Engenharia de Software

Anhanguera


AVALIE
SUA PROFISSÃO
QUANDO APARECER EM SEU PORTAL UMA AVALIAÇÃO SOBRE SEU CURSO, RESPONDA:
NOTAS
9 OU 10
SIGNIFICA QUE VOCÊ INDICA
NOTAS
7 OU 8
SIGNIFICA QUE VOCÊ NÃO INDICA
Anhanguera

Anhanguera

DESENVOLVIMENTO ORIENTADO A TESTES E FERRAMENTAS

Anhanguera

Tivemos a oportunidade de conhecer técnicas e metodologias de teste que se baseavam em uma sequência bem estabelecida de procedimentos e que eram capazes de acomodar as particularidades de alguns tipos específicos de aplicações. Essa sequência foi assim apresentada:

Implementar código.

Selecionar casos de teste.

Executar o código com os casos de teste selecionados.

Depurar o código para encontrar defeitos.

Corrigir defeitos.

Avaliar resultados.

Considerando o momento da aplicação dos testes, a fase em que ela ocorre é, tradicionalmente, uma das últimas do processo de software, logo após o completo desenvolvimento do produto e antes da entrega ao cliente. Via de regra, qualquer atraso ocorrido durante o projeto impactará no tempo disponível para os testes. Haveria então outro momento mais adequado para aplicá-los? Será que a práticado teste também não teria experimentado uma evolução, impulsionada pelo aprimoramento das metodologias de desenvolvimento? Felizmente a resposta é sim para ambas as questões.

DESENVOLVIMENTO ORIENTADO A TESTES (TDD)

O advento das metodologias ágeis, especificamente o Extreme Programming (XP), ofereceu-nos inovações também no modo como fazemos nossos testes, sendo amais importante delas o TDD. De acordo com Aniche (2014), a ideia central do TDDé a de fazer com que o desenvolvedor escreva testes automatizados de maneira constante ao longo do processo de desenvolvimento e antes mesmo da implementação do código. Ainda segundo o autor, essa inversão no ciclo compele o desenvolvedor a escrever um código de melhor qualidade, já que a escrita de bons testes de unidade requer o bom uso dos recursos da orientação a objetos.

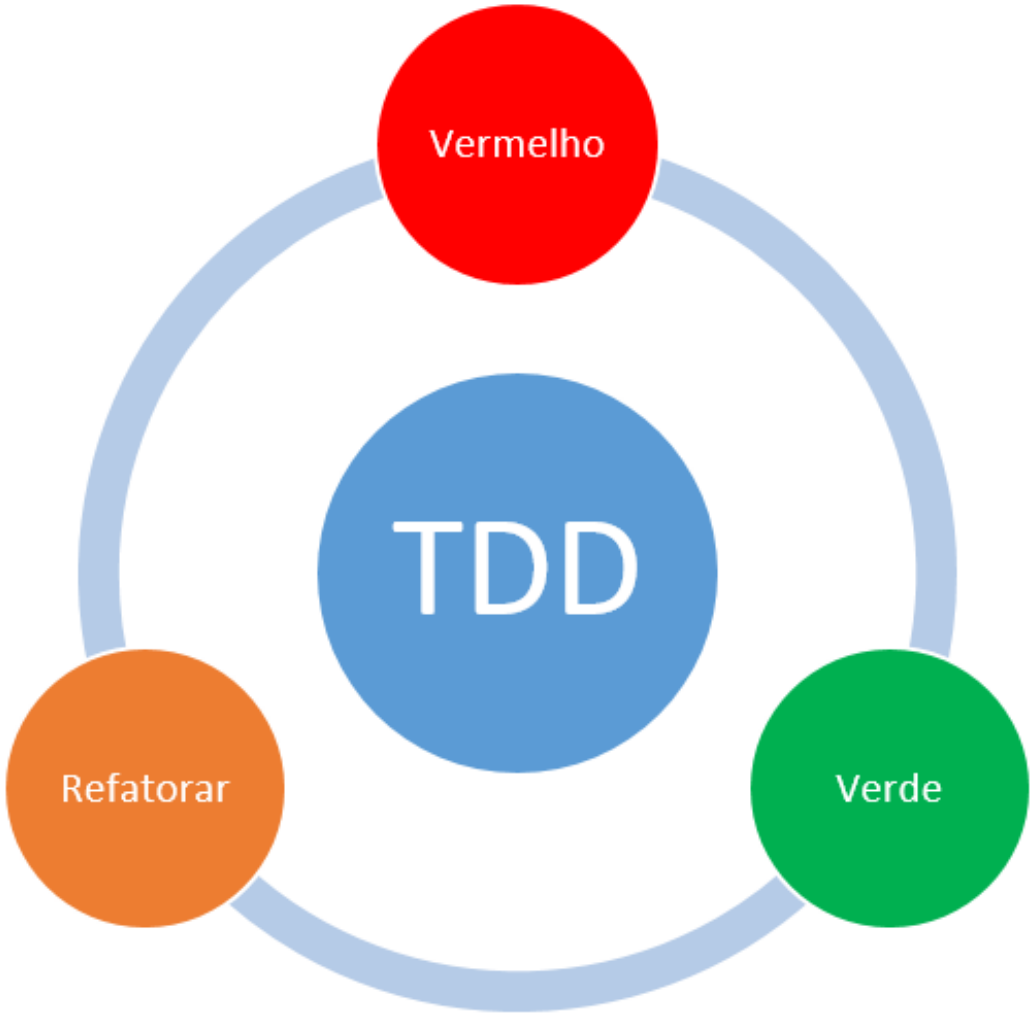
Antes de continuarmos a exploração deste conteúdo, será necessário resgatarmos alguns termos já citados para fazermos uma delimitação do nosso tema. O Desenvolvimento Orientado a Testes será abordado aqui como uma das práticas do XP e, nessa condição, usaremos o teste de unidade como nosso objeto de análise. O fato de o TDD estar situado no contexto do XP explica nossa menção ao paradigma de Orientação a Objetos feita há pouco. Por fim, vale relembrarmos queo teste de unidade é aquele realizado sobre cada classe do sistema e que os testes de aceitação são efetuados sobre cada estória (ou funcionalidade) do sistema(TELES, 2004).

