original goal which was to create a tool that facilitates software quality analysis, then it comes the maintenance phase to ensure the continuous reliable delivery of value to the users, and finally, the project can enter its end of life state if the tool is no longer needed by the main target users.

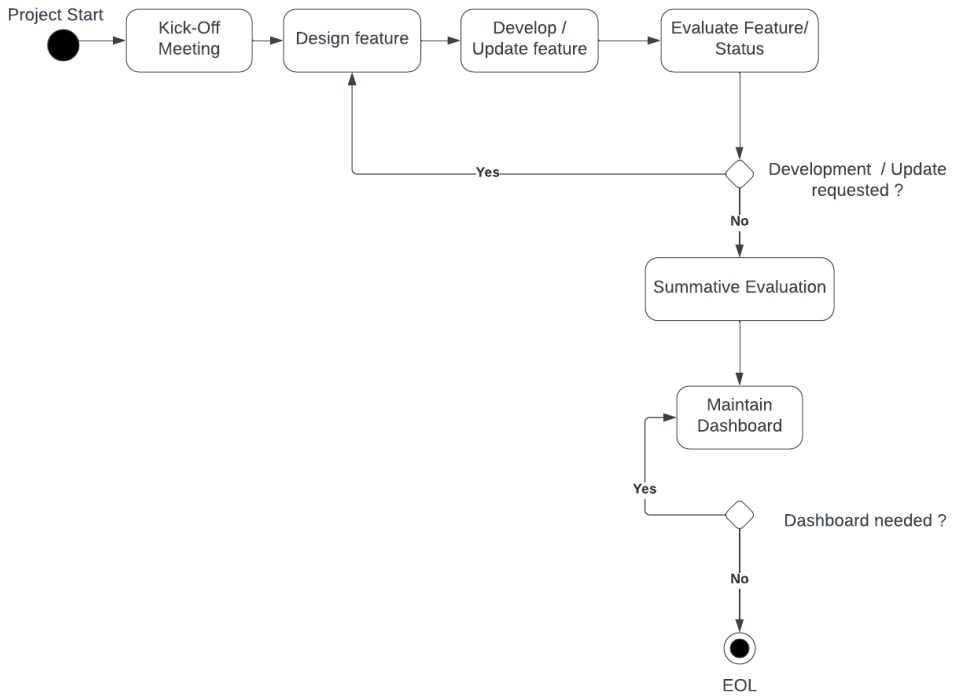

Figure 3. Development process activity diagram

Elektrobit has multiples software projects, where each one of them has numerous modules that function in different operation systems, so with the available resources , it was not feasible at all to develop a tool that analyse software quality for all the projects from the beginning, instead I had to follow the same process of iteratively and incrementally adding value to the users, as show in below diagram, I started with collecting data for one module for one project, and then slowly add more modules until one full project is finished, and then started the procedure to move to the other project

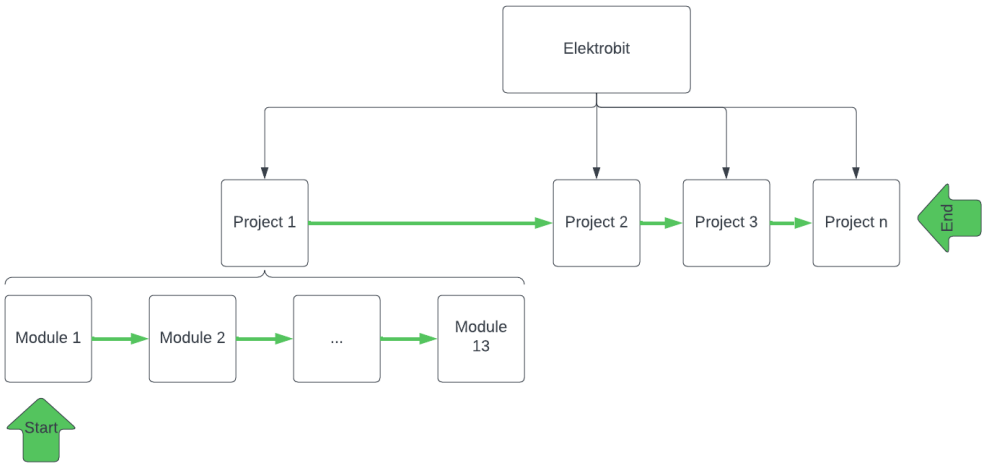

Figure 4. Organizational development process

4.3 High level architecture

The first format has two main components which is the data collection script written in python that connects to Jenkins REST API and collect necessary build information, metric values, and tests information from there, the reason python was selected due its advantageous readability, versatility, and vast library ecosystem (Babayev, 2021). Python's syntax is clean, and its code is generally easier to read and write, making it an excellent language for scripting and automation tasks. In the context of interacting with Jenkins, Python's request library simplifies making HTTP requests to the Jenkins API, providing an efficient way to retrieve necessary data. Moreover, the Jenkins API library offers a high-level, Pythonic interface to Jenkins, which abstracts away much of the low-level details of interaction, making the scripting process smoother. Then, after collecting the data from the Jenkins, I stored it in csv files due to their simplicity, human readability, and fitted the initial needs, the data was then visualized with the help of a library called streamlit which is an open-source Python library that allows developers to create web applications for machine learning and data science projects quickly and easily. It offers a powerful yet simple and clean solution for creating highly interactive web-based data applications with only a few lines of Python code (Singh, 2021), since as mentioned before, the project was a proof of concept, and I needed to provide value as fast as possible, so streamlit seemed like a great first option as it offers a straightforward Pythonic interface that lowers the barrier of entry, enabling me to focus more on data analysis and less on web programming.

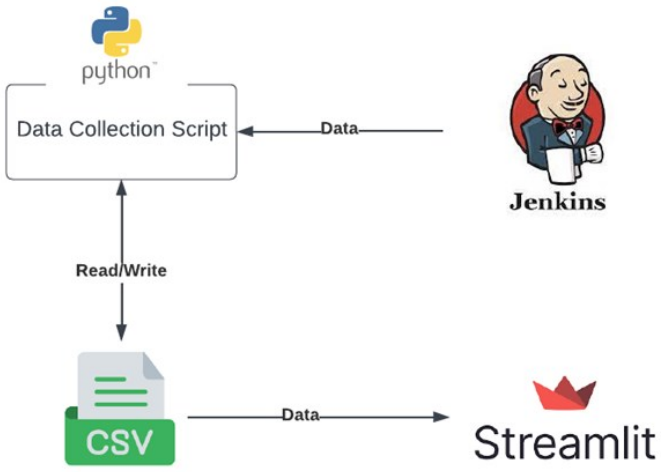

Figure 5. First version of high-level architecture

As I kept adding more features in the iterations, some of them required relational queries between different elements of the data such as builds, metrics, and tests. Furthermore, some of the features required inserting and updating data, and csv files are not optimized for this kind of operations, so I started to face difficulties to develop the requested features, and experienced decreased performance when trying to execute certain queries, so I decided to change the database technology, I wanted to go with MySQL as the relational database management system due to the relational nature of the data (a Jenkins build has tests and metric values), ease of scalability, high performance, secure access, and large active existing community.

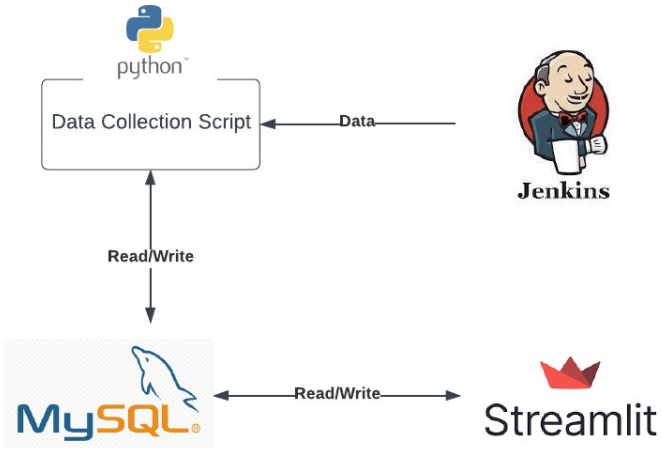

Figure 6. Second version of high-level architecture

Although streamlit is easy to use and speeds up the development process, it also had problems when it came to user interface customization, navigation, and state management, so it started to limit me when I wanted to implement features that required the aforementioned elements, which forced me to look for workarounds and complex solutions, consequently, I started seeing decreased performance and implementation obstacles, then it was more clear to me that streamlit works for best for single page applications with minimum backend logic, and user interface customization, which fitted the user's needs in the early iterations and allowed me to deliver values to customers as early as one month after starting the project, but due to new user needs, I decided to drop this library and move to something more robust, secure and performant.

Several options were considered such as Grafana, Kibana, and ReactJS, I ended up picking the latter, as it is component-based architecture which promotes reusability, maintainability, and testing of individual parts of the user interface, it also provides excellent performance (Kavinda, 2020), gave me the needed flexibility to smoothly implement the existing and planned features where the other options did not (database manipulation) , and finally because I had previous professional experience in this library, it minimized the learning process. But ReactJs alone would not work , as I needed another component that deals with the business logic such as calculating quality levels, trend data, quality table data, metric information and many more, I chose Flask to fulfill this role, which a python micro web framework that is light, very easy to use, has a small learning curve, simple and intuitive (Deery, 2022).

Furthermore, since it is a python framework, it gave me access to numerous data science libraries such as pandas, numpy, and Beautifulsoup, and finally, Flask has the blueprints feature, which gave me the ability to develop the backend application in an organized, easily readable, reusable, modular, and scalable way. I started fetching data from Jira as well , as one of new features gave the user the ability to link Jira tickets to metric values, so with each data collection script executing, I was fetching data from Jira to update ticket related details such as status, priority and assignee, but interacting directly with the Jira API from the backend web server had a slow response time, so I chose to implement the update process in the data collection script to maximum user experience and increase performance.

