# AI-Driven Continuous Integration and Continuous Deployment in Software Engineering

**Conference Paper** · March 2024

DOI: 10.1109/ICDT61202.2024.10489475

Abdul Sajid Mohammed
University of the Cumberlands
**36** PUBLICATIONS **144** CITATIONS
SEE PROFILE

Santhosh Kumar Gopal
Cvs Health Corporation
**14** PUBLICATIONS **115** CITATIONS
SEE PROFILE

Dhanasekaran Selvaraj
Sri Eshwar College of Engineering
**44** PUBLICATIONS **707** CITATIONS
SEE PROFILE

# AI-Driven Continuous Integration and Continuous Deployment in Software Engineering

Abdul Sajid Mohammed,
School of Computer and Information SciencesUniversity of the Cumberlands, Kentucky, USA
amohammed5836@ucumberlands.edu

Venkata Ramana Saddi,
Technology Lead
Raleigh, NC, USA
ramana.saddi@outlook.com

Santhosh Kumar Gopal,
Technology Lead, Charlotte NC, USA
GopalSanthoshKumar@outlook.com

S.Dhanasekaran,
Department of ECE, Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu, INDIA
dhanasekaran.s@sece.ac.in

Mahaveer Singh Naruka,
G.L. Bajaj Institute of Management, Greater Noida, INDIA
ms.naruka@glbitm.ac.in

2024 2nd International Conference on Disruptive Technologies (ICDT) | 979-8-3503-7105-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICDT61202.2024.10489475

**Abstract— AI driven Continuous Integration and Continuous Deployment is a new way of managing and continually updating a software project. This process, powered by Artificial Intelligence, automates the entire software delivery and deployment process – from code submission to monitoring and bug fixing. It eliminates manual errors and allows for multiple versions to be tested in parallel, saving time and effort. By increasing agility, it allows organizations to launch new features to production faster than ever. Continuous Integration and Continuous Deployment leverages artificial intelligence in implementation and execution. It automates the process of integration, testing, packaging and deployment. Furthermore, AI is used to detect and fix bugs which can prevent delays and costly production bugs. AI driven Continuous Integration and Continuous Deployment has become an increasingly popular development strategy. It helps reduce the overall cost and accelerate the software's production cycles, making it easier for developers to quickly get their features and services in the hands of the market.**

***Keywords— Automation , Artificial Intelligence , Machine Learning , Adaptive Systems , Cloud Computing***

## I. Introduction

AI-driven continuous integration and continuous deployment (CI/CD) are becoming increasingly popular in the software engineering industry as they offer a powerful solution to the challenges posed by the ever-changing landscape of technology[1]. By combining the benefits of artificial intelligence with traditional continuous integration and deployment, teams are able to improve their software development processes more efficiently and effectively than ever before[2]. In its core, AI-driven CI/CD is about automating the development process from end-to-end, using machine learning algorithms to detect and alert you to problems that could cause delays or impede progress[3]. By doing so it significantly reduces the amount of manual work required by teams. For example, it frees engineers from having to manually check for code changes related to their commits, as AI-driven CI/CD Tools can now automatically identify and report such changes[4]. Teams are given a great advantage of being able to work faster and more precisely than ever before. In addition, Continuous Integration (CI) is an automated process that continually checks for software quality assurance[5]. As soon as a code commit is made, it's immediately tested to ensure that it works as expected and the existing codebase is intact. The introduction of Artificial Intelligence (AI) driven Continuous Integration (CI) and Continuous Deployment (CD) are two of the most significant advancements in software engineering in recent years. CI and CD are often referred to as the two arms of DevOps, and they are important in ensuring that software releases are reliable and timely[6]. Continuous Integration (CI) is a software development practice that is used to reduce the number of errors and promote collaborative development. It involves automating the process of integrating and validating code changes into the main branch of the source code. To achieve this, developers need to have a consistent programming practice. The construction diagram has shown in the following fig.1

