МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗВО “УКРАЇНСЬКИЙ КАТОЛИЦЬКИЙ УНІВЕРСИТЕТ”

Факультет суспільних наук
Кафедра управління та організаційного розвитку

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Впровадження системи метрик якості програмного забезпечення як конкурентна перевага для сервісної ІТ компанії

Виконав: студент 2 курсу, групи СУТ22М спеціальності 073 “Менеджмент” освітньої програми “Управління технологіями” Батаєв С.В.

Керівник: Лагоднюк О.
Рецензент:

Львів – 2024

Abstract

Implementing a software quality metrics system is instrumental in enhancing the competitiveness of the IT service provider company. These metrics provide an opportunity and platform to turn qualitative data into quantitative data, allowing them to use statistics and numerical data to analyze organizational processes and procedures. The primary objective of this study is to show how quality metrics in software development lay the foundation for improving customer satisfaction and using data to make informed decisions, improving the overall competitiveness of IT companies. The study will prove that through different quality metrics such as scope and team performance and technical and financial metrics, project managers, in collaboration with programmers and project stakeholders, can gain insight into the development process and make informed decisions concerning product quality, project deadlines, and budgets. It will also underline the need for a cross-functional team to develop software, given that such project teams will have the expertise to turn customer needs into clean, efficient, and high-performing products and services. In addition, this research indicates that the unpredictability of project or program implementation is a significant challenge that can have severe consequences for organizations, including cost overruns, schedule delays, and benefits shortfalls. These consequences can lead to financial losses, customer dissatisfaction, and damage to the organization's reputation. The unpredictability of project implementation can also impact customers, who are the ultimate beneficiaries of the project outcome.

Contents

Introduction

Increased competition for the available market in the modern business environment pushes players in the software industry to provide high-quality products and services. The objective is to provide quality products that meet or exceed customer expectations, thus attracting and retaining long-term customers (Makhoul, 2022). Software metrics are one of the methodologies that software companies employ to monitor their competitive environment and performance. Software quality metrics are identified as indicators developers and teams use to assess the product's overall quality metric framework. The software quality metrics system is imperative for every IT service Provider as it helps them attune to the market requirements. These companies can measure performance weaknesses through software quality metrics analysis and invest in process optimization, providing customers with the highest-class products and services (Gruzdo et al., 2019). Ultimately, these initiatives improve client satisfaction and realize their competitive advantage. In software development, several metrics are interrelated; they are related to four primary management functions: providing the service, organizing, planning, and improving (Gruzdo et al., 2019). Given the need to closely track and monitor the individual performances of these IT companies, this study will focus on implementing a software quality metrics system to enhance the IT service provider company's competitiveness.

Project Objectives

Given the significance of having strong metrics to measure the quality of the IT services and products provided by an IT company, the aims of this project include:

a. Increasing customer satisfaction through a transparent software development lifecycle

b. Increase competitiveness by investing in continuous improvement and establishing areas for optimization
c. Data-driven decision-making for all stakeholders

1.1 Problem and Significance

Software Quality

As highlighted above, software metrics are instrumental for any IT service and product provider, considering that they are instrumental in identifying gaps and developing measures to help address the gaps, thus resulting in enhanced performance, customer satisfaction, and competitiveness (Făgărăşan et al., 2022). A software metric defines the measure of software features and characteristics that can be quantified and countable. They are instrumental in identifying and evaluating software quality. Software quality refers to the extent to which a software product fulfills its designated requirements and meets the expectations of its users. It encompasses a range of characteristics, including reliability, efficiency, usability, maintainability, and security, all of which contribute to the overall quality of the software (Sharma & Spinellis, 2018). Attaining a high level of software quality is crucial to ensure proper functionality, optimal performance, a user-friendly experience, and effective long-term maintenance and evolution of the software.

Software Development Life Cycle (SDLC) programming quality becomes a systematic process, as it is executed in every software development phase. Therefore, during the requirement-gathering phase, the team develops the project's needs elements and the processes that must be met (Ghosh et al., 2020). During the design stage, it helps them to tinder the concepts and to test their coordination, stability, and usability. The implementation stage has an impact on developers,

enabling them to build code in line with the standards of good design, efficient systems, and well-documented code to the prototyping stage of their development, and moreover helps them to have different levels of testing where they can identify and evaluate software (Akinsola et al., 2020). Continuous quality monitoring and testing at the different phases of the software systems lifecycle is normal since it allows the developers to develop software products that satisfy the consumers' requirements and make companies successful.

The Need for Reliable and Efficient Software Development

The modern business environment is highly competitive and demands more high-quality and reliable software development. Businesses and consumers need software products and services of high quality that can be trusted and thus require reliable and efficient development (Olorunshola & Ogwueleka, 2021). Several factors could define the growing need in the modern business environment, including:

The Need for More Efficiency

In today's highly competitive software development industry, one of the primary needs is reliability and efficiency because of the need to increase work efficiency. Companies are in continuous pursuit of achieving the highest levels of operational efficiency, and new approaches to achieve this continue to be searched for. This is done so that costs can be reduced and products/services can be provided to consumers on time (P. Miguel et al., 2014). Developing a reliable and productive software development strategy enables businesses to automate monotonous tasks, increase efficiency and collaboration among the workforce, reduce mistakes, and speed up the time to market. Lastly, strategizing a strong and error-free software development process allows providers to offer products and services that are simple to use and easily updated and

repaired. The flexibility is herein whereby the individual consumers can change the existing programs to enhance their service delivery to the market and improve customer needs (Idri et al., 2013). In addition, when software developers deliver reliable and efficient products, there is no need to invest excessive time in debugging, fixing defects, or deciphering complex code. This enables users and consumers to focus on meeting customer needs and desires rather than troubleshooting the software they have acquired.

Reduced Risk of Reworking

Re-working a product already developed to meet the requirements consumes much time. Moreover, this time is a loss to the developers, as they would have used it to develop other products. It also means that developers take longer to meet the deliverables of a project or a product, which amounts to a loss of revenue (Fitzgerald & Stol, 2017). It equally means that customers can distrust a company's product, as they would either receive the defective finished products or wait to be redone. Further, developing reliable and effective software the first time means that the final product will require little or no time to redo after it is released or published in the public domain. Hence, the competition implies that developers' refusal to refactor projects wastes time and money as they must put in the effort to develop software (Petersen et al., 2015). With growing customer demand for stable and effective software products, companies are under constant impetus to bring out reasonable quality solutions urgently and unfailingly. Starting from the basics and supporting it with the existence of correct methods and tools that ensure reducing the number of mistakes and failures is a great deal of the idea of cutting the significant expenses for reworking and retesting (Lee, 2014). This accelerates time-to-market, enhances customer satisfaction, and improves competitiveness by allowing companies to stay ahead of competitors with consistently high-quality software offerings.

Reducing Retention Rate and Improving Team Spirit

Today, the two main drivers for the increasing need for stable and efficient software development processes in modern software development are decreasing workforce efficiency and a dropping motivation of IT specialists. Losing workers frequently can effectively be reflected in high turnover costs, not to mention team morale, productivity, innovation, and business success in general (Lee, 2014). The developers can improve coding quality by investing in clean and well-written code. As a result, people will be more satisfied at work and frustrated less. Moreover, Clean, well-written code enhances readability, maintainability, and scalability. Also, scalability is enabled by using structured code. It decreases error risk and makes the process of project planning and development more collaborative among team members. Top-notch code might enhance effectiveness, weed out errors, and generate high-end programs (Gillies, 2011). This may result in the production of high-quality products. Hence, this trend prioritizes providing a strong organizational culture and operational methods to retain key talent and inspire personnel. Modern trustworthy and efficient software engineering approaches can significantly impact products/services and positively affect the team spirit, which results in the general improvement of the business performance.

The Challenge of Objectively Measuring Software Quality

The quality of software estimation is an objective factor since it is based on the subjective perception of quality. What is interesting about quality is that quality is more intangible and multi-focus, which makes it harder to measure software quality in a meaningful way before the attributes of the software have been unpacked (Mishra & Otaiwi, 2020). Then, each attribute was ensured to be observable. In addition, things like user experience, performance, security, and stability are usually evaluated by stakeholders differently. Besides, it is often seen that setting objective and

measurable goals that evaluate how good the software is, to some extent and in a way, is also complex. Since different software development methods (methodologies), technologies, and project requirements exist, the measurement process has become more sophisticated owing to its complexities. Hence, to effectively deal with this challenge, factors such as metrics, stakeholders' involvement in establishing suitable quality goals, and perpetually adapting the measures need to be taken into account to guarantee a veracious and relative software quality assessment (Cai et al., 2019). Multiple critical factors to consider about the challenge of objectively measuring software quality include:

Subjective Concepts

The concept of desirable code and its maintainability can be transformed into measurable, calibrated, highly quantifiable entities with the help of adequately measured metrics. Through metrics, developers can set qualitative and quantitative criteria for "setting our code in order" and "making our software maintenance-free." The highest quality software can be measured by tracking the numbers correlating with the qualitative aspect of excellent code and maintainability (Martínez-Fernández et al., 2022). For example, such metrics could be the number of lines of code or, even worse, the number of defects that have grown in the program during the creation process. This activity is unbiased as it is a software-based assessment, allowing one to check the performance and have proof of progress. Metrics function as a good tool in the analysis of features of software systems that help organizations identify substantial information and achieve sound decisions by referring to data (Serban et al., 2020). The key to successfully managing the issue of software quality assessment is objective measurement. Proper use of metrics can help teams to do this. And they can drive continuous improvement in development processes all over the organization.

Establish a Baseline

As mentioned above, software developers must track their progress to measure the quality of their output and ensure that they deliver high-quality products and services that meet customer needs and wants. However, to achieve this, they have to establish a baseline to measure the quality of their output (Rathor et al., 2023). This means that metrics can establish a baseline by providing quantifiable data on various aspects of software quality. Thus, this baseline allows the development team room to track progress and measure performance compared to industry standards or against the benchmarks. Using an objective mechanism for tracking software quality, each team can trace the changes in these metrics, detect the weak points, and make a data-driven decision to improve the quality. This interactive system stands as the basis for further solutions refinement, and the team understands how to avoid various issues as the products are being developed (Oreški, 2022). Statistical tools help to establish a reference point for product developers, enabling them to use historical trends to determine current and future market needs and product demands. This way, product developers can ensure that the quality of their output is maintained or even enhanced.

Foundation for Data-Driven Decision Making

Data associated with software quality that developers can directly use to make informed decisions is constrained by its subjective nature, which makes it impossible to use as a strategic tool to make an informed decision regarding the quality of the products and services offered by a company (Zarour, 2020). Nevertheless, metrics could be used as a foundation for making informed decisions that involve using numbers to objectively measure the state of the software. Through the fact that various aspects of building software can be quantified, metrics that aim to evaluate the efficiency of processes and quality of products serve as valuable instruments for getting insights. These metrics give the developers a sense of direction and ensure accountability, allowing them to

change existing processes or enhance their quality to lift the company and its performance (Lakra & Chug, 2021). Consequently, due to metrics, teams have a clear objective. They can easily track their progress, identify trends, and make decisions based on hard data, not just opinions and subjectivity. Objective measurement tools retain organizations, embed interventions, and drive teams with improvement as a goal. It allows developers to treat data as a fact instead of guesswork by emotions, personal opinions, and assumptions about the service they want to provide (Pargaonkar, 2023). The data-driven approach enables quality-compliant processes and facilitates measures of continuous improvement that favor lasting customer satisfaction.