EXEMPLIFICANDO

A ilustração de um teste de unidade será oferecida por meio do exemplo deuma loja virtual qualquer na web. Quando o usuário seleciona um produto para comprá-lo, o sistema se incumbe de colocá-lo no carrinho de compras, de fazer contato com a operadora do cartão de crédito, de retirar o produto do estoque, de informar o pessoal da logística para preparar a entrega e de enviar ao usuário um e-mail de confirmação de compra. É natural imaginarmos que o sistema que gerencia todas essas operações (maisaquelas que não mencionamos) seja bastante complexo e que, portanto, esteja implementado em diversas classes, cada qual com uma função específica.

Ao invés de focar o sistema todo, o teste de unidade se preocupará com cada uma dessas classes, que geralmente é a unidade de um sistema Orientado a Objetos. Em nosso sistema de exemplo, muito provavelmente existem classes como CarrinhoDeCompras, Pedido e assim por diante. Aideia é termos baterias de testes de unidade separadas para cada uma dessas classes, com cada bateria preocupada apenas com a sua classe(ANICHE, 2014). Bem, e como o Desenvolvimento Orientado a Testes funciona? Descobriremos isso ao longo desta primeira etapa da seção e a iniciaremos pela apresentação de uma figura que se tornou referência universal quando se trata desse assunto. A aplicação do Desenvolvimento Orientado a Testes é ilustrada pelo ciclo Vermelho-Verde-Refatorar, mostrado na Figura 3.11.

