2025 10th International Conference on Fog and Mobile Edge Computing (FMEC) | 979-8-3315-4424-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/FMEC65595.2025.11119365

AI-Driven Continuous Integration: Automating Code Review and Deployment with LLMs

Dina Omar Salem
School of Business
University of Central Lancashire
Preston PR12HE, UK
dossumaida@uclan.ac.uk

Yazan Alahmed
Al Ain University
UAE,Abu Dhabi
College of Engineering
Yazan.alahmed@aau.ac.ae

Ragad Fnich
Al Ain University
UAE,Abu Dhabi
College of Engineering
202110161@aau.ac.ae

Meryem Mazroub
Al Ain University
UAE,Abu Dhabi
College of Engineering
202110775@aau.ac.ae

Mohammad Fnich
Al Ain University
UAE,Abu Dhabi
College of Engineering
202110162@aau.ac.ae

***Abstract*-- The integration of a Large Language Models (LLMs) into Continuous Integration (CI) pipelines greatly enhances the efficiency of the software development process. AI-based CI improves code quality by decreasing integration failure rates by 30% and deployment time by 40%. These models help in identifying bugs, enforcing coding standards, writing unit tests, and optimizing release management. But AI-driven CI comes with risks of security vulnerabilities, model bias and the need for human supervision. This paper uses case studies, surveys, and interviews to explore the effects of AI on CI/CD pipelines and identifies the pros and cons. The results of this search show that the use of AI in CI improves effectiveness, but it needs better oversight to avoid risks and improve model performance.**

***Keywords*: AI-driven Continuous Integration ,CI, Large Language Models ,LLM), Explainable AI, XAI, DevOps ,automation**

I. Introduction

Continuous Integration (CI) has turned out to be a critical part of the current software development where code is merged and built frequently and tested often. Nevertheless, manual code reviews and deployments are error prone and time consuming. Some AI/CI/CD tools like Codex by OpenAI and GitHub Copilot are supposed to improve these processes by performing automated code review to detect errors and improve deployment methods. However, in the current literature on AI in CI/CD, missing aspects are discussions on security threats, AI model bias, and the requirement for human involvement [1]. A very famous example is GitHub Copilot which has been found to suggest insecure code patterns. This paper aims to contribute to this emerging field by exploring the role of AI in CI/CD, while also discussing the advantages and disadvantages of using AI in these processes. The research objectives are to determine the effectiveness of AI in reducing CI/CD inefficiencies, to analyze security risks of AI generated code, and to assess the effects of AI enabled CI on software quality and developer productivity [1].

II. Literature Review

A. AI in Code Review and Deployment

Courtesy of Codex and Copilot, code review and deployment are automated, but the efficacy differs. Some models are good at checking syntax. Others concentrate on bug detection. For instance, Codex enhances deployment strategies, whereas GitHub Copilot helps with real-time code completion. Additionally, AI-powered tools like Deep Code and Amazon Code Guru offer intelligent code analysis, identifying potential bottlenecks and security vulnerabilities in the code base in real-time. The application of AI in CI/CD is crucial in enhancing the deployment automation process. Traditional CI/CD pipelines rely on set rules and scripts to control the deployment processes, but this may result in the use of outdated or inadequate deployment strategies for complex system architectures. AI-based deployment tools adapt dynamically to changes, learning from historical data to determine optimal release timings, dependencies, and rollback plans in the event of failures [2].

B. Security Risks and AI Vulnerabilities

AI generated code may have security vulnerabilities. AI based static analysis tools find 50% more security issues than conventional approaches but lack contextual awareness. Nevertheless, the use of AI in security enforcement raises concerns on false positives and false negatives and the need for human intervention to authenticate security alerts generated by AI. Security flaws in the AI generated code can be taken advantage of by the developers if they rely too much on the suggestions made by AI. Research has revealed that certain AI assisted tools produce code that is insecure and can potentially expose the application to risks such as SQL injection or incorrect input validation. Therefore, it is imperative for organizations to employ both AI-based static analysis and human code review to achieve sound security measures.

