Crossover of Artificial Intelligence and Software Development Lifecycle

Amna Raza Khan[1], Javeria Fatima[2], Sumaira Ahmed[3]
Center of Computing and Research, Dept of Computer Science and Software Engineer JUW, Karachi, 74600, Pakistan
E-mail: **amnarazakhan02@gmail.com, javeriafatima438@gmail.com, sumaira.ahmed@juw.edu.pk**

Received: 19th April 2023; Accepted: 14th June 2023; Published: 1st July 2023

***Abstract*:** Artificial intelligence (AI) is a multidimensional technology that has the potential to revolutionize our environment by combining numerous algorithms and techniques to generate diverse results based on user-provided information. AI is redefining the meaning of life and human existence by offering innovative solutions to age-old problems. In this article, we have discussed the various ways in which AI can be utilized across the software development lifecycle (SDLC) and its different phases. By incorporating AI into the SDLC, we can automate many of the procedures, spending less time on development and more time on problem-solving. AI has many new emerging fields which can help us efficiently work on the different problems faced in SDLC. Neural networks, Natural processing Language, Image generation using different algorithms, using Ai in quality assurances and testing. Building integrated development environment on the basis of AI and infusing the rules of the said technology. This will lead to a faster-paced world and increase the trust of non-IT individuals in this concept, resulting in widespread adoption of AI-based systems. The development of a system that provides the foundation for automated coding and development will allow non-IT individuals to create custom applications and fulfill their own development needs. This will propel us forward to solve future problems, and it is already underway with the production of Low Code No Code apps.

Keywords: Software development, Artificial intelligence, software development lifecycle, Low code no code.

1. INTRODUCTION

Artificial intelligence primarily concerns the area in which we desire computers to function similarly to humans. Everything in our period is being developed with the conception of AI in mind. AI has permeated countless fields, including the software development cycle and software engineering. This study seeks to connect the notion of employing AI for software development. Scientists and academics anticipate that robots will take over every current job, including software engineers, and that AI will enable computers to write code and solve problems. However, it is still a work in progress. Scientists believe that using AI for the software development lifecycle can be challenging because of trust issues. Although, the significant difficulties faced are planning periods, stakeholders' engagement, debugging, human error, and testing. And this is where AI comes to the rescue. AI can increase the speed of coding, being precise and efficient for the entire SDLC. Tang [1] experimented and stated that by using neural networking for the analysis of any problem, we can improve the software development by 20%. Through this, the stakeholders can focus on the application and software design and features. The application of AI in multiple industries for the creation of various software can minimize repetition in code, remove bugs quicker, reduce human errors, reduce overall cost, and time, and manage data. As a result, there is a demand for the automation of these processes. AI technology will give easy answers and a rapid path to innovation in the relevant industry and since there is a need for custom applications, not everyone has the time to go to the developer and get the changes done and updated. AI has opened a vast opportunity to integrate many new algorithm and technologies which include neural networks for prediction and analysis for any error or bugs, while NLP can be used for requirement analysis and to reduce the gaps between communications. Prediction models are being developed for forecast telling for different approaches.

This study is motivated by the desire to address the previously mentioned challenges by shedding light on the rise and increasing sophistication of low-code/no-code (LCNC) development alternatives, such as Microsoft power applications, which bring the power of application development to users across the enterprise. Low Code helps the developers reduce the time and the lines of code while the no code part is helping the non IT personnel with no prior coding knowledge to customize their apps according to their needs. The inclusion of more people in decision-making and software development using AI, as well as to enhance their involvement and trust in AI, to encourage improved creativity and minimize the load on IT departments. There are many pitfalls in the SDLC. As a result, system needs are misunderstood, and the ultimate result is unavoidably disappointing. Furthermore, the SDLC's complexity frequently steers a protrusion to derail or teams to lose sight of specifications and needs. A project might easily fall short if all specifications and design goals are not rigorously followed. It requires immense human labor, which makes it prone to mistakes and delays, resulting in inefficiency. Bhagyashree W. Sorte, [2] discussed the use of Artificial intelligence, Expert System, and Knowledge Engineering for the automation of the numerous processes in the development phase.