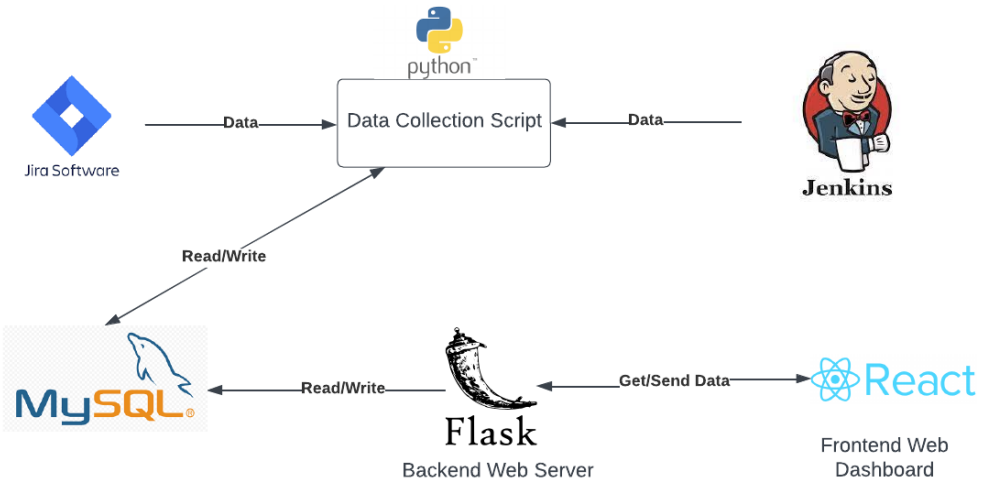

Figure 7. Current version of high-level architecture

4.4 Data collection script design

The script responsible for collecting the data is being executed in the start of every hour and was developed in an object oriented way so the main components of the code can be reusable ,modularized, and easy to read and maintain, it includes five classes : Metrics, Jenkins, Database, Jira and Main which is the class that is responsible for initializing all the previous classes, each one of them have their own purposes, with attributes and functions that only relate to the parent class,

This design made it easy to develop new features and debug potential problems, so if I wanted to start collecting data for a new metric, I would just need to create a function inside the Metrics Class that deals with this task and not worry about other classes. Furthermore, with this design I was making sure that the connection objects for different external tools such as the MySQL database, Jenkins, Jira are only accessible through their respective classes, which makes the script secure.

This design aligns well with Agile and Design Science Research methodologies, primarily because of its focus on modularity, reusability, and adaptability. Agile, known for its iterative and incremental development, is complemented by script encapsulation and modularity, allowing for components or features to be developed, tested, and improved in isolation. This enhances the efficiency of iterations and facilitates the continuous integration and delivery fundamental to Agile. Moreover, the emphasis on code reusability aligns with the Agile principle of maximizing work not done, enabling faster iterations, and reducing code redundancy. Within the Design Science Research context, this design assists in the creation and refinement of artifacts in iterative cycles, enabling easy modification and extension of features, which is central to the methodology.

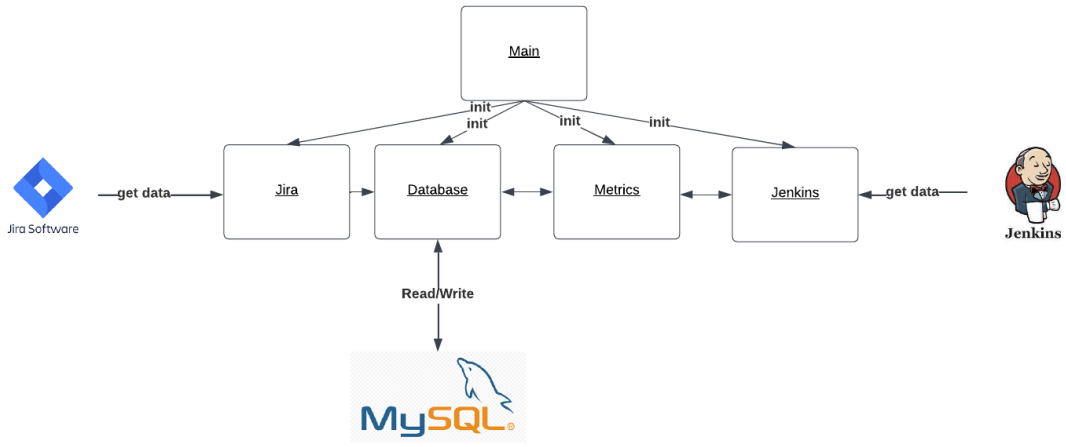

Figure 8. Data collection script design

4.5 Client Server Architecture

The architecture of the Web Dashboard is a client server architecture as shown in the diagram below, where the Flask framework is used for the server and the ReactJs Library is used for the client, the former deals with data retrieval, model processing, and business logic, and the latter is responsible for presenting data to the user, and the communication between the two is done with following REST methods such as POST, GET and DELETE, This separation made the dashboard more manageable, maintainable, and allowed me to focus on one part of the software at a time.

This architecture also enhances scalability, as the user base grows and add more projects, I can scale the server-side and client-side independently according to their respective resource demands, so if the server is experiencing heavy loads due to increased data processing requirements, the team can scale up the server resources without affecting the client-side. Furthermore, this separation also improves security of the dashboard since sensitive data and business logic are stored and processed server-side, reducing the risk of client-side data exposure or manipulation, it also gives me the ability to implement authentication and authorization measures at the server level to control access to the data. This can be further enhanced with role-based access control (RBAC) or other access control models to finely tune who has access to what data.

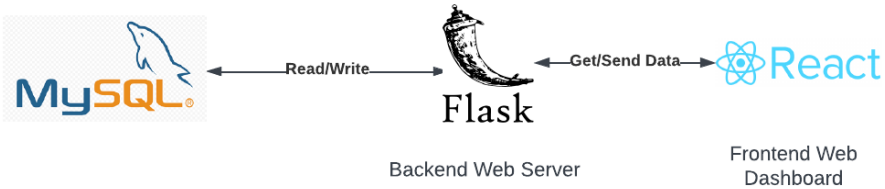

Figure 9. Dashboard main components

4.6 Backend application design

As mentioned earlier, one of the main reasons I chose Flask was because of its blueprints feature, which helped me organize and structure the application by grouping related API routes, vies, templates, static files together (Garcia, 2020). This helped me create the backend API with modular and reusable components, making it easier to manage and scale.

The concept of blueprints shares some similarities with the microservices architecture, since they both insist on modularity, separation of concerns, independent deployment, API contracts, flexibility, and scalability. Some of the main differences between the two are that Flask blueprints are primarily designed to organize and structure Flask applications internally, while microservices are an architectural approach for building distributed systems where each service is an independent and autonomous component. Furthermore, microservices often involve separate codebases, databases, and deployment environments, while Flask blueprints are typically part of a single codebase.

The main modules functioning in the application were Builds, Metrics, Tests, Jira, Comments, RF, and Monitoring. Each module had their own single purpose, API Route prefix, REST Methods, and related functions. Here is how the API URL was constructed:

Hostname : Port/ module / functionAlias

The diagram below explains the general process of communication between the backend and the frontend, where it starts with server launching, then its listen for incoming API Calls, executes a target function based on the above URL structure, and finally sends the data once function execution is finished.

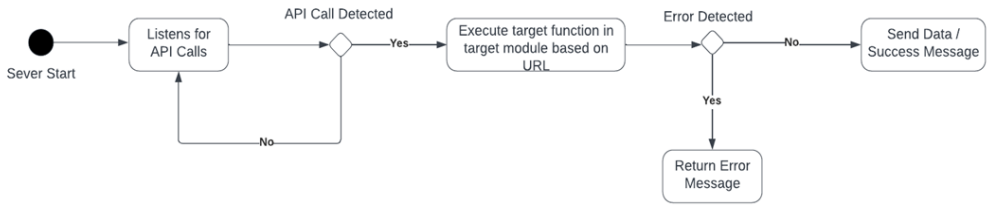

Figure 10. Backend general activity diagram

Here below is an example of the general process when a user wants to delete a comment related to a metric problem.

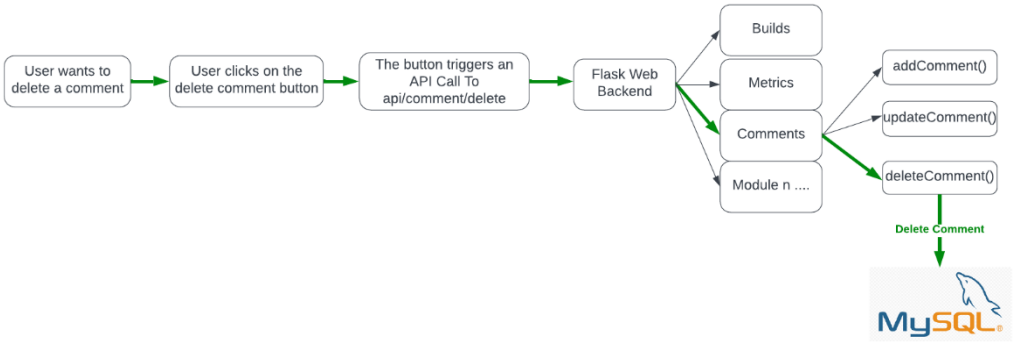

Figure 11. Example Backend process when a user wants to delete a comment.

4.7 Frontend application design

The development of the front-end application using ReactJS was largely centered around the principles of modularity and component reusability, ReactJS allows for the creation of UIs using a component-based architecture, meaning the UI is split into isolated, reusable pieces called components. These components can be composed to form complex UIs. Each component maintains its own state and logic, and can be reused throughout the application, promoting DRY (Don't Repeat Yourself) principles.