Fosters Effective Communication at all Organizational Levels

The principle of effective communication is the foundation of every successful organization. It provides a communication channel to share and broadcast data, ensuring all stakeholders are on the same page. Therefore, effective communication helps avoid unclear messages that can lead to different interpretations, thus making it a harmonious workforce (Pargaonkar, 2023). Hence, metrics serve as a jargon that unites technical and non-technical employees regarding software quality. Indicators that come in are precise measuring units. Metrics provide both camps a means of quantifying data points and making them easy to comprehend. With their help, the users and the developers can evaluate the performance and quality of software products or processes plainly and without hidden agendas (Gao et al., 2022). This allows them to generate more meaningful decisions and smoother collaboration and alignment with the organizational goals and objectives. Overall, metrics facilitate effective communication and mutual understanding between technical experts and business stakeholders, ultimately leading to improved software quality and successful outcomes for the organization.

Increase Performance and Decrease Operating Costs.

Every organization focuses on improving its profits while reducing operating costs, increasing the stakeholders' wealth, and cementing its place in the competitive market. Software quality metrics can significantly improve performance and reduce companies' escalating software development costs (Gao et al., 2022). Companies can objectively assess the quality of their software products and processes by measuring specific metrics such as technical, financial, scope, and performance metrics. These metrics are also instrumental in identifying and evaluating flaws in the development process, facilitating the development of preventive and intervention measures to address the flaws and enhance organizational performance. This data-driven approach enables companies to identify performance gaps, optimize resources, and make informed decisions to enhance quality and productivity. By continuously monitoring and analyzing software quality metrics, companies can effectively address challenges related to software quality, increase performance, and ultimately reduce costs associated with rework and maintenance (Pargaonkar, 2024). Notably, by cutting operational costs, the developers can maximize their profits, stay on top of current expenses, and save on costs, money that can then be invested in quality standards and control, ensuring that the developers provide only high-quality products and services.

1.2 Proposed Solution: Implementing a Software Quality Metrics System

Implementing a software quality metrics system is vital for IT service providers to enhance competitiveness and deliver high-quality products. By measuring critical indicators, companies can identify the need to put quality and performance metrics in place and demonstrate competitiveness and credibility, which is instrumental in attracting and retaining customers in the long run (Lee, 2014). This data-driven approach offers quick insight into the status of software

testing efforts, leading to better control through intelligent and effective decisions. However, individual organizations must find the right interventions and solutions, allowing developers to monitor their work closely. One way to achieve this is through a Software Quality Metrics System (SQMS).

Those managing and overseeing new software development must be competent in doing everything systematically and critically, thinking about the final quality before release. Those managers must have a profound knowledge of software quality metrics to describe the progress of different software development processes (Lee, 2014). A Software Quality Metrics System (SQMS) is a structured approach encompassing software development processes and collecting, analyzing, and presenting tangible data related to these processes. The management process is the software developers' avenue for exploring software quality and development. This process aims to meet the quality and customer expectations while complying with the regulatory standards and policies (Kassie & Singh, 2020). It involves measuring and evaluating key performance indicators (KPIs) to assess the quality of software products, processes, and outcomes. These indicators generally safeguard the development process by tracking it and ensuring it is completed on time and within budget.

Furthermore, the SQMS solution allows enterprises to access data concerning code complexity, code coverage, test coverage, defect density, and customer satisfaction with continuous improvement tracking (Kassie & Singh, 2020). Usually, patterns recognized in the SQMS prove beneficial as they showcase some best practices. More specifically, SQMs provide:

Data-Driven Insights into the Development Process

By implementing an effective SQMS, companies can collect, analyze, and leverage data to improve software quality and development efficiency (Chakravarty & Singh, 2021). SQMS can also track metrics such as total scope, project costs, unit test coverage, and other quality indicators to provide a comprehensive view of the development process. By analyzing these metrics, companies can identify their current status concerning developing a particular software and where it wants to go. This provides the baseline to measure performance and progress while also providing a sense of direction, considering that the developers and companies can develop clear goals and objectives that are not only specific, measurable, achievable, relevant, and time-bound but can also provide a clear of path of what needs to be done (Ulan et al., 2021). Overall, SQMS empowers IT service provider companies to make data-driven decisions, continuously improve development processes, and enhance software quality to stay competitive in the market.

24/7 Access and Real-Time Data (Metrics) for Customers

Apart from providing data to make informed decisions, SQMS facilitates access to real-time customer data, laying the pillars and foundations for improving customer satisfaction and experience. By offering round-the-clock access to quality metrics, customers can see the performance and quality of software products and services, enabling them to make informed decisions and monitor progress continuously (Ulan et al., 2021). Real-time data helps the customer negotiate the effectiveness of the actions undertaken to improve quality, identify the issues before they become a problem, and discuss them with service providers to tackle the arising difficulties in time. This covers them against inconvenience associated with the software upgrades and protects them against system breakdowns. With 24/7 access to SQMS, customers can obtain information anytime, allowing them to track and analyze performance indicators (Masmali &

Badreddin, 2020). Thus, customers will understand the deliverables' quality and how the standards are met. The intransigency creates trust and openness in the communication and helps form a healthy (with service providers) - relationship. Finally, real-time data enables companies to respond to customer feedback quickly, make data-driven decisions, and demonstrate their commitment to quality and customer satisfaction (Masmali & Badreddin, 2020). Customers can contact the service provider any time of the day and get the help they need, considering that the service providers have real-time data about the performance of a particular software and know how to address the issue in question.

Address the Estimated Time and Cost for Backlog Completion and Highlight the Risks Related To the Timeline, Budget, or Technical Quality

Estimation plays an essential role in agile software development to satisfy the quality standards and requirements, which are the critical points for customer satisfaction. Such a process of estimation in software development delivers an insight into how much time and money is necessary to create programs and software (Masmali & Badreddin, 2020). It helps consumers and companies to make anticipatory solutions so that these products are delivered on time, within the set budget, and in such a manner they are all within a specific frame of expectations. There is a method - SQMS can significantly facilitate estimating time and cost for backlog completion in IT projects. The project managers can make more precise decisions on the time and funds needed to achieve backlog relief upon evaluating such metrics and criteria. Therefore, the project manager and team would be unable to manage the time and cost without using metrics such as time and cost estimation, which means the team could spend more time than needed or, conversely, take fewer days than the specified project duration (Gordieiev & Kharchenko, 2020). Although the lengthy SQMS process is worthwhile because it allows project managers to prepare budgets, timelines,

and plans, organize the tasks, and successfully execute projects, all the additional procedures are demanding.

However, timeline, budget, and technical quality are associated with risks when estimating backlog completion. One common risk is underestimating the time needed. Time risks are the risks of project tasks taking longer than estimated. It is non-negotiable that delayed timelines are the factors that lead to over-budgeting, late delivery, and below-par projects (Jain et al., 2014). Budgeting risks also play a crucial role in project management. Here, a lot of resources needed to solve quality issues are overpassed. Budget problems can stem from internal and external factors, including changes in revenue, cost, and demand or factors such as technology and regulations. Risk factors are the reason for budget failings, often causing project delays and projects exceeding the budget. Lastly, technical risks in software development include the uncertainty and magnitude of differences between the real and the best software concepts that can lead to defects in the artifacts and processes (Akinsola et al., 2020). This business's inherent risks lie in estimation errors, scope creep, and inaccurate project assessment. If the metrics expressing software quality standards are inaccurate or interpreted with doubt, then there is a possibility of adversely affecting performance, reliability, and customer satisfaction.

1.3 Software Quality Metrics System Approach

As mentioned earlier, a software quality metrics system approach should focus on defining quality policies, choosing related metrics, measuring processes, involving people, and creating a culture of continuous improvement. Businesses must control, track, assess, and review their metrics to improve quality outcomes, optimize lean processes, and attract more customers (Cai et al., 2019). SQMS can be utilized by an organization to deliver high-quality products and services

to the customer through limiting product defects, end-user confusion, and customer dissatisfaction. Through such an approach, businesses can boost their productivity and gain a competitive edge as recognized and reliable IT service providers.

Metrics Selection

The selection of software quality metrics is crucial for measuring and improving the quality of software products and processes. Factors to consider include alignment with business goals, relevance to customer requirements, measurability, reliability, and actionability. Metrics should be meaningful, specific, and consistent to provide valuable insights for decision-making (Chakravarty & Singh, 2021). The various metrics that software developers should focus on when selecting or implementing metrics include:

Scope and Team Performance Metrics

Scope and team performance metrics are crucial for measuring the effectiveness and efficiency of project teams in achieving project goals within defined scope boundaries. Scope metrics assess the alignment of project deliverables with initial requirements and objectives. In contrast, team performance metrics evaluate the performance and collaboration of team members in executing tasks and meeting deadlines (Fitzgerald & Stol, 2017). By tracking scope and team performance metrics, including total Scope, Scope completed, scope planned, scope difference (increased/decreased), iteration velocity trend (average team based on previous five iterations), and prediction on deadline based on scope planned and iteration velocity trend. Combined, these metrics enable project managers to identify areas for improvement, optimize team performance, and ensure successful project delivery.

a) **Total scope** – scope metrics appraise the project's deliverables and the desires for completion. It provides insight into whether the target goals meet the expectations of the technical and end users. Total scope metrics define what is used to evaluate the whole size and the complexities of the software project. These metrics help identify requirements, features, and tasks (Fitzgerald & Stol, 2017). Through the summary of total scope metrics, the project managers will have a clearer idea of themselves, from the early-stage planning, setting realistic timelines to managing scope changes. This is a way of delivering optimum performance.

b) **Scope Completed** – this scope helps establish a defined boundary for all project activities and tasks. Everything to be done will eventually lead to delivering a product or service based on the predefined goals and objectives. Performance metrics utilized by scope completed assist in establishing the measurement of the extent to which all the intended activities and deliverables within the scope of a job/ task have been accomplished or completed within the specified indicators (Făgărăşan et al., 2022). The data that resulting metrics yield becomes valuable for revealing the status of a project, its ability to stick to scope requirements, and eventually for evaluating the outcome. Through tracking the scope completed, managers of the projects can study performance, detect deviations or certain risks in advance, and take corrective measures to keep the projects within the scope borders.

c) **Scope Planned –** the project scope identifies what the product will do, what it will not do, and what the end product will have and will not have. As a result, the scope of planned metrics for the software quality metrics system includes defining clear quality goals and objectives, selecting appropriate metrics such as defect density and code complexity, and

establishing measurement processes and tools for data collection and analysis (Gao et al., 2022). These planned metrics aim to drive quality improvements, optimize processes, enhance customer satisfaction, and demonstrate competitiveness in the market for the IT service provider company. It can also list necessary resources, chart project milestones, and identify potential roadblocks that inhibit the project team's ability to attain the predetermined goals and objectives.

d) **Scope Difference (Increased/Decreased)** – one of the ways of exploiting and using scope difference is by knowing what scope is, why it was initiated, and what it is meant to achieve. This allows the company to develop solutions for mistakes and findings, which helps them align the project components and deliverables with the already defined scope. Hence, the scope metric has a massive difference in dimension since it helps set scope boundaries, which are the initial features or functions (Ghosh et al., 2020). The increased scope peculiarities point to additional specs or functions, which could be the reason for a targeted schedule. Therefore, a re-adjustment of the project budget may be required. Increased gap difference means that the project is managed effectively based on the interpreted initial requirements and constitutes delivery on time and within the limitations of the budget.

e) **Iteration Velocity Trend (Average Team Based on Previous Five Iterations)** – the trend report underlines a team's historical velocity in a given time. Its purpose is to underscore what the team has achieved over the past period and quantify their performance regarding critical points. Therefore, iteration velocity metrics show the team's average velocity over the last five iteration velocities (Gillies, 2011). By studying this pattern, a group can find out if it has been improving or worsening in its productivity rate

and forecast performance levels in the future. Keeping the measure of iteration velocity and analysis of such trends is part of the continuous journey of improvement aimed at delivering high-quality software products that meet or beat the customer's expectations.

f) **Prediction on a Deadline Based on Scope Planned and Iteration Velocity Trend –** This ensures that the project team, including other stakeholders, understands the deadline, improving team morale. It lets them complement each other and contribute to the project's success. Hence, predicting deadlines calculated by the planned scope and iteration velocity trend is a significant part of the project management process (Gordieiev & Kharchenko, 2020). Considering the scope planned for each iteration and evaluating the tracked velocity trend over time, the project managers can estimate when the project will be completed. Ideally, the well-established scope and the continuous trend regarding the velocity will make the deadlines more accurately predicted. While not frequent, sometimes variations in scope, unclear velocity trends, or other unexpected issues happen, affecting the precision of the prediction (Gruzdo et al., 2019). Therefore, in addition to the fixed deadlines, it would be more useful to re-evaluate and change the schedule whenever the project circumstances become different to ensure the timely completion of the deliverables.