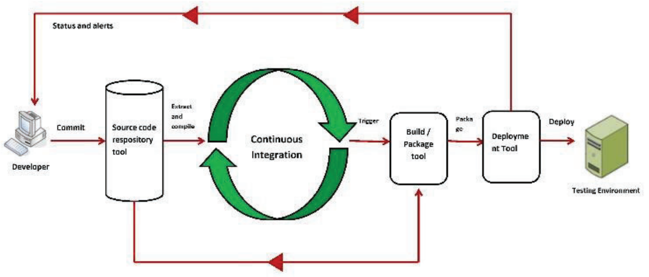

Fig. 1. Construction diagram

This includes coding mechanisms such as version control, automated build and test systems, and automated deployment systems. The aim of CI is to detect integration problems early on and resolve them quickly[7]. Continuous Deployment (CD) is a practice that promotes the automated and rapid delivery of software applications. It involves automating the deployment process, such as the creation of virtual machines, application provisioning, and database updates[8]. CD eliminates the potential for errors related to manual processes and allows the developer to focus more on enhancing the application. It is often seen as a way to reduce risk and accelerate the delivery of software applications. The main contribution of this paper has the following,

1. Automated Testing and QA: AI-driven Continuous Integration and Continuous Deployment allows for automation of various testing processes such as unit testing, integration testing, regression testing, and end-to-end testing. AI can integrate test results and analyze them in order to identify issues and suggest the most optimum solutions.

2. Source Code Versioning: Continuous Integration and Continuous Deployment provides an automated version

control system to easily track changes across different versions of the code. AI can help identify code changes that can break functionality and reduce risk of issues that may arise due to versioning.

3. Automated Deployment: AI-driven Continuous Integration and Continuous Deployment help in automated deployment of new versions of the application to multiple environments including staging, production, etc. This helps reduce the amount of time and effort taken to deploy changes and ensures a faster and smoother release process.

4. Dependency & Build Management: AI-driven CI/CD tools can help manage the various dependencies such as software components, libraries, templates, etc. and quickly identify any potential compatibility issues that may arise. This helps in ensuring that every build is compliant with the specified standards and requirements.

## II. Related Works

The advancements in artificial intelligence (AI) have enabled the automation of continuous integration (CI) and continuous deployment (CD) processes in software engineering[9]. This has progressed the field of software engineering dramatically as it improved the speed and accuracy by which software products are being developed and delivered. Moreover, by automating manual processes, more development teams are capable of maintaining the quality of their works without sacrificing velocity[10]. Despite the many benefits provided by AI-driven CI and CD, there are several issues that must be considered before adopting such automation. First, developers must determine the appropriate points in their development cycle that are most suitable for AI-driven automation. Determining these points requires careful consideration of the impact of automated processes on the development process and the accuracy of the output produced from those processes[11]. Second, AI-driven automation has the potential of introducing errors due to its reliance on rules and code conventions that are not necessarily accurate in every development context. For example, AI-driven linting tools may not be able to detect bugs and violations of coding conventions if they are not specifically told to do so in specific contexts. As such, developers must be vigilant in terms of determining the points at which automated processes should be used and in which ways they should be employed[12]. AI-driven continuous integration and continuous deployment is the process of integrating software development and testing with deployment, while taking advantage of artificial intelligence technologies. This process entails automating software builds and deployments, and allowing software development teams to incrementally release new versions of their product [13]. The technology aims to make the software development process more efficient and release deployments faster and with more predictability. While this can yield great benefits, there are some issues associated with this technology[14]. One of the main problems with AI-driven continuous integration and continuous deployment is that it is often too complicated for many users. As the software development environment is rapidly changing, this kind of technology is complex and hard to understand for those who have not had experience working with these technologies before [15]. This can create a barrier to the use of automation, thus decreasing its effectiveness. In addition, as this process requires increased attention to detail and proper configuration, problems may arise from user error when something is written incorrectly or configured incorrectly[16]. The main novelty of this research is to provide the optimal solution of the above mentioned problems. They are, AI-driven Continuous Integration and Continuous Deployment (CI/CD) are revolutionizing how software is developed and deployed. AI-driven CI/CD provides significant improvements in both speed and reliability [17. It automates the build, test, and deployment process in an agile environment, allowing developers to focus on code quality rather than worrying about manual steps, enabling shorter development cycles and faster response to user feedback[18]. AI-driven CI/CD systems use predictive analytics to proactively identify issues with code changes, suggest alternative approaches, and optimize the software delivery pipeline. Even more significantly, AI-driven CI/CD can run in production, ensuring that software can be quickly adapted to changing customer requirements and use cases[19]. This new approach to software development and deployment significantly increases the speed and effectiveness of software delivery, improving customer satisfaction and shortening release cycles.

## III. Proposed Model

AI-Driven Continuous Integration and Continuous Deployment in Software Engineering is the process of introducing automation into the software development life cycle. The use of AI-driven solutions is gaining traction as it increases the efficiency of the software development process by increasing the speed of changing code and ensuring that quality standards are met consistently. Continuous integration (CI) involves the merging of multiple changes into a single unit of code and then testing this unit as often as possible. The ability to integrate code from multiple developers is essential for efficient software development. AI-driven CI solutions are able to make complex decisions quickly, allowing developers to focus more of their time on coding. Continuous deployment (CD) is the process of automatically releasing software deployments whenever they reach a certain quality level. AI-driven solutions can detect the quality of code and pinpoint faulty code before it gets deployed. This helps reduce deployment cycles and software bugs, and speeds up the deployment time. AI-driven CD solutions can also be used to keep track of changes across different development environments and ensure that the software development process is well-managed and organized.

AI-driven continuous integration and continuous deployment (CI/CD) is a software engineering methodology that can help software developers create, test, and deploy applications faster and with greater efficiency. It involves automating the process of integrating code changes, testing and releasing software updates, and automating deployments to ensure that the latest changes are live in production as soon as possible. AI-driven CI/CD combines code repository monitoring, running automated tests, and automating the deployment process. This results in rapid and reliable updates to the existing application or its new versions. The code repository can be stored in a cloud platform such as GitHub and monitored by a continuous integration (CI) system. The CI system will track the code base and detect any changes or new features that added and automatically builds and tests the code. The build output is then used to automatically deploy the application to the test environment, test for errors, and then deploy the application to the production environment with no manual intervention. Automated tests typically involve automated unit tests, system-level / end-to-end tests, and performance tests. The functional block diagram has shown in the following fig.2

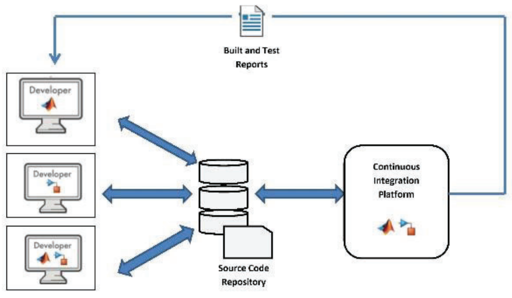

Fig. 2. Functional block diagram

AI-driven continuous integration and continuous deployment (CI/CD) is an application development practice that streamlines the workflow cycle and enables faster, automated delivery of software applications. By leveraging advanced AI, CI/CD is capable of rapidly introducing and deploying code changes, leading to high-quality and reliable production deployments of applications that can improve customer experience and satisfaction. CI/CD consists of two main components. Continuous integration (CI) is the practice of regularly merging code from multiple developers into a central repository, which runs automated tests on their code changes to identify any problems or issues. Continuous deployment (CD) is the automated deployment of changes, such as new features, bug fixes, and updates to a production environment. AI-driven CI/CD uses AI models such as machine learning and natural language processing to automate the workflow cycle. By predicting potential bugs and quality issues, AI models can streamline manual processes and pinpoint areas of risk before code is even committed to the repository. Additionally, AI can analyze changes and detect patterns that could lead to potential problems in the future. The operational flow diagram has shown in the following fig.3

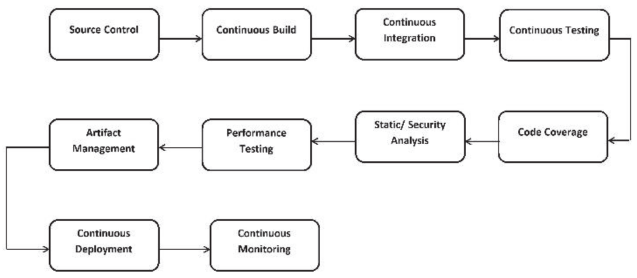

Fig. 3. Operational flow diagram

AI-Driven Continuous Integration and Continuous Deployment (CI/CD) are software engineering methodologies that enable developers to rapidly and predictably deploy software into production.

$$\partial P_1 = -\partial O + \sum_{o=1} \partial P_o = 0 \Rightarrow \frac{\partial P_o}{\partial O_p} = 1 \tag{1}$$

$$f'' = \lim_{e \to 0} \left( \frac{g^e * (g^f - 1)}{f} \right) \tag{2}$$

$$f'' = \lim_{e \to 0} \left( \frac{g(e+f) - g(e)}{f} \right) \tag{3}$$

The main principle behind CI/CD is to build on a continuously changing, automated codebase by repeatedly running unit tests, integration tests, and releasing new features to customers. In CI/CD very small increments of changes are made to applications in a short range of time. APIs are often used as an interface between stages. Normally, developers work on an individual feature, then perform a local build to verify that their contribution works. This build is then tested by unit tests to reduce development risks. Upon successful completion, the code is shared with the rest of the team through a source repository.

$$\partial P_2 = O + \sum_{o=1} \omega_o * P_o = 0 \Rightarrow \frac{\partial P_2}{\partial O_o} = 1 \tag{4}$$

$$\frac{\partial \ln(\omega_p)}{\partial O_p} + \psi_1 \frac{\partial P_1}{\partial O_p} + \psi_2 \frac{\partial P_2}{\partial O_p} = 0 \tag{5}$$

$$\ln(P_o) - \ln(O_p) + \psi_1 - \psi_2 O_p = 0 \tag{6}$$

The team can then immediately start to incorporate it into the main trunk of the application, alongside the changes made by other developers. After a successful build, an automated integration test suite is triggered. This verifies that no unexpected problems have been created by the integration of the changes. Finally, if successful, the changes are released in an automated fashion. Each update is automatically tested, deployed and monitored, ensuring an efficient, seamless process.

## IV. Results And Discussion

AI-driven continuous integration (CI) and continuous deployment (CD) are becoming increasingly popular in software engineering. This type of CI/CD leverages the power of artificial intelligence and machine learning to automate many of the process steps for CI/CD. AI-driven CI/CD replaces manual processes with automated ones, increasing the speed and reliability of system updates. In addition, AI-driven CI/CD helps to ensure compliance with industry standards and regulatory requirements. Performance analysis of AI-driven CI/CD involves examining the overall efficiency of an organization's CI/CD processes. The proposed model has been compared with the existing Natural Language Processing (NLP), Cuckoo Search Algorithm (CSA), Randomized Optimization Algorithms (ROA) and Particle Swarm Optimization (PSO).

### *A. Evaluation of Deployment time*

Different factors will be taken into consideration when evaluating performance, including the time it takes to execute a CI/CD operation, the number of times an operation has to be executed, the number of bugs or defects found in the system, and the number of defects that require manual intervention. Table 1 given the comparison of various algorithm for Deployment time.

TABLE I. COMPARISON OF DEPLOYMENT TIME (IN %)

| No.of rounds | NLP | CSA | ROA | PSO | Proposed |
|---|---|---|---|---|---|
| 200 | 88.90 | 86.24 | 83.77 | 84.44 | 93.30 |
| 400 | 89.87 | 87.24 | 85.37 | 86.28 | 94.08 |
| 600 | 90.51 | 87.79 | 90.16 | 87.16 | 94.70 |
| 800 | 91.55 | 89.64 | 91.17 | 87.83 | 95.97 |
| 1000 | 92.87 | 90.02 | 92.46 | 88.87 | 97.28 |

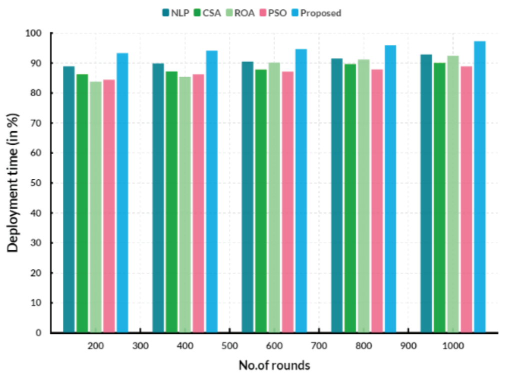

Fig. 4. Comparison of Deployment time

Overall, AI-driven CI/CD provides organizations with increased agility and reliability. It not only improves the overall speed and accuracy of system updates, but also reduces manual effort and errors. By ensuring that systems comply with industry standards, it also helps organizations to stay up-to-date with regulatory changes. In order to ensure that AI-driven CI/CD is efficient and effective for an organization, performance analysis and evaluation should be routinely performed.

