Tuhina Das

ISM II

Research Assessment #7

Date: October 6, 2024

Subject: Full-Stack Software Development

Assessment: Research Assessment #7 - CI/CD Principles and Best Practices

In Vamsi Krishna Thatikonda's journal publication, "Beyond the Buzz: A Journey Through CI/CD Principles and Best Practices", Thatikonda writes a general overview of the organizational ideology behind continuous integration and continuous development (CI/CD) in software development. I was curious about what CI/CD was in principle, as well as how it worked, due to my recent summer experience interning with BigCommerce's Mobile Apps team, where CI/CD was heavily relied upon to manage development and deployment workflows for the app I built upon. While Thatikonda's publication doesn't fully delve into the technical aspects of how CI/CD pipelines work, I now better understand why these structures commonly exist in development environments, and this leaves me room to consider what I can do to streamline the development of my final product.

Thatikonda initially discusses the basics and key principles of continuous integration (CI) and continuous development (CD), taking extra care to highlight the differences between the two concepts. CI is defined as a practice where developers continue to merge their code into a primary "master" branch, while continuous development focuses on the automation of software update releases. The publication highlighted three key principles, including the first two: consistency and automation. These two principles align heavily with the software industry's changing standards in efforts to make workflows more efficient, and it's interesting to think

about how artificial intelligence is playing an increasingly important role in these processes, something Thatikonda also brings up in his discussion of improvements being made to current CI/CD softwares. Unfortunately, this publication does not dive into the technical aspects of how CI/CD works at a micro level; thus, I am left curious about how artificial intelligence can further streamline an already automated development process. The last principle of CI/CD in Thatikonda's list is fast feedback loops: the idea that through cyclic integration and development methodologies, a team of developers can continuously receive feedback on features that they are rolling out to their project. Since CI/CD pipelines are loaded with scripts designed to catch flaws related to build code, they are essential to catching the first wave of bugs that developers of large-scale projects typically miss: environment and dependency bugs, which can rapidly deteriorate the quality of the software on the user's end. What is interesting to me is seeing how this final principle aligns so closely with the Agile methodology of software development, which stresses the importance of continued development through iterations known as 'sprints'. Repeatedly showing up in my research, it might be worth considering using a tool to plan app development like Jira to help streamline my development process.

Two of Thatikonda's best practices in CI that interested me included the mentioning of frequent code commits and self-testing builds. When developing my project last year, I committed frequently. But I didn't commit with regulations or a development process in mind, so keeping CI in mind, I hope to set up a regulatory development process going forward. Additionally, the concept of self-testing builds is one I have not fully explored yet. Unit testing is something commonly discussed in software development, but this is something that I still have little practice with, especially in the realm of web development – perhaps this is something I can

look into in the near future as I continue to research best development processes that'll help me solidify my workflow for my final product.

APA citation(s):

Thatikonda, V. K. (2023, September 1). Beyond the Buzz: A Journey Through CI/CD Principles and Best Practices. *European Journal of Theoretical and Applied Sciences*, *1*(5), 334-340. https://doi.org/10.59324/ejtas.2023.1(5).24