We first identified common UI patterns that could be abstracted into reusable components. For instance, metric data or Jenkins build information could be encapsulated in their own components, each responsible for rendering in a specific way. These individual components could then be reused wherever that specific visualization was needed. Here below is an example showing how the same DataChart component was used in 4 different views, and the same DataTable and TopCards Components were used in three different views, which made the code cleaner and more organized.

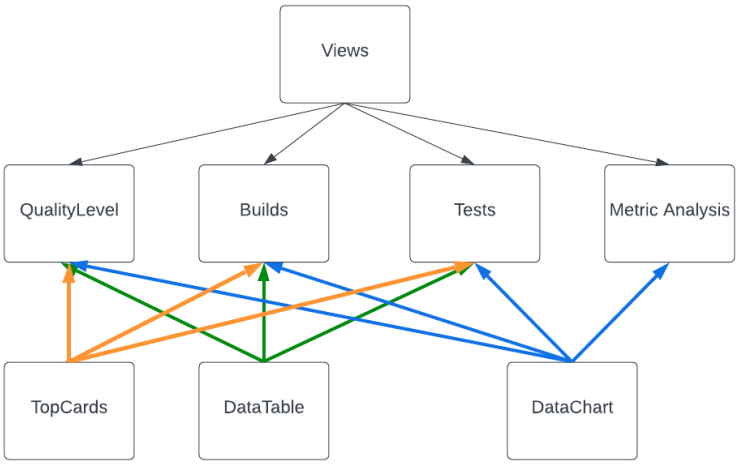

Figure 12. Diagram showing component reusability.

We also leveraged ReactJS's built-in component lifecycle methods to manage data fetching and state updates. Data fetching from the Flask API was handled in the React hooks useEffects method, ensuring that the necessary data was fetched from the server as soon as the component was inserted into the DOM.

For more complex interfaces, I used higher-order components and the composition model of ReactJS. Where these components take a component as an argument and return a new component with additional props or behavior. This allowed the application to share common functionality between components without duplicating code. Here below are the components building the whole User Interface from the beginning until the end.

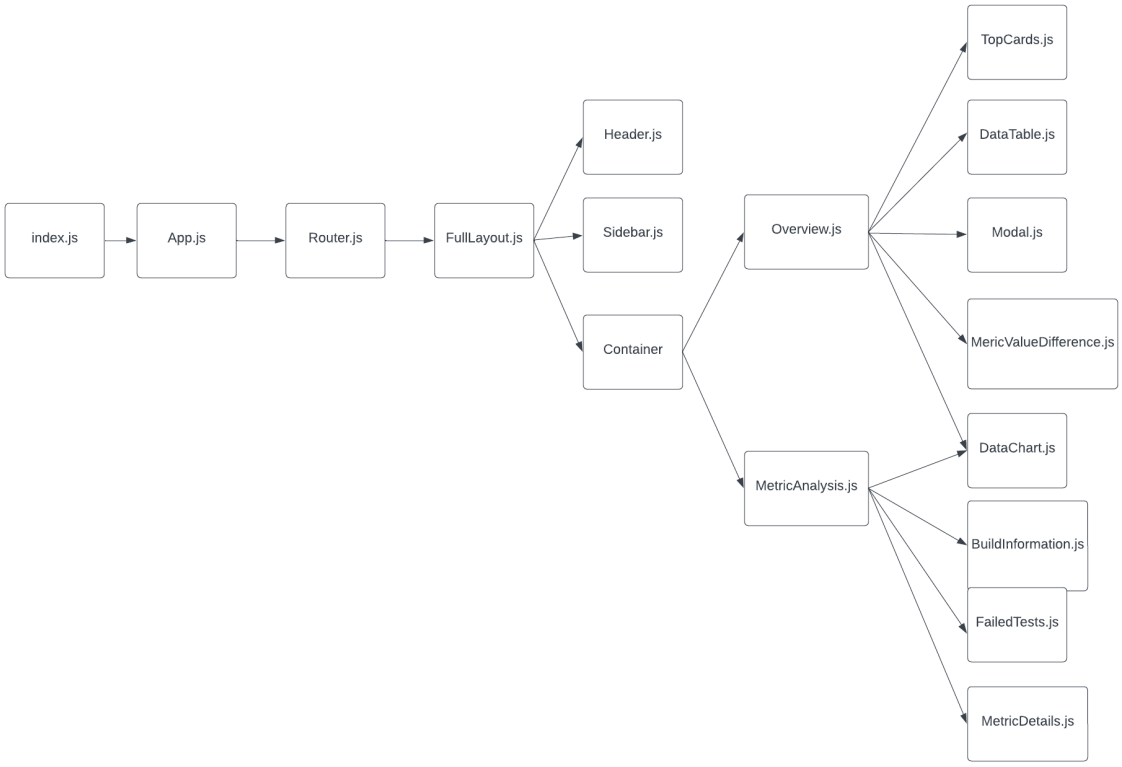

Figure 13. Diagram of all user interface components implemented in the frontend.

4.8 Database schema

We mentioned earlier that the nature of the data is relational, and that was one of the main reasons why I selected MySQL as the database system, so I used the foreign key constraint in every table, in the diagram below, I present the schema of the database, and at the center of it, is the MetricValues table , since the dashboard is mostly focused on software quality metric analysis. The metric values come from the Jenkins builds, and it is also related to JiraTickets and comments tables, since it helps the developers to know if a metric is being dealt with and its resolution progress. Furthermore, each metric has its own description, different quality level conditions, and associated categories, which is why the central table was linked to the metrics information table. The JiraTicketMetricRelation acts as a pivot table since the relation between MetricValues and JiraTickets is many to many, so each metric value can have different Jira tickets

linked to it and each Jira ticket can be linked to many metric values. Finally, I linked the tests table to the builds table because each Jenkins builds has several executed tests associated with it.

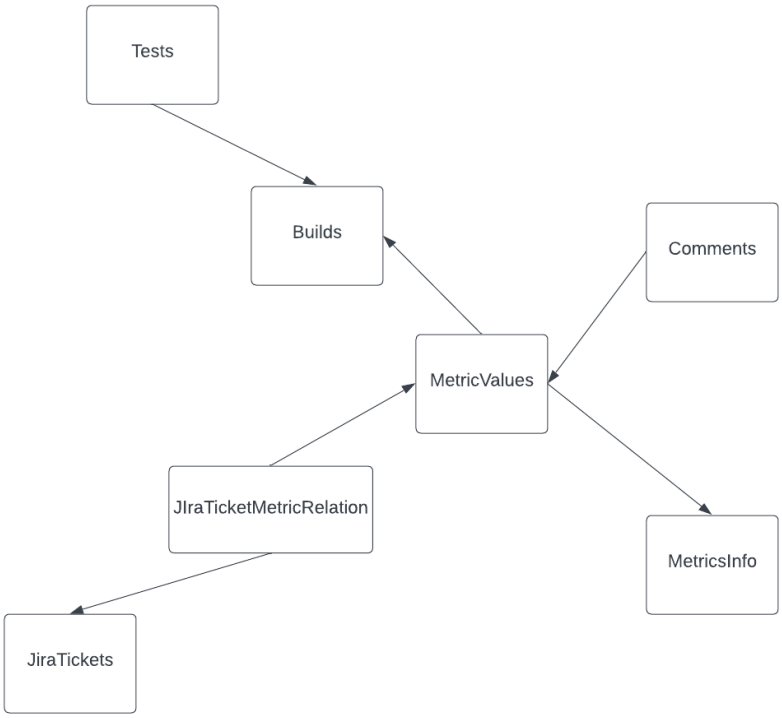

Figure 14. Database High level Schema.

5. Results

In this section, I will be presenting the current state of the dashboard that was designed and developed to optimize software quality analysis at target projects in Elektrobit, I will present different components and core features that are incorporated into the application and explain their functionalities and behaviors.

As mentioned in the previous section , the dashboard follows a layout that is composed of a header, sidebar and a container component that changes based on the page selected, in the following paragraphs, I will go through the core components and sub-components that are included in the layout, please keep in mind, the data is used in this section is template data, and does not represent the company's current quality levels.

5.1 Header

The header components is composed of four main elements, the title of the dashboard, which is the quality dashboard in the case, two major filters which lets the users select their desired module and operation system to track, mainly because the metric values are different between each module and operation system, and the final element of the header, is the data of the last successful data collection, so the user is aware if the data is up to date or not.

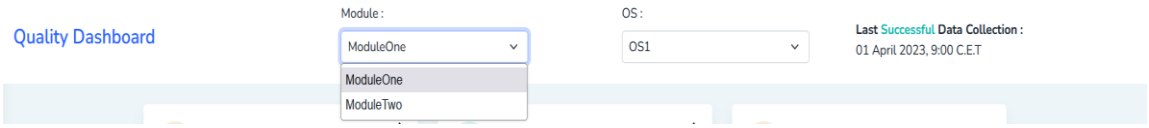

Figure 15. Screenshot of the header component.

5.2 Sidebar

The sidebar components is composed of five main elements, the first one is the company's logo, the other four elements are navigation links, the overview and metric analysis takes you to their respective pages where the current selected page is highlighted in blue, and the remaining two elements takes the user to the Jira's platform, where they are directed to Jira tickets template for either requesting a feature or reporting a bug depending on their selection.

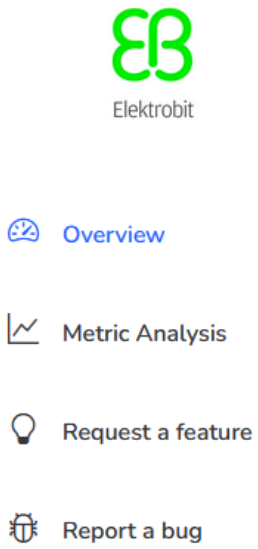

Figure 16. Screenshot of the sidebar component.

5.3 Overview page

5.3.1 KPI Cards

Currently, the main view has three KPI cards, where it shows the most important information about project quality, three of them come from one component with different data passed into it. Once a user, clicks on one of these cards, it changes the data of below components to match the selected card, so for example, if a user clicks on the build result KPI card, the table and trend chart component data will be Jenkins build related (job name, build result ,…).