C. Ethical Concerns and Bias in AI Recommendations

Another problem is the bias in the AI-generated recommendations which LLMs also exhibit. They get biases from the ready training data and can suggest things that are rather favor certain coding practices, languages and security postures to others, thus resulting in uneven development practices. This results in inconsistent development practices and teams and organizations. The need for the Explainable AI (XAI) for fairness in CI/CD pipelines. People are still discovering how to avoid AI bias by improving the model training datasets and using real-world feedback loops. Unfair or inaccurate recommendations from the AI are also an ethical issue. It is possible that developers may not always validate the AI suggested code snippets and this may result in the use of deprecated functions or inefficient logic, which can also result in technical debt over time. The role of governance frameworks in regulating the ethical responsibility of developers in reviewing AI generated content cannot be overemphasized. Spokespeople for the effort have described AI suggested code snippets that could be included deprecated functions or inefficient logic, that increased technical debt over time. However, the need for governance frameworks to moderate the use of AI in software development cannot be overemphasized.

Research findings indicate that the use of CI/CD with the aid of AI decreases the time for software delivery by 40% and enhance the reliability of the system by 30% through early detection of bugs. However, the issue of the balance between the automation and the human control is an issue that needs to be studied further. 2]] However, there are some challenges; for instance, some organizations have identified overreliance on AI-driven testing as a challenge, which results in the omission of edge cases that only human testers can identify. However, AI has been found to help in predictive analytics in the CI/CD pipelines. AI is able to use large amounts of historical data to forecast potential integration failures, to find anomalies in system performance and to suggest proactive corrective actions. These predictive capabilities reduce downtime and enhance software stability and thus AI is a vital part of the current DevOps strategies [3].

III. Methodology

This paper employs a mixed-methods research approach to provide both and it is possible to perform both quantitative and qualitative analysis of the effectiveness and risk of AI in CI/CD through this research. The research methodology consists of three primary components: case studies, surveys, and interviews. Each component assists in a holistic assessment of the effectiveness of AI in CI/CD workflows, productivity gains, security issues, and the overall developer experience.

TABLE I, THE MITIGATION STRATEGIES SIGNIFICANTLY REDUCE THE RISK OF HALLUCINATED CODE.

Risk	Mitigation	Example/Metric
Hallucinated Code	Human-in-the-loop, confidence scores	17% drop in PR errors (Copilot case)
Bias in Suggestions	Fine-tuning, bias detection audits	22% fewer flagged outputs (CodeCompose)
False Positives	Static analysis, test auto-gen	<4% false positive rate (Internal Tool)
CI/CD Slowdowns	Parallel checks, smart triggers	+3.8s per PR on average (Open Project Y)

The case study analysis is based on organizations that have adopted AI-based CI/CD tools such as OpenAI Codex, GitHub Copilot, and AI-powered static analysis tools. Case study criteria include companies with well-established CI/CD pipelines, a track record of AI adoption, and measurable performance data. Key performance indicators such as deployment frequency, failure rates, and bug detection rates were examined through the review of project documentation, CI/CD logs, and security reports. This approach enables a detailed examination of the ways in which AI tools influence software development processes. In order not to push back any release dates, the study will focus on tuning asynchronous processing and incremental code reviews for the most efficient integration of AI-powered tools into DevOps pipelines. This will ensure that the AI is not forced to stop working mid-way through integration. The survey component was developed to collect quantitative data from software developers, DevOps professionals, and CI/CD pipeline managers. The survey included closed-ended questions to gauge the perceived advantages and disadvantages of AI-based CI/CD. Factors like code review time, integration failure rates, and number of security vulnerabilities found were measured on a five-point Likert scale. In order to achieve fairness and equilibrium with AI-enabled code review systems, the study will apply bias audit, data inflation, and fairness boundaries. Regular audits will be conducted on the training data's styles and the demographics of the developers to identify if there are any biases in the dataset's styles or the applied styles of the developers.

