Jordan Mitchell

9/5/2024

Professor Farley

The Role of Testing in the Software Development Life Cycle

The Software Development Life Cycle (SDLC) provides us with a structured approach to software development, guiding developers through the process of planning, designing, building, testing, and deploying software. Each step plays an important role in creating and delivering a functional, high-quality product. The testing stage is especially important for ensuring the software meets requirements. Testing helps identify bugs and defects as well as performance, security, and the final user experience. Key activities involved in the testing stage of the SDLC are essential to successful development and testing may occur at an earlier or later point than usual.

During the testing stage, the software will undergo evaluation to detect errors, bugs, or any issues in functionality, performance, or security. This stage usually involves many different types of testing, like modular testing, integration testing, system testing, and security testing. Each test helps to ensure that individual components work correctly, that the components function properly once integrated, and the entire program meets functional requirements that were set earlier. In a past project I'd worked on, I performed unit tests after each build to make sure each function worked properly on its own. Once they passed, I moved to integration testing, which revealed unexpected bugs and errors when each module interacted. Catching those bugs before submitting my final saved my grade from taking a hit, as my program wouldn't have been able to run properly.

Testing is vital to the SDLC for several reasons. First, it ensures the quality and reliability of the software. If not thoroughly assessed and tested, software can end up releasing with bugs that are detrimental to the user experience or even security vulnerabilities. Second, testing provides a platform to verify that the product meets the specified requirements and works as intended for real-world applications. Software that fails to meet consumer needs can lead to dissatisfaction and even lost business, so testing is a great tool to safeguard against that. In my own experience, when I thoroughly test, I have fewer issues throughout the entire development process. It also provides me with an opportunity to refine my code's quality to make sure the software performs well under various conditions. If I skipped testing, I would inflict hours and hours of debugging and rewriting upon myself. From the user perspective, I've also encountered software suffering from a shaky release that eventually pushed me away from the product altogether. A great example of this in the videogame industry is the company Ubisoft. This game developer often promises countless features and unparalleled user experiences that more often than not fall flat. Initially, I was a fan of their Assassin's Creed series, until each new release showcased glitchy, unpolished, and overall infuriating gameplay that made me feel like they were exploiting the series' name to get money from their devoted fanbase. Being on the receiving end of janky software serves as an example of what the kind of software I don't want to create.

Although testing traditionally occurs after the building stage, there are exceptions where it might happen earlier or later. In agile software development, testing is incorporated throughout the overall process. This approach ensures that problems are proactively found and addressed, which can also be thought of as continuous testing. In my experience with a smaller project, I practiced test-driven development (TDD), where tests were outlined before the actual code. This process resulted in the testing stage occurring earlier and influenced how I wrote and designed my code. On the other hand, there are situations where testing can be delayed, like when a project has a tight deadline. Unfortunately, I’ve experienced firsthand how delayed testing at the last moment leads to significant problems since it reduces the time to find and fix bugs. This can lead to the submission of a flawed product or require a post-release patching process.

In conclusion, testing is an essential part of the SDLC that helps guarantee the reliability and functionality of software as well as user satisfaction. While it normally takes place after the building stage, instances like continuous testing or delayed testing are examples of the flexibility of the SDLC depending on the needs of your project. From my own experience, I can confidently say that utilizing the testing stage early and effectively results in better outcomes, both in terms of product quality and development speed.