The first card represents for a selected date , the quality status percentage which is calculated based on the values of a variety of software quality metrics that the project is currently tracking, the arrows in the card give the possibility to switch to another quality level, which could have different conditions that they need to respect, mainly because not all projects and modules are pursuing the same quality level, each project depending on its phase, is aiming at a specific level with its own metric value conditions, which why this arrow component is a key feature of the dashboard, as it gives to the user the ability to monitor the quality status of their project from different perspectives.

The second KPI card by default shows the percentage of successful Jenkins builds for a selected date, it is based on the build result attribute of the tracked Jenkins builds, as in this case, it shows the ratio of number of successful builds and all builds, the arrow components in this card let the user switch to other build results related percentages such as percentages of unstable, failed, and aborted builds.

The final KPI card, tracked the count of failed tests for a selected date, it does not have the arrows components, as currently, the users are only interested in tracking failed tests

in the quality dashboard, once users are interested in tracking passed, skipped, and other test statuses, I will add the arrow components to this card.

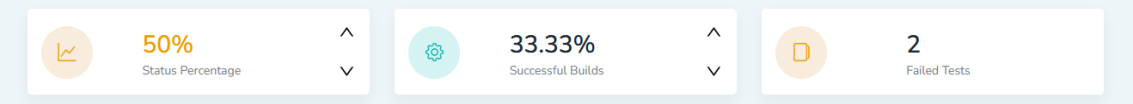

Figure 17. Screenshot of the KPI cards.

5.3.2 Quality status view

As mentioned earlier, the content of the table changes based on KPI card selection, so if the user selects the first KPI card, the table and the trend chart show the data and components that are related to the card, the purpose of this view is to show information about quality status and all metrics tracked by their project for a selected date. For instance, on the left side, the user can find a table that shows for each metric, its Jenkins verification source, value and status based on the selected quality level. In the value column, it shows the difference between the current value and the previous one, this helps the user to know if a metric value is getting worse or better, if a user clicks on one of the table rows, it takes into the metric analysis page where they can analyze the selected metric in more details. The component in this view has four filters, the RFM and not RFM checkboxes are responsible for showing metric that respects or does not respected the selected quality level conditions, which in this case the selected quality level to track is RFM, the date select box let the user consult metric statuses for different dates, and the final filter is the Jira View checkbox which changes the structure of the table to allow users to assign Jira tickets and comment to their metric values as shown in figure 19.

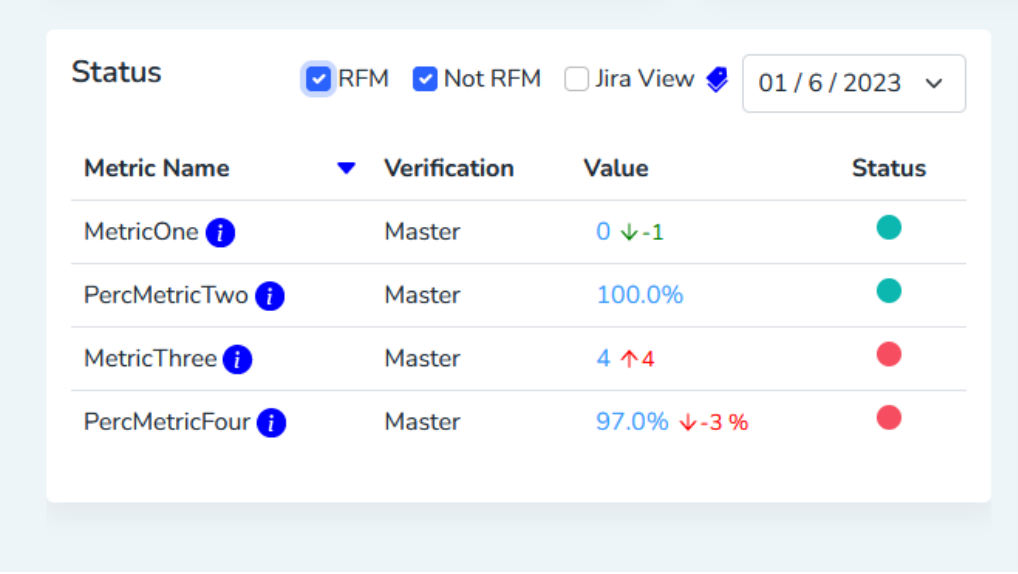

Figure 18. Screenshot of metrics status table.

The Jira view make the table width equal to the whole parent container width, so it fits all needed elements in easily readable way, it shares the same four columns as figure 18 add two new columns that let the users assign one or multiple Jira Tickets and comments to different metric values.

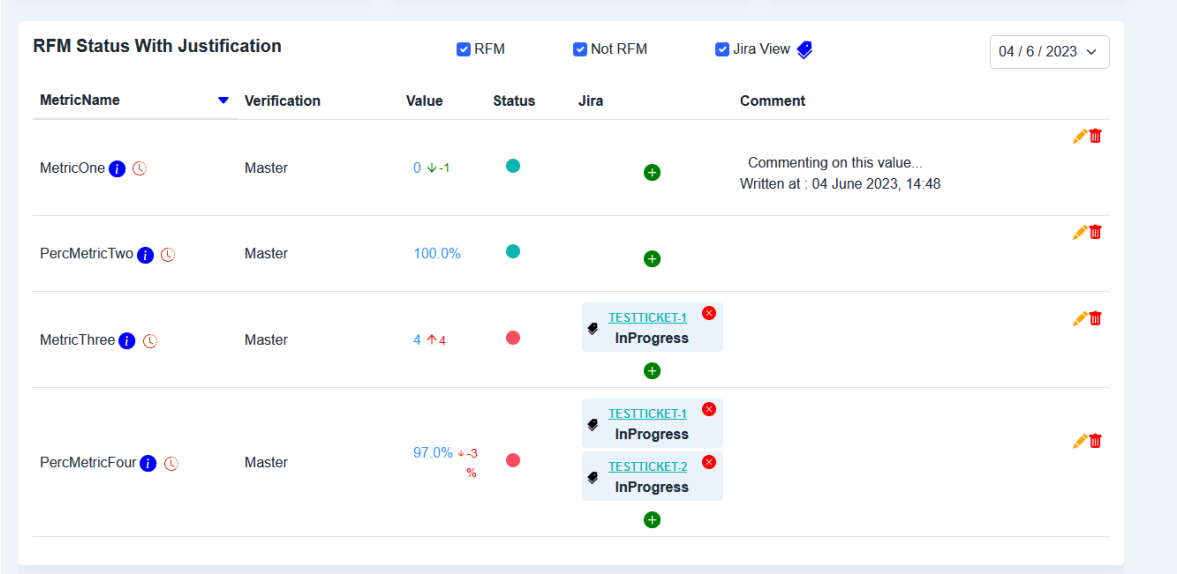

Figure 19. Screenshot of metrics status table with Jira filter selected.

To assign a Jira Ticket to a metric value, the user clicks on the plus button, and a modal shows up that lets them specify the name of the ticket, in case the ticket exists in the Jira platform, it is added to the database, if not, an error message is displayed. The user can later delete the relation between the metric and the Jira ticket by clicking on the red remove button located in the top right corner of the ticket.

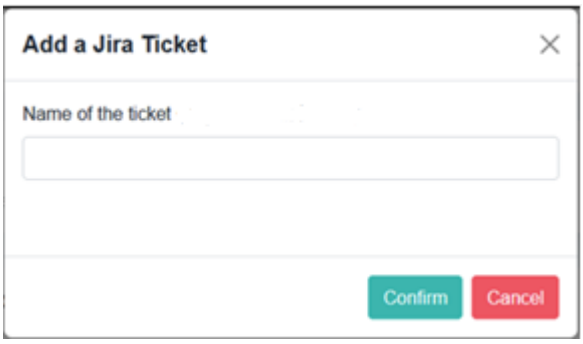

Figure 20. Screenshot of a modal allowing the user to link a Jira ticket to a metric value.

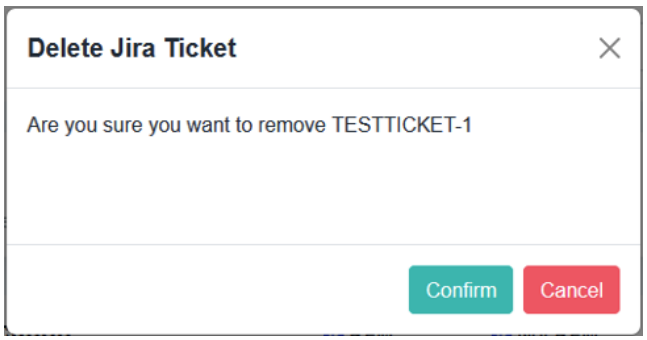

Figure 21. Screenshot of a modal allowing the user to remove existing relation between a metric value and a Jira ticket.

In case the Jira view is not selected, to the right side of the container, the user can find a trend chart showing the progress of quality level percentage for a selected date range, where it is also dependent on the quality level selected, the figure below shows that the user can track the RFM quality level for a selected date range. This component is important because it lets the user monitor and verify that the quality status is always improving and moving towards the right direction.

