Examining The Role of Software Maintenance in Ensuring Software Quality

Nashali Perera[1], D. I. De Silva[1], Chanukya Serasinghe[1], M. P. Gunathilake[1], Sahan Perera[1], and Devindu Samarasinghe[1]

[1]Sri Lanka Institute of Information Technology

May 1, 2023

Abstract

This study investigates the relationship between software maintenance practices and software quality as well as the impact of maintenance budgets and schedules on software development. Corrective, adaptive, perfective, and preventive software maintenance are the four basic categories identified in the article, which also emphasizes the significance of efficient maintenance in raising software quality. The report also highlights the importance of good project management, ongoing testing, and documentation to guarantee that maintenance processes are conducted properly and quickly. According to the study's conclusions, initiative-taking maintenance can save maintenance expenditures over time while also enhancing software quality. Reduced downtime, better software performance, and higher customer satisfaction are all benefits of effective maintenance, which may assist in finding and addressing problems before they grow into larger ones. The report also emphasizes the necessity for continual cooperation between development and maintenance teams as well as the difficulties in juggling maintenance expenses and schedules with software development time limits. In the end, the study emphasizes the significance of maintenance in guaranteeing software quality and the demand for continued investment in maintenance procedures to enhance software performance and reduce costs over time.

Keywords—Software quality, Software maintenance, Software development

Introduction

Today, a lot of businesses rely on software to manage their daily operations, therefore software development has turned into a necessity. The importance of software quality is growing since it directly influences how well it performs the tasks for which it was designed. Software quality is determined by several factors, including software maintenance practices, and refers to how effectively a piece of software meets the needs and expectations of its users.

The process of changing and updating the software after it has been launched is known as software maintenance, and it is a crucial step in the software development life cycle. For software to continue to be successful and practical over time, this is crucial. Bug patches, changes in user needs, technological advancements, and other reasons all contribute to the need for software maintenance.

Despite the significance of software maintenance, prior research has mostly concentrated on evaluating the expertise of the maintenance team, the software development technique employed, and the kind of software being maintained. There has been a glaring void in the literature when it comes to examining the connection between software maintenance procedures and software quality.

This study will look at how software maintenance affects software quality to close this gap. There are four main methods of software maintenance that can guarantee the most noteworthy level of software quality. They are,

1. Corrective Maintenance
2. Adaptive maintenance
3. Perfective Maintenance
4. Preventive Maintenance

Corrective maintenance points to settle defects, errors, and bugs within the program that have been identified by clients or through testing. Corrective maintenance ensures that the software package is free of issues and works as aimed. Adaptive maintenance includes altering the computer program to oblige changes within the environment in which it works. For illustration, changes in equipment, working frameworks, or lawful necessities may require alterations to the software package. Adaptive maintenance guarantees that the software is up-to-date and compatible with its environment. Perfective maintenance points to move forward the quality of the package by improving its usefulness or execution. It includes adjusting the software to include modern highlights or progress existing ones, to enhance user satisfaction and expand efficiency. Preventive Maintenance includes taking proactive measures to avoid issues from happening within the package. This may incorporate standard framework reinforcements, execution observing, and security overhauls. Preventive maintenance guarantees that the program framework is solid, steady, and secure.

The purpose of the study is to discover how different software maintenance techniques affect software quality. It specifically seeks to ascertain whether there is a substantial relationship between the frequency of maintenance tasks and the caliber of the software. The study also attempts to investigate how various maintenance practices, such as corrective, adaptive, and perfective maintenance, affect software quality.

Software maintenance is essential for ensuring software quality, yet little is known about how maintenance procedures affect program quality. Despite the significance of software maintenance, there is a lack of knowledge regarding the difficulties, ideal procedures, and trade-offs involved. As a result, research is required to determine how software maintenance contributes to software quality assurance and to propose efficient management solutions for software maintenance that would maximize software quality. To increase software quality, this research intends to examine the connection between software maintenance and quality and to pinpoint the best strategies for handling software maintenance.

This study has significance because it can assist teams working on software development in identifying and putting into practice maintenance strategies that will ultimately result in software of higher caliber. We can better comprehend the elements that go into creating high-quality software and devise tactics to streamline the maintenance procedure by looking at the role that maintenance plays in software quality. Both theoretical and practical ramifications for the larger software engineering community as well as practical ramifications for software development firms may result from this research.