Low code technology has the potential to be a catalyst for company innovation and business agility by combining technology and culture. It is more than just a tool for increasing productivity [3]. Low-code development systems emphasise visual interfaces to allow users with no technical knowledge to easily construct and deploy commercial programmes [4]. As the pace of digital transformation quickens, there will be an even greater need to rely on non-developer staff to manage business applications without contacting IT teams. Using a low-code platform makes empowering the citizen developer much easier to implement due to its simplicity and quickness. Low-code development is a method that allows people who lack extensive computer or coding abilities to create unique apps and digital solutions.

2. LITERATURE REVIEW

Sabeer Saeed and Asif Varol, [5] proposed a novel approach for the verification and validation of autonomous vehicles for the market. So that the public will be able to access high-quality AI-driven vehicles. Robert Feldt [6] provided a basis for software engineers to consider the risk of implementing AI. The risk analysis and development of a strategy for its adoption in various disciplines need to be studied. AI plays an important role in the evolution of Software Engineering. Shyamal, [7] claimed that there is a line between ML and software engineering which has opened a huge gate for future studies. The evolution of knowledge-based systems for learning software architecture, as well as the development of computational intelligence, can result in a wide range of advancements and study topics important to many professions. There are many imitations and potential risks of applying AI at the six stages of the software development lifecycle. This study was concluded by Marco [8]. AI assists both game developers and linguists in the creation of AI translation algorithms and computer languages. A systematic literature review on software engineering challenges, processes, methods, approaches, and tools to address the problems and lack of knowledge about SE methods applied in AI/ML development in the industry conducted by Elizamary [9] in the context of the development of AI/ML systems. This study aimed to analyze the gap between the challenges that programmers are confronting and their different solutions accessible. Different software agents and knowledge-based systems were discussed for the betterment of the SDLC by the author Jorg Rech [10]. Hazrina Sofian [11], through their research, concluded that machine learning can be used to automate processes in different software engineering steps using a variety of algorithms.

3. METHODOLOGY

Software engineering involves knowledge-intensive processes that revolve around human beings, while artificial intelligence (AI) leverages the computational intelligence of computers. By utilizing self-optimizing algorithms, AI can enhance software engineering processes by optimizing occupation. AI, which refers to machine-based discernment, thinking, and understanding of natural and mental constructs, has the potential to improve various software engineering techniques, including analysis, design, assessment, implementation, testing, maintenance, and reengineering of software.

AI can be used to plan software development, which can improve productivity and efficiency in the software development life cycle (SDLC). AI-powered testing reduces the chances of errors and bugs, as well as the burden on testers, while enhancing the overall development process. In different software engineering environments, we are witnessing AI executing the job of programmers by utilizing various algorithms and techniques such as natural language

processing and machine learning, which can enhance the productivity of developers' work. AI provides AI-based assistants for different types of testing, enabling programmers to code in any language through testing software that is based on AI. These machines are trained and can learn on their own. AI is also used for code compilation, integration, diagnosis, and testing. Machine learning is utilized to create tools that reduce code typing and enhance production efficiency. Additionally, AI assists with cost analysis by analyzing historical data and providing cost estimates based on this information [8]. Based on this paradigm, AI tools can forecast future outcomes and challenges, improving decision-making and supporting businesses in making swift changes.

