CT60A4800 Fundamentals of smart systems – Assignment 5

Name: Trieu Huynh Ba Nguyen Student number: 000405980

Artificial intelligence (AI) has developed at an extraordinary rate over the last few years. Nowadays, advanced AI could help us in software development, debugging, and testing. In this article, the pros and cons of this practice are discussed.

First, by using an AI-power programming assistant, such as Tabnine, GitHub Copilot, or Kite, the coding process will be much faster. They can help us automate repetitive tasks, speed up our workflow, and improve code quality by generating long sections of code created using machine learning. Additionally, they can provide you with valuable insights into the public codebase and help us name potential problems. Some advanced AIs, such as the one powering GitHub Copilot, could even solve complex algorithm-related problems. On the other hand, there are several risks associated with these tools. If the AI is not programmed correctly, it could make errors in its code that could cause problems for the user. Furthermore, as these AIs learn from the public codebase, if the AI is not given enough data to work with, it could make incorrect assumptions about the user's needs and preferences, which could lead to a sub-optimal experience. Lastly, if the AI is not updated regularly, it could become outdated and unable to keep up with the latest changes in the programming language or development environment. Therefore, for most software development activities, AI could only augment - hence the term "assistant" - a human developer. The code of most programs nowadays is too complicated and elaborate that an AI could never understand. We are still a long way from fully automated software development.

Next, AI-power debugging tools include the ability to find and fix errors more quickly and efficiently. Additionally, AI debugging tools like Intel ControlFlag can help to identify potential issues, such as security risks, code anomalies, performance shortfalls, etc., before they become actual problems. Since debugging takes a lot of time during the development process, this would save time and money in the long run. However, given that most programs nowadays have such complicated code, it is still too early to let AI take over some parts of the debugging process. An AI might not be able to show all errors in the code, especially those concerning the logical operation of the program. Even when an error is caught, the AI could not supply an ambiguous explanation and solution to that error. If we force an AI to do debugging, this will not only slow down but also damage the quality of the program. As a result, AI again could only help human debuggers. It is not possible to fully automate the debugging process as of now, and AI has only been able to offer suggestions for debugging.

When it comes to testing, AI-powered tools like Testsigma could speed up the process. This is because AI can be used to automate certain simpler testing tasks, such as unit or integration testing. This means that less time needs to be spent on these tasks, which can free up time for other aspects of the testing process. In addition, AI can also be used to help find potential bugs and errors, since they could analyze data more effectively than humans. As a result, it can help to improve the accuracy of software testing. Nevertheless, many software tests are yet to be completed by AI, and it is not wise to automate all testing tasks. Some test cases, like user experience, should not or even could not be done by AI, due to the complex perception and sentient intelligence of human nature. AI could help us to generate comprehensive test cases based on real-life user data, and analyze the results to suggest code improvement, but could not emulate how an end-user sees the system. If we ever deploy AI to such a complex test case, the results would be far from correct. Consequently, AI still could not automate every aspect of software testing and could augment a human only. Some straight-forward test cases could be performed by AI, but the rest needs to be worked on by a human tester.

In conclusion, although AI-based technologies have progressed so far, it is still incapable of replacing a human during software development. AI could augment us in creating, debugging, and testing software, but could not do most of these activities on its own.