International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-12 Issue-2, July 2023

The Effectiveness of Code Reviews on Improving Software Quality: An Empirical Study

D. I. De Silva, W.A.C Pabasara, S.V Sangkavi, Wijerathne L.G.A.T.D, Wijesundara W.M.K.H, Reezan S.A

Abstract: This study intends to investigate how well code reviews contribute to higher software quality. A group of developers working on the study examine source code to find flaws, improve readability, and guarantee compliance with coding standards. The research on the effects of code review on defect discovery, defect prevention, and code maintainability is given together with a thorough overview of the literature on code review and software quality. This study has demonstrated that code review is an effective method for enhancing software quality. According to several studies and trials, code review has been shown to reduce flaws and improve code maintainability significantly. Code review helps increase client satisfaction by ensuring the product meets their needs. The goal of this study is to highlight the value of code review as a quality assurance technique in software development workflows. The study's findings offer valuable insights for software development teams, highlighting the benefits of code review in enhancing software quality and customer satisfaction. The results of this study can help software development teams incorporate code review into their workflows as a standard procedure, thereby improving software quality and reducing errors. In conclusion, this study demonstrates that code review is an effective method for enhancing software quality. In terms of fault detection and prevention, code maintainability, and customer satisfaction, the study underlines the benefits of code review. This study can influence software development teams to make code review a common practice by highlighting its advantages, which would increase product quality and client satisfaction.

Keywords: Code review, Software quality

Manuscript received on 06 May 2023 | Revised Manuscript received on 17 May 2023 | Manuscript Accepted on 15 July 2023 | Manuscript published on 30 July 2023.

*Correspondence Author(s)
Dr. D. I. De Silva, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka E-mail: dilshan.i@sliit.lk, ORCID ID: https://orcid.org/0000-0001-6821-488X
W. A. C Pabasara, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka E-mail: chamali.p@sliit.lk, ORCID ID: https://orcid.org/0009-0000-6431-8251
S. V Sangkavi*, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka E-mail: vithyasangkavi@gmail.com, ORCID ID: https://orcid.org/0009-0002-6350-6714
Wijerathne L.G.A.T.D, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka E-mail: thilakna@gmail.com, ORCID ID: https://orcid.org/0009-0001-7635-7431
Wijesundara W.M.K.H, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka E-mail: kaushanihw@gmail.com, ORCID ID: https://orcid.org/0009-0007-1133-9842
Reezan S.A., Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, Sri Lanka E-mail: imreezan@gmail.com, ORCID ID: https://orcid.org/0009-0003-5104-2059

I. INTRODUCTION

A. Background

Code review is a widely used method for identifying errors and enhancing software quality during the development process. A code review is a procedure where a piece of code is examined by someone other than the author(s) [1]. Before it is included in the main codebase, software code must be thoroughly reviewed for errors, bugs, and other potential issues. The purpose of code review is to raise the general level of the software's quality, lower the number of errors, and boost its dependability. Code review can be conducted in two ways: face-to-face review or online review, such as via Google Meet, email, and other methods.

Every sophisticated software development project requires a code review process that evaluates code submitted by developers. One of the best QA procedures for software projects is code review, which is seen as being quite expensive in terms of time and effort but offers significant value in terms of spotting errors in code modifications before they are committed to the project's code base [2].

According to a survey conducted by our team, 75% of the experts frequently conduct code reviews, while 25% conduct them rarely. Code reviews can be carried out either manually or automatically using tools. Developers or team members review the code manually and offer feedback based on their knowledge and experience. Software tools are used in automated code review to examine the code and find any potential problems. According to [3] website 60% of Developers are using automated tools; 49% are using them at least weekly.

Numerous studies have been conducted to investigate the effectiveness of code review in improving software quality. These studies have examined various aspects of code review, including its impact on maintainability, developer productivity, and problem identification. As stated in a blog, high quality, onboarding a new employee, knowledge exchange, increased consistency, and time saving are some advantages of code review [4].

Creating a robust code review procedure lays the groundwork for ongoing development and stops unstable code from being released to users. To enhance code quality and ensure that another team member reviews every piece of code, code reviews should be integrated into the software development team's workflow.

Another crucial step in transferring knowledge throughout a company is the code review procedure. In addition to these factors, 76% of developers who participated in the 2022 *Global DevSecOps* Survey indicated code reviews are "very valuable" [5].

Despite the potential advantages of code review, there are also some drawbacks to this process. For instance, code review can require a significant amount of time and effort from developers. The size of the codebase, the complexity of the code, and the reviewers' level of experience are other variables that may affect how effective code review is [6].

Given these factors, additional research is required to examine the effectiveness of code review in various situations and to determine how to best utilise this technique across several quality criteria, including maintainability, testability, dependability, security, usability, correctness, efficiency, and scalability. The goal of the current study is to advance this field of research by systematically reviewing the existing literature and analysing the effectiveness of code review in improving software quality.

B. Problem Statement

Software development is a process that involves several stages, including requirements gathering, design, coding, testing, and maintenance. The quality of the software produced is crucial to its success, and several quality assurance practices are employed to ensure that the software meets the established quality standards. Code review is one such practice that is used in the software development industry to detect errors and improve software quality [9].

Code review is a process in which a group of developers checks the source code to identify defects and suggest improvements. Code review can be performed manually, where developers review code by reading through it, or it can be automated, where tools are used to analyse the code and identify defects automatically. Code review is also performed at different stages of the software development process, such as during coding, before testing, or during maintenance [9].

Although code review is commonly practised in software development, its effectiveness in improving software quality remains a topic of debate among researchers. There's very little research that was conducted to address this problem. While some studies have reported significant improvements in software quality metrics such as defect density, code maintainability, and overall software quality, others have found little or no improvement [7,10].

