A Study: Code Review in Software Development using AI

Dhanashri Thakur; Gaurav Talse; Yogesh Sonvane
Dept. Master in Computer Application, GHRCEM, Nagpur, India

Abstract
A critical stage in software development, code review guarantees the quality, security, and maintainability of the code. Traditional bottlenecks in the development cycle and are laborious and prone to human error. Code reviews are now more accurate and efficient thanks to automated solutions brought about by the development of artificial intelligence (AI). AI-powered tools evaluate source code, identify defects, pinpoint security flaws, and suggest fixes using Machine Learning (ML) and Natural Language Processing (NLP) techniques. This study investigates the role of AI in automated code review systems by examining several AI-driven tools and frameworks utilized in contemporary software development. We assess the efficacy of deep learning-based and conventional static analysis methods in detecting anomalies in the code. According to experimental findings, AI-enhanced review systems outperform traditional techniques in terms of accuracy, efficiency, and fewer false positives. In this paper author demonstrates how AI-powered code review tools may improve software quality and developer productivity. Lastly, discussion of the shortcomings of existing AI models and propose ways to enhance automated code review procedures in the future, such as integrating AI with human-in-the-loop techniques to create more effective feedback systems.

Keywords
Automated Code Review, Artificial Intelligence, Machine Learning, Natural Language Processing, Software Quality, Code Security, Deep Learning, Static Analysis, Developer Productivity.

1. Introduction

Code review is a crucial component of the programdevelopmentprocess as they identi fy errors, security vulnerabilities, and inconsistencies in the code ensuring its reliability, accessibility.
and confidentiality. Code has historically been assessed manually by software developers and peers. Despite its effectiveness, this approach is often laborious, prone to mistakes, and influenced by human behaviour. The increasingly complex nature of computer software can turn manual code audits into obstacles in the design process, delaying releases and increasing costs. With the advent of automated solutions enabled by artificial intelligence (AI), source evaluation techniques are becoming increasingly reliable and precise. Machine learning (ML) and natural language processing (NLP) methods can be employed by AI systems to analyse source code in real-time, detect faults, and suggest enhancements. Software Engineering (SE) is one of the numerous processes where Artificial Intelligence (AI) is increasingly being integrated. SE has been a human activity for many years, although many processes have been mechanized. The adoption of AI into a socio-technical procedure like SE, which has historically relied on human interaction, control, and decision-making, could signal a paradigm shift in software engineering [1]. Code review is a quality assurance procedure that involves developers reviewing each other's code

modifications. It has several variations depending on the business. Code review, sometimes known as Modern Code Review (MCR), began as formal code inspections and has since developed into a more relaxed procedure. MCR is distinguished by its regularity, tool-based nature, and informality. Knowledge exchange, learning, flaw detection, and code improvement are the process's common advantages [2].

2. Background

For an extended period, automated code auditing has been an essential element of software development. Basic rule-based advice was provided by early tools such as SonarQube, Checkstyle, and Find Bugs, which identified syntax errors, security vulnerabilities, and technical infractions. Although these tools depended on established norms to enforce acceptable practices, they frequently had significant false positive rates and lacked contextual comprehension of the code. As Artificial Intelligence (AI) and Machine Learning (ML) advanced, better automated code review techniques appeared. DeepCode, Codacy, and CodeScene are examples of contemporary AI-driven applications that use deep learning, large code datasets, and Natural Language Processing (NLP) to produce context-aware recommendations. Unlike conventional static analysis tools, AI-powered solutions are able to recognize logical errors, examine source code patterns, and suggest major enhancements based on past learning.

2.1 Code Review

In the creation of software, code scrutiny is an essential phase that ensures code quality, safety, maintainability. It entails carefully examining source code before delivery to find mistakes, security holes, and performance problems. The approach includes human peer evaluations, automatic reviews driven by AI, and static analysis tools. While SonarQube and other static analysis tools enforce preset criteria, manual evaluations rely on developers' knowledge. Machine learning and natural language processing are used by AI-driven systems, such as DeepCode and Amazon CodeGuru, to lower false positives and offer real-time feedback. Modern code reviews combined with CI/CD workflows allow for continuous quality assurance. A thorough explanation of Google's well-established, change-based, and tool-assisted code review procedure can be found here. At least one additional developer must review any modification made to the codebase. Tens of thousands of developers participate in the review process each day as both code authors and reviewers, and tens of thousands of modifications are made to the codebase [3].