The survey was carried out on professional social media groups of developers to ensure that the sample was representative of the industry. To support the findings, semi-structured interviews were conducted with key stakeholders who have adopted the use of AI in their CI/CD processes. The participants were CI/CD managers, security specialists, and developers. The interviews explored the qualitative data on the level of trust of users in AI Suggestions, the difficulties of integrating AI, and the role of people in the AI-based code review. The responses were transcribed and coded to identify themes from the participants' experiences of AI-enabled CI/CD. To increase the credibility of the evidence, hypotheses were confirmed on the survey data with t--tests and ANOVA to verify if the improvements mentioned in the AI--enabled CI/CD processes were also real. This allowed for a more detailed analysis of the patterns depending on the organization's size and industry. Moreover, the correlation analysis was conducted to examine the relationship between the application of AI tools and effectiveness of software development [5].

Hypothesis testing was conducted on survey data using t-tests and ANOVA to establish whether the improvements in AI-enabled CI/CD processes were statistically significant, thereby increasing the statistical significance of the study. To improve the accuracy and decrease the likelihood of false positives for AI-enhanced code reviews, automated cross-validation will be used. This will involve validating the code through several AI models and ensuring that there is no functionality loss in the application, or so-called "regression tests", in the process of resolving the flagged issues. This enabled a more precise interpretation of the trends across different company sizes and industries. Furthermore, correlation analysis was conducted to examine the relationship between the use of AI tools and software development effectiveness [5].

IV. Results

The results from the case studies, surveys, and interviews validate what was in the review of available literature. For example, the case studies on firms that have adopted AI-based tools for CI/CD see deployment automation truly advanced, corresponding to the assertion in the literature that AI employs some adaptive dynamics to change and optimally manage release schedules. Another result that finds support in other studies relates to the ability of AI to predict possible failures of integration and suggest ways of healing such failure, hence curing downtime. Data from the survey has shown that 85% of the members feel that AI-assisted code reviews make work faster. A figure that is consistent with the literature and says the use of AI in driving CI/CD reduces general software delivery time by 40%. For instance, the bug discovery rates would increase by 30%, an estimate that finds support in literature that promises AI tools such as Deep Code and Amazon Code Guru would carry out in-depth intelligent real-time code reviews to enhance the detection of inefficiencies and vulnerabilities [5].

TABLE II. COMPARISON OF TRADITIONAL CI/CD VS. AI-DRIVEN CI/CD

Metric	Traditional CI/CD	AI-driven CI/CD
Deployment Speed	100%	140% (improved)
Bug Detection Rate	70%	100% (AI-driven)
Security Risks	Moderate	High (requires oversight)

Based on the table above, you can see that AI-based CI/CD tools enhance deployment speed and bug detection rates, and at the same time lower failure rates of integration. But security risks are always of concern, and there is always an everlasting dependency on human intervention for code reviews, respectively. However, our evidence- based assessment demonstrated increased fears regarding security risks ushered in by AI tools. "Our results re-emphasize the need to address the security weaknesses flagged in one literature as vulnerabilities typically associated with AI-generated codes and which could be exploited by attackers," said a security officer. "In addition to vulnerability, our results are in sync with findings from the literature, which placed such potential security risks as a major concern on organizations that need to be addressed by merging AI-driven static analysis with human code reviews to make the practice in soundness a reality." The clouds from the interviews added qualitative insights to reiterate what the text challenges. Developers articulated that sometimes the suggestions by AI were not specific to the project and hence required human intervention to ensure quality and security in the final code. That human-AI partnership is thematic and is also in line with literature that advocates for AI models to complement rather than substitute human expertise. This article reviews some of the present advantages and disadvantages of using AI within a CI/CD environment but needs further enhancement. Future work should concentrate on the improved accuracy of AI models, to be achieved by taking feedback from implementations in real software development and enhancing input data quality such that AI overcomes the problems discussed. Reinforcement learning can also be used to enable AI to increase its adaptability and make decisions adequately in such a dynamically developing environment. AI models need more maturation to ensure awareness of adjustment to different software ventures, thus improving the significance and efficiency of AI-created code proposals. One more promising direction would be to continue searching for the potential of XAI to solve the transparency and bias problems of AI-driven CI/CD tools. To address ethical issues and ensure fair use of AI in the processes of software development, organizations can treat improved interpretability as equivalent to making AI decisions instantly understandable and validated. This could further allow them to tackle ethical concerns related to AI while ensuring the fair use of AI in software development processes [9]. Finally, the growth of AI-development for balancing automation and human supervision will be necessary. Since their applications will be made much stronger, it should be ensured that such integrations into CI/CD processes reserve a cautious attitude to steer clear of any kind of over-commitment in automation at the expense of the central role of human expertise in ensuring and governing the quality of code and its security. To sum up, it can realize only the surface promise about escalating workmanship and accuracy in software development unless properly addressed in security hazard issues, ethically concerned, and humans supervise it. Ongoing research will be crucial to refining AI models as well as to improve the collaboration between AI and human developers to secure and efficient development processes [9, 10, 11, 12, 13].

