Journal Publication of International Research for Engineering and Management (JOIREM)
Volume: 10 Issue: 01 | Jan-2025

AI In Modern Software Development: Current Practices, Challenges and Future Possibilities .

Sunil Medepalli

IBM

Abstract - Artificial Intelligence is fast emerging as an inseparable part of software engineering. Its impact on modern software development is significantly high. Though writing code or to be more precise, generating code through AI systems and tools facilitates the development of a wide variety of products, there are ethical considerations, risks, and limitations that ultimately affect the overall quality of the software. This research paper highlights how AI is used in modern software design, development, testing, security, and management processes and they are flawlessly streamlined for enhanced productivity and quality. The research describes how AI is important in software engineering as firms without employing AI in SDLC are susceptible to underperforming. It provides insights into, what it could help with in terms of developing modern and scalable applications, and the ethics involved in AI-driven software engineering. It also highlights limitations underlying the applications of AI in the past, present, and future.

1.INTRODUCTION

Software Development is rapidly changing, growing in strength with the power of new technologies. Earlier, it was hard for anyone to imagine a bot could be capable enough of simplifying human tasks (Hernández-Ledesma et al., 2017). No doubt, the modern world is software-obsessed, running various types of software and programs that make work more efficient and reliable (Farley, D., 2021). Seemann (2021) states that the primary goal of software engineering is to develop a more reliable, efficient and user-friendly system. Whether it is a small application or a big application, it must address the goals. Different technologies from time to time have successfully been implemented, and have greatly influenced the way the imaginations are programmed (Pantiuchina et al., 2017). Indeed, technologies such as cloud computing, data science, blockchain, the Internet of Things, and others have significantly influenced software engineering in various ways. However, still, the role of experimental AI is more vital. It has ushered the software industry into a new phase of "anything possible" where any ideas could easily be turned into a solution.

Artificial Intelligence is now widely used by corporate, and small and big companies in simplifying their software development processes. It enables developers to overcome a plethora of problems in core areas of coding, code refinement, automation, code optimization, debugging, deployment, and project management (Barenkamp et al., 2020). ML, an AI's subset helps analyze large sets of data and information from previous projects. The predictive analytics guide teams in making well-informed and insightful decisions to further improve their current or ongoing projects. NLP (Natural Language Processing) is a key to building more meaningful interactions. A wide range of code development tools streamline team collaboration and improve efficiency and transparency in project management (Amershi et al., 2020). Apart from this, Marar (2024) explains how AI as a tool is more powerful when it comes to CI/CD (continuous integration and continuous development). Shneiderman (2020) describes how identifying small and big changes in code at any stage like designing, development, testing or security has always been more crucial and how AI makes it easier with minimal or even without human intervention. Adopting AI strategies, developers can take on agile methodologies to swiftly respond to market requirements and deliver more intuitive and robust products.

Software Engineering Evolution

Software history is certainly amazing from the first program designed by Ada Lovelace in the mid-19th century to the birth of ENIAC (Electronic Numerical Integrator and Computer) programming in the 1940s. Though it moved through various evolutionary phases, it reached new levels of possibilities with the introduction of AI in the mid-20th century. Early AI-based systems were good at performing a wide range of tasks in problem-solving and logical reasoning (Philipson, 2004). AI techniques continued to evolve since then. The use of symbolic reasoning and heuristic search became a little practical. ELIZA an AI program, a natural language processing program could be an example of its growing influence then. Mijwel (2015) explained how with the development of programs that could show human-like intelligence in the 1980s (an example of it could be MYCIN) new milestones in AI in software engineering were reached. Currently, popular techniques such as neural networks facilitate the process of software testing and development process optimization. The real impact of AI surfaced when AI was preferred as a technology to generate code for small and big software programs. Bot-driven, AI-powered coding, debugging, application security, testing, monitoring, and management are now common. But, yes, all these have certain limitations. Some problems and challenges exist and they largely impact the development of worthier, more user-friendly, and business-friendly applications (Vaidya et al., 2023).

SDLC and Automation Possibilities

Software engineering involves a large number of tasks. De Silva et al. (2022) described the complexities involved in different phases in the Software Development Lifecycle (SDLC) and how each of these phases will have small goals. Liu, et al., (2020) stress on importance of AI applications or approaches which are known to be more beneficial in terms of developing smart solutions within less time. The use of AI in software engineering increased over the past two decades. It is now a technique commonly employed in designing, building, and testing various software components with ease. This in turn enables a smooth development process. But, still, there is also a supposed gap between the research in AI applications and the actual implementation of AI in the software programming processes.

Generating, Analyzing, and Revamping Code: Synching AI with Human Intelligence for Better Results

Creating the required code as per the project goals is one big thing for all developers. Ideas are turned into real applications only with proper coding. If it goes wrong, the entire application goes wrong. But, writing the code is now a thing of the past. We use the term “code generation” as these are AI tools that help with creating the right code snippets. Now tools come with the power of AI, trained on a large set of programs, code structures, architecture, designs, and patterns which also seem to have the potential to solve even complex problems (Kulkarni et al., 2017). Human-like intelligence was limited in the past but AI technology made it all possible for systems and machines to think and act like humans and generate code at speed. Analyzing the context, the programs now generate code snippets that depict the developers’ and the stakeholders’ thoughts.

No doubt, it is the power of NLP (natural language processing) that has made this task much simpler since it allows the users to drape their thoughts with the real code, as they could provide instructions in plain language to get the right or the preferred output. By describing code requirements in natural language, coders can simplify their task of coding. For instance, developing a more secured and scalable REST API endpoint to extract, define, and

refine the user data is now made simple with instructions given to the AI-code generation tool. Similarly, getting suggestions, and recommendations on the next block of code or to put it simply, code completion support increases productivity greatly. For instance, systems trained in Python, Java, and other coding languages provide developers with suggestions or what could be the next line of code taking into account the context. Earlier, coders do not have such an option. They were supposed to write at length what they needed to build functional components of the software without real-time support. But, now with AI, it has become possible. Now, many programming platforms have implemented AI intending to support developers in developing the code they need with suggestions and recommendations (Padmanaba et al., 2019).

AI Assistants: Convenience vs. Risks

According to Liang et al. (2024), there has been a great increase in dependence on AI assistants. A quantitative study on software development practices and the use of AI assistants such as GitHub Copilot and OpenAI's ChatGPT and other similar tools that provide transformative insights indicates how developers prefer to consult one of such tools to code programs and how such tools also help increase efficiency, quality, and productivity. It is a well-known fact that AI tools are a little biased in producing code that could be insecure too as tools are trained without much due attention to the sources. The sources for the data could be verified or unverified. Therefore, there is a high risk of code going wrong, which might disrupt the work of a developer who depends entirely on the AI output for the code. Apart from this, research conducted by Pinto et al. (2024) demonstrate how concerns are prevailing in areas of plagiarism and copyright as the code generated by such tools could match the existing pieces of code or copyrighted programs. Apart from this limitations in the accuracy and correctness of such code snippets are also one of the biggest issues.

AI's capabilities to identify and analyze patterns and deliver more powerful predictions enable a team of developers to overcome barriers in development, reduce or overcome inconsistencies or errors, and ultimately improve deployment and delivery. It is worth mentioning that developers save time and allocate more time to brainstorming and innovation by doing away with repetitive tasks (Maninger et al., 2024). Technology firms relying on AI tools for coding and code generation also gain competitive advantages as their teams become much smarter with AI assistance and begin to produce code at a great speed which helps complete projects on time. One of the key aspects of an AI-driven environment of coding is that it enhances team collaboration, brings different teams from different platforms to one single and unified platform where all the team members, leaders, and decision-makers can find it easy to code in collaboration, and develop various components of the software without compromising on quality. This collaborative approach not only reduces the time needed for the development of the product but also makes the product more powerful. Research also showed that AI and Human collaboration yield better results. AI-generated code could create problems if it is not paired with human intelligence. Refining, validating, and rewriting it to fit the purpose is crucial. Research emphasizes the importance of humanized systems which ultimately help with enhancing software productivity and reliability (Perry, et al., 2023).