Furthermore, the effectiveness of code review might depend on several factors, such as the type of code review, the expertise of the reviewers, the complexity of the code, and the tools used [8]. The lack of general agreement on the effectiveness of code review in improving software quality, as well as the factors that influence its effectiveness, underscores the need for further research in this area. Therefore, the goal of this research paper is to investigate the effectiveness of code review in improving software quality and to identify the factors that influence its effectiveness. To achieve this goal, the following research questions will guide the study:

1. What is the impact of code review on software quality?
2. What are the factors that affect the effectiveness of code review in improving software quality?
3. How can code review best be implemented to improve software quality?

To answer these questions, the study will conduct a literature review of articles published over the last decade that investigate the effectiveness of code review in improving software quality.

C. Significance of Study

In software development, code review is a critical practice where developers examine the code for potential security and quality issues [15]. The review process can take several forms, including tool-assisted (TA), over-the-shoulder (OTS), peer-to-peer, and pair programming [14]. Despite its widespread use, there is still a lack of empirical evidence on how different code review methods impact software quality [12].

Previous research, including Bosu *et al.'s* work, has examined some effectiveness factors of code review, such as review duration, size, and the number of flaws found [16]. However, additional studies are required to analyse the influence of various techniques and review size on code review effectiveness, as pointed out by a survey conducted by Jureczko *et al.* [14].

The present study aims to contribute to the field of software development by analysing the effectiveness of various code review methods in identifying quality attributes that impact software quality. The authors surveyed software engineers at *hSenid Mobile Solutions* to identify the most critical quality attributes for code reviews, and selected eight attributes that were highly rated for this study. The study aims to compare the effectiveness of different code review methods in addressing these quality attributes.

Code review methods can vary in their impact on the selected quality attributes, as each method is conducted distinctly. Tool-assisted code review involves specialized tools to facilitate the review process [17], while other methods like over-the-shoulder, peer-to-peer, and lead code reviews involve direct interaction between the author and the reviewer [18]. By comparing the effectiveness of these different methods, the study aims to determine which method is most suitable for detecting and addressing quality issues in software development.

The expected outcomes of this study are:

- A better understanding of the strengths and weaknesses of different code review methods in terms of software quality and security.
- A set of guidelines and best practices for choosing and applying code review methods in different contexts and projects.
- A contribution to the improvement of software development processes and practices that can enhance the quality attributes discussed in this journal.

This study will benefit the academic community by addressing a gap in empirical research on code review methods. In addition to benefits for academia and the software industry, this work will have larger impacts on society.

High-quality software is crucial in numerous aspects of modern life, such as healthcare, transportation, and banking [17]. Improving code review procedures can enhance the effectiveness, reliability, and safety of software products, benefiting both users and society as a whole [18].

Software errors can have significant consequences for patients, including incorrect diagnoses or inadequate therapy. Software faults in the transportation industry may result in mishaps or system breakdowns with lasting effects [17]. This study offers the ability to raise software's safety and quality across a wide range of sectors by enhancing code review procedures [18].

Moreover, this study may also have economic benefits. Software flaws can be expensive, both in terms of the expense of repairing them and the impact on users [19]. This study can decrease the number of errors that reach the production stage by identifying the most efficient code review techniques, saving both software development teams and end users time and money [19]. High-quality software development can also give businesses a competitive edge.

D. Objectives

The primary objective of this research study is to thoroughly investigate the effectiveness of code reviews in improving software quality and to identify the different types of code reviewing methods that are commonly used in the industry. Generally, a code review intends to identify and rectify errors and other issues in the code, thereby helping to maintain and improve the overall quality of the software.

In this study, the effectiveness of code review will be thoroughly discussed, while existing studies are comprehensively searched and examined. Additionally, a survey will be conducted among software development professionals to gather information on their opinions regarding code reviews. The survey will include questions about the preferred methods of conducting code reviews, the benefits of code reviews that most affect software efficiency improvement, and the effectiveness of code reviews in improving software quality.

Another key objective of this study is to evaluate the effectiveness of various types of code reviews on software quality improvement. Several code review methods are typically employed, including tool-assisted reviews, automated reviews, peer reviews, walkthroughs, and formal inspections.

Analysing the effectiveness of these different code review methods is crucial for learning how to improve software quality.

This can help reduce development time and improve team collaboration and communication as well. The ultimate goal of analysing the effectiveness of different types of code reviews is to identify the most efficient method suitable for a particular development environment.

Another goal of this research study is to identify the critical components that affect the success of code reviews. Other factors may also influence the success of code review. One of the primary key factors is reviewer expertise. Many studies show that code reviewer expertise is a significant factor that affects the success of code reviewing as well [15]. Other key factors include code review frequency, the code review process, and the tools used for code review.

The different types of code review methods that are commonly used will be examined through this research paper as well. Several code review methods are typically employed, including tool-assisted reviews, over-the-shoulder reviews, walkthroughs, and formal inspections.

The impact of code review on various software quality attributes, such as code maintainability, testability, reliability, security, and usability, will also be discussed in this review.

Also, another objective of this study is to analyze and discuss the relationship between code review and other software engineering practices.

This study aims to provide software developers and managers with alternative advice on optimising their software development processes and maintaining software quality through the collaboration of code review and other software engineering practices.

E. Research Questions

1. What is the impact of code review on software quality?
2. What are the factors that affect the effectiveness of code review in improving software quality?
3. How can code review best be implemented to improve software quality?

The first research question will be addressed by examining the impact of code review on software quality. To determine the advantages and disadvantages of code review and to assess its effect on software quality, it will be necessary to analyse the current literature and case studies.

The impact of code review on raising software quality will be examined in the second research question. To learn more about software developers' experiences with code reviews, including the types of reviews utilized, the experience of reviewers, and the degree of automation employed in the process, a survey of software developers will be conducted. The factors most strongly linked to successful code review will be identified through statistical analysis of the data. The third study topic will focus on the most effective way to utilise code review to enhance software quality. To achieve this, best practices for code review will be developed based on the answers to the first two research questions. A controlled experiment will test these best practices to determine whether they can raise software quality.