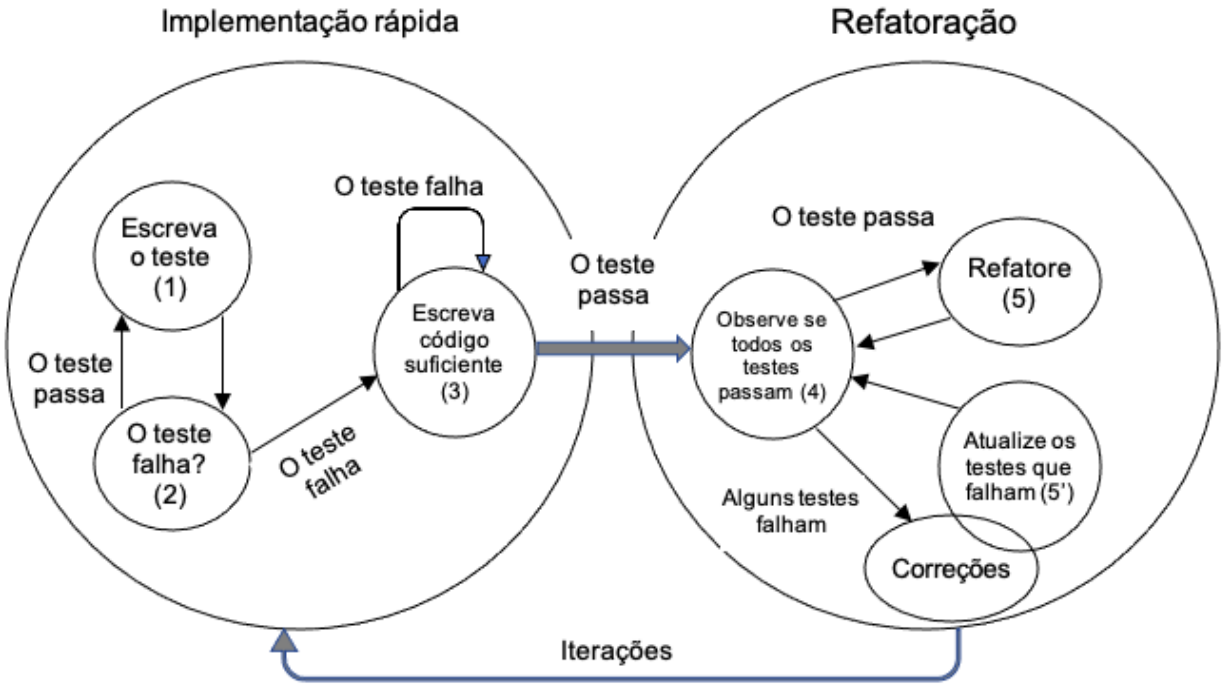
Anhanguera



Essa representação pode ser traduzida de modo textual da seguinte forma:

1. Na primeira fase do ciclo, o desenvolvedor escreve um teste queintencionalmente falhará. O código em que ele está inserido provavelmente sequer compilará.
2. Na sequência, o desenvolvedor escreve um código – referente a uma novafuncionalidade do sistema – que fará o teste escrito na etapa anterior passar.
3. Como última dessas três etapas, o desenvolvedor aplicará a refatoração nocódigo, eliminando eventuais duplicidades, estruturando melhor o código,mudando o nome de variáveis, entre outras providências.

Embora essa ilustração nos dê ideia de um ciclo, ela não deixa claro como ele seefetiva, principalmente para quem se propõe a usar o TDD pela primeira vez. AFigura 3.12 nos oferece visão detalhada do procedimento. Na verdade, o TDD contém duas partes: implementação rápida e refatoração e, na prática, o teste para implementação rápida não se limita ao teste de unidade. Pode ser um testede aceitação também.



Observe que, na parte de implementação rápida, o procedimento executado pelo desenvolvedor é visualmente descrito da mesma forma que o fizemos textualmente, ou seja, (1) o desenvolvedor escreve um teste que deverá falhar e (2) verifica se, de fato, o teste falha. Caso não falhe, o teste deverá ser reescrito. Caso falhe, (3) o desenvolvedor deverá escrever o código para que ele passe. Neste ponto um registro deve ser feito: a expressão “código suficiente” indica que o desenvolvedor só deverá codificar o suficiente para que o teste passe, nem mais, nem menos. Quando o teste passa, a parte da refatoração começa e, como primeira providência dessa etapa, (4) o desenvolvedor deve verificar se todos os testes passam e (5) aplicar a refatoração no código. Caso um ou mais testes falhem, (5’) eles devem ser atualizados e eventuais correções devem ser feitas. Aseta com a legenda de “Iterações” indica que esse procedimento deve serexecutado até o fim do procedimento.

Neste ponto cabe um exemplo da aplicação do TDD e novamente usaremos a loja de comércio eletrônico para desenvolvê-lo, segundo Aniche (2014). A classe exibida no código do Código 3.7 aponta o produto de maior valor e o produto de menor valor no carrinho e o faz percorrendo a lista de compras e comparando valores.