Testing With AI

AI now plays a critical role in testing. Software developers find it easy to generate and execute a wide variety of tests, which help identify bugs. AI algorithms are employed to effectively overcome challenges that largely impact the overall software

quality. Study conducted by Heusser and Larsen (2023) show that with proper analysis, potential vulnerabilities are weeded out. This in turn enhances the testing lifecycle efficiency. It is applied to enhance exploratory testing processes. Similarly, it is employed to analyze the data, client cases, and development patterns. Furthermore, AI-trained systems and processes reduce the time needed to heal the programs as they can automatically detect the bugs and fix the systems. Kasowaki and Akara (2023) describe AI-driven systems as more adaptive as they can adapt to ongoing software development processes too, and suggest what is working and what is not. Apart from this, AI synced with cloud technology allows the users to scale systems up and down as required with extensive modifications in codebases.

With the resurgence of AI in software development, corporate firms prefer to employ AI Agents for development. Companies employing AI systems to develop different components of the software notice a greater level of efficiency in all phases of SDLC. Each AI agent will be responsible for carrying out its process of design, coding, re-coding, fixing the code, debugging, testing, and management. Though it is considered risky, technological advances, particularly in AI will make this vision more achievable and this will further reduce the necessity of hiring multiple employees for the task (Donvir et al., 2024). Even if AI moves to this level, there will be a need for human intervention as this will help organizations ensure that the products are designed, developed, and tested as per the standards and it is ready for use or to make an impact (Korzeniowski and Goczyła 2019).

Transformative Role of AI in Development: Current Challenges and Possible Solutions

Though AI plays a transformative role in SLDC there are challenges that it grapples with. AI models are trained on huge datasets but still, poor-quality data is one big problem that escapes attention of the organizations. AI systems trained on non-verified databases might produce unsuitable results. Developers using such output might be at risk of developing a system that doesn't align with the project goals or at least it will disrupt the development process or might increase the time for production (Hossain Faruk et al., 2022).

Implementing AI in SDLC requires a considerably greater initial investment. Enterprises might have to bear the losses if it doesn't deliver the expected results. According to Khaliq, Z et al. (2022) technically complex would be the process of integrating AI into legacy software development systems. It is one big issue that hinders organizations to experiment or innovate. Hence, companies prefer to revert to their old systems or methodologies of developing the software. Furthermore, Fischer et al. (2022) stressed the importance of developing AI models that could sync perfectly well with the SDLC and other integrated systems but creating such models would need unparalleled expertise. A lack of technical understanding of how AI could be implemented to streamline the SDLC might stop organizations from adopting it. Machine intelligence can't be superior to human intelligence. Therefore, relying only on the analyses generated through AI systems is never recommended as such analytics might lead to unfair results or might disrupt the process of development.

Leveraging AI's Potential

AI systems are more vulnerable to attacks and they could malfunction if they are fed with the wrong data. The scalability of such systems is also one big problem since it invites investment from time to time to comply with modern development in this field. Managing the systems also adds to the increasing costs (Prather et al., 2024). Though there are small and big challenges in implementing and integrating AI into SDLC to achieve much better results faster, they could be mitigated with proper planning. Promoting AI literacy should be the first