This study has the following precise goals:

- Examine the connection between software quality and maintenance procedures.
- Determine the best software maintenance procedures for preserving software quality.
- Examine the effect of software maintenance on the price and duration of software development.

The following are the study's hypotheses:

Good software maintenance procedures have a favorable effect on software quality.

Time and money spent on software development are cut down due to efficient maintenance procedures which in turn affect software quality.

The following research queries will be addressed in this study:

- Which methods of software maintenance ensure the highest level of software quality?

- How do software maintenance costs and schedules affect software development so that it will affect the software quality?
- What connection exists between software maintenance procedures and software quality?

Before beginning this research study, a thorough review of the body of knowledge on software maintenance practices and software quality will be conducted. The publication will go into great depth about the research design, data collection techniques, participant selection, and data analysis procedures. In the paper, the results of the data analysis will be given together with a discussion on how they could influence the practices used in software development. Following a discussion of the study's flaws, the report will offer suggestions for additional research.

literature review

Software maintenance is an ongoing process that takes place after software development is completed. Its goal is to keep the software functioning as expected and in line with users' evolving needs. The importance of software maintenance is often underestimated, and it is often not given the attention it deserves. Software maintenance accounts for more than 50% of the overall cost of software development, according to Lehman and Belady (1985). This demonstrates how crucial software maintenance is to maintaining software quality. In this literature review, we will examine the role of software maintenance in ensuring software quality.

Software Maintentance Procedures

The case study conducted by Hassan and Holt(2003) in order to test the connection between software quality and software maintenance of a large software system in a financial institution was done by gathering both subjective and objective software quality measures. They then analyzed software system maintenance data for two years to establish the association between software quality and software maintenance. The study includes 23 empirical investigations that used various software quality measures, such as defect density, reliability, maintainability, and customer satisfaction .

Similarly, T. Mens and S. Demeyer (2008), proposed a reuse-based strategy to software maintenance and evolution instead of making random modifications to the code, they advised identifying common functionalities and developing reusable components that could be used to modify the source. This method entails identifying and abstracting common functionality in the software system, as well as developing reusable software components. It underlines the significance of reusing code in maintenance because it can save the time and effort necessary for bug repair and testing. A huge Java-based software system utilized by a Belgian bank was the subject of the case study. In the codebase, authors recognized specific common functionalities and constructed reusable components to implement these functionalities.

Software Maintenance Costs And Schedules

It is crucial to balance the need for maintenance with the need for new development to ensure that software remains stable and up-to-date while meeting the changing needs of its users.

Development cost: Software maintenance expenses can have an impact on a project's development expenditures. The ability of the development team to allocate resources to the development of new functions or other projects may be impacted if the maintenance expenses are significant. This can cause software development to take longer and cost more money to complete. As depicted by Fig. 1, the cost of fixing bugs in a typical software development cycle. The cost of fixing bugs in the maintenance stage looks to be substantially higher than the cost of fixing issues in all other stages combined, as seen in the figure. As a result, proper software development requires solid design and ensuring that the software is extensively evaluated to remove problems as much as possible before the product is given to the market or customers so that the costs of maintaining the software are reduced.

Hosted file

`image1.emf` available at https://authorea.com/users/613328/articles/640784-examining-the-role-of-software-maintenance-in-ensuring-software-quality

Schedule: The software maintenance schedule can have an impact on software development and software quality. Overly frequent maintenance may cause development delays or possibly program instability. On the other side, if maintenance is performed infrequently, bugs and other issues may build up, increasing the probability of software failure.

Izquierdo-Cortazar et al. (2011) investigated the effect of corrective maintenance on software development. The process of finding, analyzing, and repairing problems or faults in software programs is referred to as corrective maintenance. It emphasized the importance of early detection and rectification of issues in order to keep the project on schedule. Bugs that are discovered early can be corrected before they cause major delays. On the other side, if defects go unnoticed for an extended period of time, they might pose substantial issues that cause significant delays in the project schedule. The researchers employed a case study approach to perform the study and analyzed data from a software development project that required corrective maintenance. They analyzed project data and ran computer simulations to examine the influence of various corrective maintenance procedures on schedules for the project .

Similarly, Edward E. Ogheneovo(2016) explored the relationship between software complexity and maintenance costs. Specifically, the study aims to investigate whether there is a positive relationship between software complexity and maintenance costs, and to identify the factors that contribute to this relationship. The study focuses on software systems developed using the Object-Oriented Programming (OOP) paradigm, which is widely used in modern software development.