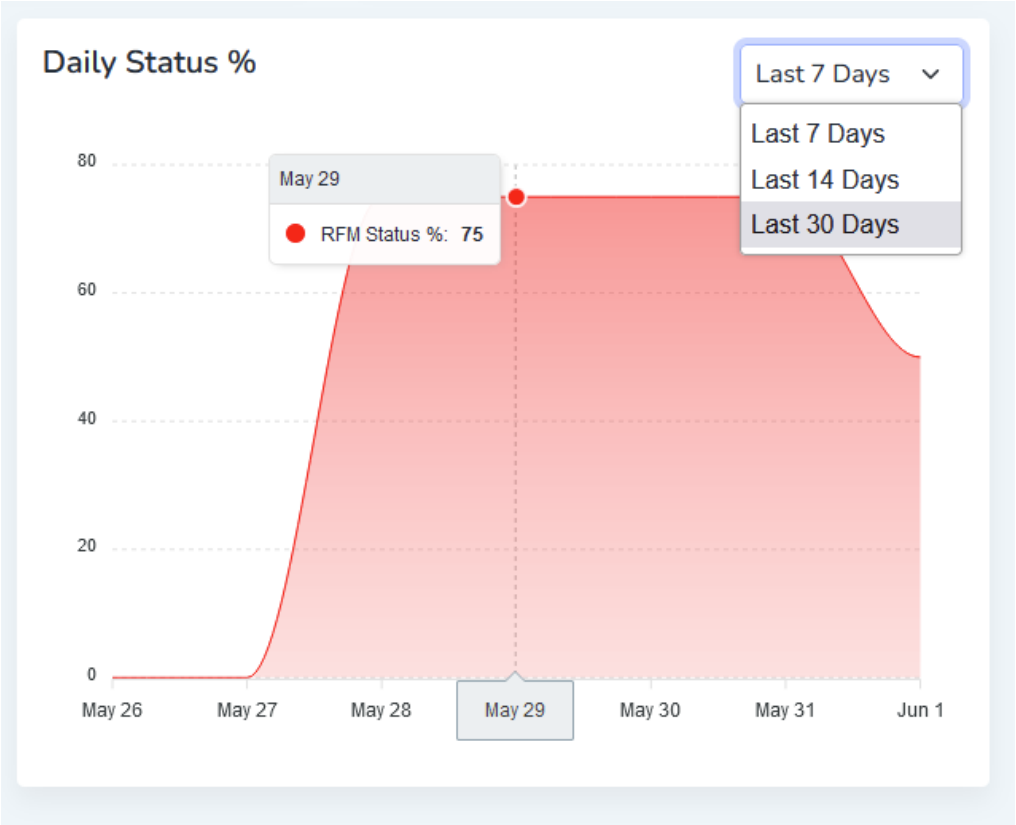

Figure 22. Screenshot of a quality status trend chart.

5.3.3 Build view.

Once a user selects the second KPI Card, the view in the below figure is shown, the same components of the previous are used, but different data and subcomponents are passed to it. In this view, the table component is showcasing Jenkins build related information for the selected date, such as the date, job name, build number, build result and the verification type, if a user clicks on one of the rows, it takes them to the Jenkins page of the selected build where they can dive into more details .Two filters are included in this table, once the user selects the “Most Recent” filter, the table only shows most recent builds for the same job, the second filter can be accessed by clicking the blue arrow in the result column, which brings up three checkbox that allows the user to show Jenkins builds based on the build results.

The trend chart component shows for a selected date range the count of executed Jenkins builds, categorized by their build results, the green line represents the number of successful builds, the orange line represents the number of unstable builds, and the red line represents the number of failed builds. This graph helps users to identify patterns and relations in Jenkins builds executions.

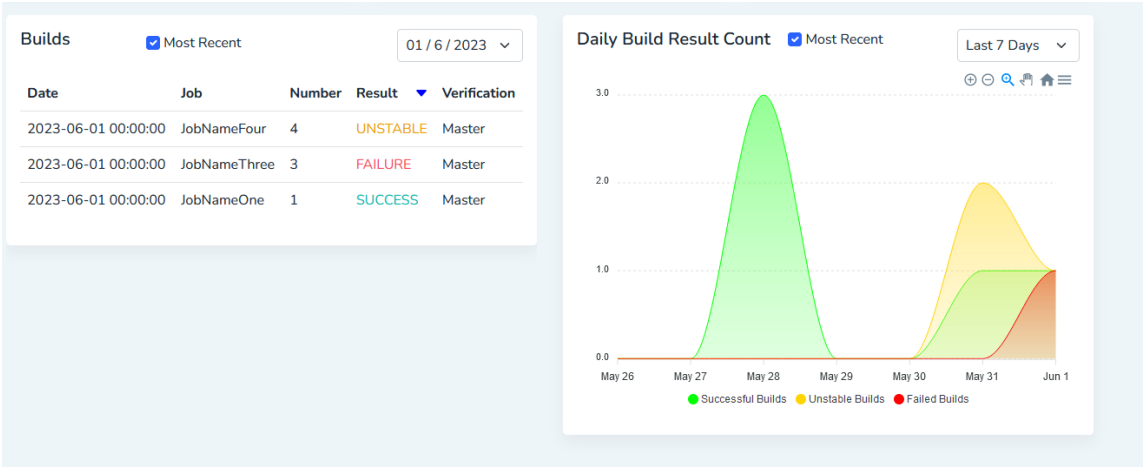

Figure 23. Screenshot of build table and trend chart.

5.3.4 Test view

In case the user selects the third KPI card, the view in the below figure is shown, where the same components of the two previous views are used, the table component in this view show the name of all failed tests for the selected date, it also has the same “Most Recent” filter as the previous and it functions the same way, as if selected, the table only shows the most recent version of a failed test. If the user clicks on one of the rows, it takes them to the Jenkins platform for the selected test, so they can access the full output of the test. The trend chart in this view shows the count of failed tests for a selected date range, this helps the user identify spikes and unusual patterns faster.

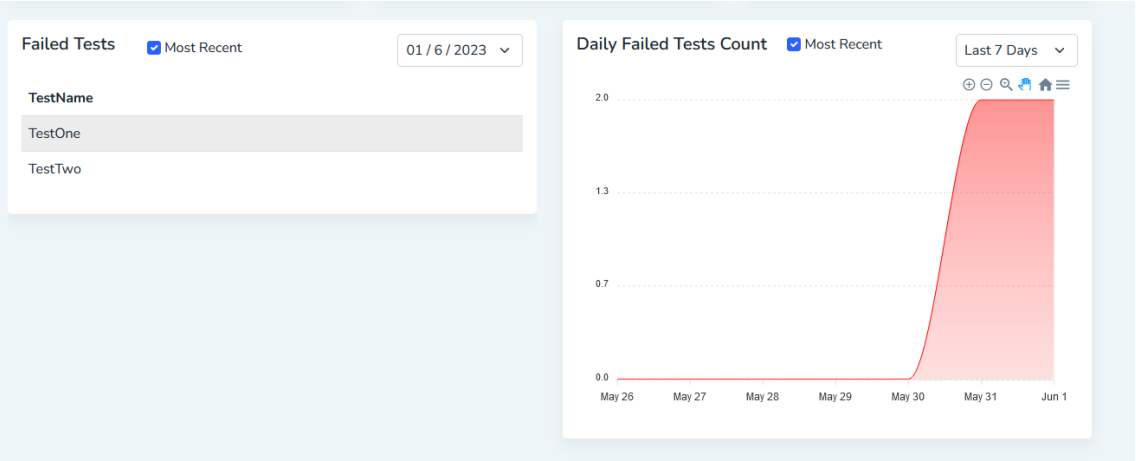

Figure 24. Screenshot of test table and trend chart

5.4 Metric analysis page

The metric analysis page can be accessed either by selecting it in the sidebar, or by selecting one of the rows in the metrics table in the overview page. The role of this page is to let the user either analyze metric values independently or stacked in different categories (coverage metrics, process metrics…)

The first component of this page contains two radio buttons and one select box, where the selected radio button directly affects the values of the select box and below content, if "Metric by Metric" is selected, then the select box options represent individual quality metrics and the content below only concerns the selected metric. If the "Metric Stacks" is selected, the select box options represent different metric categories that incorporates similar metrics together (coverage, system, requirements…) and the content below also is related to the selected metric category.

Figure 25. Screenshot of metric analysis filter bar.

5.4.1 Metric by metric view

This view incorporates two main components, the component on the left presents information about the source of the selected metric including official metric name, the source's job name, build number, build date, build result, verification type, and the metric value, the component has also a select box filter that lets the user select different dates which gives the user access to all historical values of the metric, this has been deemed

important by many users as it help them solve metric problems faster by following the same resolution method for previous cases.

The component on the right lets the users track the progress of a single metric value for a selected period, this can help the user identify patterns, spikes and anomalies faster.

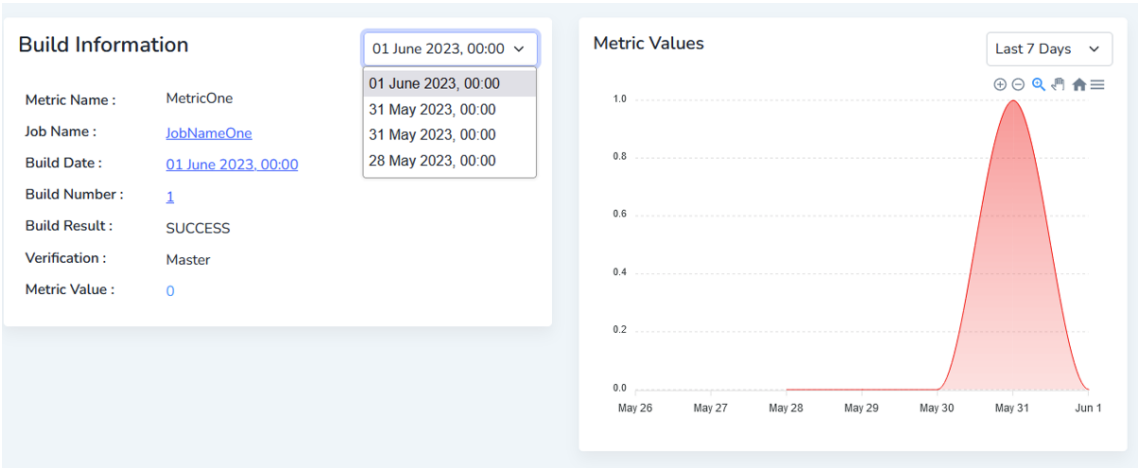

Figure 26. Screenshot of build information card and trend chart for the selected metric.