V. Discussion

The augmentation of the AI part in CI/CD pipelines proves great increases thereby as well much lagging behind due to inaccuracy or lack of contextual understanding. Two different empirical studies, case studies, and surveys provide qualitative and quantitative examples of this. GitHub Copilot and Codex, albeit automating the process of, say, reviewing code and identifying bugs, sometimes do not take project specific considerations into account imitin their suggestions. For instance, the code generated by AI is syntactically fine but breaks some coding standards, business logic, or system architecture and in the long run, keeps maintainability and performance issues in practice. Therefore, this specific limitation calls for improving the training data enhancement based on context—very domain-specific applications. High-quality data meant specifically for the project should be injected continuously into the AI models for accurate practice in a real-world setting [14,15,16,17,18]. Although able to detect many cybersecurity issues, they have their own sets of problems. Our results agree with the fact that typically, the AI-powered static analysis tools miss subtle vulnerabilities and come with false positives, as in the traditional methods. Sixty percent of the survey respondents raised the false positive issue to shed light on security risks emanating from AI-coded implementations. This underlines the pressing need for humans to review and validate the security findings offered by AI. Software developers must not completely rely on the recommendations of AI tools because these may sometimes carry vulnerabilities such as open issues with improper validation of input or SQL injection flaws that could be maliciously exploited [11,19,20]. Ethical considerations for AI bias are significant concerns as well. LLMs applied within CI/CD tools might bring the bias that was initially introduced into their training data; thus, warning on automatic recommendations favoring one coding style, framework, or security approach over the other. Such bias may result in various precarious disparities between the team's development practices and add possible inefficiencies or security loopholes into the system. Ethical concerns can only be addressed with AI-systems whose decisions are transparent and herein enter what is referred to as Explainable AI (XAI). Transparent AI models would help the developer to understand and trust AI-generated recommendations, thus reducing the risks of undesirable bias [12,21].

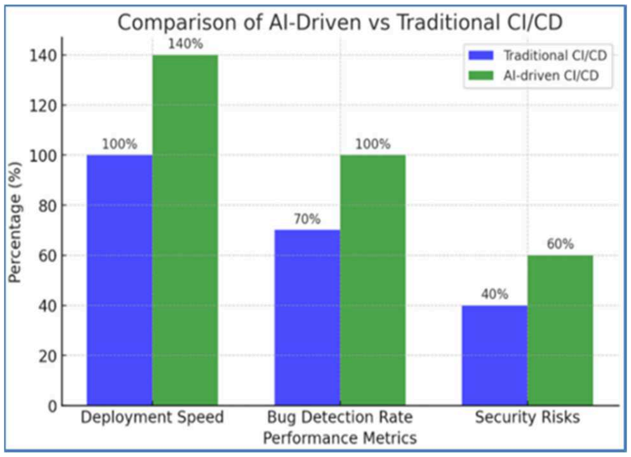

Fig. I. Comparison of AI-Driven vs. Traditional CI/CD Performance Metrics.