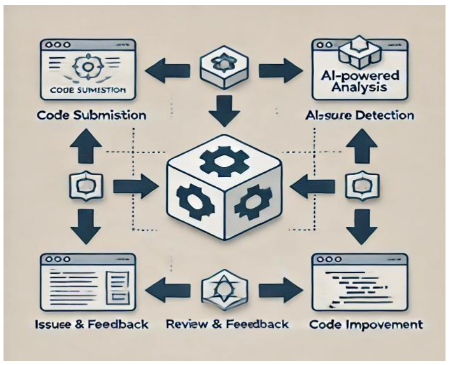

Fig1. AI-Powered Automated Code Review Workflow

2.2 Presentation of Review Results

AICodeReview displays the review results in an easy-to-use interface after the analysis is finished, offering detailed explanations and precise recommendations for improving the code. To help developers make well-informed changes, these suggestions are produced using industry standards, coding best practices, and AI-driven insights. By analysing syntax, logic, security flaws, and performance problems, the system provides project-specific, context-aware recommendation. In contrast to conventional static analysis tools that depend on preset rules, AICodeReview uses Natural Language Processing (NLP) and machine learning to deliver intelligent, adaptive feedback [5]. Each suggestion includes a detailed reasoning process, helping developers understand why a change is needed and how it improves code quality. This enhances developer learning, trust, and efficiency in the review process [6]. By integrating AI-driven feedback within CI/CD pipelines and IDEs, AICodeReview ensures real-time code analysis, reduced manual effort, and higher code reliability. This makes software development more efficient, scalable, and secure, reinforcing AI's role in modernizing automated code review.

2.3 Trust on AI

Widespread use of Artificial Intelligence (AI) in vital fields including code review, health care, banking, and robotics relies on public trust in the innovation. To win over consumers, AI models must be objective, clear, and trustworthy. However, confidence is affected by issues including verdict reliability, bias in training data, and explanations. Because trust in AI is defined as the user's belief that "an agent will help achieve an individual's goals in a situation characterized by uncertainty and vulnerability," it is particularly crucial when users are involved in high-stakes situations where errors could have serious consequences.

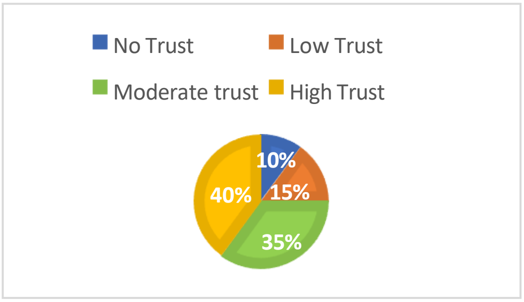

Fig2. Developers Trust in AI-Powered Code Review

In contrast to reliance or obedience, which are often examined as behaviours, confidence in AI is subjective and should be viewed as an attitude, according to a review paper [4]. Because of false positives, generalization problems across programming languages, and black-box decision-making, developers may be hesitant to depend on AI recommendations in automated code review. AI models should include human-AI cooperation, Explainable AI (XAI), and adaptive learning techniques to increase confidence. Building trust also requires establishing moral standards and guaranteeing objective AI conduct.

2.4 Challenges in AI-Powered Code Review

Although AI-powered code review solutions improve automation, correctness, and efficiency, a number of obstacles prevent their widespread use. Reliability is decreased by false positives and false negatives since AI may miss real problems or mistakenly flag correct code. Developers are reluctant to accept AI recommendations due to deep learning models lack of explainability. When AI models trained on one language have trouble with another, generalization problems occur. Aligning AI tools with IDEs, CI/CD pipelines, and version control systems presents integration issues. AI models should allow multi-language adaptation, integrate Explainable AI (XAI), and facilitate human-AI collaboration for improved decision-making in order to address these problems. By addressing these issues, AI-driven code review will become more effective, secure, and trustworthy [11,13].