In case, there are failed tests related to the metric values, or additional details, the additional components in the below figure are shown. The component to the right helps the user get to the bottom of the problem faster by showing them the exact source of the problem along with useful error descriptions. Meanwhile, the component to the right showcasing all failed tests related to the metric problem, by resolving those failed tests, users can quickly bring the metric value to the required target.

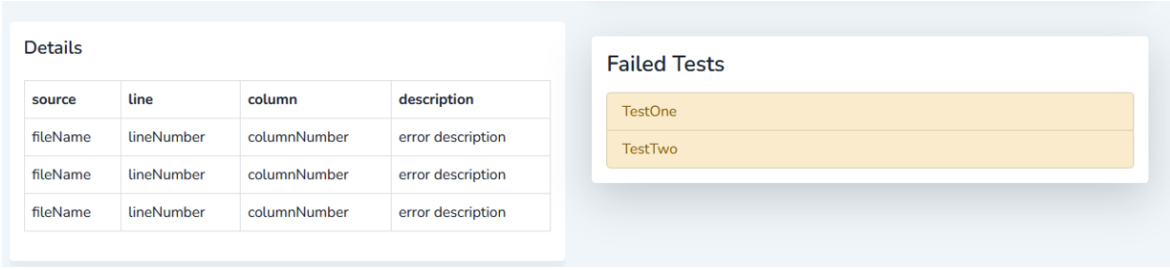

Figure 27. Screenshot of details table and failed tests.

5.4.2 Metric stacks view

In this view, the user can visualize the progress of metric values for a selected date range of different metrics that belong to the same category, this helps identifying possible relationships between different metrics, patterns, and anomalies.

Figure 28. Screenshot of a trend chart for metrics that belong in the same category.

6. Evaluation

In this section, I will provide a comprehensive report on the evaluation results of the software quality analysis dashboard. the focus is on illuminating the users' experiences, opinions, and overall satisfaction with the application, as gauged through structured interviews. The insights gathered from these interactions have been crucial for assessing the efficacy, usability, and impact of the developed tool on users' work processes, productivity, and team collaboration.

The goal of the user interviews was to explore eight key aspects: the frequency of use, impact on work and productivity, usability, effect on team collaboration, thoughts on design and layout, features considered unnecessary, difficulties encountered, and overall satisfaction with the tool. A thorough analysis of the gathered data has been performed to provide a deeper understanding of users' interactions with the dashboard.

Let's now delve into the specific results derived from structured interviews and the subsequent analysis. This will provide a valuable understanding of how the software performs in a real-world setting, and its alignment with the initial objectives set for its development.

6.1 Analysis of answers for Q1:

The first question to the participants was "**How often do you use the dashboard?**", I decided to represent the answers with a bar graph since I categorized the answers into three different categories (daily, weekly, monthly) and this type of graph allowed me to easily compare the frequency of usage between participants, as I can see from the graph below, nine participants reported that they are using the quality dashboard daily and only one participant reported using it weekly.

The frequency of dashboard usage reported by the participants signifies its relevance and success. With a significant majority using the dashboard daily, it demonstrates its high relevance to their work tasks and robust user adoption. This consistent daily usage indicates a strong user reliance on the tool, suggesting it is reliable and meets performance and utility expectations. The singular weekly user does not necessarily reflect a drawback of the dashboard but may instead represent job role variations or differing responsibilities among users. Overall, these findings denote that the dashboard effectively delivers valuable, frequently required information or functionalities.

Furthermore, the answers directly correlate with the richness and depth of the evaluation responses. As users interact with the dashboard more frequently, they gain a more nuanced understanding of its features, providing the evaluation with more detailed and insightful feedback.

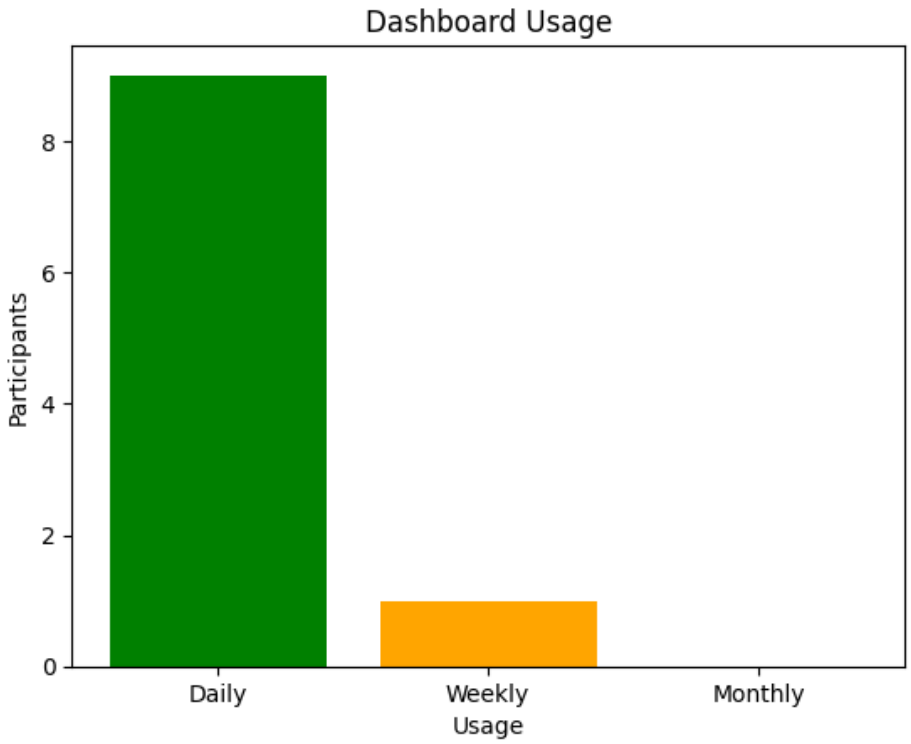

Figure 28. figure of a bar chart representing the frequency of usage among interview participants.

6.2 Analysis of answers for Q2 to Q5

As mentioned in the research methodology section, after conducting the interviews, I transcribed the participants answers, cleaned it and removed non-relevant information, and for each question, I selected themes that represent the answers, and also determined for each answer if it is a positive or negative answer. Thanks to themes classification, now the team knows for each aspect of the dashboard (usefulness, usability, communication,) what key features are the most important to the users, which will help the development team when they are prioritizing the backlog and improve overall decision making in the future. Furthermore, most answers to these questions were notably positive, indicating that the tool is successfully serving its intended purpose.

Table 1. Common themes and their representative quotes table for Q2 to Q5

Question	Most common themes	Representative Quotes
Can you describe the impact of the dashboard on your work and productivity?	Efficiency and Timesaving, Improved Quality Analysis, Data Accessibility, Improved Workload Management	*"All the information in one place"* *"It is efficient at highlighting problems at hand."* *"It makes understanding problems quicker and easier."*
What are your thoughts on the usability of the dashboard?	Ease of use, Performance, Navigation	*"It is easy to use, and it does not have performance issues."* *"There are no complex interactions."* *"I did not need instructions to use it"*
What was the impact of the dashboard on the collaboration and communication among team members?	Information Sharing, Communication efficiency, work distribution, team alignment	*"Everybody looks at data same way."* *"It opens discussions about problems and who will fix them."* *"I can communicate easier the root cause of the quality problem to my colleagues "*
What is your opinion on the overall design and layout of the application?	Aesthetics, Layout and Organization, Information density, Clarity	*"Good style and design"* *"Clear information positioning"* *"Organized without distractions"*

Then, I classified the participants answers into positive and negative attitude responses based on manual word frequency counting for the terms that related to their respective category.

Table 2. Participant attitude classification for Q2 to Q5

Question	Positive Attitude	Negative Attitude
Can you describe the impact of the dashboard on your work and productivity?	10	0
What are your thoughts on the usability of the dashboard?	9	*1*
What was the impact of the dashboard on the collaboration and communication among team members?	10	*0*
What is your opinion on the overall design and layout of the application?	9	1

6.3 Analysis of answers for Q6 and Q7

The unanimous feedback from participants regarding questions 6 and 7 demonstrates that the dashboard's design and functionality align well with the user needs and expectations, The fact that no user suggested removing any feature indicates that all existing features are providing value. This suggests that the initial design process, which presumably included careful consideration of the users' needs and a thoughtful selection of features, has been successful.

The lack of reported difficulties also speaks to the robustness and stability of the dashboard. It implies that the tool performs consistently without technical glitches or errors that can disrupt the user experience. This feedback suggests a strong alignment between the dashboard's features and the users' needs and workflows. If users find all features useful and encounter no challenges in using them, it implies that the dashboard has been successful in meeting its intended purpose and fitting seamlessly into the users' workflows.

6.4 Analysis of answers for Q8

The consistent positive feedback regarding satisfaction with the dashboard is a compelling testament to its success. It implies the tool effectively meets user needs, indicating a high rate of user adoption and an accomplished design process. Users' satisfaction signifies that the dashboard's features are beneficial and pertinent, delivering valuable information without unnecessary clutter. Furthermore, it suggests the tool performs reliably and consistently, free of major bugs or errors. This feedback also shows that users see the tool as a worthwhile investment of their time, further fostering continued use. Nevertheless, it's essential to sustain attention, regularly collecting user feedback and refining the tool, as user needs and technological landscapes evolve over time.