AI is also helping in decision-making for the planning process. Developers can use AI and ML techniques to probe and develop appropriate decision-making processes that aid in making sound decisions. However, while AI and ML have the potential to transform the process of making strategic decisions, these algorithms may not be flawless and may struggle to adapt to changing programming environments. AI algorithms can be used to analyze historical data on software development projects and provide insights that can aid in project planning. By using machine learning algorithms, AI can identify patterns in data that can help predict potential challenges and estimate timelines and resource requirements more accurately. AI can assist with software design by providing recommendations on code architecture, design patterns, and other best practices. This can save time and improve the quality of the final product. AI can assist with software design by providing recommendations on code architecture, design patterns, and other best practices. This can save time and improve the quality of the final product. AI can assist with software design by providing recommendations on code architecture, design patterns, and other best practices. This can save time and improve the quality of the final product. One area where AI can bring significant innovation to software engineering is in the area of software maintenance. Maintenance is a critical part of the software development life cycle that involves modifying, updating, and improving software to meet changing user needs and requirements. AI-powered maintenance tools can help reduce the time and effort required to maintain software systems while improving their quality and reliability. For instance, AI algorithms can be used to analyze user feedback and identify patterns and trends that indicate common issues or bugs in the software. This data can then be used to automatically generate bug reports and suggest fixes, reducing the time and effort required to identify and resolve issues. AI can also be used to automate routine maintenance tasks such as software updates and backups, freeing up developers' time to focus on more complex tasks [12]. Machine learning algorithms can learn from past maintenance tasks and apply this knowledge to future tasks, improving their efficiency and effectiveness. Every methodology in SDLC have its own benefits and drawbacks. Integrating artificial intelligence and its different aspects can help us to reach our desired outcome, Agile, Waterfall, Scrum, Lean all have their advantages and their comparative analysis on the basis of their requirements and procedures tends to help but in low code no code, none of these are followed. The comparative analysis of LCNC would be on the basis of different platforms and software which provide an interface and application to build desired applications on the basis of requirements, their cost and development time. Both of the SDLC and LCNC can be integrated and are being integrated with AI to make efficient changes and give more ease of access to the developers and users. In the next section, we have discussed how by using artificial intelligence we can streamline the software development lifecycle.

3.1 Artificial Intelligence in Software Development Lifecycle

The purpose of AI in the initial phase of the SDLC can be adept through an expert system that may be used to anticipate concerns in the planning stage. The ability of AI to analyze data can provide rapid insights into project parameters. It enables project managers to make decisions based on past data. Continuing, in the second phase of the SDLC process AI can be implemented to produce automated designs and UML diagrams. AI can also provide different aspects for the implementation of the blueprint. Since AI is an excellent tool for prediction analysis and drawing patterns from a huge amount of data, it can create an amazing model. The machine can be trained to be a better learner. In the coding phase of the SDLC, through automated methods, AI can assist with self-adaptive routines and automated code integration. These assistants may be able to review the data provided by the user to tailor a software design and code based on the preferences. Testing is the most popular phase which is automated through Artificial intelligence. Automated test routines, code diagnosis, and implementation helps the developers with efficient SDLC [13]. Software maintenance is another process that is well-automated with the help of AI. Predictive maintenance software revolves enormous volumes of data into usable insights and data points using artificial intelligence to sustain in evading data overload.

3.2. Automated software development through Low-Code No Code

The low code part of this strategy is for the developers [14]Alamin et al. It helps the developer to reduce the time

required to code. It also helps the software engineer to reduce the lines and codes. This strategy helps with the reusing of different code which is already written and executed. This helps the developer to save time and can efficiently move to the part where he needs to apply his knowledge. This is intelligent for enabling people across an enterprise to customize and build their software and application. Through this, many organizations can improve the burden of their IT staff. This strategy can be used by the staff to make the application tailored according to their needs. It is possible by drag and drop capabilities and visual coaching of the individuals. It is a clever method to construct an enterprise-worth business application with little to no coding expertise. LCNC improves the lifecycle of software development (SDLC) by drastically minimizing the number of stakeholders throughout the process, increasing speed and productivity. Figure 1 shows the comparison of the typical agile methodology and the low code no code development.

