What Do We Know About Test-Driven Development Review

Test-Driven Development (TDD) is a development approach that is emphasized by many developers, but it is also a highly controversial agile development approach. This is because there is no strong research evidence to prove that TDD can effectively improve productivity and quality in software development (such as maintainability and reliability). Moreover, although TDD has always been a highly discussed topic, its actual usage rate is very low. According to surveys, only about 10% of the entire development process follows TDD specifications, and this 10% is often not the most important part of software development.

Research shows that the lack of a complete understanding of TDD is the biggest reason for its failure. Most engineers who use TDD for development do not understand the correct usage methods. For example, test-first is often seen as the focus of TDD, or even the only standard, but other key aspects of TDD such as maintaining the red-green cycle are often ignored. Another reason for failure is that writing test code requires additional time, which seriously affects the productivity of engineers. Therefore, many engineers do not like to use TDD for development. In a study, it was found that, in addition to the order in which tests are written, a short and stable average development cycle, and clear task definitions are the factors that truly affect the quality and productivity of agile development methods.

Since TDD has not achieved the expected results, should we completely abandon the practice of TDD? Or when should we use TDD? As mentioned in the conclusion of this article, this should depend on the habits and ideas of the engineers. Some people maintain discipline in writing unit tests for every small function through TDD, or some people are already accustomed to TDD, so continuing to use TDD for development is also possible. However, it is important to pay attention to the order of writing tests, not modifying the main program when the tests do not fail, and maintaining a high-quality level of tests and a short and stable development cycle to truly improve the maintainability and coevolution of the main program.

Test-Driven Development, as the name suggests, emphasizes the importance of testing in the development process. This approach begins by assuming specific scenarios and expected results, then writing and modifying the main program so that the tests for the assumed scenarios can pass. Then, different and more complex situations are considered. If the tests do not pass, it means that the main program has a problem that needs to be fixed. This process is repeated continuously, known as the red-green cycle. Through this testing, we can ensure that

each function operates smoothly. When problems arise in the program, we can also identify which part is wrong without difficulty, avoiding difficulties in debugging due to dependencies between functions.

However, based on my own experience, TDD is not easy for a junior engineer. A junior engineer who can complete the main program independently is already impressive. Engineers need to come up with more test data and consider all possible situations to ensure that the tests are effective. Writing test code is also time-consuming and burdensome for engineers who need to meet specific goals within a set timeline.

Many companies have a position for a test engineer, which I believe is a reasonable practice. Having a specific team to write test code can reduce the pressure on program development engineers. Through a team approach, all situations that should be tested can be identified completely. Relying solely on individual thinking can lead to blind spots, which can result in significant issues in product development and negate the benefits of TDD.

Furthermore, this article mentioned that the focus of TDD should not be on program coverage but on ensuring the quality of test code. However, it is difficult to measure the quality of test code with numbers, while there are many tools to calculate program coverage. Therefore, when engineers need to report in meetings, program coverage may be the only value that can be calculated in the test code. Engineers tend to focus more on coverage than quality.