Aline Cristina Pinto
Eduardo Cotta Guimarães França
Luiz Gustavo de Paiva Matos
Pedro Lucas Silva
Ruan Bertuce de Carvalho Souza

TDD - Test Driven Development

Belo Horizonte

Abril de 2018

Aline Cristina Pinto
Eduardo Cotta Guimarães França
Luiz Gustavo de Paiva Matos
Pedro Lucas Silva
Ruan Bertuce de Carvalho Souza

TDD - Test Driven Development

Trabalho acadêmico da disciplina *Linguagem de Programação II* do curso técnico de Informática do Centro Federal de Educação Tecnológica de Minas Gerais.

Belo Horizonte
Abril de 2018

Faça ou não faça. Não há tentativa.
Yoda

Resumo

O TDD é uma prática que visa a criação de um teste e, em seguida, um código de produção. Possui um ciclo de funcionamento com etapas bem definidas e regras a serem seguidas. Este trabalho, além de descrever a metodologia e o funcionamento do TDD, também destaca o método de testes automatizados e apresenta análises sobre quando o uso do TDD pode não ser eficiente e sobre os benefícios que podem ser gerados ao utilizá-lo.

Palavras-chave: TDD. Test Driven Development. Testes. JUnit. Metodologia. Programação.

Abstract

The TDD is a practice that aims the creation of a test and, after that, a production code. It has a working cycle with well defined steps and rules to follow. This work, besides describing the TDD's methodology and working, also highlights the automated testing method and shows analysis about when using TDD may not be efficient and about the benefits that can be generates by using it.

Keywords: TDD. Test Driven Development. Tests. JUnit. Metodology. Programming.

Sumário

1 Introdução

Uma prática com raízes antigas mas utilidades muito atuais, o TDD, Test Driven Development, evoluiu para um dos conceitos mais importantes da programação. Kent Beck, considerado o criador da prática, fez sua primeira apresentação dela em 1995 na conferência da OOPSLA, dando o primeiro passo na divulgação do TDD pela comunidade da área de Tecnologia da Informação.

Test Driven Development é o Desenvolvimento Guiado por Testes, em que testes, as expectativas sobre eles e seus resultados são os elementos principais no direcionamento do desenvolvimento do código. TDD é uma das técnicas mais utilizadas atualmente pois visa, a partir do método já mencionado, entre outros benefícios, reduzir o custo da criação e manutenção de determinado código em relação a outros métodos de programação.

Neste trabalho de pesquisa são tratados assuntos relevantes sobre o TDD, como sua importância, sua implementação, os momentos em que ele pode ou não ser utilizado e considerações finais sobre o uso desta prática como um todo.

2 Test Driven Development

2.1 A importância de testes e o TDD

Erros de sistemas são comuns e podem ocorrer como consequência da falta de testes no momento de seu desenvolvimento. A falta desse processo muitas vezes justificada pelo custo e demanda de tempo ocasiona programas de baixa qualidade e difícil manutenabilidade. A inserção do Test Driven Development trás uma nova perspectiva a esses empecilhos visto que o desenvolvimento orientado a testes exige que os testes sejam escritos antes da implementação do sistema, consequentemente, o processo de produção se torna mais confiável e menos custoso pois além de se conseguir testar constantemente a aplicação, a criação dos testes afirma que as regras de negócio foram compreendidas durante a fase de criação dos testes unitários.

2.2 Metodologia Test Driven Development

No livro ***JUnit em Ação*** (MASSOL; HUSTED, 2005), os autores definem Test Driven Development como uma prática de programação que ensina os desenvolvedores a escreverem novos códigos, somente se um teste automatizado falhar e a eliminar a duplicação, sendo que a meta dessa metodologia é o "código limpo que funciona". Logo, trata-se de uma metodologia que preza pela criação de um código teste e em seguida o código de produção.

Nesta metodologia, a criação de teste unitários ou de componentes é parte crucial pois os módulos do sistema devem ser testados individualmente para garantir que operem corretamente para que quando sejam integrados, a aplicação possa ser testada como um todo obtendo a menor possibilidade de erros.

2.2.1 Ciclo de Funcionamento

O ciclo de funcionamento desta metodologia é simples e se baseia em três etapas: A criação de um teste, A codificação para passar neste teste e a Refatoração do código.

No primeiro ciclo, (Vermelho), também chamado de Test, o desenvolvedor cria um código que dê um erro,pois as funcionalidades do teste ainda não foram implementada.

No segundo ciclo, (Verde), também chamado de Code, o desenvolvedor escreve um código com as funcionalidades que consiga passar no Test.