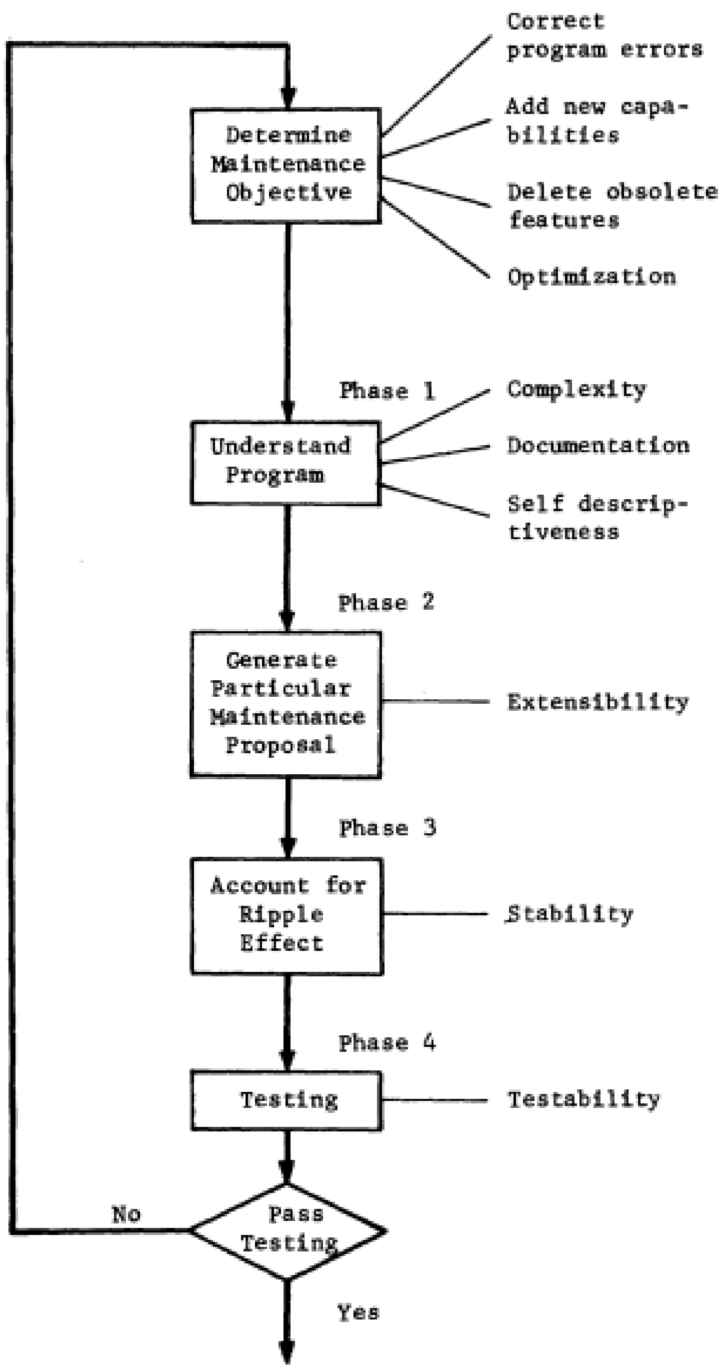
Ogheneovo conducted the study by analyzing data from five software systems built by a single organization. He measured the complexity of software systems using many software complexity metrics, such as the number of classes, the number of methods, and the depth of inheritance. He also gathered information on the maintenance costs for each software system, such as the number of repair requests and the time and effort required to complete each one. The research emphasizes the significance of smart software design in reducing software complexity and lowering maintenance costs. Modularization and encapsulation are two design principles that software development organizations can use to reduce reliance among software components and simplify program development and maintenance .