step as knowledgeable employees with a sound understanding of AI in SDLC can ultimately outperform in development. Organizations should focus on small projects first where they could use the AI to discover its potential and the inherent challenges and as they reach new milestones of success they can move to bigger projects. Furthermore, problems existing in core areas such as communication, collaboration, and project management can be resolved with the integration of modern M2M (machine-to-machine) strategies. Building such M2M APIs could be challenging but it will have the potential to facilitate communications greatly as the agent could interact seamlessly. An agent developing the code will share the code with the agent responsible for analysis. As the code is analyzed it could be shared with the agent responsible for debugging. AI agents will work perfectly well when there is human intervention. Though the AI agents will act autonomously, a human agent responsible for the management of the process will have a bigger responsibility. There are bigger challenges in developing such multi-agent platforms where these AI-driven agents can take the task of developing the software on their own. There can be challenges in terms of defining the roles and objectives for every agent. Designing AI architectures and developing systems with different capabilities might pose bigger challenges for the firms but with proper planning, organizations could achieve these goals and thus streamline their software development processes. No doubt, combining the output of AI with human intelligence should be prioritized as it is the only way to generate flawless solutions. Without human intervention, AI technology in programming solutions might sound meaningless or superficial. Updating the AI-driven systems should be a top priority concern. Continual data training will make the AI system more powerful and this way it will generate the right output and software development will be an error-free process.

3. CONCLUSIONS

Though AI has been in use for over two decades and has successfully transformed the world of technology, it still has a long way to go. With the development of more modern super-intelligent systems, it could be possible to automate software engineering or at least core activities that fall within areas of design, recoding, reuse of code, generating code snippets, deploying, testing, debugging, and management. It is possible that shortly AI systems integrating with software development technologies will open new ways of autonomous coding. It will be possible to generate completely error-free code with descriptions in natural language. The developers will have more time to resolve problems and innovate while the AI systems will simplify small and big repetitive tasks. Manual coding will be reduced to a greater extent. Personalizing the software programs with AI is limited now but training verified data systems will power the AI systems greatly and AI systems could dynamically adapt to software development processes and yield better results. Apart from this, by reducing errors, saving time, and improving the collaboration among teams, it will continue to be the first choice for the SDLC management.

REFERENCES

1. Hernández-Ledesma, G., Ramos, E. G., y Fernández, C. A. F., Aguilar-Cisneros, J. R., Rosas-Sumano, J. J., & Morales-Ignacio, L. A. (2017). Selection of Best Software Engineering Practices: A Multi-Criteria Decision Making Approach. Res. Comput. Sci., 136, 47-60.

2. Farley, D. (2021). Modern Software Engineering: Doing What Works to Build Better Software Faster. Netherlands: Pearson Education.

3. Seemann, M. (2021). Code That Fits in Your Head: Heuristics for Software Engineering. United Kingdom: Pearson Education.

4. Pantiuchina, J., Mondini, M., Khanna, D., Wang, X., & Abrahamsson, P. (2017). Are software startups applying agile practices? The state of the practice from a large survey. In Agile Processes in Software Engineering and Extreme Programming: 18th International Conference, XP 2017, Cologne, Germany, May 22-26, 2017, Proceedings 18 (pp. 167-183). Springer International Publishing.

5. Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications of AI in classical software engineering. AI Perspectives, 2(1), 1.

6. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., ... & Zimmermann, T. (2019, May). Software engineering for machine

learning: A case study. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 291-300). IEEE.

7. Marar, H. W. (2024). Advancements in software engineering using AI. Computer Software and Media Applications, 6(1), 3906.

8. Shneiderman, B. (2020). Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI systems. ACM Transactions on Interactive Intelligent Systems (TiiS), 10(4), 1-31.

9. Philipson, G. (2004). A short history of software. In Management, Labour Process and Software Development (pp. 13-44). Routledge.

10. Mijwel, M. M. (2015). History of Artificial Intelligence Yapay Zekânın T arihi. Computer Science,(April 2015), 3-4.

11. Vaidya, J., & Asif, H. (2023). A Critical Look at AI-Generate Software: Coding with the New AI Tools is Both Irresistible and Dangerous. Ieee Spectrum, 60(7), 34-39.

12. De Silva, D., & Alahakoon, D. (2022). An artificial intelligence life cycle: From conception to production. Patterns, 3(6).

