Curso: Testes Automatizados

Micro 6 - TDD X BDD

Olá! Seja bem-vindo!

Durante esse curso, você conheceu individualmente as principais metodologias para desenvolvimento ágil: Test Driven Development (TDD) e Behavior Driven Development (BDD), está lembrado? Então, nessa aula, você vai descobrir a diferença entre essas metodologias e também as vantagens de implantar cada uma delas em projetos de software.

Já adianto que, para um melhor aprendizado de testes automatizados, é essencial colocar em prática os comandos que serão abordados.

Bons estudos!

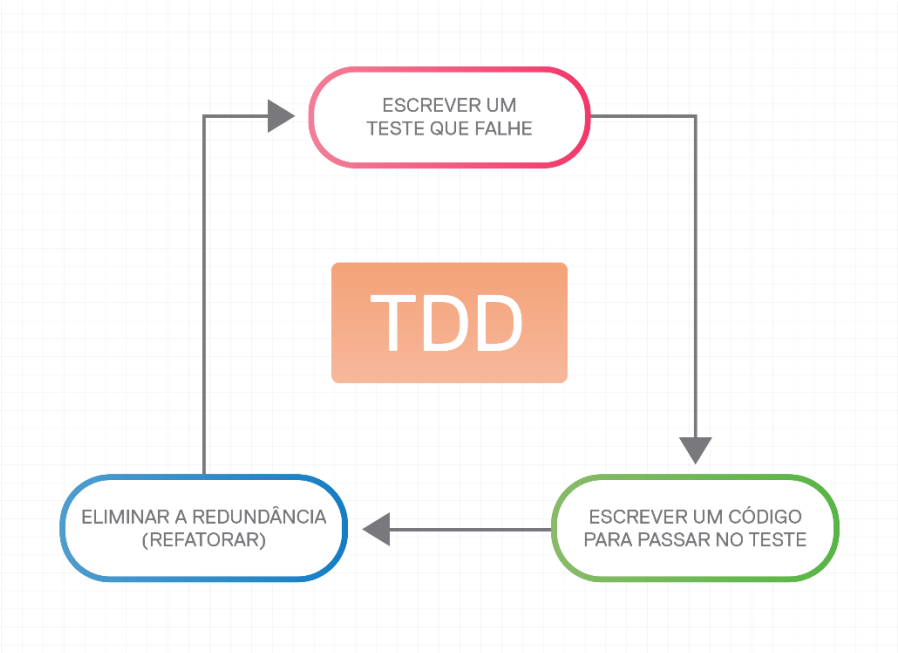
Para começar, recapitule o que seria o TDD. O Desenvolvimento Orientado por Testes (TDD) é uma metodologia ágil baseada na técnica de programação que visa entregar um produto com uma qualidade esperada pelo cliente em um curto período. Dessa forma, as principais metas da técnica do TDD são agir de forma preventiva, ou seja, prevenir possíveis erros e facilitar o entendimento dos requisitos do sistema.

O TDD é a prática ágil mais utilizada dentro do time de desenvolvimento de software, pois o ciclo de vida de desenvolvimento de um sistema é um processo natural no time de desenvolvedores.

Baseado nesse processo natural, imagine o ciclo de vida de um teste, escrito usando a técnica do TDD. Você deve se perguntar agora, o que exatamente propõe o ciclo no TDD. Então, resumidamente, ele propõe criar um teste para a funcionalidade desejada, que inicialmente vai falhar. Logo após a falha, desenvolver um código que execute corretamente o resultado esperado, de modo que o teste

passe e em seguida, pretende refatorar o código de uma forma que a redundância seja eliminada e este código fique o mais simples e funcional possível.

Esse exemplo é representado na imagem a seguir, em que um ciclo ilustra o Test Driven Development (Desenvolvimento Guiado por Testes). No ciclo, as três fases do processo, são representadas por três retângulos coloridos e interligados. Na primeira fase, o retângulo vermelho está preenchido pelo texto: “Escrever um teste que falhe”. Na segunda fase, o retângulo verde está preenchido pelo texto: “Escrever um código pra passar no teste”. E na terceira fase do ciclo, o retângulo azul está preenchido pelo texto: “Eliminar a redundância (refatorar)”. No centro do ciclo, há um retângulo laranja, preenchido pela sigla “TDD”.



Com o uso continuado desta técnica, com esse ciclo de vida, o tempo de depuração do software tende a diminuir, minimizando assim o esforço do time, pois os desenvolvedores saberão se a funcionalidade implementada está conforme as

regras esperadas, ou se o código escrito vai apresentar erros e/ou problemas no futuro.

No TDD é fundamental refatorar os cenários de testes, para que estes testes também venham a ter uma maior qualidade. Uma vez que, o teste vai garantir que o comportamento continue de acordo com o esperado.

Já no BDD, é possível realizar a extração das funcionalidades de um sistema através do levantamento de requisitos com o cliente e através desse levantamento de requisitos, escrever os cenários de teste se baseando no comportamento do usuário no sistema. É importante que a escrita do teste esteja focada no comportamento do usuário. Neste caso, também é válido ressaltar que a escrita dos cenários de teste usando o BDD, é realizada enquanto o time de desenvolvedores está implementando o código.

