TDD – TEST DRIVEN DEVELOPMENT

TDD, “test driven development” ou em português “desenvolvimento baseado em testes”, se baseia em ciclos pequenos de repetições, onde para cada funcionalidade do sistema um teste é criado antes. Este novo teste criado inicialmente falha, já que ainda não temos a implementação da funcionalidade em questão e, em seguida, implementamos a funcionalidade com o objetivo de fazer o teste passar.

O método de desenvolver uma funcionalidade ou uma aplicação baseada em teste surgiu com o objetivo de simplificar o código, visando que o mesmo seja criado com um único propósito, passar no teste criado para tal. Com isto, temos um código mais simples e mais rápido de ser desenvolvido, sem funcionalidades adicionais desnecessárias ou uma complexidade exagerada.

O TDD possui diversas características e benefícios, tais como:

1. **Cobertura de código**: como existe um teste associado a cada funcionalidade, é possível ter a certeza de que todo o código foi executado e os erros são encontrados no início do desenvolvimento.

2. **Segurança**: a implementação de novas funcionalidades é um processo delicado, pois há o risco de que a alteração gere bugs no sistema. Com o TDD, esse risco é reduzido, dando mais segurança para quem está desenvolvendo o software.

3. **Feedback rápido:** visto que as funcionalidades são testadas logo após a sua criação, é possível obter um feedback quase instantâneo do seu funcionamento.

4. **Depuração simplificada**: o processo de depuração se torna mais intuitivo, uma vez que, quando um teste falha, é mais fácil identificar onde se encontra o problema.

5. **Códigos mais limpos**: o TDD segue a filosofia do KISS, "Keep It Simple, Stupid", com isto os testes tendem a ser os mais simples e enxutos possíveis e quando passam pelo processo de Refactoring, se unindo ao sistema principal, o código em geral acaba ficando mais limpo.

6. **Maior produtividade**: como o código é flexível e limpo, gasta-se menos tempo com a correção de bugs e implementação de novas funcionalidades, consequentemente, a produtividade da equipe é maior.

7. **Aplicações flexíveis**: com o código sendo separado em diversos pedaços para as aplicações dos testes, o código acaba sendo mais flexível.

Uma das principais diferenças em relação aos outros métodos é o ciclo de desenvolvimento que começa com os testes e não com a implementação. Isso gera uma outra grande diferença que é uma maior segurança na correção de bugs, pois já que cada implementação só é feita para que se passe o teste e novos testes só são feitos quando a implementação anterior já está funcional. Além disso, a documentação são os próprios testes que mostram como o sistema está funcionando naquele momento, e é preciso percorrer os testes para entender o funcionamento geral do sistema.

Um outro método que é similar ao TDD é o BDD mas eles também têm suas diferenças: TDD é se importa mais com o design e o código e BDD nos desenvolvedores e testadores; BDD foca no nível da aplicação e dos requisitos e o TDD foca no nível de código que implementa esses requisitos; o TDD pode ser usado no BDD para passagem de testes e o BDD pode ser usado no TDD para melhorar o nível de abstração; e o BDD não tem fase de refatoração.

Alunos: Lucas Lobato, Luís Tiago Sena e Rodrigo Bezerra.