13. Liu, B., Li, G., Zhang, H., Jin, Y., Wang, Z., & Shao, D. (2024). The Gap Between Trustworthy AI Research and Trustworthy Software Research: A Tertiary Study. ACM Computing Surveys, 57(3), 1-40.

14. Kulkarni, R. H., & Padmanabham, P. (2017). Integration of artificial intelligence activities in software development processes and measuring the effectiveness of integration. Iet Software, 11(1), 18-26.

15. Padmanaban, P. H., & Sharma, Y. K. (2019). Implication of Artificial Intelligence in Software Development Life Cycle: A state of the art review. 2019 IJRRA all rights reserved.

16. Liang, J. T., Yang, C., & Myers, B. A. (2024, February). A large-scale survey on the usability of AI programming assistants: Successes and challenges. In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (pp. 1-13).

17. Pinto, G., De Souza, C., Rocha, T., Steinmacher, I., Souza, A., & Monteiro, E. (2024, April). Developer Experiences with a Contextualized AI Coding Assistant: Usability, Expectations, and Outcomes. In Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI (pp. 81-91).

18. Maninger, D., Narasimhan, K., & Mezini, M. (2024, April). Towards Trustworthy AI Software Development Assistance. In Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results (pp. 112-116).

19. Perry, N., Srivastava, M., Kumar, D., & Boneh, D. (2023, November). Do users write more insecure code with AI assistants?. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (pp. 2785-2799).

20. Heusser, M., & Larsen, M. (2023). Software Testing Strategies: A testing guide for the 2020s. Packt Publishing Ltd.

21. Kasowaki, L., & Akara, N. (2023). Exploratory Testing Strategies for Software Quality Assurance (No. 11360). EasyChair.

22. Donvir, A., Panyam, S., Paliwal, G., & Gujar, P. (2024, October). The Role of Generative AI Tools in Application Development: A Comprehensive Review of Current Technologies and Practices. In 2024 International Conference on Engineering Management of Communication and Technology (EMCTECH) (pp. 1-9). IEEE.

23. Korzeniowski, Ł., & Goczyła, K. (2019). Artificial intelligence for software development: the present and the challenges for the future. Biuletyn Wojskowej Akademii Technicznej, 68(1).

24. Hossain Faruk, M. J., Pournaghshband, H., & Shahriar, H. (2022, October). AI-oriented software engineering (AIOSE): challenges, opportunities, and new directions. In

International Conference on Software Process Improvement (pp. 3-19). Cham: Springer International Publishing.

25. Khaliq, Z., Farooq, S. U., & Khan, D. A. (2022). Artificial intelligence in software testing: Impact, problems, challenges and prospect. arXiv preprint arXiv:2201.05371.

26. Fischer, L., Ehrlinger, L., Geist, V., Ramler, R., Sobiezky, F., Zellinger, W., ... & Moser, B. (2020). Ai system engineering—key challenges and lessons learned. Machine Learning and Knowledge Extraction, 3(1), 56-83.

27. Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., ... & Briggs, B. (2024, August). The widening gap: The benefits and harms of generative ai for novice programmers. In Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1 (pp. 469-486).

28. Santa Barletta, V., Cassano, F., Pagano, A., & Piccinno, A. (2022, November). New perspectives for cyber security in software development: when end-user development meets artificial intelligence. In 2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT) (pp. 531-534). IEEE.

29. Artificial Intelligence Methods For Software Engineering. (2021). Singapore: World Scientific Publishing Company.

30. Lean and Agile Software Development: 6th International Conference, LASD 2022, Virtual Event, January 22, 2022, Proceedings. (2022). Germany: Springer International Publishing.

31. Lee, K., Qiufan, C. (2021). AI 2041: Ten Visions for Our Future. United States: Crown.

32. Artificial Intelligence Methods For Software Engineering. (2021). Singapore: World Scientific Publishing Company.