AI's Impact on Traditional Software Development: A Technical Look

Bhanuprakash Madupati

MNIT, MN

JULY 2024

Abstract

The application of artificial intelligence (AI) has brought key shifts in conventional tactical software development, including code generation, testing and debugging, and deployment. Waterfall and Agile development approaches, which have been used for a long time, also widely employ manual and well-planned steps. However, with the help of automated tools and models such as OpenAI Codex and GPT-4, many aspects of the Software Development Life Cycle (SDLC) have been made possible. This paper examines the technical aspect of integrating AI into prior traditional software development life cycle methodologies, emphasizing code automation, intelligent testing frameworks, AI-based debugging, and continuous integration and deployment pipelines. The analysis is also based on the advantages of utilizing AI for optimizations in efficiency, accuracy, and development speed alongside issues like over-dependence on AI, ethical questions, and technical constraints. Based on the case and example given in this paper, it is clearly shown that the self-improvement of AI in software development makes the process more dynamic, autonomous, and optimized.

Keywords: AI, conventional software development, AI-based tools, software testing, code generation, debugging, continuous integration/ deployment (CI/CD), software development life cycle (SDLC).

1. Introduction

1.1Background

The Waterfall, Agile, and DevOps models dominate the IT industry and have become the grounds for constructing advanced software solutions. These approaches follow a linear or circulatory model where coding, testing, and deployment steps are usually involved. Albeit efficient, these methods may have issues concerning timeliness, expansiveness, and errors due to human interference. With changing business environments and technologies, keeping pace and delivering the software in record time has become essential. As a result, the software development life cycle's (SDLC) dynamism has been recorded by incorporating Artificial Intelligence (AI).

Artificial intelligence has advanced much in integrating key segments of software processing, namely code generation, testing, debugging, and deployment. Through the use of OpenAI Codex and GPT-4, among other technological advancements, it is now possible to write functional code from simple text inputs, thereby greatly cutting down on the time required to code and minimizing human errors. Similarly, AI-assisted testing tools improve traditional testing techniques by detecting bugs and security breaches faster and more efficiently. AI in these areas enhances both efficiency and the quality and accuracy of software applications.

1.2 Purpose of the paper

Technically, this paper investigates how AI affects the conventional software development paradigms, specifically focusing on how AI tools and approaches disrupt several misery steps of code generation, testing, debugging, and deployment. The objectives of this paper are as follows: The objectives of this paper are as follows: