ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/srija-saha/249

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 2 & Iss: 1

Review

Improving Software Development Using AI Enabled Predictive Analytics

Srija Saha*

School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ, USA

Citation: Saha S. Improving Software Development Using AI Enabled Predictive Analytics. *J Artif Intell Mach Learn & Data Sci 2024*, 2(1), 1050-1053. DOI: doi.org/10.51219/JAIMLD/srija-saha/249

Received: 03 January, 2024; **Accepted:** 28 January, 2024; **Published:** 30 January, 2024

***Corresponding author:** Srija Saha, School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ, USA, E-mail: ssaha35@asu.edu

ABSTRACT

The incorporation of Artificial intelligence into software development is moving rapidly with predictive analytics playing a crucial role in better decision making, maximising resources, and increasing product quality. The use of AI powered predictive analytics, and several phases of the software development life cycle has gained considerable traction. However, there is still a lack of exploration in implementing a comprehensive strategy to its application. This paper provides a thorough investigation of the incorporation of Artificial intelligence (AI) across the whole software development process, including requirement analysis, design, coding, testing and maintenance. Using predictive models, artificial intelligence allows for the anticipation of the changes in requirements, discovery of design fault, detection of code defects, optimization of test coverage, prediction of maintenance needs. This paper discusses the comprehensive integration of AI across the whole project lifetime, leading to improved results by enhancing individual stages and creating synergetic benefits. The proposed solution aims to tackle key challenges such as data quality, model interpretability, and tool integration. These solutions include implementation of explainable AI approaches, adoption and strong governance practices. Concrete examples from real world scenarios showcase the practical advantages of using AI driven predictive analytics. These examples highlight the ability to reduce project timelines, enhance code quality, and raise fault detection rates. The importance of AI in software development is anticipated to increase as it continues to improve, presenting enhanced prospects for innovation and efficiency. This paper offers a roadmap for enterprises aiming to use AI to revolutionise the software development processes and achieve exceptional results.

1. Introduction

The evolution of software development processes has been characterised by the ongoing pursuit of tools and techniques that improve efficiency and speed. Conventional approaches like Waterfall and even Agile, often face challenges in meeting the requirements of constantly increasingly complex software systems and need for faster deployment. Consequently, there has been an increasing interest in using Artificial Intelligence (AI) to tackle these challenges. AI driven predictive analytics, using algorithms and machine learning models, has emerged as a promising method for forecasting future events based on previous data.

In recent years, much research has been carried out on the use of Artificial Intelligence (AI) in many phases of the software development process. Defect prediction has been a significant area of research, where studies have shown that machine learning models can effectively forecast software defects. This improves code quality and reduces debugging time[1,2]. AI has also been used in automated testing, demonstrating its ability to improve test case development and prioritising testing efforts[3,4]. AI also had a positive impact on project management, namely in the areas of resource allocation and risk management. Predictive analytics has been used to enhance decision making processes in these areas[5].

There has been an increasing desire in recent years to use AI extensively across the whole software development process. Aryyama and Srija[6] investigated how AI may increase efficiency in smart buildings by using AI powered controls systems. They

found that this approach led to substantial improvement in resource usage and operational efficiency. Their research focused on using AI to optimise financial aspects of electrical power networks, namely in project planning and risk management, demonstrated the adaptability and efficiency of AI in several fields[7]. Furthermore, the capacity of AI to analyse extensive quantities of data instantaneously has become a significant asset in a dynamic environment, where prompt and precise decision making is essential[8]. The capacity to handle the growing needs of modern communication networks is especially evident in network management since AI driven solutions are created for this purpose[9].

In addition, their research on enhancing the resilience and security of infrastructure using AI-powered threat detection and prevention emphasised the crucial role of AI in making smart grids more efficient[10]. By utilising Artificial Intelligence (AI) to identify potential risks in real time, previous research works have demonstrated the significant role that AI can play in preserving the stability and reliability of critical services such as PowerGrid which are continually susceptible to cyber-attacks and other forms of disturbances[11,12]. For sectors that depend on continuous operations, the capacity to proactively detect and address potential risks prior to their ability to cause substantial harm is a game changer.

Although there are significant improvements, the possibility of integrating AI across the whole software development process, from analysing requirements to maintenance, has not been fully explored. Prior studies have often concentrated on individual uses of AI such as network packet switching[13] or decentralised finance Defi security[14], without fully acknowledging the wider consequences and advantages of a comprehensive AI driven strategy across all stages of software development.

This paper aims to address this gap by presenting a complete methodology for incorporating AI driven predictive analytics into all aspects of software development. There uniqueness of this technique is comprehensive viewpoints, which considers the interdependencies between all phases of the software development lifecycle and utilises AI to enhance the whole process. This framework utilises predictive analytics across the whole software development life cycle including requirement analysing, planning, design, development, testing, deployment and maintenance. Its objective is to optimise decision making, boost resource utilisation, and minimise time to market, all while ensuring exceptional software quality.

This paper endeavours to make a scholarly contribution to the continuing academic discussions about the integration of artificial intelligence in software development. Its objective is to provide a complete methodology that explores the role of AI in refining software engineering techniques and achieving better project outcomes.

2. AI Integration in Different Stages of Software Development

The incorporation of Artificial Intelligence (AI) into the software development lifecycle has fundamentally transformed the processes of software design, development, testing and maintenance. The capacity of AI to evaluate extensive amounts of data, forecast subsequent outcomes, automate various operations and augment decision making processes has rendered it a vital tool in tackling the intricacies and requirements of modern software systems.

2.1. Requirement gathering

The phase of requirement analysis holds significant importance in ensuring the success of software systems projects. In the conventional approach, this stage involves collection, documentation and verification of the requirements and expectations of the relevant stakeholders. Nonetheless the procedure might be tedious and prone to errors due to ambiguity and constant evolution of the requirements. The integration of natural language processing (NLP) in requirement analysis can greatly augment the process by enabling the analysis and interpretation of various forms of communications such as requirement documents, emails and textual materials.

Artificial intelligence (AI) tools provide the capability to autonomously extract essential requirements, identify inconsistencies, and predict modifications based on historical data. This approach not only yields a reduction in time and effort spent on manual analysis, but also guarantees that the requirements are unambiguous, thorough and aligned to project objectives. Moreover, AI has the capability to rank requirements by considering variables such as feasibility, cost and impact. This empowers teams to concentrate on the most critical aspects of the project right from the beginning.

2.2. Design

The design phase includes the development of the software's architecture and formulation of its detailed design. This phase holds significant importance since it establishes the fundamental base for the entire development process. Incorporating Artificial Intelligence (AI) into the design phase can effectively augment decision making capabilities and optimise decision processes. One example of the machine learning models is their ability to analyse historical design patterns and outcomes to propose the most optimal design strategies for the project. The utilisation of Artificial Intelligence (AI) can also facilitate the identification of potential design flaws by comparing the proposed design with both successful and unsuccessful designs from past projects.

Furthermore, AI powered technologies have the capability to automate the generation of design artifacts including UML diagrams, flowcharts and architectural blueprints. This automation significantly reduces the time and effort needed to manually construct these crucial documents. The use of these technologies can effectively promote uniformity and precision in design, hence assisting teams in avoiding prevalent challenges and guaranteeing that the design aligns with established norms and industry benchmarks. By providing immediate feedback and ideas, AI empowered designers to examine various design choices and make well informed judgments that augment the overall quality and efficacy of the product.

2.3. Development

During the coding phase, software developers convert the design specifications into the executable code. AI can assume a crucial role during this phase by automotive repetitive coding processes, detecting and rectifying software defects, and enhancing code efficiency for improving performance. AI driven code generation systems can have the capability to aid developers by producing code snippets, proposing enhancements, and even composing whole functions based on developer input. These software applications employ machine learning algorithms that have been trained on extensive collections of code to offer suggestions that are contextually appropriate and adhere to establish coding conventions and best practices.

AI has the capability to improve the quality of code by proactively identifying and forecasting potential defects and vulnerability prior to their escalation into significant problems. Through analysis of code metrics and previous bug data, AI has the capability to detect code segments that are prone to errors and suggest corrective actions. The capacity to foresee future events allows developers to proactively identify and resolve errors at an early stage of the development processes and hence minimising the necessity for prolonged debugging and subsequent rework. Furthermore, AI has the capability to enhance code performance by suggesting more effective algorithms and data structures, thereby guaranteeing that the end output satisfies performance criteria while minimising resource usage.