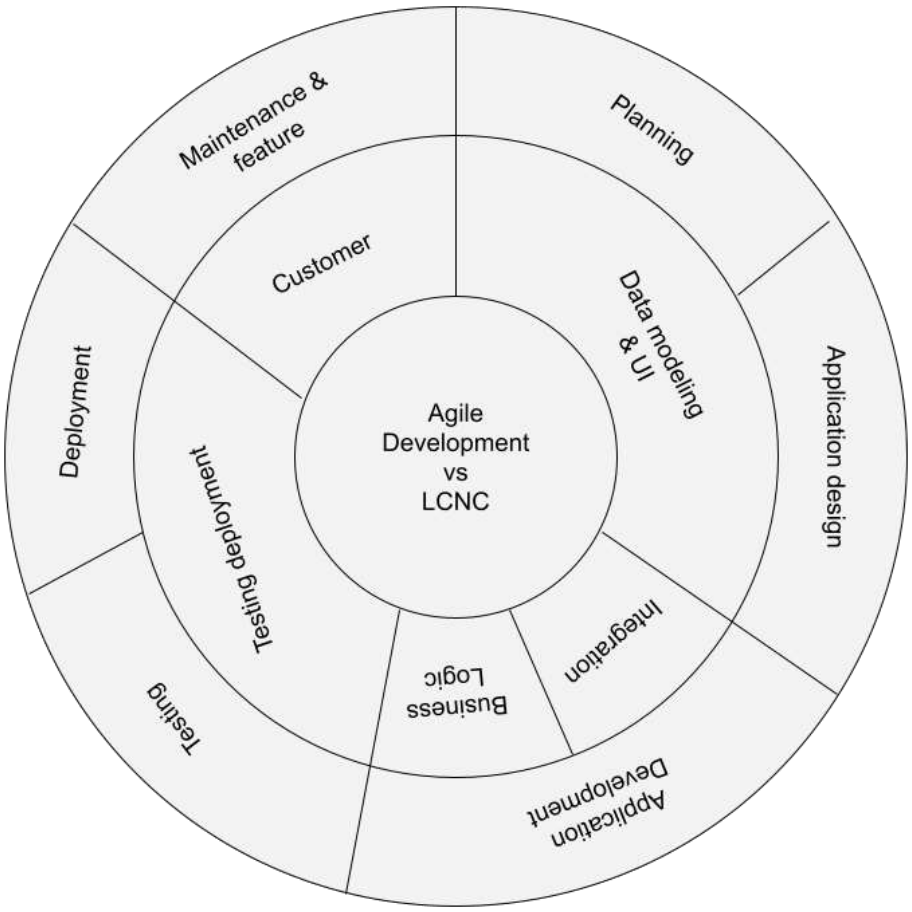

Figure 1- Agile development vs Low Code/No Code Development

Elshan et al. [13] stated on the challenges and answers to the LCNC development. The comparative analysis of agile methodology and the low code no code methodology are done below. On the different phases of the Agile and how different it is from the usual low code/no code methodology. Table 1 shows how LCNC is different from the agile methodology. After all the requirement gathering and feasibility analysis this process is started and all the stages and explained.

Traditional Agile Methodologies	Typical Low-Code/No-Code Software Development lifecycle	Description
Planning	Data Modeling/ UI design	The team uses the platform tool to design or model the database of the application. The data model serves as a base for the application. Similarly, there are many no code solutions for UI design where drag and drop feature helps the non-technical staff. They may use the platform's coding skills to construct personalized components.
Application design		
Application development	Integration of external services	For extra functionality the integration of APIs and external services will help with the development of such applications. And if the required functionality is not available, the team may need to code it.
	Business logic implementation	Creation of workflows, decision trees and sets of rules on the basis of how the application manages and process data. Typical low-code no-code platforms provide drag and drop options.
Testing	Testing and deployment	This is the same as the traditional Agile methodology.
Deployment		
Maintenance & additional features	Customer features / additional features	This includes the customer feedback and how additional features and techniques will help the application and the users in the future and present.

