Journal of Academic Inquiry and Knowledge Volume 2024 (3) 130-135
ISBN 979-8-9888908-1-2

THE IMPACT OF ARTIFICIAL INTELLIGENCE ON SOFTWARE ENGINEERING

Eliot Heo

Los Angeles, CA, USA

The Impact of Artificial Intelligence on Software Engineering

Artificial Intelligence (AI) has been widely used since the public release of ChatGPT in November 2022. People are using ChatGPT to write essays and AI programs to create art; people are even using AI chatbots to alter syntax to create a different "voice" in text. As AI is becoming more commonplace, there is growing concern about its appropriate use. However, in software engineering, AI enhancement will have a positive impact. Introducing AI to software engineering will lead to increased productivity, cost-efficient methods, enhanced software quality, and assistance in the development process.

AI is used in many ways in software engineering. For example, the software engineering process consists of planning, problem analysis, software design and development, implementation, testing, integration, support, and maintenance. These steps all conform to creating software that functions properly. AI assists with three major groups of these steps: development, testing, and maintenance (*Applications Of AI In Classical Software Engineering.pdf*, n.d.). In the development stage, AI can assist the developer by doing routine things and carrying out tasks. In the testing stage, AI can look for errors, bugs, or optimization

for the developer. In the maintenance process, AI can continue code writing and examine existing code to improve the software.

With a knowledgeable programmer, AI will benefit software engineering tremendously. However, a developer needs several requirements to successfully use AI in software development (*What Is It Like To Program With AI.pdf*, n.d.). First, the developer must have sufficient background knowledge of programming a solution to a problem to determine the problem for AI to solve. Second, the developer must understand AI's limits as AI must be given material to understand or replicate code. AI cannot make its own perfect code for the developer, just as humans using conventional engineering cannot make the ideal code themselves. AI must be used to grasp knowledge of code, replicate code, proofread, or expand upon the script given to them.

Software engineering is a lengthy process that requires trial and error, and AI will increase productivity in this process. AI can automate functions, allowing developers to rewrite code repeatedly (*Interaction Between Software Engineering And AI.pdf*, n.d.). AI can also analyze code and suggest software outcomes or alternative solutions. The developer can then learn from AI's observations and autonomous learning (*Interaction Between Software Engineering And AI.pdf*, n.d.). Moreover, genetic programming, a form of AI, can generate computer code and take on redundant tasks, allowing the developer to focus on prioritizing steps, thereby enhancing productivity (*AI Techniques in Software Engineering.pdf*, n.d.).

AI can create cost-efficient innovations that will open up more business models. AI can use automatic programming to generate reusable code for the developer (*AI Techniques in Software Engineering.pdf*, n.d.), saving time and money to test new lines of code and find newer or better innovations for the code. AI can also help with risk management by using reusable

code, allowing developers more time to focus on effectively using their budget in maintenance or testing (*AI Techniques in Software Engineering.pdf*, n.d.). Both of these strategies will give long term benefits for software engineering.

AI will also enhance software quality, leading to the mass production of low-effort software with fewer bugs; AI can predict software bugs or errors by using *explainable AI*, which discovers code errors and how they can be fixed (*Explainable AI For Software Engineering.pdf*, n.d.). LIME (local, interpretable, model-agnostic explanations), a form of explainable AI, is used to explain errors to the developer to improve software quality. Again, this helps developers improve their skills and increases the software's overall quality.

AI will also help in the many stages of software development. Machine learning (ML) can be used for decision-making and assisting or creating code from scratch by giving them a basis (*Systematic Mapping Artificial Intelligence Techniques in Software Engineering.pdf*, n.d.). This allows developers to gain clear objectives through coding and prioritize optimization and edits to the code. AI also helps in software development's maintenance or testing phases as it can check code and identify its trends, errors, bugs, or possible optimizations (*The Role Of AI In Software Engineering.pdf*, n.d.). AI can accomplish these functions more readily than a human programmer, speeding up the software engineering process and improving productivity and quality.