In general, the research aims to shed light on the effectiveness of code review in improving software quality and to identify strategies for enhancing its application. The study will advance the discipline of software engineering and contribute to current efforts to enhance software quality by providing insights into these research topics.

II. LITERATURE REVIEW

Previous research has examined the methods of code inspection and code review, and how they help raise the calibre of software. Using the CAIS (Collaborative Asynchronous Inspection of Software) tool, Stein *et al.* did research on distributed, asynchronous code inspections [18].

They discovered that asynchronous inspections using CAIS were successful in locating flaws and improving the quality of software. As participants could evaluate code at their own pace and on their schedule, the authors discovered that asynchronous inspections were more flexible and effective than conventional synchronous inspections.

In a study he conducted on code inspection practices, Laitenburger produced a taxonomy of different code inspection methods [19]. Johnson looked into how open-source software project managers' judgments were affected by code review processes [19].

The study in question examines various facets of code review; therefore, it has been divided into several subsections, each of which discusses a separate field of research. As was done in the study by Jureczko *et al*. [14], the term "patch" has been utilized in this study to refer to changesets for consistency's sake.

A. Effectiveness of the Review Process

The success of code reviews is influenced by several factors, which have been thoroughly investigated in numerous studies. According to Porter *et al*. [17] and Sauer *et al*. [21], the essential components in enhancing inspection efficacy are the skill of the code reviewer and the adoption of superior flaw detection tools. Sauer *et al*. found that two reviewers are the ideal number for the most efficient code review, and this result is in line with behavioural studies that imply that expert pairs outperform larger groups [20].

According to Jureczko *et al*., over-the-shoulder code reviews produce a higher rate of knowledge transfer than tool-assisted code review approaches, which can ultimately lead to superior software quality. To identify the proper areas to which any of these strategies can contribute, further study is required [14]. This study will be an extended version of the original research, and some of its facts and data may be derived from it.

In a study of code reviews in two significant open-source projects, Thongtanunam *et al*. [22] found a connection between software quality and both code ownership and reviewers' skill. Code ownership refers to the concept that some developers have greater expertise and responsibility for specific parts of the codebase. The study found that when code ownership algorithms that consider review activity are used, there is a positive impact on software quality. Additionally, the study found a substantial and upward-trending link between the likelihood of post-release errors and the percentage of reviewers lacking knowledge. This suggests that when reviewers lack knowledge or expertise in the area of code they are reviewing, it can increase the likelihood of errors being introduced into the software after its release.

These findings suggest that the effectiveness of code reviewers and their involvement in the process are key factors in enhancing software quality through code reviews. Reviewers must possess the necessary knowledge and expertise to review the code and provide valuable feedback effectively.

In a separate investigation into the peer review procedures used by the Apache HTTP server project, Rigby *et al*. [23] found that more minor fixes were more likely to result in higher-quality code and were therefore more likely to be accepted.

McIntosh *et al*. looked into the relationship between the quality of software and the involvement in and coverage of the modern code review process [24]. They found a strong correlation between code review participation and program quality. There may be up to two and five more post-release issues for components with low code review coverage and involvement, respectively. This suggests that poorly reviewed code in a complex system can compromise the quality of software.

Similarly, Morales *et al*. investigated the relationship between software design quality and code review practices. They found that software components with less review coverage or involvement are typically more prone to anti-patterns than those with more active code review methods when measuring software design quality by the prevalence of seven different types of anti-patterns. The open-source programs Qt, VTK, and ITK were the basis for this case study [22]. In a study of a major commercial project, Dos Santos and Nunes [23] found that when there were a large number of altered lines of code, the review process took longer and received fewer comments. This suggests that when developers make numerous changes to the code, it requires more time for reviewers to review all the changes and provide feedback.

The study also discovered that having more participating teams and reviewers increased contributions to the review process, but at the cost of time. This suggests that while involving more people in the review process can lead to increased feedback and discussion, it may also prolong the review's completion time.

Ultimately, the study highlights that for larger code patches, reviewers were less engaged and provided fewer feedback comments. This implies that when there are many changes to review, it can be more challenging for reviewers to examine all the changes and provide detailed feedback thoroughly. Effective communication during code reviews is another crucial factor to consider. In their study on communication in code review, Ebert et al. [24] found that confusion was a common issue. The study explains that this confusion often arose from a lack of clarity and explanation of non-functional aspects of the solution. Non-functional aspects refer to characteristics of the software that are not directly related to its functionality, such as performance, security, and maintainability. If these aspects are not clearly explained and understood, it can lead to confusion during the review process. Additionally, the study found that issues with tools and communication could also contribute to confusion during code reviews. For example, if there are disagreements or unclear communicative objectives between developers, it can lead to confusion and delays in the review process.

This confusion can have several adverse effects on the code review process. It can result in delays, less effective reviews, increased discussions, and lower quality software. To address these issues, developers need to communicate clearly and effectively during code reviews, thereby minimising confusion and ensuring that the process is both efficient and effective.

Additionally, some discussions may lead to a higher number of rejected comments. Thus, to some extent, this study provides a calculated advantage that code reviews can have on determining software quality.

B. Classification Of Reviews

The study's research has shown that different methods of code reviews produce different amounts of pointless comments. An extensive examination into the incidence of such remarks in various code review methodologies was done by Jureczko *et al.* According to the study, there were significantly more or fewer nonsensical comments for various types of reviews. In comparison to over-the-shoulder reviews, the researchers found that tool-assisted review procedures produced fewer pointless comments [14]. These findings suggest that the choice of review methodology can significantly influence the effectiveness and efficiency of the code review process.