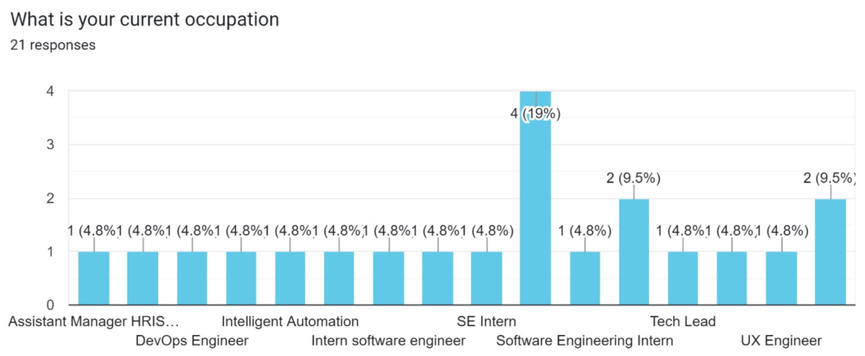
The authors define software complexity as the difficulty of understanding and modifying a software system. They claim that complicated software is more expensive to maintain since it demands more effort and time to comprehend, modify, and test. The authors studied a sample of COBOL programs from three distinct organizations. To assess the program's complexity, they utilized a statistic known as cyclomatic complexity. They also gathered information on the program's maintenance expenditures, such as the time spent on maintenance, the number of lines of code updated, and the number of errors made during maintenance. Figure 2 depicts the costs of fixing bugs in a typical software life cycle. The cost of fixing bugs in the maintenance stage looks to be substantially higher than the cost of fixing issues in all other stages combined, as seen in the figure. As a result, there is a need to properly build software by having solid design and ensuring that the software is fully tested to remove flaws as much as possible before the product is published to the market or customers in order to lower the expenses of maintaining the software.

Methodology

In this study, it has reviewed the literature to examine the relationship between software maintenance practices and software quality as well as the effects of software maintenance expenditures and timelines on software development. It has looked through a variety of sources, including online databases like IEEE Xplore, ResearchGate, and Google Scholar, to find pertinent research papers, books, and conference proceedings on software maintenance and software quality. To focus the search, it has added terms like software development, software quality, and maintenance costs. Following a preliminary inspection of the search results, the chosen papers are further assessed considering their relevance to our study goals. Studies that were authored in English and released between 1990 and 2021 were selected. Studies unrelated to software maintenance and software quality or not pertinent to the study goals were excluded.

Moreover, the chosen papers are analyzed to determine major themes and conclusions about software maintenance practices, software quality, and the effect of software maintenance budgets and schedules on software development. To synthesize the data from the chosen studies and make judgments regarding the study issues, a qualitative methodology is adopted. To approve the viability of the investigation results, a few case studies are investigated and examined included within the chosen to inquire about articles. This case is used considers demonstrating the key themes and discoveries of our investigation and to supply real-world illustrations of software support methods and their effect on its quality and development.


Determine Maintenance Objective
Correct program errors
Add new capabilities
Delete obsolete features
Optimization
Phase 1
Understand Program
Complexity
Documentation
Self descriptiveness
Phase 2
Generate Particular Maintenance Proposal
Extensibility
Phase 3
Account for Ripple Effect
Stability
Phase 4
Testing
Testability
Pass Testing
No
Yes



One of the approaches used in this study was to conduct a survey to gather information from experts in the field of software engineering about their experience with software maintenance best practices to Ensure software quality at the highest level. The survey was conducted online, and participants were recruited through professional networks and social media platforms as shown in Fig. 2. Survey data was collected and analyzed using statistical software to identify trends and patterns in the responses. The survey results are then used to draw conclusions about the most effective software maintenance methods to ensure software quality. This survey was conducted with the aim of collecting valid and reliable data from experts in the field of software engineering to provide insight into the role of software maintenance in ensuring quality.

To focus the search, it has added terms like software development, software quality, and maintenance costs. Following a preliminary inspection of the search results, the chosen papers are further assessed considering their relevance to our study goals. Studies that were authored in English and released between 1990 and 2021 were selected. Studies unrelated to software maintenance and software quality or not pertinent to the study goals were excluded.

Moreover, the chosen papers are analyzed to determine major themes and conclusions about software maintenance practices, software quality, and the effect of software maintenance budgets and schedules on software development. To synthesize the data from the chosen studies and make judgments regarding the study issues, a qualitative methodology is adopted. To approve the viability of the investigation results, a few case studies are investigated and examined included within the chosen to inquire about articles. this case is used considers demonstrating the key themes and discoveries of our investigation and to supply real-world illustrations of software support methods and their effect on its quality and development.