## B. Evaluation of test execution time

Software engineering is a complex process that involves building new software or products to fit requirements for a company or organization. For successful software engineering, it is important to use the right tools and processes. AI-driven continuous integration and Continuous Deployment (CI/CD) is a popular and efficient way of managing software engineering project implementations. Continuous Integration (CI) allows teams to quickly try software implementations by breaking them down into small, manageable parts and integrating them together into the main software project. Table 2 given the comparison of various algorithm for test execution time.

This allows teams to continuously update software applications based on valid code and environment changes, which is critical for effective software engineering. Continuous Deployment (CD) refers to the process of automating the delivery of software applications to end users. This allows teams to quickly deliver new features and providing services quickly to customers, without the manual effort of deploying an application to a production environment. Performance optimization of AI-driven CI/CD is necessary to ensure that the software development process remains efficient and stable. Optimization involves continuous monitoring of CI/CD pipelines, ensuring that the proper events are triggered and proper actions are taken. Monitoring allows teams to identify issues at an early stage, reducing potential risk to the system. Performance optimization also involves using intelligent routing algorithms to route code changes and improve code quality. Additionally, automation should be used to reduce time consuming manual steps, improving overall efficiency of the software development process.

TABLE II. COMPARISON OF TEST EXECUTION TIME (IN %)

| No.of rounds | NLP | CSA | ROA | PSO | Proposed |
|---|---|---|---|---|---|
| 200 | 84.90 | 90.24 | 80.77 | 80.44 | 88.30 |
| 400 | 85.87 | 91.24 | 82.37 | 82.28 | 89.08 |
| 600 | 86.51 | 91.79 | 87.16 | 83.16 | 89.70 |
| 800 | 87.55 | 93.64 | 88.17 | 83.83 | 90.97 |
| 1000 | 88.87 | 94.02 | 89.46 | 84.87 | 92.28 |

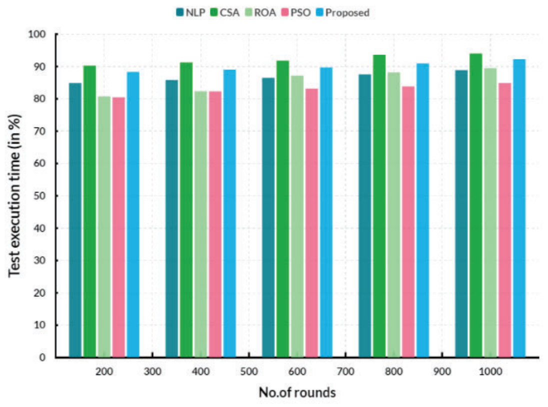

Fig. 5. Comparison of test execution time

## C. Evaluation of readability

Software engineering is a highly complex discipline and is composed of a variety of tasks. One common task is the continuous integration and deployment (CI/CD) of software changes. CI/CD automates the process of integrating and testing new software changes to an existing software package and deploying them to production servers. AI-driven CI/CD makes use of artificial intelligence (AI) to automate the entire CI/CD process, from monitoring the codebase and source code repository to creating builds, integrating changes and deploying the resulting artifacts. Table 3 given the comparison of various algorithm for readability.

TABLE III. COMPARISON OF READABILITY (IN %)

| No.of rounds | NLP | CSA | ROA | PSO | Proposed |
|---|---|---|---|---|---|
| 200 | 80.90 | 86.24 | 76.77 | 74.44 | 91.30 |
| 400 | 81.87 | 87.24 | 78.37 | 76.28 | 92.08 |
| 600 | 82.51 | 87.79 | 83.16 | 77.16 | 92.70 |
| 800 | 83.55 | 89.64 | 84.17 | 77.83 | 93.97 |
| 1000 | 84.87 | 90.02 | 85.46 | 78.87 | 95.28 |