In a study by Mäntylä and Lassenius, which examined 23 student Java projects and nine commercial C/C++ programs, flaws were categorised. The study identified 388 and 371 deficiencies in the student and industrial projects, respectively, and revealed that 75% of the flaws found during code review had no impact on the software's apparent functionality. These flaws were discovered to improve software evolvability by increasing its understandability and modifiability. Code reviews can thereby increase software quality by discovering flaws that enhance program evolvability. Code reviews can help to increase the software's evolvability and, as a result, its overall quality by locating and fixing flaws that make it easier to comprehend and modify.

An exhaustive mixed-methods study by Bosu, Greiler, and Bird entitled "An Empirical Study at Microsoft" sought to identify the elements that contribute to effective code reviews. Their study consisted of three parts: first, they conducted a qualitative analysis to identify the features of code reviews that engineers found most helpful. Based on their qualitative findings, they subsequently developed and validated a categorisation model that could distinguish between relevant and non-relevant code review feedback. To study the elements that contribute to more effective code review feedback, they utilized their classifier to analyze 1.5 million review comments from five Microsoft projects [13].

The researchers identified several variables that influence the value of code review feedback. For instance, they discovered that during a reviewer's first year at Microsoft, the proportion of valuable remarks they make climbs dramatically, but after that, it tends to stagnate. They also found that the proportion of code review comments that will help the author of the change is negatively correlated with the number of files in the modification. In addition, they discovered that for some projects, comparing changes that span more than 40 files to those made up of only one file could result in a 10% drop in the number of meaningful comments [14]. These results highlight the importance of identifying the key elements that facilitate efficient code reviews, thereby increasing their adoption and ultimately enhancing the quality of the software being developed.

C. Transparency of code

The knowledge transparency of the code across various reviewers and programmers is one of the critical elements contributing to well-defined code reviews, according to an extensive study in this area. This suggests that code reviews facilitate knowledge sharing among developers and encourage a deeper understanding of the codebase among teams. Developers can improve the overall quality of the product and significantly reduce the number of coding errors by exchanging expertise and spotting potential problems in the code.

When compared to tool-assisted code reviews, Jureczko *et al.* found that the over-the-shoulder method enhances knowledge transfer in MCR. They did, however, note that this does not always result in evaluations of a better calibre. In contrast to tool-assisted code reviews, they found that most of the comments gathered through the over-the-shoulder method were minor, indicating that they did not address any significant functional changes that needed to be made to the code [14]. However, the authors emphasise the importance of considering knowledge transfer and review quality when selecting a code review technique, as each method may have its benefits and drawbacks, depending on the demands placed on it by the project and the team. The expectations, results, and difficulties of the existing code review process were examined by Bacchelli and Bird [11]. They emphasised the importance of team members sharing knowledge and working together during the code review process, although they did not directly address knowledge transfer through code review. They found that code review promotes knowledge sharing, high-quality code, and teamwork in addition to finding problems. Additionally, they identified several issues with code reviews, including the need for practical tools and procedures, overcoming reviewer biases, and striking a balance between code reviews and other development tasks.

In their study "Knowledge Transfer in Modern Code Review," Caulo *et al.* looked into whether developers' contributions to open-source projects can get better over time thanks to the code review procedure. They examined numerous peer-reviewed pull requests that developers had posted to GitHub, assuming that if a developer's pull request had previously been reviewed, knowledge had been passed on to that developer during the code review process. They then evaluated whether, as more of their pull requests were examined, the developers' contributions to open-source projects got better over time. They failed to uncover proof, nonetheless, that the code review procedure improved the calibre of developers' contributions [25]. However, Jureczko *et al.* noted that Caulo *et al.* studied tool-assisted reviews while Jureczko *et al.*'s tests showed that over-the-shoulder reviews have a considerably bigger influence on knowledge transfer [14] and that tool-assisted reviews do enhance knowledge transmission to some level. As a result, depending on the specific approach taken, the efficiency of the code review process in improving software quality and promoting knowledge transfer may vary.

The present study aims to provide insight into selected quality attributes of software, informed by the findings of the literature studies reviewed during its conduct.

III. METHODOLOGY

Based on a thorough analysis of the current literature, the goal of this study is to determine how well code reviews contribute to higher software quality. Below is a description of the study's methodology.

Survey: An online survey was administered, consisting of questions that pertained to how code review influences various software quality attributes, including maintainability, testability, usability, correctness, reliability, security, and efficiency. Additionally, participants were asked to provide their demographic information, including job role and level of experience. Some of the common questions that this study covers through the journal are,

1. How often does the participant participate in code reviews?
2. What is the most effective way to integrate code review into the development process for maximum efficiency?
3. Which benefit of code review do participants consider the most significant for enhancing software efficiency?
4. How much time does code review typically add to the testing process for scalability?
5. When working on a security-related aspect of a software application, how helpful have code reviews been in helping the participant understand their mistakes and adopt safer coding practices?
6. How do you track and measure the correctness of your software over time?
7. Which aspects of software reliability have participants noticed the most significant improvements due to code reviews, based on their personal experience?

Search Strategy: To conduct a systematic review of the literature, a structured search strategy is employed to identify relevant studies, utilising Google Scholar, Scopus, and other relevant journals and articles. Inclusion Criteria: Articles must adhere to the following inclusion requirements to be considered for this study:

1. Accentuate how successful code review is at enhancing software quality.
2. Describe empirically how code review affects software quality.
3. Published in conference proceedings or a peer-reviewed journal.
4. Presented in English

Exclusion Criteria: The following exclusion criteria will prevent articles from being included in this study:

1. Consider subjects other than code quality and code review.
2. Don't include empirical information about how code review affects the calibre of software.
3. Not included in conference proceedings or peer-reviewed journals
4. Not in English but written in another language.