Financial Metrics

Unlike scope metrics that focus on team performance and set project boundaries and deliverables, financial metrics are more interested in the project's financial performance. They are essential for monitoring projects and teams' financial performance and profitability. Key financial metrics include total project costs, total spent to date, forecast to planned date, and whether the project is within the set budget (Gruzdo et al., 2019). These metrics help organizations track and

control project costs, allocate resources effectively, and make informed decisions to maximize financial outcomes.

a) **Total Project Cost (Planned)** – the total project estimate includes the base estimate and the estimate for all project costs at the time of the estimate. As a proactive approach, the project managers shall develop contingency plans to ensure that the total project cost rate does not affect project completion timelines or performance standards. Therefore, the total cost of the project concept represents the planned estimate that must be implemented for a project to be completed as expected (Idri et al., 2013). Financial metrics like budget variance, cost performance index, and return on investment are vital for tracking and containing the cost, which makes them a must-have component of every project.

b) **Total Spent to Date (Fact on Payments)** is another significant financial metric in any project. Like other financial metrics, beyond saving cash, it shows how the project team has spent the funds until a specific period, keeping them within their spending limits. With a payment monitoring approach, businesses can see how their finances are in the long run, how to stay within their budget plan, and make the best spending choices for the future (Jain et al., 2014). This performance measure endlessly monitors cash flow balance, expense budget, and financial health. The team analyzes the cost up to date, allowing it to identify trends, allocate resources appropriately, and ensure it remains financially stable. It provides financial health metrics, financial accountability, the efficiency of the targeted plans, and strategic financial decision-making.

c) **Forecast To Planned Date (Total Project Cost Projection Based on Total Scope and Project Duration)** – project forecast allows the project managers to forecast or project the financial resources they will need to complete a project successfully.

The primary perspective that this approach provides is to address the whole field, including the budget, labor, equipment, and overhead costs necessary for a project so that one can have the right estimate for this subject (Kassie & Singh, 2020). Therefore, these forecast-to-planned-date financial metrics translate to the cost forecast of the entire project considering the total scope and length. This indicator gives a financial picture showing the budgeting and resource management implications. The budget analysis in which the actual expenditures are compared and contrasted against the projected costs of the end-point date gives project managers the needed information to make decisions about the project to ensure it stays within budget and is delivered on time (Kumar et al., 2019). Monitoring and modifying these financial indices during the project's lifespan will help prevent risks, optimize expenses, and be oriented toward accomplishing objectives and profitability.

d) **Is Project In Budget (Yes/No)** - It involves reviewing the actual costs versus the budgeted amount and explaining financial indicators. To know whether a business is consistent, the project manager checks if the budgeted value matches the actual expenses. Financial indicators help them solve the question of whether the business is effective.

Technical Metrics

Technical metrics are for project managers who track or develop software in project management. They represent the project manager's quantitative and essential qualities for the quality of products or services that may be provided. They measure the software development procedure's quality, production, and effectiveness (Lakra & Chug, 2021). They include unit test coverage, number of unit tests, code duplications, technical debt, and bugs (in source code).

a) **Unit Test Coverage** - Unit test coverage is a technical metric measuring the percentage of code that unit tests cover. It illustrates the ability of testing procedures to find potential faults. If the unit test coverage is high, it underlines the thorough testing of the code and efficiency (Lee, 2014). To determine which parts of code are adequately covered by test units and need extra testing, the programmers pay attention to the code coverage metrics.

b) **Number Of Unit Tests** - The number of unit tests is a crucial technical metric used to measure the quality and reliability of software code. On the one hand, more unit tests indicate good test coverage with consequent perfect error detection, speedy debugging fixes, and superior software quality (Lee, 2014). By tracking and reviewing the number of test units, developers can make their programs much more stable and accurate, which means that end users will be exposed to more efficient and effective software.

c) **Code Duplications** – code duplication calculates the amount of replicated code. Constant repetitiveness of the code may become a source of maintenance issues, reduce effectiveness, and lead to increased errors (Makhoul, 2022). These metrics identify duplicated code, which will eventually help to refactor these code sections, thus facilitating better code quality, simplifying the technical debt, and increasing maintainability to the core. (Makhoul, 2022). Tracking the code duplication amounts will enhance the software development routine, guarantee the codebase maintainability, and keep software projects sustainable for a long time.

d) **Technical Debt** - Technical debt constitutes the lowered level of correctness achieved due to the accepted half-step compromises during software development. It may be used as a technical metric to monitor product complexity and ineffectiveness and find any possible software risks (Martínez-Fernández et al., 2022). Technical debt allows

organizations to do the essential work first, which, in turn, minimizes future developmental expenses and optimizes software quality and maintainability.

e) **Bugs (In Source Code)** - Bugs in source code are the primary technical metric used to measure the quality of the software. They may lead to the app being branded as fuzzy and unprofessional (Masmali & Badreddin, 2020). Through the count and levels of bugs, the developers can identify corrections and then prioritize bug fixes that will contribute to better performance and reliable software. Tracking bugs as a technical metric is used for evaluating development efficiency, making improvements related to better coding, and guaranteeing a good user experience. The development of applications is an unending process of improving the quality (Masmali & Badreddin, 2020). It becomes possible to achieve higher customer satisfaction and be competitive in the market when bugs are dealt with instantly.

Data Collection, Analysis, and Implementation:

Data collection, analysis, and implementation are vital aspects of software development. To provide real-time data, the following tools must be integrated: Atlassian Jira, Azure DevOps Services, and Sonarqube.

a) **Atlassian Jira - Atlassian Jira** is a project management and issue-tracking service provider used as the delivery medium among organizations and team members by visualizing, tracking, reviewing the progress, and ensuring the completion of the projects. Projects can be planned, tracked, and reported through timelines, and some diverse project management environments can also be covered (Mittal & Mehta, 2020). Software like Jira provides numerous instruments to facilitate daily routine management. Jira's adaptability

and customization capabilities have been critical factors in its success. By way of this, the teams may adapt the platform to their workplace practice.

b) **Azure DevOps Services** - Azure DevOps Services is an application service designed by Microsoft that is tailored to software developers working in a cloud environment and has all the necessary tools and services for revising and carrying out project planning, software building, software testing, and delivering the finished product. It entails various features, including Agile planning, version control, CI/CD deployment, automated testing, and DevOpers infrastructure options, enabling developers to develop under proper guidelines (Soni, 2020). Among the benefits of Azure DevOps Services is its integration with other popular development tools, including Visual Studio, GitHub, and Jenkins, which provides a versatile and flexible platform for many projects.

c) **Sonarqube** – Sonarqube is one of the most popular static code analyzer tools everybody uses nowadays to analyze the efficiency and security of code. It detects defects and security threats automatically, thus suggesting remedies to enhance code quality by providing actionable insights (Lalic et al., 2022). SonarQube supports a wide range of popular programming languages and allows for integration into the development process, thus providing dev team members with detailed insights on the bugs that might have been overlooked earlier when the development cycle was nascent. This software allows users to navigate from a high-level overview of code quality metrics.

1.4 Identify The Team For A Software Quality Metrics Product.

In search of a team for a quality metrics product, it is critical to focus on people knowing software development, quality assurance, data analysis, and project management. The team

formation should encompass people with different qualities but equipped with tools to facilitate effective communication, togetherness, and the respective goal performance (Lee, 2014). For a given software development team to be effective., it is essential to consider software engineers and developers, middle management, senior management, site reliability engineers, and support engineers.

Software Engineers and DevOps

The software engineers, as well as those with the DevOps job roles, work hand in hand in the development of software products by making sure the delivery of quality software is streamlined and, at the same time, optimized. Software engineers play a role in designing, building, testing, and maintaining software applications, while DevOps specialists work on implementing the automation and integration of software development, testing, and deployment processes (Zarour,2020). Software engineers are professional coders with expertise in programming languages, algorithms, and design principles. They can diligently handle customer requests and turn them into a working masterpiece that is programmed well and beautifully and has thorough testing done to ensure that the software will perform optimally. Software engineers happen to be the most critical professionals who work together with various persons of contact, among whom are product managers, designers, quality assurance persons, and other stakeholders, in the process of creating solutions and services that are meant to meet the needs of clients as well as the goals of the organization. (Venters et al., 2018).

In contrast, DevOps specialists strive to automate and orchestrate all stages of the software development, testing, and deployment procedures, aiming at faster and more reliable software delivery. They take full advantage of tools and technologies like CI/CD pipelines, containerization, and infrastructure as code to decrease the work to be done manually, increase collaboration

between the developers and operation teams, and identify any errors (Venters et al., 2018). In addition, the DevOps professionals concentrate on monitoring, logging, and performance optimization to maximize software application scalability, security, and availability once the applications are deployed in production environments.

Fundamentally, programmers and DevOps engineers collaborate to drive innovation, speed, and quality in creating software. They act as real-time reviewers and guide developers throughout the project as their code maturity is assessed by listing down code complexity, bugs, and standards the code needs to comply with (Ulan et al., 2021). These developers can also achieve the same outcome by detecting any quality issues even during the initial coding process, which makes it easier for them to avoid reworking the codes that would be needlessly time-consuming. Besides, metrics serve likewise as drivers of the refactoring efforts and product of resource allocation. Therefore, when these two teams integrate their efforts, they can develop and implement effective practices in software engineering (Ulan et al., 2021). This, in turn, will speed up the time to market, increase software reliability, and improve customer satisfaction in a world where new technologies are changing rapidly.

Middle Management (Scrum Masters, Project Managers, Portfolio Managers)

Middle management is crucial in software development, bridging upper management and development teams. This group typically includes Scrum masters, project managers, and portfolio managers, each with unique responsibilities contributing to the success of software projects. Scrum masters ensure the Agile scrum framework is correctly implemented during the development team's work process (Sharma & Spinellis, 2018). They conduct daily review meetings, remove blockers, and create an atmosphere of Agile culture among team members. Scrum masters can be considered servant leaders who assist the team in getting through their action plans and help them

bring their process up to date. The primary role of a project manager is to take care of various aspects of software projects, like planning, execution, and handing over the completed deliverables. They provide systems that ensure equal resource allocation, budget administration, reporting, and stakeholder communication. The project managers supervise the projects to ensure success in meeting the timelines, spending within the budget, and meeting the highest quality expectations (Serban et al., 2020). They play a vital role in risk management, conflict resolution, and business objectives, as well as the objectives of the project. However, a portfolio manager implements several projects, ensuring all these objectives are well-linked and result in high-value output, products, and services. They influence the project management process by allocating resources, following up on progress, and making sound decisions to improve the status of the project portfolio (Rathor et al., 2023). Portfolio managers, who balance the organization's project portfolio to reach planned objectives, are no less vital in helping the organization ensure adequate return on investment.