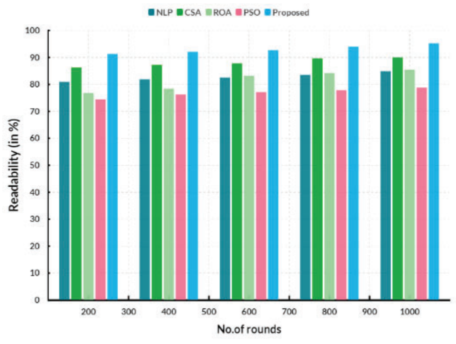

Fig. 6. Comparison of readability

AI-driven CI/CD can provide numerous benefits over traditional CI/CD, including faster delivery of software updates, improved quality of releases, and fewer manual processes. One common advantage of AI-driven CI/CD is that it can automate the entire process, making it more efficient and reducing the need for manual interventions. AI algorithms can scan the codebase and identify potential issues or changes that could break the application. This can lead to faster builds and more accurate quality control. Another advantage of AI-driven CI/CD is that it can drastically reduce the amount of time needed to perform the build and deploy process. AI-driven CI/CD can identify changes between versions of the codebase and trigger the build and deploy process automatically. This can lead to faster turnaround times, resulting in quicker product releases and improved customer satisfaction.

### D. Evaluation of code average

AI-driven continuous integration and continuous deployment in software engineering is an approach to improving the speed and accuracy of software releases. It combines the use of an artificial intelligence tool with a continuous integration and continuous deployment process to monitor software builds and deployments in order to anticipate and address any potential issues. By applying an artificial intelligence algorithm to the continuous integration and continuous deployment process, AI-driven CI/CD can identify potential development issues and problems quickly—even before they become noticeable to developers and testers. Table 4 given the comparison of various algorithm for code average.

TABLE IV. COMPARISON OF CODE AVERAGE (IN %)

| No.of rounds | NLP | CSA | ROA | PSO | Proposed |
|---|---|---|---|---|---|
| 200 | 73.90 | 77.24 | 73.77 | 68.44 | 79.30 |
| 400 | 74.87 | 78.24 | 75.37 | 70.28 | 80.08 |
| 600 | 75.51 | 78.79 | 80.16 | 71.16 | 80.70 |
| 800 | 76.55 | 80.64 | 81.17 | 71.83 | 81.97 |
| 1000 | 77.87 | 81.02 | 82.46 | 72.87 | 83.28 |

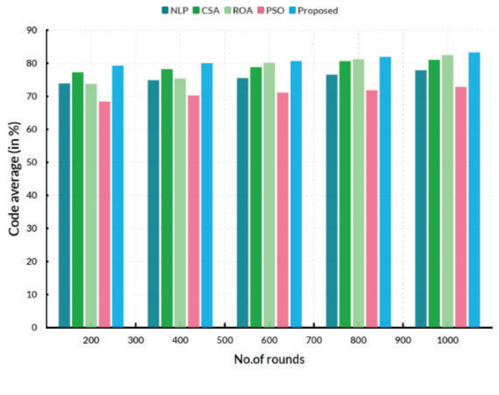

Fig. 7. Comparison of code average

It can also detect any code modifications or additions that could potentially affect the software's performance and alert those responsible for its development so they can take corrective action swiftly. This approach both reduces the time it takes for software to be released and boosts its overall quality by detecting potential errors earlier on. Additionally, by keeping track of changes and development in the software, AI-driven CI/CD makes it easier to identify any best practices or trends emerging within software engineering teams. This can help development teams understand how their work affects end-users and can help them incorporate those ideas into the final product.

## V. CONCLUSION

AI-driven Continuous Integration and Continuous Deployment (CI/CD) are important software engineering practices that offer a systematic way for companies to continuously deliver applications reliably. AI can be used in CI/CD to create automated pipelines that identify problems faster and give developers more control over the process. AI can identify unexpected bugs or problems in a build, detect changes to the environment, and recommend changes to a deployment plan in order to increase efficiency. AI can also help to automate the task of running tests and make better use of the resources available. By leveraging AI in CI/CD, software engineers can automate the process and increase the pace of software delivery. This reduces manual labor, speeds up delivery time, and reduces the risk associated with shipping a bad software product. Ultimately, AI-driven Continuous Integration and Continuous Deployment can enable companies to focus on developing applications with better results and fewer issues.

## REFERENCES

[1] Singh, A., & Singh, P. LEVERAGING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING IN SOFTWARE ENGINEERING: CURRENT TRENDS AND FUTURE DIRECTIONS.