Ethics: This study adhered to ethical standards for human subjects' research, which include gaining informed consent from each participant and respecting their anonymity and privacy. In conclusion, the study employed a mixed-methods research design to investigate the contribution of code review to higher software quality. Data was gathered through a survey and analysis of the code review process, which will be evaluated using a combination of qualitative and quantitative techniques. The study's findings will enrich the existing knowledge on the significance of code review and provide suggestions on how to use it most efficiently in software development.

IV. RESULTS

The study of source code by one or more people to identify and correct errors is known as code review, and it is a crucial step in the software development process. Through error detection and improved maintainability, code reviews can raise the quality of software. The efficiency of code reviews has been the subject of numerous studies, which have also revealed several variables that may affect the quality of the review process. A survey was conducted to gather insights from software developers and practitioners on various code review methods, their benefits and challenges, and the impact on software quality. The poll included several inquiries about code review and its effect on software quality. 'What is the most effective way to integrate code review into the development process for maximum efficiency? 'was one of the main questions posted. Five alternatives were given to respondents:

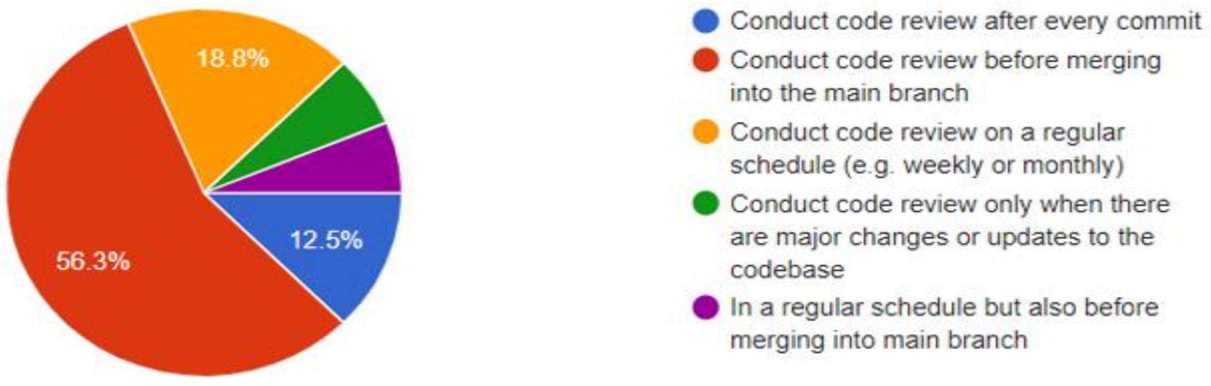

Fig 1. Survey results on code review

56% of those polled said the best way to incorporate code review into the development process was to evaluate the code before merging it into the main branch. This method involves analyzing code modifications before they are merged into the main branch, assisting in the early detection of bugs and issues.

18.8% of respondents thought that performing code reviews frequently (weekly or monthly, for example) was the best strategy. With the help of this method, which entails periodically reviewing a set of changes, the codebase can be reviewed in greater detail over time.

12.5% of those surveyed said that performing code reviews after each commit was the best strategy. This method requires analysing each modification to the code as soon as it is committed, providing quick feedback, and identifying mistakes early.

The survey's findings can provide valuable insights into how to effectively integrate code review into the development process, maximising its effectiveness and enhancing software quality.

Another central question that was asked, focusing on the quality attributes, was 'Which of the following benefits of code review do you think is most important for improving software efficiency?'. Respo.ndents were given four options

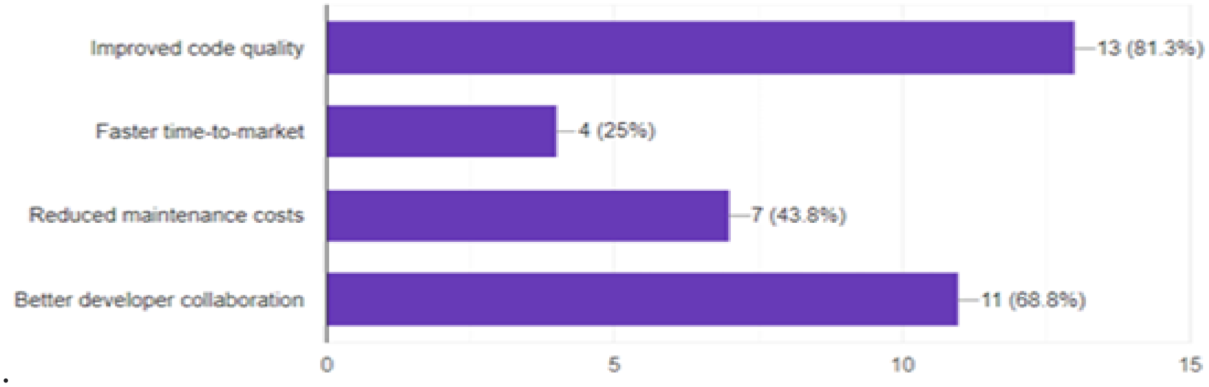

Fig 2. Survey results on quality attribute efficiency

According to 81.3% of respondents, improved code quality is the primary advantage of code review for increasing software effectiveness. This advantage includes identifying flaws and problems at an early stage of development, ensuring the code is maintainable, and enhancing the quality of the software product.

According to 68.8% of respondents, better developer collaboration is the main advantage of code review. This advantage entails encouraging developer collaboration and communication, promoting knowledge sharing and learning, and developing a sense of shared accountability for code quality.

Most responders (43.8%) thought that lower maintenance costs were the most significant advantage of code review. Early bug detection reduces the need for time-consuming debugging and maintenance, resulting in time and cost savings.

Faster time-to-market was deemed the most significant benefit of code review by 25% of respondents. This advantage includes early error and problem detection, decreased rework requirements, and eventually faster software delivery.

Another critical question that was asked in the survey was 'How can code review help to improve software maintainability. Respondents were given three options.