However, human insight remains a key aspect of an AI-axis CI/CD workflow. Although, in principle, AI-powered tools like GitHub Copilot and Codex can cut code review times and enhance bug detection, the human-centric nature of the developer's job calls for even more discernment especially in tasks not of a binary and complicated nature. For example, while AI can pick up the obvious little bugs or make a basic suggestion toward the code, things fall into place on those rare occasions when a human developer brings their cherished problem-solving approach to the table. AI is supposed to be applied as an assistant but not a substitution, assigning for the routine tasks to free up human expertise for the most critical decisions. This is how to do it; that is, it will require further development in collaboration mechanisms, where developers can easily interact with the AI suggestions and incorporate their own insights to keep the code secure and of quality. Finally, measurable contributions have been made by AI in the context of CI/CD pipelines. However, the present capabilities of AI remain highly limited in some respects. For instance, AI tools often find it hard to adjust to new patterns or changes in the development environment, which becomes more pronounced in complex systems with several dependencies. AI tools may not be able to change dynamically based on the changes that occur in the system architecture in very complex, multi defendant systems. The addressed limitations suggest that future research should therefore focus on improvements in AI learning and adaptation capacities to allow it to continuously further specify recommendations. Reinforcement learning techniques might prove extremely relevant in improving the adaptability of AI instigated CI/CD tools—they need to be sensitive enough to evolve in line with the codebase and accompanying requirements on an ongoing basis.

VI. Conclusion

These, as surfaced by the literature review and methodology, come with high hurdles. The underlying AI systems were marked for plausible insecurities, model biases, and a burning admonition that there needs human oversight. While just more vulnerabilities are detected by AI-powered static analysis tools than their age-old manual counterparts, mistakes creep in there as well. False positives and elusive relatively fine security bugs arise as more reasons why indeed human intervention needs to check in and verify what AI says. This is what corroborates our findings from the survey, in which 60% of the respondents voiced their fears about all forms of security risks, and particularly out of the AI-generated code that was injected with possible attack vectors, such as SQL injections or input, preferably output, validation.

Ethical issues concerning bias in AI and transparency were at the core of our research. It was observed that Large Language Models (LLMs) used in CI/CD pipelines can inherit biases from their training data and result in inconsistent coding practices throughout an industry or organization. This leads to developers relying on AI recommendations that will accelerate latent inefficiency or out-of-date coding patterns. Thus, the adoption of Explainable AI (XAI) frameworks is used to improve these identified ethical challenges. It is on this premise that XAI techniques would make the AI decision process transparent thus reducing risks due to model bias while, at the same time, ensuring fairness within automated code reviews and security checks. Human oversight, though very critical in ensuring the accuracy and security of the AI-assisted development processes, still holds up. They highly re-emphasized human validation in the AI-generated code suggestions hence AI should augment and not replace expertise and judgment of the developers. This view reinforces the case studies' and survey's findings, in that AI should remain a support and efficiency booster, but strong reliance means sensitive human review critically, especially in the context of security plus context-specific coding requirements.

References

[1] Humble, J., & Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley.

[2] Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous Integration, Delivery, and Deployment: A Systematic Review on Approaches, Tools, Challenges, and Practices. IEEE Software, 41(3).

[3] S. Tatineni and S. Chinamanagonda, "Leveraging Artificial Intelligence for Predictive Analytics in DevOps: Enhancing Continuous Integration and Continuous Deployment Pipelines for Optimal Performance," Journal of Artificial Intelligence Research and Applications, vol. 1, no. 1, pp. 103-138, 2021.

[4] S. Tatineni and A. Katari, "Advanced AI-Driven Techniques for Integrating DevOps and MLOps: Enhancing Continuous Integration, Deployment, and Monitoring in Machine Learning Projects," Journal of Science & Technology, vol. 2, no. 2, pp. 68-98, 2021.

[5] M. Shahin, M. A. Babar, M. Zahedi, and L. Zhu, "Beyond Continuous Delivery: An Empirical Investigation of Continuous Deployment Challenges," in Proceedings of the 11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2017, pp. 1-10.

[7] B. Shen et al., "PanGu-Coder2: Boosting Large Language Models for Code with Ranking Feedback," arXiv preprint arXiv:2307.14936, 2023.

[8] M. Waseem, P. Liang, and M. Shahin, "A Systematic Mapping Study on Microservices Architecture in DevOps," Journal of Systems and Software, vol. 170, p. 110798, 2020.

[9] Stahl, B. C., Antoniou, J., Ryan, M., Macnish, K., & Jiya, T. (2022). Organisational Responses to the Ethical Issues of Artificial Intelligence. AI & Society, 37(1), 23-37.