Anhanguera

```java
public class MaiorEMenor {
	private Produto menor;
	private Produto maior;
	public void encontra (CarrinhoDeCompras carrinho) {
		for (Produto produto : carrinho.getProdutos()) {
			if (menor == null || produto.getValor() < menor.getValor()) {
				menor = produto
			}
			else if (maior == null || produto.getValor() > maior.getValor()) {
				maior = produto;
			}

			}
		}
	}
	public Produto getMenor() {
		return menor;
	}
	public Produto getMaior() {
		return maior;
	}
}
```

O método encontra() recebe um CarrinhoDeCompras e percorre a lista deprodutos desse carrinho, comparando sempre o produto atual com o menor emaior já encontrados. Ao final da execução, temos nos atributos maior e menor osprodutos desejados. O Código 3.8 exemplifica o uso da classe MaiorEMenor.

Observe que o carrinho contém três itens: o que tem preço menor é o jogo depratos e o que tem preço maior é a geladeira. Ao executar essa aplicação, a saída será:

O menor produto: jogo de pratos

O maior produto: geladeira.

Anhanguera

```
1  public class TestaMaiorEMenor {
2  public static void main(String[] args) {
3
4  CarrinhoDeCompras carrinho = new CarrinhoDeCompras();
5  carrinho.adiciona(new Produto("Geladeira", 450.0));
6  carrinho.adiciona(new Produto("Liquidificador", 250.0));
7  carrinho.adiciona(new Produto("Jogo de pratos", 70.0));
8
9  MaiorEMenor algoritmo = new MaiorEMenor();
10 algoritmo.encontra(carrinho);
11
12 System.out.println("O menor produto: " +
   algoritmo.getMenor().getNome());
13 System.out.println("O maior produto: " +
   algoritmo.getMaior().getNome());
14     }
15  }
16 }
```

Sendo assim, podemos concluir que a aplicação funciona, certo? Ainda não. Precisamos verificar se ela funciona também em outro cenário. Se o código do desenvolvedor fizesse a inserção dos mesmos produtos em ordem diferente(geladeira, liquidificador e jogo de pratos), a saída gerada seria a seguinte:
menor produto: jogo de pratos.
Exception in thread "main" java.lang.NullPointerException atTestaMaiorEMenor.main(TestaMaiorEMenor.java:5).
É fato que, se os produtos forem adicionados em ordem decrescente, a classe não retorna a saída esperada e o cliente não conseguirá efetuar sua compra. Essa situação nos leva a duas conclusões:
1. Testar constantemente, considerar vários cenários e repetir o procedimento a cada mínima alteração feita são providências indispensáveis em um procedimento de teste.
2. Proceder manualmente (ou seja, sem uma ferramenta de teste), conforme determina o item anterior, é impraticável.

Ainda nesta seção trataremos de testes automatizados de software como forma de apresentarmos uma solução possível para este caso. Antes, porém, colocaremos como tema algumas questões sobre o teste de unidade e seu gerenciamento, sempre no âmbito do Desenvolvimento Orientado a Testes, e, na sequência, será apresentada uma solução viável para a situação que acabamos de tratar.

GERENCIAMENTO DE TESTES

Na maioria dos casos, criar testes antes para codificar depois não é exatamente o modelo de desenvolvimento com o qual um profissional de TI está acostumado. Como se a falta de familiaridade com esse estilo não fosse obstáculo importante o suficiente, há questões relacionadas ao teste de unidade que demandarão odevido gerenciamento por parte de quem estiver à frente deles. É bom lembrarmos que o teste de unidade é o elemento central do TDD, embora ele possa ter como objeto também o teste de aceitação. Na visão de Teles (2004), as situações a seguir requerem bom gerenciamento para que a condução do TDD seja satisfatória.

Como escrever testes quando o sistema faz acesso a um banco de dados?

Nesta questão, o desempenho é um aspecto a ser seriamente considerado no procedimento de teste. É necessário que os testes de unidade sejam executados muito rapidamente, de modo que o ciclo “testar-codificar-refatorar” seja completado também rapidamente. O desempenho do teste será tanto mais significativo quanto mais numerosas forem as classes que acessam um banco dedados. O gerenciamento da situação inclui fazer com que uma boa quantidade detestes que usam dados do banco passe a acessar arquivos na memória volátil, emvez de o fazerem no registro em disco. Uma boa alternativa seria escrever uma classe “falsa”, que atuasse como um banco de dados, interceptando os comandosSQL enviados pela aplicação. Embora essa solução seja viável, em algum momento o acesso ao banco real deverá ser testado, mas o bom gerenciamento da situação deverá providenciar que tais acessos sejam feitos com a menor frequência possível.