No terceiro ciclo,chamado de Refatorar, o objetivo é melhorar código que está funcionando, atendendo os requisitos, eliminando redundâncias e deixando-o mais limpo.

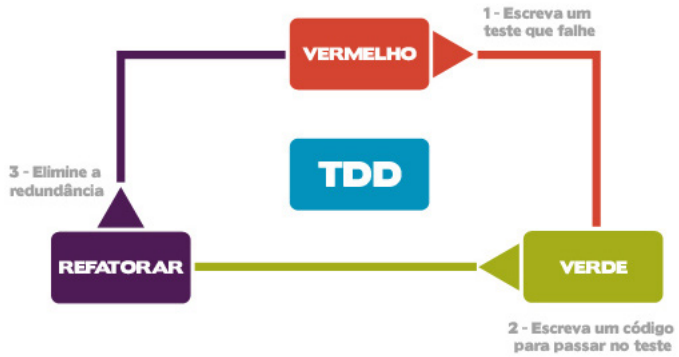

Figura 1 – Ciclo do TDD

2.2.2 Requisitos de Teste

Para a criação de teste deve-se compreender com detalhes a especificação do sistema e as regras de negócio. Além disso, os testes devem seguir o modelo F.I.R.S.T.

F (Fast) - Testes unitários devem ser rápidos;

I (Isolated) - Testes individuais das unidades e não sua integração;

R (Repeatable) - Repetição nos testes, com resultados de comportamento constante;

S (Self-verifying) - Auto verificação relatando a falha ou sucesso do teste;

T (Timely) - O teste deve ser por unidade.

2.2.3 Implementação do TDD

Uma maneira de conseguir testar o sistema por completo é automatizando os testes, utilizando frameworks, executando um teste de uma funcionalidade (teste de unidade) de um código . Existem diversas ferramentas que possibilitam esta prática, como o:

JUnit: Framework de teste para Java, que permite a criação de testes unitários que está disponível como plug-in para IDE'S como Eclipse e Netbeans;

Jasmine: Framework para teste unitário de JavaScript;

CUnit: Ferramenta para os testes unitários disponível para Linguagem C;

PyUnit: Framework Xunit para testes na linguagem Python;

PHPUnit: Framework XUnit para teste unitário em PHP que está disponível como plug-in para algumas IDE'S.

2.2.4 Quando não usar o TDD

A utilização do método TDD trás grandes vantagens pois torna o código mais seguro (além de ele já nascer testado) e mais simples (pois o processo de refatoração busca pela solução mais simples). Ele é muito importante para facilitar códigos de alta complexidade. Entretanto, em ***Test-Driven Development Teste e Design no Mundo Real*** (ANICHE, 2012) considera que o TDD não se mostra tão eficiente durante a implementação de um DAO (classe que faz integração com outros sistemas externos como banco de dados e servidores web), porque neles não existe a necessidade de teste de qualidade do código visto que se trata de scripts de funcionalidades simples.

3 Conclusões

Visto tudo o que foi exposto, entende-se que a metodologia Test Driven Development é feita para atender os requisitos do sistema que está proposto a resolver trazendo também outros benefícios, como a confiabilidade do código e o baixo custo de tempo despendido na construção do sistema. A TDD é baseada em um ciclo de aplicação de três etapas, as quais identificam, solucionam e melhoram a solução respectivamente. Além disso, sempre busca atender os requisitos de teste, o qual pode ser automatizado, e por isso é utilizado em problemas que necessitem de testes de solução.

Apesar de parecer trabalhoso sempre desenvolver os testes para posteriormente desenvolver o código, a TDD encurta bastante o processo de construção de um sistema reduzindo códigos desnessários pelo fato de encerrar em si o objetivo de atender requisitos específicos. No final das contas, a sua motivação é o menor esforço.

Referências

ANICHE, M. *Test-Driven Development Teste e Design no Mundo Real*. 1. ed. São Paulo: Casa do Código, 2012. Citado na página 15.

GOMES, F. *TDD: fundamentos do desenvolvimento orientado a testes*. 2017. Último acesso em 27/04/2018. Disponível em: <https://www.devmedia.com.br/tdd-fundamentos-do-desenvolvimento-orientado-a-testes/28151>. Nenhuma citação no texto.

MASSOL, V.; HUSTED, T. *JUnit em Ação*. 2. ed. [S.l.]: Ciência Moderna, 2005. Citado na página 13.

NAZáRIO, R. L.; RUFINO, R. *O conceito de TDD no desenvolvimento de software*. 2015. Último acesso em 27/04/2018. Disponível em: <http://web.unipar.br/~seinpar/2015/_include/artigos/Renan_Leme_Naz%C3%A1rio.pdf>. Nenhuma citação no texto.