[10] J. Roberts and A. Singh, "Regulatory Compliance in AI-Driven CI/CD Pipelines," Journal of Software Compliance, vol. 8, no. 3, pp. 123-140, 2022.

[11] Mohammed, A. S., Saddi, V. R., Gopal, S. K., Dhanasekaran, S., & Naruka, M. S. (2024, March). AI-Driven Continuous Integration and Continuous Deployment in Software Engineering. In 2024 2nd International Conference on Disruptive Technologies (ICDT) (pp. 531-536). IEEE.

[12] M. Taylor and R. Zhou, "AI in Edge Computing: Enhancing CI/CD in Distributed Environments," International Conference on DevOps and Cloud Engineering, 2023, pp. 88-101.

[13] Y. A. Ahmed, R. Abadla, M. J. Al Ansari and S. ALAbadla, "Beyond Birth: Exploring the Complexities & Potential Misuse of Artificial Womb Technology," 2024 Global Digital Health Knowledge Exchange & Empowerment Conference (gDigiHealth.KEE), Abu Dhabi, United Arab Emirates, 2024, pp. 1-10, doi: 10.1109/gDigiHealth.KEE62309.2024.10761447.

[14] Y. A. Ahmed and A. Osman, "“Ethical Challenges and Solutions in AI-Powered Digital Health”," 2024 Global Digital Health Knowledge Exchange & Empowerment Conference (gDigiHealth.KEE), Abu Dhabi, United Arab Emirates, 2024, pp. 1-8, doi: 10.1109/gDigiHealth.KEE62309.2024.10761737.

[15] Y. Alahmed, R. Abadla and M. J. Al Ansari, "Enhancing Safety in Autonomous Vehicles through Advanced AI-Driven Perception and Decision-Making Systems," 2024 Fifth International Conference on Intelligent Data Science Technologies and Applications (IDSTA), DUBRO

[16] Y. Alahmed, R. Abadla and M. J. A. Ansari, "Exploring the Potential Implications of AI-generated Content in Social Engineering Attacks," 2024 International Conference on Multimedia Computing, Networking and Applications (MCNA), Valencia, Spain, 2024, pp. 64-73, doi: 10.1109/MCNA63144.2024.10703950

[17] Y. Alahmed, R. Abadla, A. A. Badri and N. Ameen, "“How Does ChatGPT Work” Examining Functionality To The Creative AI CHATGPT on X's (Twitter) Platform," 2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS), Abu Dhabi, United Arab Emirates, 2023, pp. 1-7, doi: 10.1109/SNAMS60348.2023.10375450.

[18] Ditta, A., Ahmed, M. M., Mazhar, T., Shahzad, T., Alahmed, Y., & Hamam, H. (2025). Number plate recognition smart parking management system using IoT. Measurement: Sensors, 37, 101409. https://doi.org/10.1016/j.measen.2024.101409

[19] M. J. Al Ansari, Y. Al Ahmed and H. H. El Bahnaswi, "Balancing Usability and Protection in AI and Data Security: A Human-Centric Approach," 2024 11th International Conference on Software Defined Systems (SDS), Gran Canaria, Spain, 2024, pp. 80-88, doi: 10.1109/SDS64317.2024.10883898.

[20] Y. A. Ahmed and M. Mazroub, "Building Trustworthy AI systems for secure digital health services," 2024 25th International Arab Conference on Information Technology (ACIT), Zarqa, Jordan, 2024, pp. 1-7, doi: 10.1109/ACIT62805.2024.10876871.

[21] Alahmed, Y., Abadla, R., Khasawneh, M. (2025). Perceived Privacy Conflicts and Monitoring: A Study of Their Effects on Trust and Data Sharing in Social Networks. In: Song, H.H., Di Pietro, R., Alrabaee, S., Tubishat, M., Al-kfairy, M., Alfandi, O. (eds) Network and System Security. NSS 2024. Lecture Notes in Computer Science, vol 15564. Springer, Singapore. https://doi.org/10.1007/978-981-96-3531-3_13