The middle management in software development represents a critical element of the wheel driving project success, providing room for collaboration, efficient resource utilization, and aligning projects with organizational goals (Rathor et al., 2023). They contribute to the actual visualization of the development state, the team performance, and the likelihood of delays affecting the project's timely completion. In these cases, they tend to concentrate on metrics that allow for assessing and calculating resources, allocating tasks to the most important, and risk reduction prospects that create a proper basis for decisions made by leaders and developers. Furthermore, they help collect concrete figures and data about a project's progress and when completion is expected (Petersen et al., 2015). In such a way, thanks to their competencies in leading, communication, and organization skills, middle managers play the role of bridging the gaps and

thus aiding in meeting the organization's requirements for providing quality software products and services.

Senior Management (Directors, VP, C-Level in Small Companies)

Senior management, including directors, VPs, and C-level executives in small companies, play a crucial role in the success of software development projects. Their leadership, strategic decision-making, and support can significantly impact software development initiatives' quality, efficiency, and outcomes (Pargaonkar, 2024). Senior management of small companies often does not settle for merely defining the strategic direction and objectives of software development projects but also acts as project managers. They are responsible for tailoring software development to the company's business goals so that projects follow the strategic direction and the business is organized around the mission, vision, and values. Senior management, therefore, plays an essential role when it comes to assigning resources, putting a plan in chronological priority, and, in particular, making critical decisions about the successful implementation of initiatives that have been developed in software engineering (P. Miguel et al., 2014). Besides the guidance and aid provided to the software development team by the senior management in small firms, they coordinate closely with the heads of the project, developers, and workmates in general, where they provide the needed direction, remove any hurdles, and ensure that goals are delivered on time as initially planned (P. Miguel et al., 2014). Thus, senior management develops and implements a culture of teamwork and innovation, letting cross-teams from different functions work together and enabling more effective and efficient teamwork.

Furthermore, senior management in small companies is responsible for championing a culture of quality, continuous improvement, and accountability in software development. They set the tone for the organization by emphasizing the importance of quality standards, best practices,

and adherence to industry regulations and standards (Oreški, 2022). They also encourage learning and development opportunities for team members, promote knowledge sharing, and foster a culture of transparency and open communication within the organization.

In conclusion, the managerial responsibilities of directors, vice presidents, and C-level executives in small businesses are crucial. Their authority and role are not limited to providing leadership, strategic direction, support, and assistance but also maintaining an innovative culture mainly focused on quality (Olorunshola & Ogwueleka, 2021). They use data-driven expertise that works across the whole software development process to improve the implementation of tasks and the program's performance. They base their opinion and decisions on metrics analysis to identify the need to rebalance and optimize the use of available resources across the projects and the organization. Aside from this, they are quality-oriented and offer tangible evidence, which is always suitable for having a clear idea of what still needs improving and how that could or should be done (Mittal & Mehta, 2020). This results in redolence in the success of a software development project and the organization's performance in the marketplace competition.

Site Reliability Engineers and Support Engineers

Site Reliability Engineers (SREs) and Support Engineers play crucial roles in software development by ensuring the reliability, availability, and performance of software systems and applications. While they have distinct responsibilities, they contribute to the overall success of software projects by managing technical operations, resolving issues, and supporting the smooth functioning of systems.

Site Reliability Engineers' primary duty is devising, starting up, and maintaining dependable and transferable infrastructure and systems. They collaborate with the development

team to meet the performance and reliability regulations. SREs employ monitoring, alerting, and automation systems, which help them detect and resolve issues promptly, optimize the overall system performance, and avoid downtime (Mishra & Otaiwi, 2020). With the software engineering approach to the tasks, SREs make the system more fault-tolerant, efficient deployment, and deliver high-quality service. Developers are the ones who write the codes for the applications and test and debug them.

On the other hand, support engineers are responsible for providing technical support to users, solving problems, and answering users' questions and incidents. They become the customer's initial contact point, where they can get help explaining software features and finding the root of a problem (Masmali & Badreddin, 2020). Support engineers interwork with cross-group teams to focus on development, testing, and product forwarding to ensure that issues are appropriately prioritized, customer concern is appropriately communicated, and resolving them is done on time.

SREs and Support Engineers complement each other's roles to guarantee the successful performance of the software product. SREs involve themselves in constructing and maintaining resilient systems, excluding the critical support provided by Support Engineers who directly attend to the users' needs. As a whole, they provide the necessary support to ensure that software development groups can fulfill the quality standards for their products, fulfill the customers' expectations, and facilitate growth in this business (Martínez-Fernández et al., 2022). Their primary focus is to ensure that the project team's scope is in the current version, contributing to the attainment of project deliverables and mission. By participating in these collaborations, software companies embrace new technologies, remain competitive in a highly changing environment, and continue to improve their business performance.

1.5 Unpredictability of Project/Program Implementation

In the current industry age, where every organization competes for supremacy, projects must be impeccably executed, meeting the set timeframe, budgetary needs, and expected results. The greatest problem for software developers and project managers, the greatest problem they might face when they try to implement a certain project or program is its unpredictable nature (Ciric et al., 2019). The fact that project implementation may fail unexpectedly can be disruptive to the organization, resulting in financial losses and customer dissatisfaction, making it the most serious challenge for the companies. The uncertainty of the implementation processes has always been among the industry's leading problems. While establishing the road map and implementing it accurately will minimize some ambiguity associated with the project budget, timeline, and outcomes, the numerous discrepancies might result in project delay, over-budget, and poor performance (Mashiloane & Jokonya, 2018). For businesses, one of the critical issues is that it is hard to predict and forecast the project's costs, schedule, and advantages. This lack of predictability can result in cost overruns, schedule delays, and benefits shortfalls, significantly impacting the organization's bottom line. According to Bloch et al., (2012), half of IT projects often and substantially overrun their budgets. On average, these IT projects often overrun their budgets by 45% and their time by 7% while delivering 56% less value to their customers than expected, underlining the significance of proper project planning and execution.

Impact on Customers

It is terrifying that despite the widely recognized need to ensure that projects deliver benefits to the customer/user, attaining this goal in the modern world is getting scarcer, and a lot of money spent on projects continues to be squandered. Business projects are implemented to address business problems (Dlamini & Cumberlege, 2021). This, therefore, means that business problems

affecting the customers are not addressed when business projects fail to implement and utilize their results. Most researchers and professionals in project management agree that business projects do not fulfill their objectives until project results are used to benefit customer organizations (Paraskevopoulou et al., 2022). Yet the customer/user is one of the most important factors affecting the success of implementation and utilization of project results. The benefits derived from project results arise from the change that the project causes in the customer's environment. This means that the benefits that derive from project results are associated with the customer/user rather than with the immediate objectives of the project. Project management literature also indicates that the customer's effective use of project results measures project success (Dlamini & Cumberlege, 2021). Obviously, for a project that has delivered results and yet the results are not being used, something must be wrong. Indeed, this defines a project that is failing.

The unpredictability of project or program implementation can significantly impact customers, who are the ultimate beneficiaries of the project outcome. Customers expect projects to be delivered on time, within budget, and with the expected benefits. However, when projects experience cost overruns, schedule delays, and benefits shortfalls, customers are dissatisfied and may lose trust in the organization (Paraskevopoulou et al., 2022). This project experienced substantial losses due to cost overruns, schedule delays, and benefits shortfalls. The financial implications of this project were enormous, with a cost overrun of $66 billion, representing a 44% increase from the initial budget. Additionally, the project experienced a schedule overrun of 7%, leading to project completion and delivery delays (Bloch et al., 2012). Furthermore, the benefits shortfall of 56% indicated that the project failed to deliver the anticipated benefits to the organization.

1. **Cost Overruns:** Cost overruns are among the most common consequences of project unpredictability. Cost overrun is best described as a situation where the actual costs of a given project exceed the budget set. Compared to other organizational projects and programs, these cases are common in large-scale IT project implementations. The cost overflows of the projects is 44%. Some of these costs are higher even though the average is 44% (Bloch et al., 2012). The cost overruns can also reduce profits, cause bleeding of resources, and taint organizational image. Price overruns might result from various circumstances during the project cycle, including the increased cost of raw materials and project scope change.
2. **Schedule Delays**: Customers often endure long wait times and delays because of project cost overruns. In other words, a schedule delay is any unexpected situation, which may not allow the project team to stay on the schedule or complete the project in the set time (Ciric et al., 2019). This may lead to incomplete project tasks and milestones on time, resulting in project extension due to late project activity. In general, most projects face schedule variability of about 7%, making some projects miss deadlines, resulting in product launching delays and missed commercial opportunities (Bloch et al., 2012). Customers who conditioned their timely project execution may be perplexed that the timeline is becoming far longer, thereby unpredictably impacting their operations. The delays may be due to planning, resource variables, project scope changes, poor communication, task dependency, skills, training, and complexity.
3. **Benefits Shortfalls:** The essential idea of project uncertainty is the missed achievement of timely gains. This is the difference between a project's successes weighed against the set expectations. Some come way short, with a potential shortfall of 56% at maximum (Bloch

et al., 2012). Such a situation can be very frustrating for the clients who have dedicated time and resources to the project and now realize that the expected results are unmet. A project not meeting the planned goals is a failure that brings financial losses, damages the business's reputation, and harms the company's strategy. In addition, the clients are also typically annoyed by the final results resulting in a loss of customers and business (Ciric et al., 2019). The inability to achieve results can be attributed to losing focus, working on content issues, dealing with skill problems, or dealing with execution failure.

Addressing the unpredictability of the project

Organizations must take proactive measures to improve project management practices to address the unpredictability of project or program implementation and mitigate its impact on customers. Some key strategies include:

1. **Robust Planning:** Organizations must allocate adequate time and resources for established project management practices, such as detailed risk assessment, stakeholder engagement, and resource allocation. Robust planning comprises the roles of the teams unified concerning the main goal with a shared vision as the foundation, team process, and a good culture (Simushi & Wium, 2020). A crystal clear project plan is essential in dealing with unpredictability's impact and controlling the project's timelines. As a matter of fact, through strategic planning and follow-up of activities, the manager can create a direction and make sure that the project goals and objectives are in tandem with the overall organizational objectives and the company vision.
2. **Agile Approach:** Agile project management techniques like Scrum or Kanban can assist organizations in managing unpredictability and aspirations, and it lets organizations

stream their progress and produce value by adapting flexibly. When the project timeline is not running as planned or has a high risk of dramatic delay, the project manager should adopt a flexible approach to avoid major delays (Kerzner, 2017). Organizations can divide projects into smaller, better-maintained tasks to quickly respond to any changes or delays, helping projects to be completed on time and within budget. Moreover, the project team can collaborate with the project leader in undertaking the progress follow-up through a dashboard that measures the system's activities or lifecycle conditions (Kerzner, 2017). This, in turn, guarantees that the minor scopes of the projects are achieved in time by monitoring closely, any possible delay is addressed appropriately, and any issue is addressed on time.