Table 1 Traditonal Agile Methodology versus LCNC software development

Low-code development does have a substantial positive influence on organizations, particularly those in the process of digitization. Various members of the organization can work together. In the manual development only the pro members who have the expertise to develop a software can work whereas in the LNCN even the non-IT folks can build an application based on their need. In the manual development, the developer follows a different model which helps throughout the process, but the main problem with it is that the specification which is defined by the client might be understood differently by the developer. There may be gaps between the final product which is showcased by the developer and what the client defined as the requirements. Even in the more modern models such as agile, there may still be a gap and the development process may need much longer time then apprehended. With Low Code No Code apps, a person can easily develop applications without the need of a developer with expertise knowledge. Users get to use the software early in the development process, and if new requirements arise, new features can be added to the following edition.

4. DISCUSSION

The ability of AI to evaluate data can give instant insights into project components in the planning stage, enabling project supervisors to reach educated decisions according to previous experience. Using NLP, we can generate more accurate and contact aware code for the LCNC platforms which can be immensely beneficial for the developer and the team. NLP can also streamline all the requirement gathering for code creation in LCNC. Furthermore, LCNC platforms may provide a more straightforward and user-friendly interface with which NLP models may interact, making it simpler for NLP algorithms to create more precise and perspective-aware code. Automated systems will swiftly analyses input in

natural languages from individuals to develop software tailored to their specific requirements. The most famous AI approaches, such as artificial intelligence (AI), natural language processing (NLP), and automation/robotics, may be utilized to aid with developing software chores, replicating behavioral patterns and completing jobs in the best way feasible. Neural networks in AI can help to find different problems and bugs in the process in both Traditional methodologies and the LCNC methodologies. Users can face problems in any step and bugs can be faced. Neural networks are a great way to integrate AI and solve tons of problems. Furthermore, by integrating data mining techniques and different algorithms for dashboard analysis and easier generation of the application based on the software is what the future holds.

5. CONCLUSION AND FUTURE WORK

The combination of previous research, the software engineering process, and AI crossover in this work will assist developers to achieve the desired product in time expeditiously. AI and LCNC platforms have the potential to revolutionize SDLC by shortening development timelines, enhancing code correctness and context-awareness, and democratizing access to AI and machine learning. However, there are drawbacks to employing respective systems, such as shadow IT and vendor lock-in. Overall, the benefits of adopting AI and LCNC platforms exceed the drawbacks, and their adoption has the potential to alter the SDLC. NLP derived LCNC platforms can improve by reducing the efforts and time needed for requirements translations. It can also be used to improve the user experience of the non-technical users who use the LCNC platforms. However, it is important to note that this field of study is still in its early stages, and there are challenges to overcome, such as ensuring that the artificial intelligence models are developed on diverse and representative data to prevent bias, and facilitating that the LCNC tools can handle the complexity of natural language input.

As a result, further research and development will be necessary to fully grasp the potential benefits of this combination concentrate on the development of AI, SDLC, and LCNC platforms to guarantee that they remain current and relevant in a quickly changing technological context.

REFERENCES

[1] Y. Tang and X. Chen, "Software Development, Configuration, Monitoring, and Management of Artificial Neural Networks," *Security and Communication Networks*, vol. 2022, p. 11.

[2] B. W. Sorte, P. P. Joshi and P. V. Jagtap, "Use of Artificial Intelligence in Software Development Life Cycle: A state of the Art Review," *International Journal of Advanced Engineering and Global Technology*, vol. 03, no. 03, 2015.

[3] R. Sanchis, Ó. García-Perales, F. Fraile and R. Poler., "Low-Code as Enabler of Digital Transformation in Manufacturing Industry," *Applied Sciences*, vol. 1, no. 12, 2019.

[4] L. Wang, S. Guan, W. Deng, P. Lu and H. A. Khattak, "RP System Design for Hydrogen Equipment Manufacturing Industry Based on Low Code Technology," *Mobile Information Systems*, 2022.