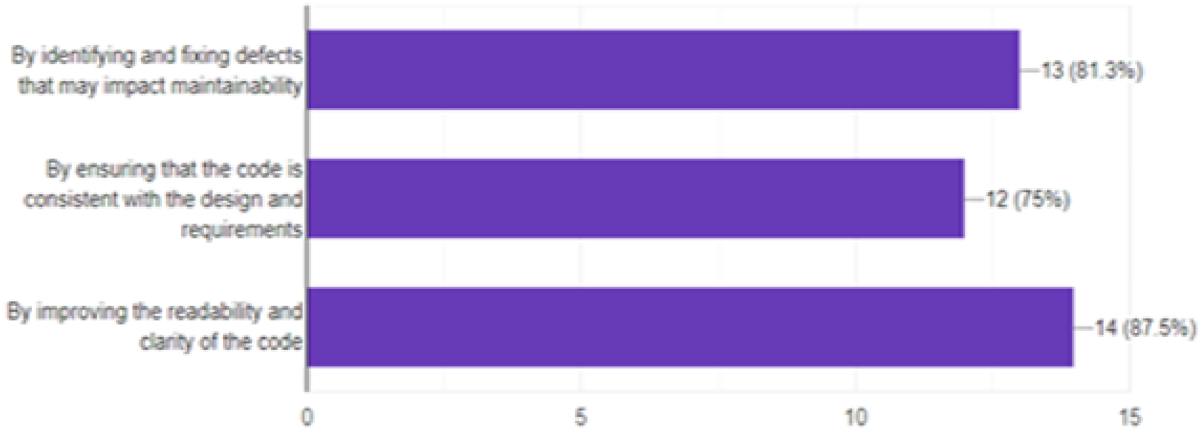

Fig 3. Survey results on quality attribute maintainability

According to the respondents, 81.3% stated that the most crucial method code review can increase software maintainability is by locating and addressing errors that may impact it. This entails reviewing code updates to identify any potential issues, including complex code or unclear logic, that may affect the software's long-term maintainability and stability. Code review can help ensure that the codebase is manageable and easy to work with in the long run by identifying these problems early.

According to 87.5% of respondents, one significant way that code review can increase software maintainability is by making the code easier to read and understand. This entails checking updated code to ensure it is simple to read and comprehend, and that it utilises clear and brief naming conventions, comments, and documentation. Code review can make the codebase easier to read and understand, which in turn makes it easier to maintain and work with over time. According to 75% of respondents, one significant way that code review can increase software maintainability is by ensuring that the code is compatible with the design and specifications. This entails checking the updated code to ensure it adheres to project specifications and is consistent with the software's overall design. Code review can simplify maintenance and updates over time by verifying that the codebase is consistent and aligned with the project goals.

The findings of this survey can provide valuable insights into how code review can enhance software maintainability, and they can help firms streamline their code review procedures to achieve this crucial objective.

Overall, by considering numerous elements such as the reviewer's experience, the use of diverse approaches, and the involvement of reviewers in the process, the effectiveness of code review can be increased. Code reviews are becoming increasingly crucial as software systems become more complex, ensuring the quality and maintainability of software.

V. DISCUSSION

The results suggest that code review is a crucial process that can enhance the quality of software. According to the research analyzed in this paper, code review can reduce post-release errors, help uncover issues, and increase knowledge transfer.

The reviewers' level of experience is one of the primary variables that affects the efficiency of code review. Expert reviewers are more adept at spotting errors and enhancing software quality, according to the studies examined in this paper. The number of reviewers also affects the effectiveness of a code review. Two reviewers are the ideal number for a fruitful code review.

Asynchronous code review has the potential to be more effective and adaptable than synchronous code review, a significant discovery. Code reviews may be of higher quality when conducted asynchronously, as participants can evaluate the code at their leisure and on their schedule. Additionally, depending on the circumstance, it may be beneficial to employ various code review approaches such as tool-assisted and over-the-shoulder reviews.

The results of this study suggest that code review is a valuable technique for improving software quality, although certain limitations and challenges should be considered. One drawback is that the size and complexity of the codebase being examined can impact the effectiveness of code review. Another drawback is that the quality of the review can be significantly affected by the reviewers' experience and engagement in the review process. Implementing code review in practice can also be challenging due to time constraints and resistance to change.

The study's findings demonstrate conclusively that code review is an effective strategy for enhancing software quality. The findings suggest that asynchronous review and expert reviewers can lead to higher-quality reviews. However, it's crucial to consider the restrictions and difficulties associated with implementing code review in practice. Future studies could examine the usefulness of code review in various situations and investigate methods for resolving implementation difficulties.

VI. CONCLUSION

This study has produced convincing evidence in favour of using code review as a valuable strategy for improving software development quality, following a thorough analysis of its effectiveness in enhancing software quality. The study's findings suggest that code review can significantly enhance software quality, particularly in terms of reducing defect rates, improving maintainability, and increasing productivity.

The advantages of code review are numerous. First and foremost, the code review procedure aids in the early detection of errors in the development process, before they can evolve into expensive and time-consuming issues. This is crucial in the software development industry, as even the most minor mistakes can quickly accumulate to become serious failures. Code review can help identify problems early and prevent them from escalating, ultimately leading to a more effective development process.

Additionally, it has been demonstrated that code review is a powerful tool for lowering software flaws. This is because code review promotes developer collaboration and information sharing, which helps identify mistakes and potential defects that might have gone unnoticed during individual coding efforts. Additionally, the comments and recommendations made during a code review can help raise the calibre of the code and guarantee that best practices are being followed.

Finally, code review can enhance the software's maintainability, making it more straightforward for engineers to update the code over time. This is due to code review's role in ensuring that code is well-structured, well-documented, and written clearly and understandably. Code review can help extend the life of software and ensure that it remains valuable and relevant over time by enhancing maintainability.