3. **Effective Communication:** Good communication with the key partners is vital to the project's success. Good communication can sometimes factor in a team's winning or losing. A good communication tactic allows communication to be effortless; hence, messages, information, and details are exchangeable in both directions (Idrees & Shafiq, 2021). Furthermore, effective communication, continuous and timely information, reports, and reception of opinions support all parties in sticking to the tasks and conforming to the project's stated goals. Lastly, communication remains one of the keys to harmonizing all the project stakeholders to gain the same target and goal.
4. **Continuous Monitoring:** Supervision and tracking the performance of the project can assist companies in identifying some phenomena that are potential warning indicators and correcting them before they escalate. Therefore, looking closely and constantly evaluating the changing cost is the key to limiting the huge inflation. Even though the cost change must be noticed even if it may sound small. When these costs are identified, the project

team should update the master project budget to account for these additional costs by changing the appropriate budget items (Idrees & Shafiq, 2021). Also, continuous monitoring of hidden costs like large tasks related to project overhead, company overhead, additional operations, and project length should be on the project team's agenda.

5. **Post-Implementation Review:** Conducting a thorough post-implementation review can help organizations assess the project's success, identify lessons learned, and adjust for future projects. This feedback loop can help improve project predictability and customer satisfaction over time.

Apart from the customers, the organization suffers greatly from failures and delays. The influence of such happening as the project is incomplete is likely to have a large negative effect on the organization's activities. Higher cost fluctuations are challenging for the organization's financial basis, leading to poor business performance and increasing the risk of project failure (Bloch et al., 2012). The overrun in a budget of $66 billion was a burdensome financial expenditure to the organization, which meant there was a need for more funds and resources to be allotted to the project to complete it as intended. Organizations can also be affected by schedule delays with resulting production stoppage, missed deadlines, and lost revenues. Also, the 7% overrun of the schedule caused a delay in the project completion, leading to operational disruptions that might inflict the organization a damaged reputation. Additionally, the project could not meet the benefits shortfall of 56% which meant that the expected advantages were not being delivered to the organization (Bloch et al., 2012). This leads to the loss of stakeholder confidence and causes a decrease in their morale, which affects the company's functions.

The Goal of the Quality Metrics

Quality metrics are crucial in ensuring the effectiveness and efficiency of software development processes. By measuring and analyzing specific aspects of software development, quality metrics provide valuable insights into areas where improvements are needed (Garousi et al., 2020). The goal of the quality metrics system is to focus on and provide enhancements in two key areas: content issues and execution issues.

Content Issues

Changes in scope can cause content problems in software development. This change can result in one of many outcomes, including creating, facilitating, or even distributing flawed content. These issues include violating the formatting, broken links, messages, or date inconsistencies of the content, resulting in errors (Garousi et al., 2020). The content issues highly influence the entire software experience, including usability, functionality, and credibility. They can be categorized into two main groups: shifting requirements and technical complexity.

In our ever-present, fast-paced software world, requirements for changing conditions, such as market trends and emerging information, are often the subject of change. Notably, frequent updates can impair development, cause delays and rework, and incur extra expenses (Ehrlinger & Wöß, 2022). This is where the quality metrics system comes by, to prioritize content criteria. Then, the system will identify the root cause of shifting demands and generate best practices to deal with these changes and requirements effectively.

In addition, lack of know-how is also something else that could be problematic for software teams. While the software is becoming increasingly smart and integrated, the technical problems and complexity may raise and complicate the development of quality products and on time and

demands (Zuse, 2019). The quality reliable measures try to focus on the technical complexities by examining and analyzing things like code complexity, interdependencies of the systems and integration problems. Attaining an understanding of the project's complexity is key, as this can allow development teams to solve technical challenges and improve the overall quality of the software.

One of the main reasons for addressing content issues in software development is the need always to enhance users' experience levels. Notably, every time a user finds themselves dealing with 404 errors, searching for current relevant details, or just generally being confused, users feel frustrated and annoyed. This might force or motivate them to uninstall the software and look for other, more convenient solutions and software (Zuse, 2019). In both cases, users will be no longer interested, negatively impacting engagement and retention. In addition, misleading or stale content information or features will lead to a fall in the users' trust and harm a company's brand as non-valuable (Kerzner, 2017). It is worth noting that failing to address the issues promptly can potentially harm the software's reputation and brand position, reducing the number of customers attached to the company and causing a loss of revenue by the company.

Solving the content problems of software development is about making content management the focus and enforcing content quality standards at each step of the development process. Among other things, the content management team is expected to perform content audits and strategy protocols and establish rules to be followed during content creation and editing (Ma et al., 2018). Before the software is released into the market, developers should take another look at its content to avoid situations where the users face content problems and compatibility failures.

Execution Issues

Execution issues are subject to time and date constraints and emergency fixing. Their essence lies in these problems occurring when the software design development process becomes a functional product. Data and code quality, technical bugs and errors, and ineffective resource allocation often arise during the execution process (Dingsøyr et al., 2017). How execution issues are handled is paramount to completing any software development project, irrespective of the nature of the challenge. Mainly, these issues are the cases of imperfect planning or just-in-time planning. These execution issues can be divided into two main categories: unrealistic scheduling and reactive planning.

Unrealistic core schedules may force the team to cope with great stress, which, in turn, will lead to rapid burnout, low morale, and diminished quality. Through monitoring and evaluating failure to meet schedule, the quality metrics system can indicate where the work is being delayed, wasting resources, or being affected by other environmental factors (Dingsøyr et al., 2017). With this data, projects' status can be monitored better, schedules changed, resources allocated more effectively, and projects are on time and within budget targets.

Reactive planning, inefficient communication, and coordination among stakeholders involved in software development can often impede successful projects, too. Often, the teams have to show the agility to adjust to the effects of the unforeseen and issue changes while the development process is still running. Unfortunately, this may lead to confusion, delay, and bad decision-making (Edison et al., 2022). The metric system for quality supports proactive planning as it provides teams with on time data, which enables them to look for possible risks for their upcoming events and to prevent them from becoming urgent. By promoting a reactive plans

culture, development teams will be capable of easier adaptions to changes. They can put less effort into developing top-notch software, adversely affecting the company's software quality.

The execution issues create many problems related to timelines in completing the projects. When delays occur at the development stage, they can break the project regime and lengthen the timeline of the whole project. As a general rule, execution issues result in project delays, poor communication, and low efficiency, affecting the final product's overall quality (Edison et al., 2022). The users may feel hurt whenever project delivery is delayed considering that this may mean lost opportunity or revenue. If software developers try to match deadlines, bypassing developmental stages, it can imply unnecessary cuts, resulting in lower code quality. The inefficient and often problematic software development with bugs and glitches may lead to instability and security issues, affecting the overall benefits of the software product. Below-standard software can adversely affect a company's image, cause customer dissatisfaction, and otherwise result in costly debugging and support efforts (Ur et al., 2019). The world of software development can be a stressful place when problems and bugs arise. Team members may turn to the blaming game instead of working together to find a solution. This can result in communication and trust issues. This will create a bad atmosphere, making collaborating more difficult and influencing the whole group's performance.

As such, software development process effectiveness and efficiency are achieved through various criteria, such as the quality metric system that works through data collection, analysis, and interpretation concerning content and execution issues. In the same case, metrics such as code and requirements can measure software code's degree of complexity and maintainability and track any changes in the project's requirements (Ur et al., 2019). Similarly, metrics on schedule allow

developers to track milestones and deadlines, while risk metrics identify factors that could thwart the project's success.

Through utilizing those metrics, development teams will possess the unique opportunity to obtain valuable insights into the reasons for content issues and poor execution. The specific code metrics analysis may reveal that there is one module with too much complexity and is error-prone (Lundell et al., 2021). This will make them prioritize refactoring to gradually provide a code that is easy to read and less likely to have defects. Moreover, when the schedule analysis shows that the project is lagging, project managers can adjust resources, schedules, or priorities to alleviate the situation.

1.6 Implementation Process and Challenges

The broad software development and implementation phases include the initial stages of requirement analysis, design, development, testing, deployment, maintenance, and support. Requirement analysis, designing, and running these programs create various problems for organizations and developers, which, if immediate action were taken, the developed programs and services would hopefully yield successful results (Lundell et al., 2021). The process underlines the significance and importance of the Quality Metrics System implementation project to guarantee success. The leading steps in this process include:

Market Research

Broad software development and implementation phases, like requirement analysis, designing, implementation, testing, integration, maintenance, and support, require careful planning and execution. Designing and developing those programs involve multiple problems that officials and developers deal with in a short-term period, and hopefully, with serious actions, the problem will

be solved, and these programs work effectively (Vallon et al., 2018). By conducting market research to assess available platforms and tools, organizations can:

1. **Identify existing solutions**: Market research allows companies to explore the market by looking for the already available platforms and tools that may serve customer's purpose for meeting their needs. It is possible to achieve this by not beginning the solution development from the start but instead based on available knowledge (Vallon et al., 2018). Furthermore, it enables developers, companies, and users to gain insights into the areas that need improving, leading to better interaction between the customer and software quality.
2. **Assess features and functionality:** Market research enables the company to compare the lived experiences of different product features, capabilities, and functionality presented by the available platforms and tools. The attributes, work capacity, and other factors contributing to information technology development research are vital (Ur et al., 2019). Researching a customer's needs and amending features to allow for further user-oriented testing will help develop products that meet the users' expectations. User feedback, usability testing and examination of what competitors offer are vital while deciding what to have and what to skip in the list of functionalities. On the other hand, the developer can focus exclusively on developing and integrating the necessary features to make the service more appealing overall to the user(s) (Mashiloane & Jokonya, 2018). Hence, the strategic planning tool enables their products to triumph over competitors and get into the customers.
3. **Evaluate costs and benefits:** In software development, marketing research focuses on expenditure regarding the time and resources allocated to survey collection and the use of experts in the field. On the contrary, these benefits surpass the costs, enabling developers to recognize customers' needs, consider market demand, evaluate competitors, and follow

industry trends to determine pricing strategies (Lánczky & Győrffy, 2021). Through market research on people's choices, developers can achieve product success, determine their products' competitiveness, and receive profitability in the market. Market research assists organizations in determining which option is more viable between leveraging available platforms and tools or developing in-house solutions related to costs and benefits (Lánczky & Győrffy, 2021). This gives companies the benefit of comparing various cost options to choose the one that is the most valuable in terms of cost.

4. **Understand market trends:** Market research in software development includes developing products based on potential customers' feedback, needs, and expectations. The merits focus on identifying customer needs, appraising market demand, judging competition status, knowing the industry trends better, and finding the best pricing strategies (Lundell et al., 2021). This has to be done with balance. The profits from these activities distinguish a successful brand and shelf positioning. Market analysis offers information about the market trends, technologies in development, and the common practices that exist in the industry (Ma et al., 2018). Since their managers are well-versed in the latest market trends, organizations can make strategic moves on technology adoption and innovation by increasing efficiency and productivity.
5. **Foster collaboration and partnerships:** With the help of market research, organizations can find reliable partners for example, suppliers and vendors, that give attractive deals or have other beneficial propositions. Allying with external providers helps in quick rollout and reduces development expenditure, and organizations can connect with the world of special knowledge. Collaboration and synergetic approaches to software development are vital since they contribute to greater leadership, knowledge sharing, and

market outreach (Dingsøyr et al., 2017). Through collaborations with other entities, pooling resources, and working together towards the same aim, developers can highlight what is ideal and strong in each other and build on their growth while offering value to customers.