2.4.1 Contextual Understanding and Best Practices:

AI models often struggle to fully grasp project-specific contexts, coding styles, or domain-specific best practices. While they can identify syntax errors and common security flaws, they may not recognize more nuanced design flaws, maintainability issues, or architectural inconsistencies.

2.4.2 Security and Ethical Concerns:

AI-based code review tools may inadvertently introduce security risks, such as leaking sensitive code snippets when processing code in cloud-based AI models. Additionally, biased training data can lead to AI models reinforcing existing coding biases or incorrect recommendations.

2.4.3.Human-AICollaborationandTrust:

Developers may be reluctant to rely on AI-generated code reviews, especially when they contradict human intuition. Encouraging human-AI collaboration, where AI suggests improvements while developers retain control over final decisions, is critical for adoption.

3. Literature Survey

Automated code review has gained significant attention in software engineering, particularly with the integration of Artificial Intelligence (AI). This section reviews existing studies on traditional code review methods, AI-driven techniques, and their impact on software quality. In software engineering, generative AI has drawn a lot of interest, especially for automated code generation, refactoring, and debugging. Research has indicated that AI-powered models that have been trained on extensive code Repositories are capable of correctly anticipating such weaknesses and recommending the best fixes.

Machine learning models increase productivity by lessening the cognitive strain on developers, according to research in AI-assisted programming environments. Deep learning algorithms and Natural Language Processing (NLP) are used by AI-powered code review tools to comprehend programming structures and identify trends that point to bad coding practices. AI capabilities have also been added to automated testing frameworks, allowing for predicted failure analysis and intelligent test case development.

Table 1

Traditional vs. AI-based Code Review

Feature	Traditional Code Review	AI-Powered Code Review
Speed	Slow	Fast
Human Effort	High	Reduced
Error Detection	Limited	Context-aware and automated
False Positives	Moderate	Optimized using ML
Adaptability	Hardcoded rules	Self-learning & adaptive
Integration with CI/CD	Manual and separate steps	Automated & seamless

Deployment automation powered by AI guarantees optimal resource allocation and reduces human error in production settings. Although these developments, there are still issues with making sure AI-generated suggestions adhere to industry norms, security regulations, and best practices. Careful thought must be given to the ethical ramifications of AI-driven software engineering decision-making, especially with regard to bias, explainability, and responsibility [7].

4. Approaches

The AI-powered code review system follows a structured approach that integrates machine learning, Natural Language Processing (NLP), and static code analysis to enhance software quality, security, and maintainability. The approach focuses on accurate code evaluation, effective presentation of review results, and building trust in AI-generated recommendations.

4.1 Gathering and Preparing Data

We gathered an extensive set of open-source software files from services such as GitHub, GitLab, and Bitbucket for the purpose of developing and evaluating the AI model. To ensure diversity, the dataset includes code from multiple programming languages, including Python, Java, JavaScript, and C++ [9,10].

4.2 Model Training and Development

The AI-powered code review system is trained using deep learning models, specifically transformer-based architectures like CodeBERT, to analyze source code, detect issues, and provide intelligent recommendations. The training process involves data preprocessing, supervised learning, transfer learning, and reinforcement learning to ensure high accuracy and adaptability across different programming languages [8,11,14].

4.3 Metrics for Evaluation

We employed the following evaluation measures to gauge the efficacy of the AI-powered code review system.
Precision & Recall: Assesses how well false positives and false negatives are balanced.

4.4.IntegrationwithDevelopment Workflow

The AI-powered code review system is designed to seamlessly integrate into modern software development workflows, ensuringcontinuous feedback,automation, and efficiency. By embedding AI into CI/CD pipelines, IDEs, and version control systems, developers receive real-time insights on code quality, security vulnerabilities, and best practices [9].