Results

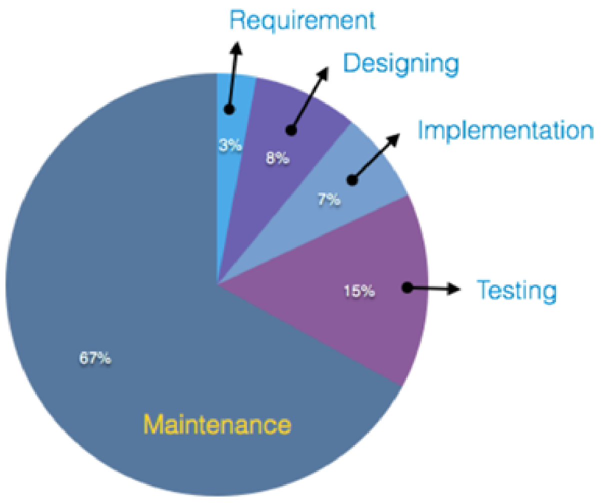
The study assessed how various software maintenance techniques, such as bug patching and code refactoring, affect the caliber of the software. The study intends to shed light on efficient software maintenance techniques that can be applied to improve software quality as depicted in Fig. 3.

In order to evaluate the hypothesis that good software maintenance methods improve software quality, Hassan and Holt conducted a study in 2003. Using data from twelve large software systems, the study investigated the impact of software maintenance on software quality. The strength of this relationship was discovered to vary depending on the precise quality measure utilized in each investigation. Those whose studies utilized fault density as a quality indicator, for example, found a stronger association between maintenance and quality than those that used other measures. Higher levels

of software quality were shown to be connected with lower maintenance costs and fewer faults, as well as increased maintainability and customer satisfaction, according to the study .

In the second study, Mens, and S. Demeyer (2008) discovered that the reuse-based approach was effective in reducing maintenance effort and time to resolution while also improving software quality, which examined the impact of software maintenance on software quality using data from ten open-source software systems .

The approach specifically enables developers to reuse existing software artifacts and components, reducing the amount of time and effort required to create new ones from scratch. Furthermore, the reuse-based strategy helped in the improvement of software quality by encouraging the usage of proven, tested components that had previously been utilized in other parts of the software system. Furthermore, the authors stated that the reuse-based strategy was well-received by the developers, showing that it was viewed as an effective and efficient way to manage maintenance and evolution responsibilities.

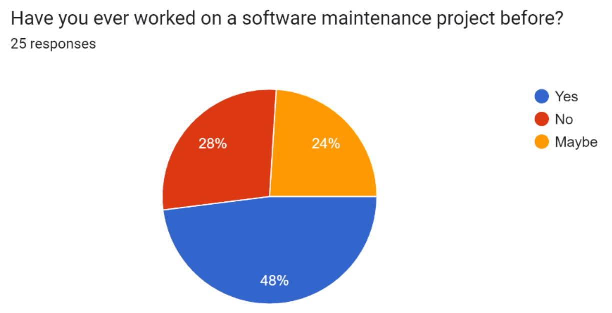


The second study topic investigates the financial and time implications of software maintenance on software development. Fig. 4, shows how the cost is distributed in the SDLC. The purpose of this was to find out how the total software development process is impacted by software maintenance costs and timelines.

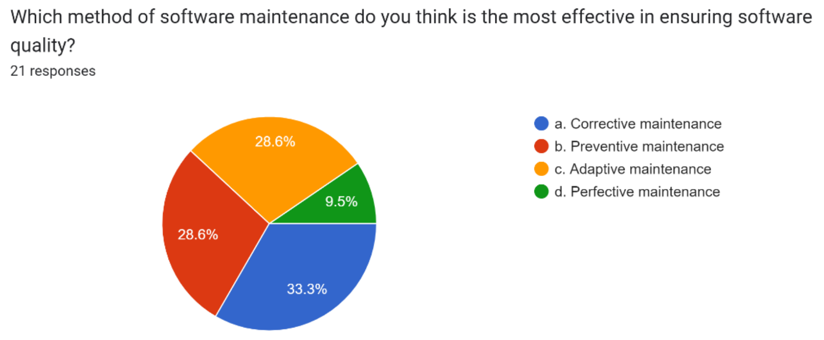
Correcting, modifying, enhancing, and perfecting an already-in-use piece of software are all included in software maintenance. The user has the choice of whether to rectify, modify, enhance, or perfect the software, excepting major faults and necessary adjustments. The upkeep task is not necessary. It is frequently possible to continue using the software in its current state, even in cases of non-critical problems. Time and money are the two factors that determine whether to complete the maintenance operation or not. When improvements and perfection are involved, time and money are even more crucial factors. Considering the current economic climate, everything comes down to weighing the costs, benefits, and priorities.