One of the other crucial questions in market research that should be analyzed is selecting the right vendors. Vendor assessment is a crucial step in IT projects, and it helps select vendors through parameters such as products they introduce to the market, their competitors, and what services they offer. The choice mechanism of this candidate is vital in MVP which screens out the best vendor to sell every software solution matched with the required support for effective implementation (Ciric et al., 2019). Considering key aspects like the supplier's reputation and experience, having the right technical skills, and offering competitive prices get priority when pre-selecting. A firm has a good chance of minimizing risks, ensuring high work standards, and compiling a task successfully by looking at a vendor through all possible sides before engaging this expert for the particular job (Ciric et al., 2019). Vendor examination, the entrepreneurial undertakings establish the basis most likely to turn the organization into a profitable unit that achieves its goals and objectives. The development team pointed out the most critical subject areas for a perfect fit for the vendor. The two main factors are the overall cost estimate of our project, services offered, main goal, and objective, as highlighted in the attached Excel spreadsheet.

Technical Assessment

Using the existing development team and saving on implementation costs is necessary to extract as much value as possible from minimal effort. However, not every instrument will do the trick. Therefore the choice of the right instruments for the integrations is prevalent. Excellent examples of technical assessment tools for software developers include Azure DevOps and SonarQube. Features and characteristics such as communication channels through APIs, custom

work items for project tasks, and agile boards for the project progress's visual flow make Azure DevOps one of the most effective Project management tools (Ciric et al., 2019). With an integrative approach with Azure DevOps, the software development team can easily administrate and control the project issues, collaborate on code development, and automate the projects' delivery process, enhancing the customers' value with minimum costs and effort.

The SonarQube integration is mostly facilitated by plugins for code analysis, custom rules devoted to code quality standards, and webhooks for automating review procedures. Integration of the SonarQube Application allows the developers to recognize and manage the code quality problems at an early stage of development, improve its maintainability, and ultimately make informed choices to optimize the software development process (Dlamini & Cumberlege, 2021). By integrating these tools, an organization can provide the highest possible value to the company, realizing the full potential of developers' skills and finding the cheapest implementation resources. This method guarantees a well-rounded consolidation of experience, ensuring better collaboration and teamwork among team members and promoting higher efficiency in the context of the software production cycle.

User Interface

Our user interface implementation consisted of us doing a sample interview with development teams, division managers, and senior managers. We collected user feedback that focused on the current user interface's uncomfortable areas or issues and included customers' favorite features in user interface design. The interviews provided the kickstart for the critical features of a user-friendly interface, like clear navigation, uniform display, and swift access to frequently used tasks (Edison et al., 2022). These guidelines will help ensure we build the user interface aimed at what our customers need. As a result, we build the following design:

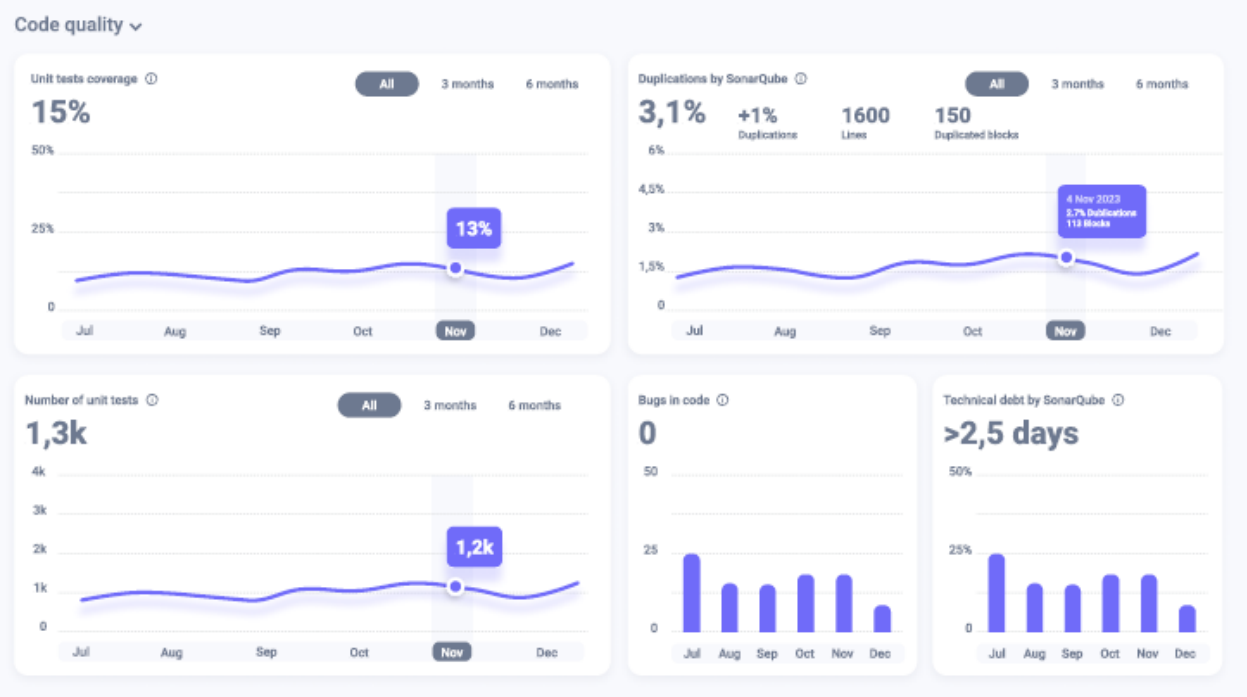
Delivery

Total Scope	Iteration velocity trend	Scope completed	Scope planned to be done	Scope Incease	In track with scope?	Total scope completed
780	30	404	376	78	Yes	53%

Sprint	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Planned to be done, SPs	704	680	671	673	666	641	611	577	544	510	478	435	392	354	309
Completed Total, SPs	0	24	50	80	111	149	185	227	271	311	350	398	444	488	535
Scope changes, SPs	0	0	17	32	24	13	6	8	11	6	7	5	3	6	2
Completed per sprint, SPs	0	24	26	30	31	38	36	42	44	40	39	48	46	44	47

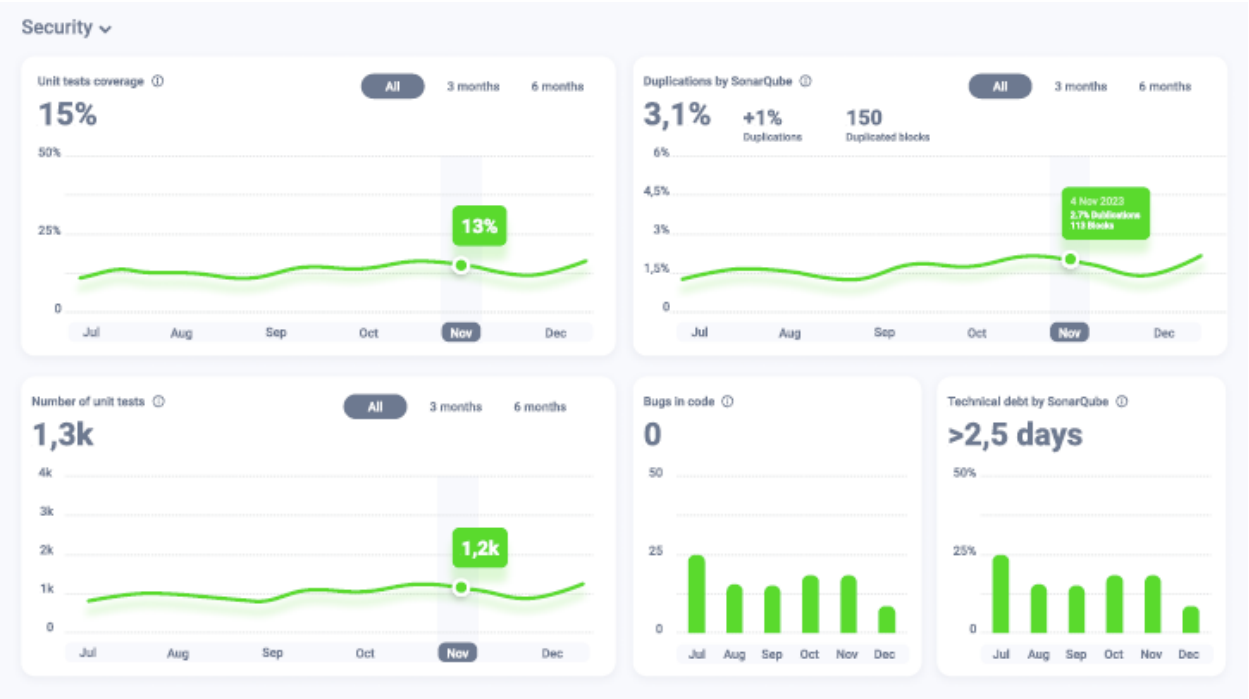
Budget

Month	May 23	Jun 23	Jul 23	Aug 23	Sep 23	Oct 23	Nov 23	Dec 23	Jan 24	Feb 24	Mar 24	Apr 24	May 24	Jun 24
Budget Aggregated, k USD	250	520	820	1150	1515	1935	2345	2735	3175	3585	3965	4289	4579	4764
Actual Costs Aggregated, k USD	174	385	674	1016	1426	1901	2366	2821	3301	3733				
Budget per month, k USD	250	270	300	330	365	420	410	390	440	410	380	324	290	185
Actual Costs per Month, k USD	174	211	289	342	410	475	465	455	480	432				


Code quality
Unit tests coverage
All
3 months
6 months
15%
50%
25%
0
13%
Jul
Aug
Sep
Oct
Nov
Dec
Duplications by SonarQube
All
3 months
6 months
3,1%
+1%
Duplications
1600
Lines
150
Duplicated blocks
6%
4,5%
3%
1,5%
0
4 Nov 2023
2.7% Duplications
113 Blocks
Jul
Aug
Sep
Oct
Nov
Dec
Number of unit tests
All
3 months
6 months
1,3k
4k
3k
2k
1k
0
1,2k
Jul
Aug
Sep
Oct
Nov
Dec
Bugs in code
0
50
25
0
Jul
Aug
Sep
Oct
Nov
Dec
Technical debt by SonarQube
>2,5 days
50%
25%
0
Jul
Aug
Sep
Oct
Nov
Dec


Code quality
Unit tests coverage
All
3 months
6 months
15%
50%
25%
0
13%
Jul
Aug
Sep
Oct
Nov
Dec
Duplications by SonarQube
All
3 months
6 months
3,1%
+1%
Duplications
150
Duplicated blocks
6%
4,5%
3%
1,5%
0
4 Nov 2023
2.7% Duplications
113 Blocks
Jul
Aug
Sep
Oct
Nov
Dec
Number of unit tests
All
3 months
6 months
1,3k
4k
3k
2k
1k
0
1,2k
Jul
Aug
Sep
Oct
Nov
Dec
Bugs in code
0
50
25
0
Jul
Aug
Sep
Oct
Nov
Dec
Technical debt by SonarQube
>2,5 days
50%
25%
0
Jul
Aug
Sep
Oct
Nov
Dec


Progress
Unit tests coverage
All
3 months
6 months
15%
50%
25%
0
13%
Jul
Aug
Sep
Oct
Nov
Dec
Duplications by SonarQube
All
3 months
6 months
3,1%
+1%
Duplications
150
Duplicated blocks
6%
4,5%
3%
1,5%
0
4 Nov 2023
2.7% Duplications
113 Blocks
Jul
Aug
Sep
Oct
Nov
Dec
Number of unit tests
All
3 months
6 months
1,3k
4k
3k
2k
1k
0
1,2k
Jul
Aug
Sep
Oct
Nov
Dec
Bugs in code
0
50
25
0
Jul
Aug
Sep
Oct
Nov
Dec
Technical debt by SonarQube
>2,5 days
50%
25%
0
Jul
Aug
Sep
Oct
Nov
Dec


Security
Unit tests coverage
All
3 months
6 months
15%
50%
25%
0
13%
Jul
Aug
Sep
Oct
Nov
Dec
Duplications by SonarQube
All
3 months
6 months
3,1%
+1%
Duplications
150
Duplicated blocks
6%
4,5%
3%
1,5%
0
4 Nov 2023
2.7% Duplications
113 Blocks
Jul
Aug
Sep
Oct
Nov
Dec
Number of unit tests
All
3 months
6 months
1,3k
4k
3k
2k
1k
0
1,2k
Jul
Aug
Sep
Oct
Nov
Dec
Bugs in code
0
50
25
0
Jul
Aug
Sep
Oct
Nov
Dec
Technical debt by SonarQube
>2,5 days
50%
25%
0
Jul
Aug
Sep
Oct
Nov
Dec

Project Implementation

As a first step of the project implementation, the group comprehensively analyzed the resource requirements and assembled all necessary schedules. The most important person in the project was the project manager, who ensured the project was on track and coordinated our work. The back-end engineer created the system core performance, whereas the front-end engineer tended to improve the user interface. The DBA engineer established the database foundation and ensured the data management infrastructure of the system worked out smoothly (Ehrlinger & Wöß, 2022). However, this whole time, the quality control engineer intently performed comprehensive testing, which made it possible to identify and resolve any bugs or issues that might occur.