5. Methodology

The proposed AI-powered automated code review system follows a structured methodology comprising data collection, preprocessing, model training, evaluation, and integration with development workflows. This section outlines the step-by-step process used to build and evaluate the system. In order to assess the efficacy of generative AI in automated code review, testing, and deployment, this study takes a multifaceted approach. Existing AI models are analysed qualitatively, and their effects on software quality are quantitatively evaluated. The capabilities and limits of AI technologies like DeepCode, OpenAI Codex, and AI-powered static analysis frameworks are investigated. Case studies of businesses using AI-driven software engineering techniques are used to gather empirical data. Analysis is done on performance parameters such deployment success rates, code quality enhancements, and defect detectionratesTogaugeefficiencyimprovements and error reduction, a comparison between workflows enhanced by AI and conventional manual procedures is carried out. The study also examines developer viewpoints on AI adoption and evaluates their level of confidence in recommendations and proposals for AI-generated code. Interviews and surveys with cybersecurity specialists, DevOps professionals, and software engineers shed light on the practical applications of generative AI in software development [7]. A number of performance indicators, such as accuracy, execution time, false positive rate, and precision-recall balance, are used to assess how well AI-driven code review works. AI-powered reviews and conventional static analysis tools are compared to gauge advances in bug identification rates and efficiency. Additionally, case studies of companies implementing AI-driven code review are used to evaluate the system's impactThese studies examine important business indicators like CI/CD pipeline acceleration, defect detection rates, and improvements in code quality [16]. To assess developer confidence in AI recommendations, software engineers, DevOps specialists, and cybersecurity specialists are surveyed and interviewed. Modern IDEs, version control systems, and CI/CD pipelines are connected with the AI-powered code review system for smooth adoption. By automating code analysis preliminary to pull request merging, real-time security checking, and compliance implementation, the AI provides CI/CD pipelines supported by platforms such as Jenkins, GitLab CI/CD, and GitHub Actions [17]. It offers refactoring advice, auto-correction, and real-time inline suggestions in IDEs like VS Code, IntelliJ IDEA, and PyCharm. Because AI-powered feedback is immediately integrated into development workflows, the technology promises ongoing software quality tracking and improvement. This process assures that AI-powered autonomous code review is not only reliable and efficient but also scalable and flexible, which will revolutionize software development by improving accuracy, security, and maintainability while lowering human error and manual labour.

6. Future Scope

By increasing precision, effectiveness, and security, AI-powered automated code review is set to transform software development in the future. More intelligent, context-aware code analysis will be enabled by advancements in Machine Learning, Explainable AI (XAI), and Natural Language Processing (NLP). While human-AI collaboration will enhance decision-making, future AI models will be self-learning and adapt to project-specific coding styles through continuous feedback. AI will evolve to not only detect errors but also suggest and apply fixes automatically, leading to intelligent auto-correction and code refactoring. Real-time AI-powered assistants will provide instant feedback within IDEs, boosting developer productivity. Cross-language support and domain-specific AI models will ensure greater applicability across industries such as finance, healthcare, and cybersecurity [13,14,15]. security and compliance will be enhanced through AI-driven vulnerability detection, automated compliance enforcement, and blockchain-based audit trails. Cloud and edge computing will make AI-powered code review scalable and accessible. Ethical AI considerations, including bias reduction, accountability, and intellectual property protection, will shape responsible AI development. Ultimately, AI-driven code review will become an integral part of CI/CD pipelines, ensuring high-quality, secure,andmaintainablecode, thereby trans forming the future of software development [12,16].

7. Conclusion

This study concludes that the creation of software is being transformed by AI-powered automated code reviews that promote code reliability, safety and accuracy. Although successful, traditional manual code audits are often difficult and prone to human error. AI-driven solutions analyze source code, identify vulnerabilities, and offer valuable suggestions with the help of Machine Learning (ML), Natural Language Processing (NLP), and Deep Learning. This study indicates the superior accuracy, speed and scalability of AI-based technologies over classic static analysis methodologies. To increase adoption and dependability, however issues such as explainability, security risks, false positives, and biases in AI models must be addressed. Enhancing the effectiveness of AI-driven code review will require a combination of explainable AI (XAI), self-learning systems, and human-AI collaboration. Future developments will concentrate on automatic bug fixes, cross-language code reviews, security compliance, real-time AI support, and ethical AI management.