[5] S. Saeed and A. Varol, "SOFTWARE ENGINEERING AND ARTIFICIAL INTELLIGENCE: REENHANCING THE LIFECYCLE," *David C. Wyld et al. (Eds): CSE, AI & FL, NLPTT* , pp. 33-44, 2021.

[6] R. Feldt, F. G. d. O. Neto and a. R. Torkar., "Ways of applying artificial intelligence in software engineering," in *In Proceedings of the 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE '18)*, New York, NY, USA, 2018.

[7] S. Prajapati, B. Prajapati, S. Vegad and G. Gohil, "Artificial Intelligence and Software Engineering:," *International Journal for Research in Applied Science & Engineering Technology (IJRASET),* vol. 10, no. III, 2022.

[8] Barenkamp, M. &. Rebstadt and O. Jonas & Thomas, "Applications of AI in classical software engineering," *Springer ,* 2020.

[9] E. Nascimento, A. Nguyen-Duc and T. C. Ingrid Sundbø, "Software engineering for artificial intelligence and machine learning software: A systematic literature review," 2020. [Online]. Available: https://arxiv.org/abs/2011.03751.

[10] Rech, Jörg and K.-D. Althoff, "Artificial Intelligence and Software Engineering: Status and Future Trends," *Künstliche Intell,* pp. 5-11, 2004.

[11] H. Sofian, N. A. M. Yunus and R. Ahmad, "Systematic Mapping: Artificial Intelligence Techniques in Software Engineering," *IEEE,* 2020.

[12] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer and S. Wagner, "Software Engineering for AI-Based Systems: A Survey," *ACM,* p. 59, 2022.

[13] E. Elshan, E. Dickhaut and P. Ebel, "An Investigation of Why Low Code Platforms Provide Answers and New," in *Hawaii International Conference on System Sciences*, Hawaii, 2023.

[14] M. A. A. Alamin, S. Malakar, G. Uddin, S. Afroz and T. B. Haider, "An Empirical Study of Developer Discussions on Low-Code Software Development Challenges," in *IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, Madrid, 2021.

[15] T. Xie, "Intelligent Software Engineering: Synergy between AI and Software Engineering.," in *In Proceedings of the 11th Innovations in Software Engineering Conference (ISEC '18). Association for Computing Machinery*, New York, NY, USA, 2018.

[16] D. P. Wangoo, ""Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, India, 2018.

[17] H. K. Dam, *Artificial intelligence for software engineering,* XRDS, 2019.

[18] F. Sufi, "Algorithms in Low-Code-No-Code for Research Applications: A Practical Review," *Algorithms,* 2023.

Authors' Profiles

Amna Raza Khan is a student currently pursuing a degree in Software Engineering at Jinnah University for Women. She is passionate about investigating the intersection of technology and innovation and have a good foundation in programming languages and software development processes. She is committed to creating significant advances in software engineering through her analytical and problem-solving skills. She adds a distinct perspective to her study by drawing on her practical expertise in the software business. She has keen interest in the field of Neural Network, Deep Learning and Data Analytics.

Javeria Fatima is a passionate software engineer and a rising researcher in the field of software engineering. She has a good foundation in programming languages, software development processes, and system architecture and she is pursuing a degree in Software Engineering from Jinnah University for Women. Her research interests include artificial intelligence, machine learning, and software analytics. This research article is her contribution to the research community, demonstrating their commitment to pursuing unique ideas and making significant contributions to the software engineering area.

Sumaira Ahmed joined Jinnah University for Women as a Lecturer in 2013 and is currently serving as Senior Lecturer in the department of Computer Science and Software Engineering. Her research interests include Deep learning, Machine Learning and Natural Language Processing. She seeks to inspire and assist the next generation of software professionals as a renowned instructor. She is dedicated to giving her pupils a thorough foundation of computer science fundamentals while also developing their critical thinking and problem-solving abilities. Her commitment to teaching quality and her ability to effectively express complicated subjects have earned her a high level of respect in the academic world.