The implementation of code review still faces difficulties despite its many advantages. Constraints of time and resources are a significant issue, especially in organizations that are already overburdened with competing agendas.

As a result, it's crucial to create methods for streamlining and increasing the efficiency of code review, such as by employing automation tools or creating explicit rules for the procedure. Future research has numerous opportunities to build on the findings of this study. For instance, further research might investigate how the effects of code review vary for other software project types, such as large-scale systems or safety-critical applications. Research may also examine the effects of various coding review methods, such as pair programming or tool-assisted code review. Future studies can help improve and modify the practice of code review to better meet the demands of software engineers and organisations by examining these and other issues. In conclusion, this study offers compelling evidence in favour of the usage of code review as an effective method for raising the calibre of software. Code review can help ensure that software is of high quality and remains relevant over time by identifying problems early on, minimising faults, and enhancing maintainability. While implementing code review still presents some difficulties, continuous research can help overcome these obstacles and improve the technique better to meet the objectives of software engineers and organisations.

DECLARATION

Funding/ Grants/ Financial Support	No, we did not receive.
Conflicts of Interest/ Competing Interests	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval and consen.t to participate with evidence
Availability of Data and Material/ Data Access Statement	Not relevant
Authors Contributions	All authors have equal contributions to this article.

REFERENCES

1. "Code Review Developer Guide," [Online]. Available: https://google.github.io/eng-practices/review/.
2. O. B. L. G. Y. C. M. W. G. Oleksii Kononenko, "Investigating Code Review Quality," p. 10, 2015.
3. "Best practices for reviewing a code," [Online]. Available: https://www.codegrip.tech/productivity/best-practices-for-reviewing-code/.
4. "Benefits of Code Review: Every Team Must Know [2022 Guides]," [Online]. Available: https://gaper.io/benefits-of-code-review/.
5. [Online]. Available:https://about.gitlab.com/topics/version-control/what-is-code-review/#:~:text=Code%20reviews%2C%20also%20known%20as,developers%20learn%20the%20source%20code, developers learn the source code.
6. [Online]. Available: https://www.freecodecamp.org/news/how-to-avoid-code-review-pitfalls-that-slow-your-productivity-down-b7a8536c4d3b/.
7. Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. IEEE Software, 30(4), 84-91. doi: 10.1109/MS.2012.175 [CrossRef]
8. Baltes, S., & Diehl, S. (2014). Improving software quality through code review: A case study. IEEE Software, 31(4), 60-67. doi: 10.1109/MS.2013.104 [CrossRef]
9. Kemerer, C. F., & Porter, L. F. (1992). Improving software quality through a formal technical review process. Journal of Systems and Software, 19(2), 119-131. doi: 10.1016/0164-1212(92)90074-f
10. P. C. Rigby and C. Bird, "Convergent contemporary software peer review practices," in Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering (ESEC/FSE), 2013, pp. 202-212. [CrossRef]
11. A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in Proceedings of the 2013 International Conference on Software Engineering (ICSE), 2013, pp. 712-721. [CrossRef]
12. Beller, M., Bacchelli, A., Zaidman, A., & Juergens, E. (2014). Modern code reviews in open-source projects: which problems do they fix? In Proceedings of the 11th Working Conference on Mining Software Repositories (pp. 202-211). [CrossRef]
13. Bosu, A., Greiler, M., Bird, C.: 'Characteristics of useful code reviews: an empirical study at Microsoft'. Proc. of the 20
14. M. Jureczko, Ł. Kajda, and P. Górecki, "Code review effectiveness: An empirical study on selected factors influence," IET Software, vol. 14, no. 7, pp. 794–805, 2020. [CrossRef]
15. J. & K. Y. Shimagaki, "The effect of reviewers' expertise on code review outcomes," 2018.
16. Devart Software, "Code Review Benefits," [Online]. Available: https://www.devart.com/review-assistant/learnmore/benefits.html.
17. M. Stein, J. Riedl, S.J. Harner and V. Mashayekhi, "A Case Study of Distributed Asynchronous Software Inspections," in Proceedings of the 19th International Conference on Software Engineering, New York, NY, USA: ACM Press, 1997, pp. 107-117. [CrossRef]
18. A. Sutherland and G. Venolia, "Can peer code reviews be exploited for later information needs?" in Proceedings of ICSE, May 2009 [CrossRef]
19. O. Laitenberger, "A Survey of Software Inspection Technologies," in Handbook on Software Engineering and Knowledge Engineering, vol. 1, World Scientific Publishing Co., 2002, pp. 517-555. [CrossRef]
20. A. Porter, H. Siy, A. Mockus and L. Votta, "Understanding the sources of variation in software inspections," in ACM Transactions on Software Engineering and Methodology, vol. 7, no. 1, pp. 41-79, Jan. 1998. [CrossRef]
21. Sauer, C., Jeffery, D.R., Land, L., *et al.*: 'The effectiveness of software development technical reviews: A behaviorally motivated program of research', IEEE Trans. Softw. Eng., 2000, 26, (1), pp. 1–14 [CrossRef]
22. R. Morales, S. McIntosh and F. Khomh, "Do code review practices impact design quality? A case study of the Qt, VTK, and ITK projects," 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), Montreal, QC, 2015, pp. 171-180. [CrossRef]
23. Dos Santos, E.W., Nunes, I.: 'Investigating the effectiveness of peer code review in distributed software development'. Proc. of the 2017 Brazilian Symp. on Software Engineering, 2017, pp. 84–93 [CrossRef]
24. Ebert, F., Castor, F., Novielli, N., *et al.*: 'Communicative intention in code review questions'. Proc. of the 2018 IEEE Int. Conf. on Software Maintenance and Evolution, 2018, pp. 519–523 [CrossRef]
25. Caulo, M., Lin, B., Bavota, G., Scanniello, G., & Lanza, M. (2020). Knowledge Transfer in Modern Code Review. In Proceedings of the 28th International Conference on Program Comprehension, pp. 230-240. [CrossRef]