The team and I worked closely through the implementation phase to ensure the system was built according to specifications and met project requirements. We regularly held meetings/ communications and updated stakeholders to keep them updated on progress (Garousi et al., 2020). After we delivered the project, the team performed the routine phase where we were tracking the system and ready to provide technical support to address any arising issues. Calculating the ownership cost of the system is a critical component of budget planning not only to comply with the organization's budget constraints but also to ensure that the system can sustain ongoing support and maintenance.

The total amount of financial support expected to be spent by the system was calculated to include all expected expenses to keep the system in operation and ensure its proper functioning. The cost of $3200-4000 per year consists of various elements; among them are the AWS expenses for the hosting and the infrastructure, which are continuously monitored so that the system can offer high reliability and scalability. Technical support staff are instrumental in the implementation process, irrespective of the cost of hiring and retaining them, considering that they are the ones

who are involved in the systems' maintenance and operation (Idrees & Shafiq, 2021). This team offers indispensable support in never-ending problem diagnosis, software update documentation and ensuring the system runs smoothly and up-to-date. Lastly, the whole project got a great boost from the passionate spirit of all its members, which was the main driving force. The system is considered functional and can be used in the long run to serve its users' interests.

Challenges and Risks

Software development and implementation are complex, often involving various challenges and risks. Some common challenges and risks in software development and implementation include tool compatibility, defining meaningful metrics, setting realistic thresholds, data overload, maintaining the system, security, and system introduction.

Tool Compatibility

Tool compatibility is a factor that can make or break a software project, as the efficacy of processes, resource utilization, and user experience can be improved through automation. Nonetheless, synchronization problems often occur when different tools are integrated into one platform, requiring time-consuming and financial efforts to adapt (Kerzner, 2017). A feasible option is given priority to integrate into customers' early used toolset while the MVP is at its initial stages. This approach speeds up the development and testing functions, and users enjoy a high-quality experience. The platform will be enhanced with more features with time, increasing its appeal and client base. Companies can avoid adding complexity and implement such tools more successfully by properly choosing just a few integrated tools and prioritizing those that positively impact users.

Defining Meaningful Metrics

Defining meaningful metrics is the most important when measuring software projects' effectiveness and relevance. The metrics, such as the number of registrations, views, and so on, necessary to be tracked, relevant, and valuable to customers, were derived from the tool analysis and stakeholder feedback (Lánczky & Győrffy, 2021). We can gauge progress, measure the attainments, and analytically make decisions to guarantee project success through such rankings. These significant figures will create a possibility to take necessary actions and ensure that the software meets the expectations. We can also build a system targeting critical areas and powering business development through measurement that aligns with customer concerns and beliefs.

Setting Realistic Thresholds

Setting realistic thresholds for software quality control and quality analysis are significant factors in software development. The right measure level has to be defined for every quality metric. Otherwise, it would not be feasible to enhance the project quality, reduce technical debt, or align with business features. The organization can achieve successful performance in product management and keep improving product quality on time through flexible quality levels set to meet project needs (Ma et al., 2018). This strategy makes the model reactive to changes and updates quality parameters as needed. By setting targets that will match the objectives and aims of the project, companies should harmonize quality assurance and the implementation of projects, and this way, they can improve the entire course of the project, leading to better customer satisfaction. The toughest part of quality measurement is determining the right measure for each metric (Mashiloane & Jokonya, 2018). Unfortunately, our strategy for improving the quality was not determined. We make recommendations that can lead to the project quality increase, the decrease of technical debt, and a balance with the business characteristics. Furthermore, every

project can specify its degree of quality and use that as a relative measure to improve gradually or worsen the quality level.

Data Overload

Software developers often find analysis problematic because of too much data. First, the appropriate metrics should be picked and categorized regarding stakeholder weight. Using the matrix and the three groups of metrics aimed at each stakeholder group will help the developers track and monitor progress toward the defined goals and focus on the most important performance indicators. It allows filtering so that reports have only relevant metrics rather than having uncountable cluttered reports (Paraskevopoulou et al., 2022). By categorizing and emphasizing essential metrics aptly, the software development teams can prevent drowning in the data and, as a result, will be able to make data-driven decisions. Data analysis provided not only the efficiency for the project but also the creation of sound, decision-making processes. Besides the traditional, there were dozens of others (200+), and not all of them deserve to be put on dashboards; however, metrics were sorted into three groups for each group of stakeholders.

Maintaining the System

The quality metrics (QMS) system is a dominating internal tool that requires increased investment in its development and support (Ur et al., 2019). This framework presupposes continuous resource allocation into upgrading and implementing the quality metrics system. However, still there were talks about possible commercialization of the product, maybe possible at the stage when additional funds will be needed. The system used in this industry must constantly be upgraded and supported to be consistent with the industry's modern standards. Owing to the growth possibility in commercial usage, continuous condition and advancement of the system should be emphasized to increase its usefulness and user-friendliness (Vallon et al., 2018). The

developers can sustain the data collection process through monitoring and continuous improvement and reach the established targets.

Security

The system will store only the metric data, including project details, executors' tasks, and codebase to ensure security. It means that the risk of access to personal information or having it compromised is effectively eliminated, thereby minimizing potential unauthorized access (Zuse, 2019). Among the options, the system avoids selling out the passwords to valuable data by saving only vital metrics in the stored database. Such prevention policy complements the data privacy and protection practices that align with the best cybersecurity practices, providing better protection of the data stored in the system.

System introduction:

Project level

Implementing a new solution at the project level might be met with resistance. Nonetheless, starting early and explaining how the tool can enhance their performance metrics can help them accept and utilize it effectively and accurately. The system can facilitate the preparation of reports, including progress and quality reports, and help the project team to save on time that could have been spent preparing these reports (Simushi & Wium, 2020). Although people may have concerns and difficulties in communication, showing the importance of the tool and its resulting efficiency would be essential for organizations to overcome resistance in project management.

Senior Management

Leadership should focus on a new system by closely examining the abovementioned areas. Organizations can communicate with customers by listing the new system's advantages and

competitive advantages. Besides the system being a monitoring point during the project's implementation phase, it also helps to enhance the efficiency of processes and the standard of the outputs (Ma et al., 2018). It also creates safe environments and carries through the operations in the most optimized manner to help the top management buy into the idea so that the implementation is effective and leads to improved outputs.

Customers

Clients can now benefit from 24-hour monitoring of the details of their projects through our new system, and yet, they do not have to be bottlenecked amid the developments as they can stay updated with real-time updates. Instantaneously, all data will be updated on which products clients are informed of the most recent information. In the process, efficiency and promptly responding to any changes are increased. In transparent metrics, trust between customers and the project is also increased because customers can follow the progress and results of the projects with simple techniques (Idrees & Shafiq, 2021). Organizations can, therefore, make progress in enhancing customer satisfaction, creating a bond with customers, and in project delivery, by giving customers data that is easily accessible together with promoting transparency in metrics.

The following issues/risks remain unresolved:

Difficulty Linking Metrics to Impact

The key issue directly ties the metrics to the outcome or the connection between quality and the underlying change affecting the project. Difficulties in linking metrics to inputs focus on the connection between the technical debt's contribution and the future tech support of the quality prospect and its effects on business value. Through a closer look at such fields, an organization can acquire valuable information concerning the impact of these quality management practices on

its short-term achievements and the long-term future (Idrees & Shafiq, 2021). However, resolving unaddressed issues should be at the forefront of improving decision-making processes and development aligned with organizational goals.

Focusing on the Wrong Metrics

The wrong metrics were a major part of the interview results, as it was realized from the participants that most of them were from service companies. For their fair processing mechanisms, the evaluation metrics may be misaligned with end-users actual requirements by being subject to a distorted representation. The development team proactively worked towards dealing with this problem by teaming up with three customers to validate the dashboards, issues raised and identifying the right metric. This framework envisions partnering with various organizations to implement policy changes, including granting access to their systems and ongoing metrics monitoring, facilitating the development of performance indicators aligned with the end-users needs and priorities (Ciric et al., 2019). Through clients' participation in dashboard development, the mitigates the risk of irrelevant and unimpactful activities attributed to irrelevant metrics and, instead of the boring data, delivers meaningful, actionable insights that will create business value and enhance customer satisfaction.

Conclusion

In conclusion, quality metrics are key to a software development system. These projects use these metrics as a means that the software engineers and other upper management teams can use to track the project development process, identify the gaps, and then design the interventions that will help ensure that these projects are delivered on time, within the budget, and meet these customers' needs and want. The main goals of these metrics are to increase customers' pleasure, strengthen the competitiveness of a single firm and business, and use accurate information to make decisions that enhance efficiency and effectiveness. The research findings identified key metrics: scope, team performance, and financial and technical. Notably, through these metrics, project managers can turn qualitative data into quantitative data, making it easier to track progress, analyze and evaluate the obtained data, and make well-informed decisions. This, in turn, ensures that companies can forecast future trends and prepare for future changes. In addition, through these metrics, companies can access data-driven insights about the entire process, provide 24/7 customer support, and address backlogin issues without affecting project completion. Organizations should focus on software engineers, DevOps, ideal management, senior management, and site reliability engineers to create a cross-functional team to deliver high-quality products. In addition, proper planning and integration of efforts from all team members is instrumental in addressing all software implementation issues and guaranteeing success.

Suggestions for Future Research

Software quality metrics systems can be introduced in IT service provider companies, allowing them to discover new opportunities, profit from efficient processes, manage operations, improve customer satisfaction, and attract prospects. The current study shows that companies can enhance quality, productivity, and market staking through metrics measuring. Thus, future studies should touch on external factors, like state requirements, industry standards, and customer expectations, and how they affect the development and implementation of software quality metrics in IT service provider companies. Comprehending how these driver external factors and how quality management practices fluctuate could lead to the companies structuring metrics systems so that those could be used for specific compliance purposes and to satisfy customers' needs.