Regarding the hypothesis that efficient maintenance procedures reduce time and money spent on software development, which in turn affects software quality, Edward E. Ogheneovo (2016) discovered a positive relationship between software complexity and maintenance costs. Ogheneovo discovered that as the complexity of the software system grows, so does the number of maintenance requests and the time and work necessary to accomplish each request. The study also discovered numerous elements that influence this relationship, such as the size and complexity of the program code, the number of features and functionalities, and the degree of interdependence among software components . In addition, Rajiv D. Banker et al. (1990) found that programs with higher cyclomatic complexity ratings were associated with higher maintenance costs. They also discovered that programs with greater maintenance expenses have a higher rate of error during maintenance. The authors examine the significance of their findings for software development and maintenance, arguing that developers should work to reduce software complexity in order to reduce maintenance costs .

With regards to address one of the research queries regarding the best methods of software maintenance that ensures the highest level of software quality, a survey was conducted. The aim of the survey was to gather information from the professionals' opinions and experiences based on their experience with projects with regard to software maintenance techniques. Almost half of those polled (47.6%) have previously worked on a software maintenance project, demonstrating that software maintenance is a popular activity in the software business.

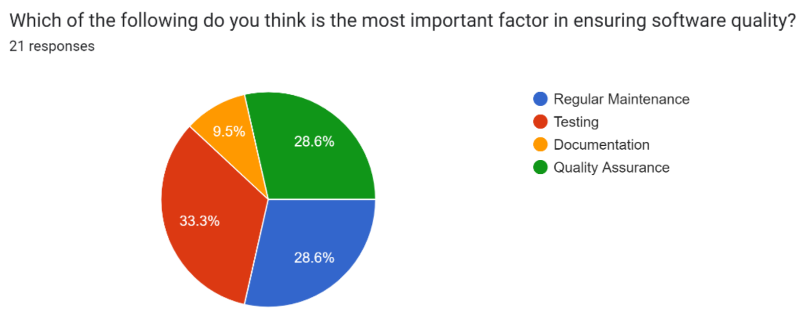


Corrective maintenance (33.3%), preventive maintenance (28.6%), and adaptive maintenance (28.6%) were chosen as the most effective methods of software maintenance by participants [Fig. 7]. The majority of participants (76.2%) cited problems with software quality owing to a lack of effective maintenance and this is depicted by Fig. 5, emphasizing the necessity of regular maintenance in preventing problems in the first place. The panelists divided the most essential components in guaranteeing software quality as regular maintenance (28.6%) and testing (33.6%) [Fig.6]. Corrective maintenance was judged to be the most time-consuming (42.9%) of software maintenance strategies [Fig. 9]. In terms of cost-effectiveness, adaptive maintenance (33.6%) was considered to be the most cost-effective, followed by preventative maintenance (28.6%) [Fig. 8].

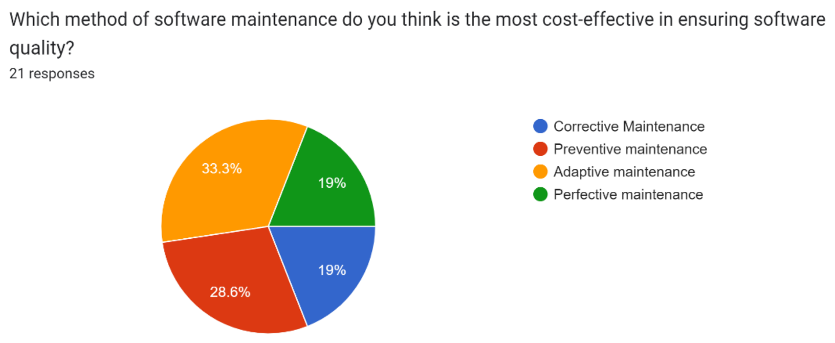


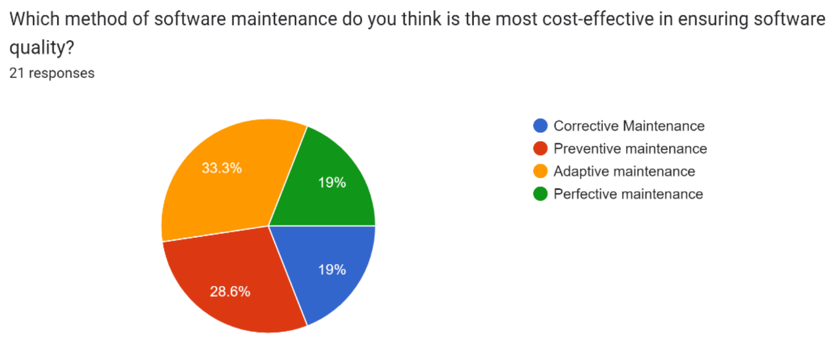
Overall, the survey findings highlight the importance of software maintenance in guaranteeing software quality and suggest that a variety of strategies may be required to achieve this aim.