O que o desenvolvedor deve fazer quando não tiver ideia de como testar uma classe?

Embora a condução dessa situação esteja diretamente relacionada à classeespecífica, ainda assim é natural imaginar que estamos diante de uma classeproblemática. Se a instanciação da classe é feita de forma complexa, é provávelque um padrão viável de codificação não foi seguido, o que torna aconselhável arevisão do código. Se a classe sob teste colabora com diversas outras classes de complexidade elevada, a providência deverá passar pela criação de mock objects, definidos como objetos falsos, os quais simulam o comportamento de objetos reais e mais complexos.

O gerenciamento dessa situação passa por reuniões com a equipe de desenvolvedores sobre o que tem sido difícil submeter ao TDD.

Como saber se foi testado tudo o que poderia dar errado?

As características de bom senso e de experiência são componentes desta questão. Quando o sistema manifesta um erro no teste de aceitação, há boa chance de estar faltando um teste de unidade. Ao escrever esse teste, o desenvolvedor deveser orientado a lembrar-se de outros testes que pode não ter escrito e essa providência tenderá a torná-lo mais desenvolto na questão da completude dos testes.

Os desafios que envolvem o TDD não se resumem a esses que abordamos. Será normal nos depararmos com quem presume ser esta uma metodologia queaumenta o trabalho da equipe de desenvolvimento. No entanto, como o TDD prevê que os testes devem ser escritos antes da efetiva codificação, a simplificação da implementação das classes deverá ser um dos bons efeitos desse estilo dedesenvolvimento. Como não podemos nos esquecer de que estamos tratando deuma prática do XP, é necessário mencionar que a programação em par constitui um ponto altamente positivo quanto à celeridade da construção dos testes, justamente pela forte interação que se estabelece entre o par.

Embora a condução dessas questões deva estar sempre na ordem do dia dos gestores envolvidos em testes, o gerenciamento do processo de teste envolve outros elementos procedimentais e outras ações, já introduzidas em conteúdos anteriores, quando tratamos de planos de testes. Cabe-nos, portanto, fazer alguns resgates de conceitos pontuados naquela seção e apresentar uma ferramenta de gerenciamento de teste bastante utilizada: a TestLink. Antes de tratarmos especificamente dela, vale a observação de que não apenas o procedimento de teste deve ser automatizado, como teremos oportunidade de constatar na sequência. O gerenciamento desse procedimento, por meio de uma ferramenta,implica facilidades que vão desde o cadastramento de projetos de teste até a designação dos testadores responsáveis por um conjunto de testes.

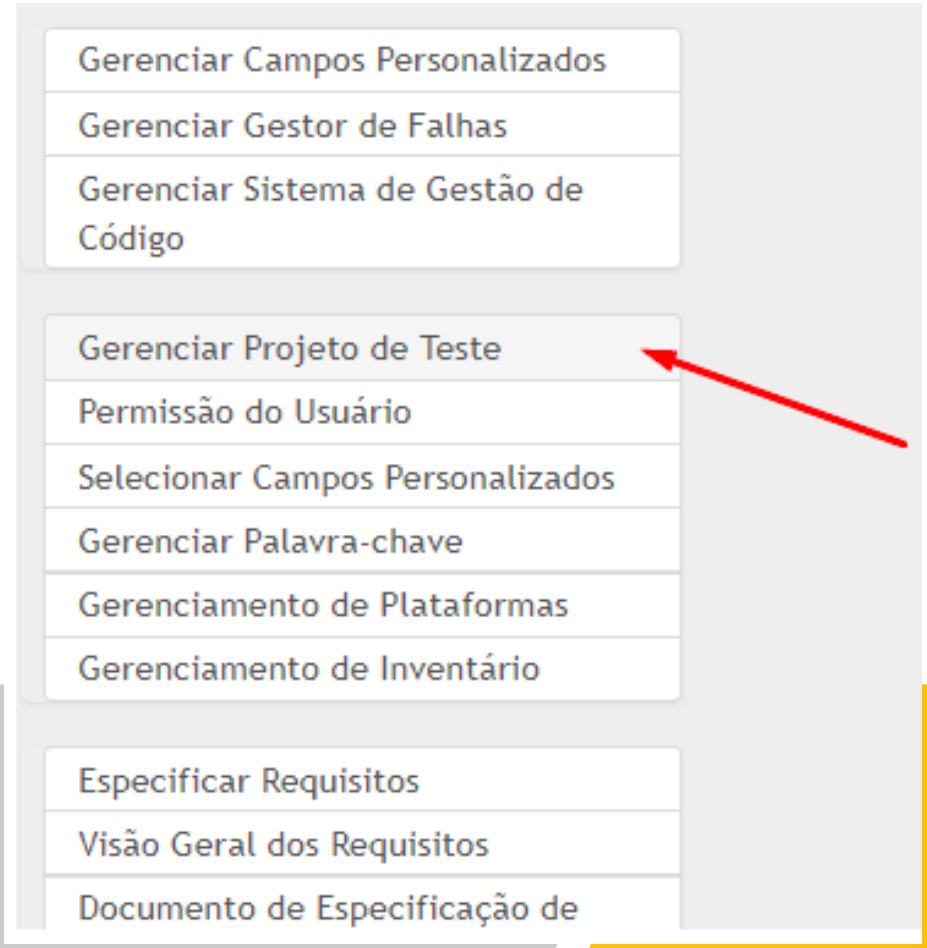
A apresentação da TestLink tem início pela menção de que se trata de uma ferramenta open source automatizada escrita em PHP e com interface web, cujo principal objetivo é dar suporte às atividades de gestão de testes, permitindo acriação de planos de testes e a geração de relatórios com diversas métricas para o acompanhamento da execução deles. Por meio dessa ferramenta, é possível associar casos de teste a requisitos específicos do produto (CAETANO, [s.d.])