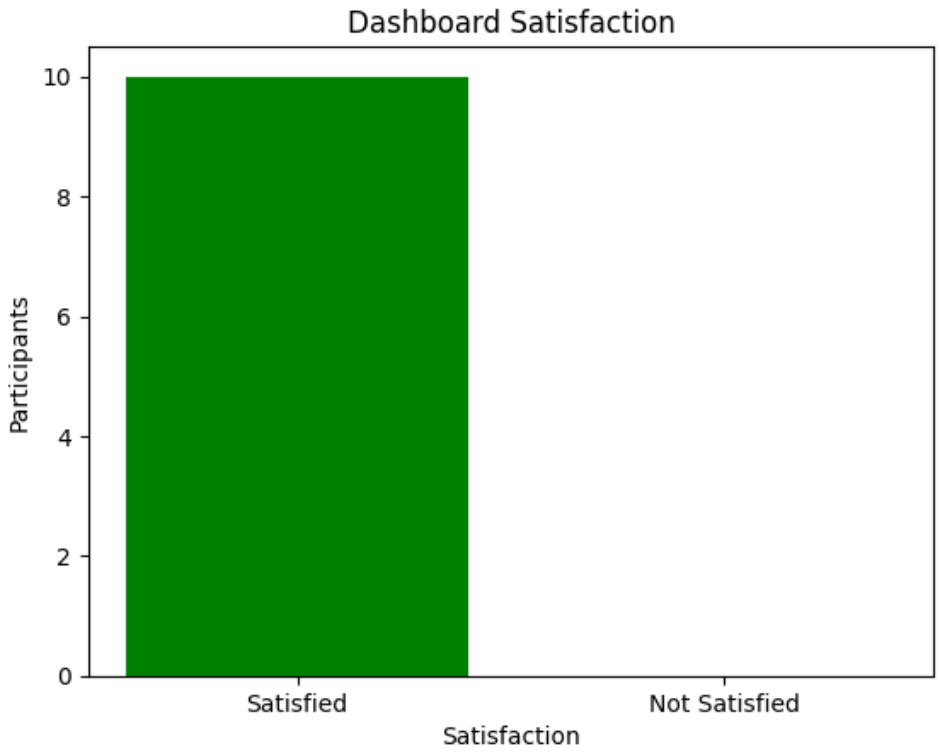

Figure 29. figure of a bar chart representing the satisfaction level between interview participants.

6.5 Evaluation Conclusion

In conclusion, the evaluation results of the software quality analysis dashboard demonstrate its significant success and effectiveness as a tool. The evidence is apparent in the user feedback collected during structured interviews. A clear majority of participants reported daily use of the dashboard, indicating its high relevance and robust user adoption. This frequent usage, combined with a lack of identified unnecessary features or reported difficulties, reveals user reliance on the tool and attests to its reliability and alignment with user needs and workflows.

Moreover, the assessment of users' experiences, opinions, and overall satisfaction strongly points to the tool's impact on work processes, productivity, and team collaboration. Themes derived from user feedback emphasized aspects such as efficiency, improved quality analysis, data accessibility, ease of use, and enhanced communication, all of which were central to the objectives for the dashboard. Notably, user satisfaction was universally high, implying that the tool not only meets user needs but also performs reliably and consistently, which further fosters its ongoing use.

The overwhelmingly positive feedback underlines the dashboard's success, and it also highlights the effectiveness of the user-centered design approach employed, where careful consideration of user needs and thoughtful feature selection played a crucial role. The ongoing refinement and adaptation of the tool to meet evolving user needs and technological landscapes showcases a deep-rooted commitment to this approach. Regular collection and response to user feedback will persist as an integral part of the process. The dashboard's success to date sets a robust foundation for its sustained development and utilization.

7. Discussion

The findings derived from the evaluation of the software quality analysis dashboard reaffirm its success as an efficient tool in the software quality assurance process at the target projects. The prominent adoption and daily usage of the dashboard by users underscore its significant value and relevance in their everyday work, revealing the dashboard's ability to streamline the process of software quality analysis.

The lack of reported unnecessary features or difficulties indicates that the dashboard aligns well with user needs and workflows. This suggests that the user-centered design approach, which prioritized addressing specific user requirements, has been effective. This aligns with the primary objective of developing a dashboard that simplifies the process of accessing and analyzing software quality data while reducing the need for manual work.

Additionally, user feedback regarding the improvement in efficiency, data accessibility, ease of use, and enhanced communication points to the impact of the dashboard on work processes, productivity, and team collaboration for the target projects. These factors were crucial to the objectives for the dashboard and the overwhelmingly positive feedback on these aspects confirms the success in achieving these goals.

In my opinion, one of the fundamental drivers behind the success of the project is the persistent engagement with the customer through weekly evaluation meetings. This close contact allowed the maintenance of an ongoing understanding of user needs, expectations, and potential concerns. Regular meetings facilitated immediate feedback, which was invaluable in ensuring that the dashboard was continually refined and improved to better meet user requirements. This approach fostered a customer-centric development process, where user feedback directly informed design decisions, resulting in a tool that effectively addressed the specific needs of the users.

The success of the software quality analysis dashboard carries significant implications for both Elektrobit and the wider field of software quality analysis. For Elektrobit, the dashboard’s success illustrates the potential value of investing in tools that streamline complex processes, improve productivity, and enhance collaboration. The overwhelmingly positive feedback could also contribute to higher morale among team members whom their project was included in the dashboard as they now have a more efficient and user-friendly tool at their disposal.

In a broader context, the findings underline the importance of user-centered design in the development of such tools. This highlights the need for other companies and developers to pay close attention to the actual needs and workflows of users during the design and development process. Doing so can result in tools that are not only more efficient but also highly adopted and appreciated by users.

Furthermore, the high user satisfaction suggests that the approach to developing the dashboard could be successfully replicated in other companies or contexts. This demonstrates the potential of such dashboards to transform the way software quality analysis is carried out across the industry.

Finally, the commitment to continuous refinement and adaptation in response to evolving user needs and technological landscapes is an important aspect that should be considered

by other companies or developers. This ensures that the tool remains relevant and effective in the long run, reinforcing its value for users.

8. Limitations and future work

Despite the success and positive feedback obtained, it is imperative to acknowledge the limitations of this study. Firstly, the current implementation of the software quality analysis dashboard remains a proof of concept. The focus of the project was to ascertain the value and viability of such a tool. Consequently, certain features like test development and an authentication mechanism, crucial for real-world deployment and usage, have not been incorporated yet.

Secondly, there exists a significant challenge associated with the maintenance of data collection. Given the heterogeneous nature of projects, modules, metrics tracked, and their disparate data sources with varying formats (CSV, XML, text), maintaining uniformity and efficiency in data collection and processing could prove difficult over time.

Thirdly, while the team maintained close contact with the customer via weekly evaluation meetings, these meetings were not recorded. Therefore, detailed insights, feedback, and discussions from these meetings cannot be disclosed or referred to in the report.

Lastly, the feedback gathered was derived from a relatively small group of participants, specifically ten in number. Although these individuals were representatives of their respective teams, the limited sample size might not fully capture the diversity of experiences and perspectives across the entire user base within the company.

Looking forward, the project presents several opportunities for improvement and expansion. The first logical progression would be to advance beyond the proof-of-concept stage. This involves fleshing out the remainder of the system, including the development of test components and the integration of a robust authentication mechanism to ensure data security.

Addressing the complexity of data collection and maintenance will be another crucial step. Future efforts should aim to streamline the data collection process, possibly through the automation of data transformation to a uniform format from different sources.

For future interactions with customers, it would be beneficial to record the evaluation meetings, given the customer's consent, to ensure that important insights and feedback can be revisited and analyzed in more detail.

Moreover, expanding the user feedback process is vital for future iterations of the dashboard. Inviting a larger and more diverse participant pool will allow for a more comprehensive understanding of the user experience. Incorporating quantitative assessment methods alongside the qualitative ones, such as usage metrics, can also provide a more holistic view of the system's effectiveness and usability.

These directions for future work offer a clear path towards the continued evolution of the dashboard. They will help address the identified limitations and unlock its full potential, thereby amplifying the tool's value and relevance to all stakeholders involved.

9. Conclusion

In conclusion, this master thesis has successfully optimized software quality analysis at target projects in Elektrobit by designing and evaluating an automated quality data collection process and visualization system. The proposed solution, a user-friendly, accessible, and interactive dashboard, has successfully streamlined the process of software quality analysis, achieving the intended objectives of saving time, reducing manual work, and enhancing user experience.

The findings derived from the evaluation of the dashboard affirm its value in the quality assurance process in the implemented projects. The favourable reception of the dashboard underscores its significant practical value, highlighting its potential for improving efficiency, productivity, and team collaboration.

Furthermore, the study showcased the importance of a user-centered design approach, regular user engagement, and iterative development methods. This led to a solution that not only met the users' requirements but also fostered a more efficient, effective, and satisfying workflow, thereby demonstrating the potential of such tools to revolutionize the way software quality analysis is conducted across the industry.

Nevertheless, it's important to acknowledge the limitations of this study, particularly in terms of the small group of participants and the current proof-of-concept stage of the dashboard. Future work will focus on these areas, refining the system, expanding the user feedback process, and addressing the challenges associated with data collection and maintenance. This will further enhance the tool's effectiveness and value, pushing the boundaries of what can be achieved in software quality analysis.

In summary, this research serves as an affirmation of the potential of automating quality data collection and using interactive visualization systems in software quality analysis. The success of this project establishes a solid foundation for future research and development in this field, benefiting not only Elektrobit but the broader software industry as well. Through persistent improvements and innovation, there is potential to further enhance productivity, efficiency, and user satisfaction, fundamentally transforming the landscape of software quality assurance.

References

Albrecht, A. J. (1979). Measuring application development productivity. In *Proc. joint share, guide, and ibm application development symposium* (pp. 83-92).

Avizienis, A., Laprie, J. C., Randell, B., & Landwehr, C. (2004). Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, *1*(1), 11-33.

Bajari, P., Chernozhukov, V., Hortaçsu, A., & Suzuki, J. (2019, May). The impact of big data on firm performance: An empirical investigation. In *AEA papers and proceedings* (Vol. 109, pp. 33-37).

Scannapieco, M. (2006). *Data Quality: Concepts, Methodologies and Techniques. Data-Centric Systems and Applications*. Springer.

Brehmer, M., & Munzner, T. (2013). A multi-level typology of abstract visualization tasks. *IEEE transactions on visualization and computer graphics*, *19*(12), 2376-2385.