É importante que você se lembre, que esses cenários de teste poderão ser automatizados e com isso garantir que o comportamento do produto permaneça de acordo com o esperado, ou seja, com o que foi definido pelo cliente, garantindo assim a qualidade do produto.

Com esse processo de teste automatizado, o time ganha agilidade e o analista de teste tem um menor esforço na execução dos cenários de teste, uma vez que não serão executados manualmente e sim de forma automatizada. Tudo bem até aqui? Então, dando continuidade, agora você vai observar as diferenças presentes entre TDD e BDD! Preparado?

No Test Driven Development, o TDD, o teste é escrito para verificar a implementação da funcionalidade, mas à medida que o código evolui, os testes podem gerar resultados falsos. O Behavior Driven Development, o BDD, também é uma abordagem de teste primeiro, mas difere ao testar o comportamento real do sistema da perspectiva do usuário final.

A seguir, você vai aprender detalhadamente, alguns pontos de diferença entre as duas metodologias ágeis, TDD e BDD. Vamos lá!

Quando o ponto de diferença é o **FOCO**, a metodologia TDD é um processo "de dentro para fora", em que o foco está na qualidade. Já na metodologia BDD, o processo é "de fora para dentro", e o foco está no valor/negócio.

No ponto de diferença "**ESCRITA"**, quem escreve os testes na metodologia TDD, são os desenvolvedores e os testes são escritos ao mesmo tempo que o código. No BDD, os testes podem ser escritos pela pessoa que entende melhor o cliente, por exemplo, o analista de negócio, o PO (product owner) ou o analista de teste.

Sendo o ponto **ESPECIFICIDADE**, a diferença é que no TDD, quando um teste de unidade falha, você sabe exatamente o que falhou, já no BDD, quando um teste falha, você sabe que algo deu errado, mas não sabe o quê.

A diferença entre as metodologias no ponto **VELOCIDADE**, é que no TDD, um teste de unidade é um teste de um comportamento isolado, portanto, são super rápidos de executar. No caso do BDD, os testes comportamentais são os testes do sistema como um todo, em que o sistema deve ser colocado em um estado conhecido antes de cada teste, portanto, são mais demorados na execução.

Quando o ponto de diferença é a **MANUTENIBILIDADE**, no TDD, qualquer alteração na funcionalidade de um sistema, exige uma alteração em um ou mais testes de unidade, e deve-se levar em consideração que os testes mudam primeiro. No BDD, nem todas as alterações funcionais impactam no comportamento externo. Como os testes neste caso, são escritos em altos níveis comportamentais, significa que eles mudam com pouca frequência.

Por último, a diferença entre as metodologias de testes no ponto **PORTABILIDADE**, no TDD, os testes de unidade são altamente específicos para o código que eles cobrem, os testes estão entrelaçados e não são portáteis. Já no

BDD, os testes comportamentais não são acoplados ao código. Caso ocorra mudanças nas regras de negócio do sistema, basta reescrever o cenário de teste, refatorando assim o cenário que já foi escrito.

Conforme os exemplos que você acabou de aprender, existem diversas diferenças entre as metodologias TDD e BDD. Tendo em mente essa constatação, você deve se perguntar então, quais seriam as vantagens de ter essas metodologias implantadas em um projeto de desenvolvimento de software?

Considere alguns pontos que beneficiam o time de desenvolvimento quando implementado o **BDD** no projeto:

1 Os cenários de teste, deixam de ser meros “testes” e são definidos “comportamentos”;
2 Melhor comunicação entre desenvolvedores, testadores, PO (product owner) e cliente;
3 Por serem de natureza não-técnica, o entendimento dos cenários pode atingir um público mais amplo;
4 Por utilizar uma linguagem simples nos cenários de teste (sintaxe Gherkin) – “Dado que, quando, então” – facilita o entendimento do comportamento esperado pelos envolvidos no projeto.

Agora, relembre algumas das vantagens de se implementar o **TDD** em projetos de desenvolvimento de software:

1. Um conjunto de testes de unidade garante feedback constante sobre o funcionamento de cada elemento do sistema;
2. A qualidade do projeto aumenta, porque um desenvolvedor pode refatorar o código a qualquer momento, garantido assim, que não serão adicionados erros/problemas na funcionalidade alterada;
3. Ajuda os programadores a realmente entenderem seu código.

Nessa aula, você aprendeu as diferenças entre as metodologias ágeis TDD e BDD e quais as suas vantagens dentro de um time de desenvolvimento de software. No decorrer do curso, você verá outros exemplos de ferramentas e técnicas para implementação de BDD. Por hoje é só! Até mais!

Referências:

Página do DevMedia disponível em:<https://www.devmedia.com.br/test-driven-development-tdd-simples-e-pratico/18533/>

NORTH, Dan. (2012) **BDD is like TDD if….** Disponível em: <https://dannorth.net/2012/05/31/bdd-is-like-tdd-if/>. Acesso em: 05 nov. 2018

Página do Medium disponível em: <https://medium.com/@jrnmagalhaes/unit-test-x-tdd-x-bdd-913278f33b80/>