References

Akinsola, J. E., Ogunbanwo, A. S., Okesola, O. J., Odun-Ayo, I. J., Ayegbusi, F. D., & Adebiyi, A. A. (2020). Comparative analysis of software development life cycle models (SDLC). *Intelligent Algorithms in Software Engineering*, 310-322. https://doi.org/10.1007/978-3-030-51965-0_27

Bloch, M., Blumberg, S., & Laartz, J. (2012, October 1). *Delivering large-scale IT projects on time, on budget, and on value*. McKinsey & Company. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/delivering-large-scale-it-projects-on-time-on-budget-and-on-value

Cai, X., Niu, Y., Geng, S., Zhang, J., Cui, Z., Li, J., & Chen, J. (2019). An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search. *Concurrency and Computation: Practice and Experience*, *32*(5). https://doi.org/10.1002/cpe.5478

Chakravarty, K., & Singh, J. (2021). A study of quality metrics in Agile software development. *Machine Learning and Information Processing*, 255-266. https://doi.org/10.1007/978-981-33-4859-2_26

Ciric, D., Lalic, B., Gracanin, D., Tasic, N., Delic, M., & Medic, N. (2019). Agile vs. traditional approach in project management: Strategies, challenges and reasons to introduce Agile. *Procedia Manufacturing*, *39*, 1407-1414. https://doi.org/10.1016/j.promfg.2020.01.314

Dingsøyr, T., Moe, N. B., Fægri, T. E., & Seim, E. A. (2017). Exploring software development at the very large-scale: A revelatory case study and research agenda for

Agile method adaptation. *Empirical Software Engineering*, *23*(1), 490-520. https://doi.org/10.1007/s10664-017-9524-2

Dlamini, M., & Cumberlege, R. (2021). The impact of cost overruns and delays in the construction business. *IOP Conference Series: Earth and Environmental Science*, *654*(1), 012029. https://doi.org/10.1088/1755-1315/654/1/012029

Edison, H., Wang, X., & Conboy, K. (2022). Comparing methods for large-scale Agile software development: A systematic literature review. *IEEE Transactions on Software Engineering*, *48*(8), 2709-2731. https://doi.org/10.1109/tse.2021.3069039

Ehrlinger, L., & Wöß, W. (2022). A survey of data quality measurement and monitoring tools. *Frontiers in Big Data*, *5*. https://doi.org/10.3389/fdata.2022.850611

Fitzgerald, B., & Stol, K. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176-189. https://doi.org/10.1016/j.jss.2015.06.063

Făgărăşan, C., Cristea, C., Cristea, M., Popa, O., Mihele, C., & Pîslă, A. (2022). Key performance indicators used to measure the adherence to the iterative software delivery model and policies. *IOP Conference Series: Materials Science and Engineering*, *1256*(1), 012038. https://doi.org/10.1088/1757-899x/1256/1/012038

Gao, C., Luo, W., Wang, J., & Zhang, Y. (2022). Software quality evaluation model based on multiple linear regression and fuzzy comprehensive evaluation method. *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*. https://doi.org/10.1109/dsa56465.2022.00058

Garousi, V., Rainer, A., Lauvås, P., & Arcuri, A. (2020). Software-testing education: A systematic literature mapping. *Journal of Systems and Software*, *165*, 110570. https://doi.org/10.1016/j.jss.2020.110570

Ghosh, S., Rana, A., & Kansal, V. (2020). A benchmarking framework using nonlinear manifold detection techniques for software defect prediction. *International Journal of Computational Science and Engineering*, *21*(4), 593. https://doi.org/10.1504/ijcse.2020.106871

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

Gordieiev, O., & Kharchenko, V. (2020). Profile-oriented assessment of software requirements quality: Models, metrics, case study. *International Journal of Computing*, 656-665. https://doi.org/10.47839/ijc.19.4.2001

Gruzdo, I., Kyrychenko, I., & Tereshchenko, G. (2019). Overview and analysis of methods for evaluating software at the design stage. *Радиоэлектроника и информатика*, *0*(2(85)). https://doi.org/10.30837/1563-0064.2(85).2019.184736

Idrees, S., & Shafiq, M. T. (2021). Factors for time and cost overrun in public projects. *Journal of Engineering, Project, and Production Management*. https://doi.org/10.2478/jeppm-2021-0023

Idri, A., Moumane, K., & Abran, A. (2013). On the use of software quality standard ISO/IEC9126 in mobile environments. *2013 20th Asia-Pacific Software Engineering Conference (APSEC)*. https://doi.org/10.1109/apsec.2013.12

Jain, S., Yadav, V., & Singh, R. (2014). A simplified formulation of predictive object points (POP) sizing metric for OO measurement. *2014 IEEE International Advance Computing Conference (IACC)*. https://doi.org/10.1109/iadcc.2014.6779526

Kassie, N. B., & Singh, J. (2020). A study on software quality factors and metrics to enhance software quality assurance. *International Journal of Productivity and Quality Management*, *29*(1), 24. https://doi.org/10.1504/ijpqm.2020.104547

Kerzner, H. (2017). *Project management: A systems approach to planning, scheduling, and controlling*. John Wiley & Sons.

Kumar, L., Satapathy, S. M., & Murthy, L. B. (2019). Method level Refactoring prediction on five open source Java projects using machine learning techniques. *Proceedings of the 12th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*. https://doi.org/10.1145/3299771.3299777

Lakra, K., & Chug, A. (2021). Application of metaheuristic techniques in software quality prediction: A systematic mapping study. *International Journal of Intelligent Engineering Informatics*, *9*(4), 355. https://doi.org/10.1504/ijiei.2021.120322

Lalic, B., Gracanin, D., Tasic, N., & Simeunović, N. (2022). *Proceedings on 18th International Conference on industrial systems – IS'20: Industrial innovation in digital age*. Springer Nature.

Lee, M. (2014). Software quality factors and software quality metrics to enhance software quality assurance. *British Journal of Applied Science & Technology*, *4*(21), 3069-3095. https://doi.org/10.9734/bjast/2014/10548

Lundell, B., Gamalielsson, J., & Katz, A. (2021). On challenges for implementing ISO standards in software. *Research Anthology on Usage and Development of Open Source Software*, 63-95. https://doi.org/10.4018/978-1-7998-9158-1.ch004

Lánczky, A., & Győrffy, B. (2021). Web-based survival analysis tool tailored for medical research (KMplot): Development and implementation. *Journal of Medical Internet Research*, *23*(7), e27633. https://doi.org/10.2196/27633

Ma, L., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., Liu, Y., Zhao, J., & Wang, Y. (2018). DeepGauge: Multi-granularity testing criteria for deep learning systems. *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. https://doi.org/10.1145/3238147.3238202

Makhoul, N. (2022). Review of data quality indicators and metrics, and suggestions for indicators and metrics for structural health monitoring. *Advances in Bridge Engineering*, *3*(1). https://doi.org/10.1186/s43251-022-00068-9

Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A. M., & Wagner, S. (2022). Software engineering for AI-based systems: A survey. *ACM Transactions on Software Engineering and Methodology*, *31*(2), 1-59. https://doi.org/10.1145/3487043

Mashiloane, R. E., & Jokonya, O. (2018). Investigating the challenges of project governance processes of IT projects. *Procedia Computer Science*, *138*, 875-882. https://doi.org/10.1016/j.procs.2018.10.114

Masmali, O., & Badreddin, O. (2020). Code complexity metrics derived from software design: A framework and theoretical evaluation. *Advances in Intelligent Systems and Computing*, 326-340. https://doi.org/10.1007/978-3-030-63092-8_22

Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. *Computer Science Review*, *38*, 100308. https://doi.org/10.1016/j.cosrev.2020.100308

Mittal, P., & Mehta, P. (2020). Optimization of software development process by Plugin integration with Jira – A project management tool in Devops. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3564436

Olorunshola, O. E., & Ogwueleka, F. N. (2021). Review of system development life cycle (SDLC) models for effective application delivery. *Information and Communication Technology for Competitive Strategies (ICTCS 2020)*, 281-289. https://doi.org/10.1007/978-981-16-0739-4_28

Oreški, D. (2022). Framework of intelligent system for machine learning algorithm selection in social sciences. *Journal of Software*, 21-28. https://doi.org/10.17706/jsw.17.1.21-28

P. Miguel, J., Mauricio, D., & Rodríguez, G. (2014). A review of software quality models for the evaluation of software products. *International Journal of Software Engineering & Applications*, *5*(6), 31-53. https://doi.org/10.5121/ijsea.2014.5603

Paraskevopoulou, C., Dallavalle, M., Konstantis, S., Spyridis, P., & Benardos, A. (2022). Assessing the failure potential of tunnels and the impacts on cost overruns and project delays. *Tunnelling and Underground Space Technology*, *123*, 104443. https://doi.org/10.1016/j.tust.2022.104443

Pargaonkar, S. (2023). Synergizing requirements engineering and quality assurance: A comprehensive exploration in software quality engineering. https://doi.org/10.31219/osf.io/g35se

Pargaonkar, S. (2024). The significance of quality metrics. *A Guide to Software Quality Engineering*, 187-198. https://doi.org/10.1201/9781032702049-16

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, *64*, 1-18. https://doi.org/10.1016/j.infsof.2015.03.007

Rathor, K., Kaur, J., Nayak, U. A., Kaliappan, S., Maranan, R., & Kalpana, V. (2023). Technological evaluation and software bug training using genetic algorithm and time convolution neural network (GA-TCN). *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*. https://doi.org/10.1109/icaiss58487.2023.10250760

Serban, A., Van der Blom, K., Hoos, H., & Visser, J. (2020). Adoption and effects of software engineering best practices in machine learning. *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. https://doi.org/10.1145/3382494.3410681

Sharma, T., & Spinellis, D. (2018). A survey on software smells. *Journal of Systems and Software*, *138*, 158-173. https://doi.org/10.1016/j.jss.2017.12.034

Simushi, S., & Wium, J. (2020). Time and cost overruns on large projects: Understanding the root cause. *Journal of Construction in Developing Countries*, *25*(1), 129-146. https://doi.org/10.21315/jcdc2020.25.1.7

Soni, M. (2020). *Hands-on Azure DevOps: CICD implementation for mobile, hybrid, and web applications using Azure DevOps and Microsoft Azure*. BPB Publications.

Ulan, M., Löwe, W., Ericsson, M., & Wingkvist, A. (2021). Weighted software metrics aggregation and its application to defect prediction. *Empirical Software Engineering*, *26*(5). https://doi.org/10.1007/s10664-021-09984-2

Ur, A., Yaseen, M., & Ali, Z. (2019). Identification of practices for proper implementation of requirements in global software development: A systematic literature review protocol. *International Journal of Computer Applications*, *177*(13), 53-58. https://doi.org/10.5120/ijca2019919533

Vallon, R., Da Silva Estácio, B. J., Prikladnicki, R., & Grechenig, T. (2018). Systematic literature review on Agile practices in global software development. *Information and Software Technology*, *96*, 161-180. https://doi.org/10.1016/j.infsof.2017.12.004

Venters, C. C., Capilla, R., Betz, S., Penzenstadler, B., Crick, T., Crouch, S., Nakagawa, E. Y., Becker, C., & Carrillo, C. (2018). Software sustainability: Research and practice from a software architecture viewpoint. *Journal of Systems and Software*, *138*, 174-188. https://doi.org/10.1016/j.jss.2017.12.026

Zarour, M. (2020). A rigorous user needs experience evaluation method based on software quality standards. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *18*(5), 2787. https://doi.org/10.12928/telkomnika.v18i5.16061

Zuse, H. (2019). *Software complexity: Measures and methods*. Walter de Gruyter GmbH & Co KG.