Discussion



The research findings demonstrated how crucial software maintenance is for maintaining the effectiveness, dependability, and quality of software systems. The research, in particular, shows that effective software maintenance practices can enhance software quality, save maintenance costs, and raise customer happiness. The research also emphasized some of the major difficulties in managing dependencies, ensuring security, maintaining performance, and controlling costs in relation to software maintenance.





A significant finding from the secondary research studies was that software complexity and maintenance costs have a positive correlation. This argues that software developers should work to simplify their products in order to cut maintenance expenses and raise software quality. By using this approach, developers can reuse pre-existing software artifacts and components, saving time and effort compared to creating new ones from scratch. Customer that increased software quality was linked to cheaper maintenance costs, fewer faults, improved maintainability, and higher customer satisfaction was another significant discovery. This emphasizes the value of spending money on software maintenance to keep software systems dependable and effective over time. Developers should carefully examine the measurements to employ when analyzing Developers should carefully examine the measurements to employ when analyzing the influence of maintenance on software quality, according to the studies, which also revealed that different measures of software quality may have differing levels of association with maintenance.

With regards to the survey conducted in order to collect primary data, the analysis showed corrective, preventive, and adaptive maintenance as the most effective hence implying that, depending on the scenario, multiple forms of maintenance are required, and that a variety of approaches may be required to assure software quality. In addition to this, regular maintenance and testing were considered extremely critical to guaranteeing software quality since regular maintenance may prevent problems from happening and testing can discover problems that need to be fixed.

Adaptive maintenance followed by preventative maintenance being the most cost-effective maintenance techniques suggests that making modifications or upgrades to the software to adapt to changes in the environment or avoid future issues may be more cost-effective in the long term than repairing problems after they arise. To elaborate more on this, preventive maintenance is making alterations or updates to the program to maintain a strategic distance from future faults or issues. Developers can avoid expensive and time-consuming challenges afterward within the software development process by proactively tending to be planned concerns. This has influence to ensure that the program fits the requirements and details and that it runs easily and without mistakes. Preventive maintenance moreover ensures that the program is up to date and able of adjusting to changes within the working framework or equipment.to adjust to changes within the working framework or equipment. Adaptive maintenance, on the other hand, involves upgrading the program to suit changes within the environment, such as working frameworks or equipment adjustments. This can be noteworthy since, as the technological landscape advances, software must adjust to stay pertinent and effective. Software may remain useable, utilitarian, and high performing by responding to changes within the environment. Adjusting computer programs to fit unused necessities or guidelines is another case of adaptive maintenance.

Preventive and adaptive maintenance must work in pairs to guarantee software quality. They help with the anticipation of issues and ensure that the program proceeds to perform legitimately in changing settings. Without this sort of support, the program can before long end up old and out of date, coming about resulting in expanded costs, lower effectiveness, and decreased execution. Besides, by diminishing the prerequisite for remedial support, both shapes of maintenance can serve to lower the fetch of software advancement. Designers can avoid more exorbitant and time-consuming corrective maintenance endeavors down the line by tending to conceivable issues early on. This will inevitably assist in lower add up to software development costs and increase the program's return on the venture. Increase the program's return on the venture. Finally, preventative and adaptive maintenance is basic for moving forward program quality. They contribute to the software's long-term ease of use, usefulness, and execution, indeed when the environment and requirements change. Designers may construct a program that is more steady, efficient, and cost-effective in the long term by prioritizing certain shapes of support.

Overall, the findings of the survey and research studies indicate that software maintenance is an important activity for ensuring the quality and reliability of software systems throughout time. Regular maintenance and testing are critical, and businesses may wish to consider spending more on preventive and adaptive maintenance in the future to lessen the need for corrective maintenance. To ensure that software systems stay dependable and performant over time, developers should seek to reduce program complexity, employ efficient maintenance techniques, and invest in continuous maintenance activities.