2.4. Testing

The testing phase holds significant importance in the software development life cycle. As it verifies the proper functioning of the software and each adherence to establish quality standards. Conventional testing methodologies can be perceived as time intensive, especially for large and complex systems. AI has a potential to significantly transform the testing phase through the automation of test case generation, test case execution and result analysis. Iterative testing solutions powered by AI have the capability of autonomously producing test cases in accordance with the software's specifications and architecture hence guaranteeing thorough test coverage.

In addition, AI has the capability to prioritise test cases by considering the probability of identifying defects, to align testers to concentrate on the most crucial aspects of the software. These prioritisation techniques are especially valuable in the context of regression testing, as they are crucial for the recent modifications in code that have not created any further defect. AI has the capability to dynamically adjust test cases in real time, considering the behaviour of the program throughout the testing phase. This capability enhances precision and operational efficiency of the testing procedure.

Furthermore, AI has the potential to be employed in non-functional testing domains including performance testing, security testing and usability testing. AI enabled tools can provide the capability to replicate various scenarios and user behaviours to evaluate the effectiveness of software under different conditions. By doing an analysis of the results, AI can detect any bottlenecks, security vulnerabilities and usability concerns. This enables teams to gain valuable insights that can be utilised to enhance the product prior to the release.

2.5. Deployment

The deployment step involves releasing the software into the production environment. This stage is frequently beset with challenges such as compatibility issues, performance limitations and unforeseen errors. Automation of deployment tasks, anticipation of potential challenges, and facilitation of seamless transition from development to production are among the ways in which AI can enhance the efficiency of the deployment process.

AI powered deployment technologies can automate the configuration, integrations and deployment of software in various environments. This can mitigate potential human errors and expedite the deployment process. In addition, these technologies can constantly monitor the operation of the software during its deployment, thereby promptly identifying and resolving any issues that may arise. The use of AI enables the anticipation of future compatibility concerns by analysing the software configurations and target environment. This empowers teams to proactively resolve these issues prior to their adverse effects on end user experience.

In addition, AI can enhance resource allocation performance throughout the deployment process, thereby ensuring the optimal efficiency of the software on a designated infrastructure. Through the examination of past deployment data and real time metrics, AI can provide recommendations for the most efficient software configuration, thereby minimising the resource usage and enhancing overall performance. Implementing this proactive strategy for deployment management enables teams to consistently produce software of superior quality that aligns with user expectations and operates reliably in the production environment.

2.6. Maintenance

The maintenance phase is a continuous process that encompasses the routine upgrading of software and enhancements of software after its first release. This phase includes tasks such as bug fixes, improving system efficiency and incorporating novel functionalities. Advanced Artificial Intelligence has the potential to significantly contribute to maintenance operations through its ability to forecast future maintenance requirements, automate repetitive procedures and uphold software security and integrity.

AI has the capability to monitor system logs, user comments, and performance indicators to proactively detect potential issues before they become critical. As an illustration AI can forecast the probable failure of a component by analysing its usage pattern and historical data. This enables the team to take proactive measures in addressing the issue. This use of this predictive maintenance strategy effectively reduces downtimes and guarantees software reliability and accessibility for users.

Furthermore, AI can automate several routine maintenance operations, including but limited to patch application, dependency updates, and system health monitoring. Through the automation of these processes, AI liberates engineers from time constraints, enabling them to allocate their efforts towards more intricate and value-added activities. Advanced Artificial Intelligence can maintain the security of software by constantly checking potential vulnerabilities and promptly implementing necessary security updates.

3. Limitations

Although the incorporation of AI into the software development lifecycle offers notable benefits, it is imperative to recognize and address several constraints associated with this integration. A significant constraint is in the reliance on good quality data. The efficacy of AI models is contingent on the availability of comprehensive and precise datasets, as any inadequacies in data quality might result in erroneous predictions and inferior results. Furthermore, the challenge of interpretability persists in AI models particularly in the case of complex machine learning algorithms such as deep learning. These algorithms often function as black boxes, posing difficulties for developers in comprehending the underlying rationale behind specific prediction or judgements. Integrating AI tools into pre-existing workflows can pose challenges, as it entails major changes to development procedures and necessitates extensive training for the development teams. Moreover, the dependence

on AI may bring about novel hazards, such as the possibility of over dependence on automated systems, which could result in complacency and a decline in critical thinking among developers. Furthermore, the expenses associated with deploying and upkeeping AI-powered systems might be a significant obstacle for smaller firms, hence restricting their ability to readily use these sophisticated technologies. The constraints underscore the necessity for meticulous deliberation and equitable execution when incorporating AI into software development procedures.

4. Conclusion

The incorporation of Artificial Intelligence (AI) into the software development lifecycle signifies a transformative advancement in the conceptualization, construction, and upkeep of software. Through the utilisation of artificial intelligence (AI) throughout many phases, including requirement analysis, design, coding, testing, deployment, and maintenance, businesses can attain enhanced operational effectiveness, enhance decision-making processes, and attain superior software product quality. Artificial intelligence (AI) has demonstrated its immense value in effectively managing the increasing complexity of contemporary software systems by automating ordinary operations, predicting potential risks, and optimising procedures. This paper presents a complete framework for integrating AI, which showcases the significant impact of AI. It provides a clear plan for enterprises seeking to improve their development procedures and maintain competitiveness in a rapidly evolving digital landscape.

Nevertheless, the completion of incorporating AI into software development is not devoid of challenges. It is imperative to effectively address the constraints related to data quality, model interpretability, integration complexity, and cost to fully harness the advantages offered by artificial intelligence. It is imperative for organisations to adopt an in-depth understanding of the issues associated with AI integration and demonstrate a commitment to ongoing learning and adaptation. To ensure responsible and effective use of AI, it is imperative for the software development community to remain well-informed regarding the newest breakthroughs and best practices in AI technologies as they continue to progress. Artificial intelligence (AI) is anticipated to assume a more prominent position in the future of software development. This advancement is expected to foster innovation and facilitate the development of intelligent and adaptable software systems that effectively cater to the ever-changing requirements of both consumers and enterprises.

5. References

1. Hall T, Beecham S, Bowes D, et al. A systematic literature review on fault prediction performance in software engineering. IEEE Transactions on Software Engineering, 2011; 38: 1276-1304.
2. Radjenović D, Heričko M, Torkar R, et al. Software fault prediction metrics: A systematic literature review. Information and software technology, 2013; 55: 1397-1418.
3. Zeller A. Why programs fail: a guide to systematic debugging. Morgan Kaufmann. 2009.
4. Kaur M, Kumari R. Comparative study of automated testing tools: Testcomplete and quicktest pro. International Journal of Computer Applications, 2011; 24: 1-7.
5. Jørgensen M. Forecasting of software development work effort: Evidence on expert judgement and formal models. International Journal of Forecasting, 2007; 23: 449-462.
6. Jana AK, Saha S. Improving energy efficiency in smart buildings with AI powered control systems. European Journal of Advances in Engineering and Technology, 2019; 6: 49-53.
7. Jana AK, Saha S. Financial optimization in electrical power systems using Artificial Intelligence. European Journal of Advances in Engineering and Technology, 2020; 7: 97-102.
8. Davenport TH. From analytics to artificial intelligence. Journal of Business Analytics, 2018; 1: 73-80.
9. Arsénio A, Serra H, Francisco R, et al. Internet of intelligent things: Bringing artificial intelligence into things and communication networks. In: Inter-cooperative collective intelligence: Techniques and applications, 2014; 1-37.
10. Jana AK, Saha S. Making Smart Grids Robust using Artificial Intelligence for Threat Identification and Mitigation. European Journal of Advances in Engineering and Technology, 2020; 7: 71-77.
11. Arghandeh R, Von Meier A, Mehrmanesh L, et al. On the definition of cyber-physical resilience in power systems. Renewable and Sustainable Energy Reviews, 2016; 58: 1060-1069.
12. Xu L, Guo Q, Sheng Y, et al. On the resilience of modern power systems: A comprehensive review from the cyber-physical perspective. Renewable and Sustainable Energy Reviews, 2021; 152:111642.
13. Jana AK, Saha S. AI-Powered Network Packet Switching-A Way Forward for Future-Ready Communication Systems. European Journal of Advances in Engineering and Technology, 2021; 8: 37-41.
14. Jana AK, Saha S. Natural Language Processing and Artificial intelligence to guarantee security in Decentralized Finance (DeFi). European Journal of Advances in Engineering and Technology, 2021; 8: 58-63.