AUTHORS PROFILE

Dr. Dilshan De Silva is a seasoned professional in the field of Information Technology (IT) with a strong background in Software Engineering. They obtained their BSc. Special (Hons.) in Information Technology (Software Engineering) between 2005 and 2009, followed by an MSc. in Information Technology (Software Engineering) from 2009 to 2012. With extensive expertise and experience, Dr. Dilshan currently serves as a Senior Lecturer (HG) in the Department of Computer Science and Software Engineering at the Faculty of Computing, Sri Lanka Institute of Information Technology. In this role, they impart their knowledge and mentor students in the field of Software Engineering. Dr. Dilshan's research interests primarily revolve around Software Complexity, Software Metrics, Machine Translations, and Augmented Reality. They are deeply committed to exploring these captivating areas and advancing the understanding of software engineering practices. With a strong academic foundation, combined with their practical experience and passion for innovative technologies, Dr. Dilshan is well-equipped to make significant contributions to the field of IT. Their expertise in software engineering, research acumen, and dedication to teaching make them an invaluable asset in academia and research projects.

W.A. Chamali Pabasara is a talented individual with a passion for the field of Information and Communication Technology (ICT). Currently pursuing a Master of Science in Information Technology at the University of Moratuwa, Chamali holds a BSc (Special) degree in ICT from the University of Sri Jayewardenepura. With a diverse academic background, Chamali has gained valuable experience and expertise in the field. They are currently working as an Assistant Lecturer in the Department of Computer Science and Software Engineering at the Sri Lanka Institute of Information Technology. Previously, they served as an Academic Instructor at the same institute and as a Computer Instructor at the University of Sri Jayewardenepura. Chamali's research interests span several exciting areas, including the Internet of Things (IoT), Data Mining, Machine Learning, Cloud Computing, and Computer Vision. They are dedicated to exploring these domains and making contributions to advance the understanding and application of these technologies. With a strong academic foundation, practical experience, and a genuine enthusiasm for emerging technologies, Chamali is poised to become a valuable contributor to the field of ICT. Their commitment to learning and drive for innovation make them a valuable asset to any research project or academic endeavour.

Srirajan Vithya Sangkavi is a highly motivated and ambitious 3rd-year undergraduate student specialising in Software Engineering at the Sri Lankan Institute of Information Technology. With a commendable CGPA of 3.4, Vithya demonstrates a strong aptitude for academic excellence. Passionate about technology and its impact on society, Vithya actively engages in research activities to broaden their knowledge and skills. Vithya's research interests encompass various aspects of software engineering, including software development methodologies, web app development, and mobile application development. They are keen on exploring innovative approaches to improve software development processes and enhance user experiences. Vithya have a proactive approach to learning and seeks opportunities to collaborate with peers and mentors to expand their knowledge further and contribute to the field. Vithya's academic achievements and dedication to research reflect their commitment to making a meaningful impact in the field of software engineering. With their strong work ethic, analytical thinking, and passion for innovation, Vithya aims to make significant contributions to the field of software engineering in the future.

L.G.A.T.D. Wijerathne is a student at the Sri Lanka Institute of Information Technology (SLIIT), pursuing a BSc (Hons) in Information Technology with a specialisation in Software Engineering. With a passion for software development, L.G.A.T.D. Wijerathne skillfully combines their technical expertise with a love for writing. Through their works, they provide unique insights into the world of coding, programming, and software engineering, aiming to inspire fellow students and technology enthusiasts alike. Join L.G.A.T.D. Wijerathne on their journey as they explore the intersection of technology and literature, illuminating the wonders of software engineering through their words. L.G.A.T.D. Wijerathne's commitment to learning and drive for innovation make them a valuable asset to any research project or academic endeavour, constantly pushing the boundaries of software engineering with their insatiable curiosity and creative problem-solving skills.

Wijesundara W.M.K.H. is a third-year software engineering student at the Sri Lanka Institute of Information Technology (SLIIT). She is highly interested in software development and all things technological. Wijesundara has developed a strong foundation in programming languages, algorithms, data structures, and software design principles throughout her education. To gain expertise in realistic situations, she has worked on a variety of projects, both independently and in teams. Wijesundara actively seeks opportunities to enhance her knowledge in software engineering. She has participated in hackathons and coding contests, showcasing her inventiveness and problem-solving abilities. She has gained teamwork, communication, and time management skills as a result of these experiences. Her primary areas of interest in study include the interactions between software engineering and modern fields, such as data science, machine learning, and artificial intelligence. She wants to help with the creation of innovative software programs that use these technologies to solve problems in the real world. As an author, Wijesundara approaches their work with precision. She carries out in-depth studies, exercises critical thinking, and maintains academic integrity. With strong attention to detail and clear writing skills, she effectively communicates her ideas to various audiences. Overall, Wijesundara is a motivated and committed student who aspires to become a successful software engineer. She is willing to share her talents and advance the technological sector.

Reezan S.A. is an undergraduate student at the Sri Lanka Institute of Information Technology (SLIIT), pursuing a Bachelor of Science degree in Information Technology with a specialization in Software Engineering. Currently in his third year of studies, Reezan demonstrates a keen interest in programming and possesses experience in various programming languages, including Java, JavaScript, and C. With a passion for software development, Reezan has completed multiple programming projects and assignments, consistently achieving commendable grades. His dedication to academic excellence is evident as he has consistently maintained a top-ranked position throughout his academic career, currently holding an impressive GPA of 3.84/4. In his free time, Reezan enjoys enriching his knowledge by reading books on technology and programming. His commitment to staying updated with the latest advancements in the field further fuels his passion for software engineering. For any inquiries, collaboration opportunities, or additional information, Reezan can be reached at imreezan@gmail.com. He looks forward to contributing to the ever-evolving world of technology and software engineering.