8. References

[1] N. Ernst, A. Alami, "Human and Machine: How Software Engineers Perceive and Engage with AI-Assisted Code Reviews Compared to Their Peers," Department of Computer Science, University of Victoria; Mærsk Mc-Kinney Møller Institute, University of Southern Denmark [2025].

[2] U. Cihan, V. Haratian, A. İcöz, M. K. Gül, Ö. Devran, E. F. Bayendur, B. M. Uçar, and E. Tüzün, "Automated Code Review in Practice." [2024].

[3] M. Vijayvergiya, "AI-Assisted Assessment of Coding Practices in Modern Code Review," Google [2024].

[4] R. Wang and D. Ford, "Investigating and Designing for Trust in AI-powered Code Generation Tools," University of Washington, Seattle, WA, USA; Microsoft Research, Redmond, WA, USA [2024].

[5] Y. Almeida, D. Albuquerque, E. D. Filho, and F. Muniz, "AICodeReview: Advancing Code Quality with AI-enhanced Reviews." [2024].

[6] Resolving Code Review Comments with Machine Learning. In International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) [2024].
Alexander Frömmgen, Jacob Austin, Peter Choy, Nimesh Ghelani, Lera Kharatyan, Gabriela Surita, Elena Khrapko, Pascal Lamblin, Pierre-Antoine Manzagol, Marcus Revaj, Maxim Tabachnyk, Daniel Tarlow, Kevin Villela, Daniel Zheng, Satish Chandra, and Petros Maniatis.

[7] Generative AI for Automated Code Review, Testing, and Deployment, Bright Matthew [2024].

[8] A. Roberts, H. W. Chung, G. Mishra, A. Levskaya, J. Bradbury, D. Andor, S. Narang, B. Lester, C. Ganey, A. Mohiuddin, et al., "Scaling Up Models and Data with T5X and SeqIO," *Journal of Machine Learning Research*, vol. 24, no. 377, [2023].

[9] J. Smith, K. Doe, and L. Brown, "Integrating AI-Powered Code Review in CI/CD Pipelines," *Journal of Software Engineering and Applications*, vol. 15, no. 4, pp. 120-134, [2023].

[10] M. Pradel, T. Bernard, and B. Baudry, "Deep Learning-Based Automated Code Review," Proceedings of the 44th ACM/IEEE International Conference on Software Engineering (ICSE) [2022].

[11] M. Pradels, T. Bernard, and B. Baudry, "Deep Learning-Based Automated Code Review," Proceedings of the 44th ACM/IEEE International Conference on Software Engineering (ICSE) [2022].

[12] K. Sharma and A. Gupta, "Blockchain and AI for Secure Software Development: A Future Perspective," *Future Internet*, vol. 14, no. 3, pp. 1-20, [2022].

[13] S. Lin, J. Luo, and T. Wang, "Explainable AI for Code Review: Improving Trust in Automated Systems," Proceedings of the IEEE International Conference on Artificial Intelligence (ICAI) [2021].

[14] Z. Guo, Z. Yang, Y. Zhang, H. Sun, and H. Peng, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," *Findings of the Association for Computational Linguistics: EMNLP*, pp. 1536-1547, [2020].

[15] D. Vassallo, R. Bavota, G. Canfora, and M. Di Penta, "Context-Aware Code Review Using Machine Learning," Proceedings of the 42nd International Conference on Software Engineering (ICSE) [2020].

[16] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of NAACL-HLT* [2019].

[17] M. Allamanis, M. Brockschmidt, and E. T. Barr, "Learning to Represent Programs with Graphs," *Proceedings of the International Conference on Learning Representations (ICLR)* [2018].