Bughin, J., Chui, M., & Manyika, J. (2010). Clouds, big data, and smart assets: Ten tech-enabled business trends to watch. *McKinsey quarterly*, *56*(1), 75-86.

Caldiera, V. R. B. G., & Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of software engineering*, 528-532.

Cappiello, C., Francalanci, C., & Pernici, B. (2004, June). Data quality assessment from the user's perspective. In *Proceedings of the 2004 international workshop on Information quality in information systems* (pp. 68-73).

Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS quarterly*, 1165-1188.

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object-oriented design. *IEEE Transactions on software engineering*, *20*(6), 476-493.

Chrissis, M. B., Konrad, M., & Shrum, S. (2011). *CMMI for development: guidelines for process integration and product improvement*. Pearson Education.

Creswell, J. W., & Clark, V. L. P. (2017). *Designing and conducting mixed methods research*. Sage publications.

DeLone, W. H., & McLean, E. R. (2003). The DeLone and McLean model of information systems success: a ten-year update. *Journal of management information systems*, *19*(4), 9-30.

Deery, M. (2022, August 1). What Is Flask and How Do Developers Use It? A Quick Guide. Career Foundry, from https://careerfoundry.com/en/blog/web-development/what-is-flask/

Eckerson, W. W. (2010). *Performance dashboards: measuring, monitoring, and managing your business*. John Wiley & Sons.

Fenton, N., & Bieman, J. (2014). *Software metrics: a rigorous and practical approach*. CRC press.

Few, S. (2006). *Information dashboard design: The effective visual communication of data*. O'Reilly Media, Inc.

Garcia, M. (2020). Use a Flask Blueprint to Architect Your Applications. Real Python, from https://realpython.com/flask-blueprint/

Glass, R. L. (2001). Frequently forgotten fundamental facts about software engineering. *IEEE software*, *18*(3), 112-111.

Gotz, D., & Wen, Z. (2009, February). Behavior-driven visualization recommendation. In *Proceedings of the 14th international conference on Intelligent user interfaces* (pp. 315-324).

Halstead, M. H. (1977). *Elements of Software Science (Operating and programming systems series)*. Elsevier Science Inc.

Halevi, G., Moed, H., & Bar-Ilan, J. (2017). Suitability of Google Scholar as a source of scientific information and as a source of data for scientific evaluation—Review of the literature. *Journal of informetrics*, *11*(3), 823-834.

Hashem, I. A. T., Chang, V., Anuar, N. B., Adewole, K., Yaqoob, I., Gani, A., ... & Chiroma, H. (2016). The role of big data in smart city. *International Journal of information management*, *36*(5), 748-758.

Heer, J., & Bostock, M. (2010). Declarative language design for interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, *16*(6), 1149-1156.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2008). Design science in information systems research. *Management Information Systems Quarterly*, *28*(1), 6.

Howard, M., & Lipner, S. (2006). *The security development lifecycle* (Vol. 8). Redmond: Microsoft Press.

ISO/IEC 9126. (1991). Information Technology - Software Product Evaluation - Quality Characteristics and Guidelines for Their Use. International Organization for Standardization.

ISO/IEC. (2011). ISO/IEC 25010:2011 Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models. International Organization for Standardization/International Electrotechnical Commission.

Jacobs, A. (2009). The pathologies of big data. *Communications of the ACM*, *52*(8), 36-44.

Jones, C. (2008). *Applied software measurement*. McGraw-Hill Education.

Jones, C. (2010). *Software engineering best practices: lessons from successful projects in the top companies*. McGraw-Hill Education.

Kallinikos, J. (2013). The allure of big data. *Mercury Magazine*, *2*(3), 40-43.

Kavinda, A. (2020, March 23). Basics of ReactJS. Medium, from https://medium.com/@ashenkavinda/basics-of-reactjs-79107d19adfb

Kitchenham, B., & Pfleeger, S. L. (1996). Software quality: the elusive target *IEEE software*, *13*(1), 12-21.

Kitchenham, B. A., Pfleeger, S. L., Pickard, L. M., Jones, P. W., Hoaglin, D. C., El Emam, K., & Rosenberg, J. (2002). Preliminary guidelines for empirical research in software engineering. *IEEE Transactions on software engineering*, *28*(8), 721-734.

Kujala, S. (2003). User involvement: a review of the benefits and challenges. *Behaviour & information technology*, *22*(1), 1-16.

Laoyan, S. (2022, October 15). What is Agile methodology? (A beginner's guide) Asana, from https://asana.com/resources/agile-methodology

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, *68*(9), 1060-1076.

Lyu, M. R. (1996). *Handbook of software reliability engineering* (Vol. 222). Los Alamitos: IEEE computer society press.

Madnick, S. E., Wang, R. Y., Lee, Y. W., & Zhu, H. (2009). Overview and framework for data and information quality research. *Journal of data and information quality (JDIQ)*, *1*(1), 1-22.

Martins, J. (2022, October 10). What is Kanban? Here's what your Agile team needs to know. Asana, from https://asana.com/resources/what-is-kanban

McCabe, T. J. (1976). A complexity measures. *IEEE Transactions on software Engineering*, (4), 308-320.

McCullough, B. D., & Vinod, H. D. (2003). Verifying the solution from a nonlinear solver: A case study. *American Economic Review*, *93*(3), 873-892.

McGraw, G. (2006). Software security. Building security in.

McCall, J. A., Richards, P. K., & Walters, G. F. (1977). *Factors in software quality. volume i. concepts and definitions of software quality*. GENERAL ELECTRIC CO SUNNYVALE CA.

Moody, D. L., & Walsh, P. (1999, June). Measuring the Value Of Information-An Asset Valuation Approach. In *ECIS* (pp. 496-512).

Nielsen, J. (1994). *Usability engineering*. Morgan Kaufmann.

Olsina, L., Lafuente, G., & Rossi, G. (2001). Specifying quality characteristics and attributes for websites. In *Web Engineering: Managing Diversity and Complexity of Web Application Development* (pp. 266-278). Berlin, Heidelberg: Springer Berlin Heidelberg.

Paulk, M. C., Curtis, B., Chrissis, M. B., & Weber, C. V. (1993). Capability maturity model, version 1.1. *IEEE software*, *10*(4), 18-27.

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. (2013). Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, *16*(1), 414-454.

Pigoski, T. M. (1996). *Practical software maintenance: best practices for managing your software investment*. Wiley Publishing.

Pipino, L. L., Lee, Y. W., & Wang, R. Y. (2002). Data quality assessment. *Communications of the ACM*, *45*(4), 211-218.

Burrus, D. (2017). *The anticipatory organization: Turn disruption and change into opportunity and advantage*. Greenleaf Book Group.

Rasmussen, N. H., Bansal, M., & Chen, C. Y. (2009). *Business dashboards: a visual catalog for design and deployment*. John Wiley & Sons.

Rind, A., Wang, T. D., Aigner, W., Miksch, S., Wongsuphasawat, K., Plaisant, C., & Shneiderman, B. (2013). Interactive information visualization to explore and query electronic health records. *Foundations and Trends® in Human–Computer Interaction*, *5*(3), 207-298.

Sacha, D., Senaratne, H., Kwon, B. C., Ellis, G., & Keim, D. A. (2015). The role of uncertainty, awareness, and trust in visual analytics. *IEEE transactions on visualization and computer graphics*, *22*(1), 240-249

.

Sadalage, P. J., & Fowler, M. (2013). *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education.

Sedlmair, M., Meyer, M., & Munzner, T. (2012). Design study methodology: Reflections from the trenches and the stacks. *IEEE transactions on visualization and computer graphics*, *18*(12), 2431-2440.

Sheth, A. P., & Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, *22*(3), 183-236.

Shneiderman, B. (1996, September). The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE symposium on visual languages* (pp. 336-343). IEEE.

Shneiderman, B., Plaisant, C., Cohen, M. S., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). *Designing the user interface: strategies for effective human-computer interaction*. Pearson.

Shull, F., Singer, J., & Sjøberg, D. I. (Eds.). (2007). *Guide to advanced empirical software engineering*. Springer Science & Business Media.

Singh, T. (2021, October 25). Streamlit: A must-learn tool for data scientist. Medium, from https://medium.com/crossml/streamlit-2256000541ad

Solove, D. J. (2006). A taxonomy of privacy. *University of Pennsylvania law review*, 477-564.

Teece, D. J. (1986). Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research policy*, *15*(6), 285-305.

Tufte, E. R. (1985). The visual display of quantitative information. *The Journal for Healthcare Quality (JHQ)*, *7*(3), 15.

Wand, Y., & Wang, R. Y. (1996). Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, *39*(11), 86-95.

Watson, H. J., & Wixom, B. H. (2007). The current state of business intelligence. *Computer*, *40*(9), 96-99.

Whittaker, J. A. (2000). What is software testing? And why is it so hard?. *IEEE software*, *17*(1), 70-79.

Wexler, S., Shaffer, J., & Cotgreave, A. (2017). *The big book of dashboards: visualizing your data using real-world business scenarios*. John Wiley & Sons.

Wang, T. D., Plaisant, C., Quinn, A. J., Stanchak, R., Murphy, S., & Shneiderman, B. (2008, April). Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 457-466).

Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, *12*(4), 5-33. Yi, J. S., Kang, Y.

Yi, J. S., ah Kang, Y., Stasko, J., & Jacko, J. A. (2007). Toward a deeper understanding of the role of interaction in information visualization. *IEEE transactions on visualization and computer graphics*, *13*(6), 1224-1231.

Yigitbasioglu, O. M., & Velcu, O. (2012). A review of dashboards in performance management: Implications for design and research. *International Journal of Accounting Information Systems*, *13*(1), 41-59.

Zaslavsky, A., Perera, C., & Georgakopoulos, D. (2013). Sensing as a service and big data. *arXiv preprint arXiv:1301.0159*.