[2] John, M. M., Olsson, H. H., & Bosch, J. (2023). Towards an AI‐driven business development framework: A multi‐case study. Journal of Software: Evolution and Process, 35(6), e2432.

[3] Fehlmann, T., & Kranich, E. (2023, August). Requirements Engineering for Cyber-Physical Products: Software Process Improvement for Intelligent Systems. In European Conference on Software Process Improvement (pp. 329-342). Cham: Springer Nature Switzerland.

[4] Amugongo, L. M., Kriebitz, A., Boch, A., & Lütge, C. (2023). Operationalising AI ethics through the agile software development lifecycle: a case study of AI-enabled mobile health applications. AI and Ethics, 1-18.

[5] Tamburri, D., & van den Heuvel, W. J. (2023). Big Data Engineering. In Data Science for Entrepreneurship: Principles and Methods for Data Engineering, Analytics, Entrepreneurship, and the Society (pp. 25-35). Cham: Springer International Publishing.

[6] Aldoseri, A., Al-Khalifa, K., & Hamouda, A. (2023). A Roadmap for Integrating Automation with Process Optimization for AI-powered Digital Transformation.

[7] Salama, R., Al-Turjman, F., Bordoloi, D., & Yadav, S. P. (2023, April). Wireless Sensor Networks and Green Networking for 6G communication-An Overview. In 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN) (pp. 830-834). IEEE.

[8] Dias, T., Batista, A., Maia, E., & Praça, I. (2023). TestLab: An Intelligent Automated Software Testing Framework. arXiv preprint arXiv:2306.03602.

[9] Chahed, H., Usman, M., Chatterjee, A., Bayram, F., Chaudhary, R., Brunstrom, A., ... & Kassler, A. (2023). AIDA—A holistic AI-driven networking and processing framework for industrial IoT applications. Internet of Things, 22, 100805.

[10] Lähteenmäki, J., Ahola, P., Baraian, A., Förger, K., Granlund, T., Hopia, J., ... & Torhola, M. (2023). Agile and Holistic Medical Software Development: Final report of AHMED project.

[11] Dittakavi, R. S. S. (2023). AI-Optimized Cost-Aware Design Strategies for Resource-Efficient Applications. Journal of Science & Technology, 4(1), 1-10.

[12] Whig, P., Jiwani, N., Gupta, K., Kouser, S., & Bhatia, A. B. (2023). Edge-AI, Machine-Learning, and Deep-Learning Approaches for Healthcare. In Edge-AI in Healthcare (pp. 31-44). CRC Press.

[13] J.Logeshwaran, & T.Kiruthiga. (2022). Interference-Resistant Communication framework for Sensor Nodes in Wireless Sensor Networks. International Journal of Research in Science & Engineering (IJRISE), 2(03), 61–75

[14] Ezhilarasi, K. K., & Rex, M. J. (2014). Reliable and energy-saving forwarding technique for wireless sensor networks using multipath routing. SSRG International Journal of Computer Science and Engineering, 1(9), 11-15.

[15] Pujahari, R. M., Yadav, S. P., & Khan, R. (2022). Intelligent farming system through weather forecast support and crop production. In Application of Machine Learning in Agriculture (pp. 113-130). Academic Press.

[16] Thamaraimanalan, T., Mohankumar, M., Dhanasekaran, S., & Anandakumar, H. (2021). Experimental analysis of intelligent vehicle monitoring system using Internet of Things (IoT). EAI Endorsed Transactions on Energy Web, 8(36).

[17] Sharif, M. H., Gupta, K., Mohammed, M. A., & Jiwani, N. (2022). Anomaly detection in time series using deep learning. International Journal of Engineering Applied Sciences and Technology, 7(6), 296-305.

[18] Salama, R., Al-Turjman, F., Chaudhary, P., & Yadav, S. P. (2023, April). (Benefits of Internet of Things (IoT) Applications in Health care-An Overview). In 2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN) (pp. 778-784). IEEE.

[19] J.Logeshwaran, & T.Kiruthiga. (2022). A Smart Server less Communication Model for Mobile Nodes in Cellular Networks. International Journal of Research in Science & Engineering (IJRISE), 2(04), 22–33