Limitations and Further Research

One restriction of the research studies presented in the literature review is that they may not be applicable to all types of software systems. The research mostly focused on large, complicated software systems and open-source software, which may not be representative of other software development situations. Furthermore, the questionnaire survey may have a sample bias because the sample of participants included undergraduate students who may not be representative of the population being studied because they are still unclear about the topic addressed, resulting in limited generalizability of the results. Response bias may also exist, in which participants may not respond accurately or choose the answer they believe the researcher wants to hear, resulting in erroneous results.

In terms of study limitations, one restriction of the studies presented in the literature review is that they may not be applicable to all types of software systems. The research mostly focused on large, complicated software systems and open-source software, which may not be representative of other software development situations. Furthermore, the questionnaire survey may have a sample bias because the sample of participants included undergraduate students who may not be representative of the population being studied because they are still unclear about the topic addressed, resulting in limited generalizability of the results. Response bias may also exist, in which participants may not respond accurately or choose the answer they believe the researcher wants to hear, resulting in erroneous results.

Researchers in the future could build on prior studies' limitations by investigating software maintenance strategies and their impact on software quality in a larger range of software development situations. Smaller and less sophisticated software systems, proprietary software, and a broader range of quality criteria could all fall under this category. Future study could also look into new and innovative software maintenance approaches that could improve software quality while reducing maintenance costs and time. Subsequently, it might look into the impact of upcoming technologies like artificial intelligence and machine learning on software maintenance and how they relate to software quality.

conclusion

Software maintenance is critical to guaranteeing the software quality throughout the program's lifetime. Maintenance efforts, which entail altering, repairing, or improving software systems to meet changing needs, address defects, and improve performance, can have a direct influence on software quality. Effective maintenance techniques keep technical debt at bay and guarantee that software systems continue to provide value overall.

Software maintenance costs and schedules can have a significant impact on software quality. Poorly managed maintenance activities can lead to delays, cost overruns, and substandard results. If not planned and executed properly, maintenance activities can introduce new bugs, create new dependencies, and have unpredictable results. This can lead to lower-quality software as systems become more complex, more difficult to maintain, and less stable.

Effective maintenance methods, on the other hand, may improve software quality by ensuring that software systems stay dependable, efficient, and up to date. Regular maintenance efforts like code reviews, testing, and refactoring can assist in detecting bugs early and preventing them from becoming severe issues. Furthermore, maintenance actions focused on enhancing software architecture and design can assist decrease technical debt and enhance system modularity, making the system easier to maintain, adapt, and scale over time. Software quality is intimately tied to software maintenance processes. Maintenance work assists in identifying parts of the software that require improvement and in keeping the program operational at all times. Maintenance methods are also critical to ensuring that software systems can adapt to changing user and stakeholder demands. Maintenance teams can discover foreseeable problems and prioritize bug fixes that enhance software quality by monitoring system performance and offering feedback.

Maintenance efforts must be carefully planned and conducted to achieve the greatest level of software quality. Maintenance teams should prioritize actions that have the greatest impact on software quality using data-driven techniques. They should also make maintenance efforts as visible and predictable as possible so that stakeholders understand what is going on and how it impacts them. Maintenance teams may develop confidence and ensure stakeholders remain engaged and involved in the software over time by incorporating stakeholders in maintenance activities and giving them timely information about changes to the program.

In conclusion, software maintenance is an important aspect of guaranteeing long-term software quality. Effective maintenance methods may help to avoid technological debt, spot problems early, and maintain software systems dependable, efficient, and up to date. Maintenance tasks that are poorly managed can impair software quality and make software systems difficult to maintain, alter, and scale. As a result, maintenance teams must carefully plan and execute maintenance tasks, prioritize activities using data, and include stakeholders to maintain transparency and involvement. Organizations may maintain high-quality software systems and offer value over time by adhering to software maintenance best practices. Effective maintenance methods may help to avoid technological debt, spot problems early, and maintain software systems dependable, efficient, and up to date. Maintenance tasks that are poorly managed can impair software quality and make software systems difficult to maintain, alter, and scale. As a result, maintenance teams must carefully plan and execute maintenance tasks, prioritize activities using data, and include stakeholders to maintain transparency and involvement. Organizations may maintain high-quality software systems and offer value over time by adhering to software maintenance best practices.