AI is beneficial in many aspects of software engineering. It increases productivity, cuts costs and time, and even enhances not only the quality of the software but the process needed to create it. Many other AI applications. For example, machine learning has been gaining traction since the introduction of AI.It can be used to a greater extent in software engineering (*AI and*

Software Engineering Rapid Process Improvement through Advanced Techniques.pdf) to improve coding productivity and software quality.

Data management and interpretability are certainly of significant significance to AI initiatives. This is especially true for jobs that require data management, balancing, mining, and large data pools (*Towards Efficient Software Engineering In The Era of AI And ML.pdf*, n.d.). AI can do much with what it is given and, in the future, could transform data or maintain it in pristine, flawless condition.

Automated jobs will become a more useful tool in software engineering. Software engineering is a lengthy and repetitive process; automation could handle the heavy lifting for a developer and allow prioritization (*Applications Of AI In Classical Software Engineering.pdf*, n.d.). Moreover, automation could become more advanced and perform functions as part of conventional software engineering, leading to the best of AI and conventional software engineering techniques. If we handle this power of automation with AI, software engineering can be transformed into a system rather than a lengthy process.

There must be several conditions to allow for these predictions of the evolution of AI in software engineering. First, there must be investment in data governance. This will enable data safety and the implementation of AI into data governance and more security. Next is to develop interpretable AI models. This will make software engineering less of a mountain and more like a wall for newer developers and conventional developers. AI requires various conditions compared to conventional software engineering, and it won't be easy to move forward and innovate if developers become divided on the process and not the outcome. Finally, the problems of AI, such as data quality, model interpretability, and ethical dilemmas in the engineering process, must be addressed. Data quality is a problem due to the code strictly written by AI being

"sloppy" or unorganized. AI cannot write a script or code the way a human can. Therefore, it lacks the human thinking to write an organized and structured code (*Applications Of AI In Classical Software Engineering. pdf*, n.d.).

Conclusion

AI is the future of software engineering. It can increase productivity, make cost-efficient innovations, enhance software quality, and assist with the software engineering process. However, it is crucial that AI is used judiciously and not heavily relied on. Overusing AI can lead to controversies and the creation of problematic software. To use AI without the consequences, its pros and cons and usefulness for software engineers must be understood. Software engineering has yet to use AI to its full potential.

To aid in the development of AI in software engineering, structural classes and software engineering courses could be developed to explain greater integration of AI in software development. AI will be greatly effective with new techniques and discoveries once software engineers realize its usefulness and implement it.

References

Bodimani, M. (2021, March). AI Techniques in Software Engineering.pdf. Kansas City; The Science Brigade Publishers.

Raza, F. N. (2009, March). AI Techniques in Software Engineering.pdf. Hong Kong; IMECS 2009.

Barenkamp, M., Rebstadt, J., & Thomas, O. (2020). Applications Of AI In Classical Software Engineering.pdf. Osnabrück; Springer.

Tantithamthavorn, C., Jiarpakdee, J., & Grundy, J. (2019). Explainable AI For Software Engineering.pdf. Melbourne; IEEE Computer Society.

Jain, P. (2011). Interaction Between Software Engineering And AI.pdf. Punjab; IJCSE.

Sofian, H., Yunus, N. A., & Ahmad, R. (2022, May). Systematic Mapping Artificial Intelligence Techniques in Software Engineering.pdf. Kuala Lumpur; Diego Oliva.

Harman, M. (n.d.). The Role Of AI In Software Engineering.pdf. London; University of College London.

Shah, V. (2019). Towards efficient Software Engineering In The Era of AI And ML.pdf. San Diego; IJCST.

Sarkar, A., Gordon, A. D., Negreanu, C., Poelitz, C., Ragavan, S. S., & Zorn, B. (2022, October). What Is It Like To Program With AI.pdf. (n.p)