O download da ferramenta deve ser feito no site TestLink (TESTLINK, [s.d.]) e, emvez de apresentarmos aqui os procedimentos de instalação, apresentaremos algumas telas da versão 1.9.13, que reproduzem a criação de um plano de testes, segundo Reis (2018). A Figura 3.13 exibe a tela de login da ferramenta.

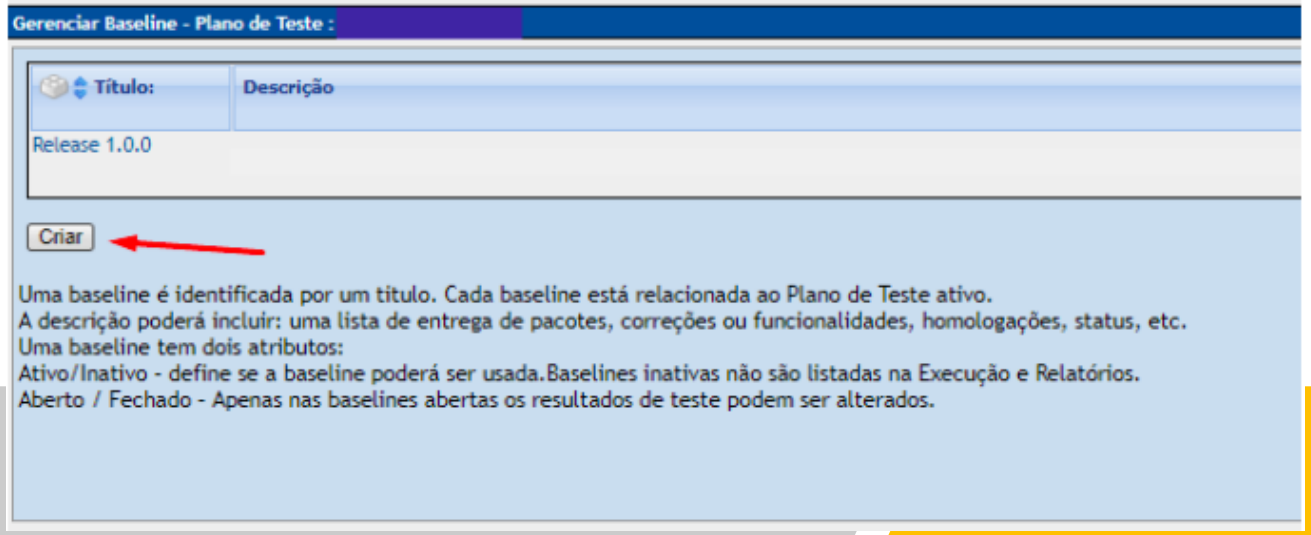
Anhanguera


TestLink
1.9.13 (Stormbringer)
Please log in ...
Login
Password
Log in

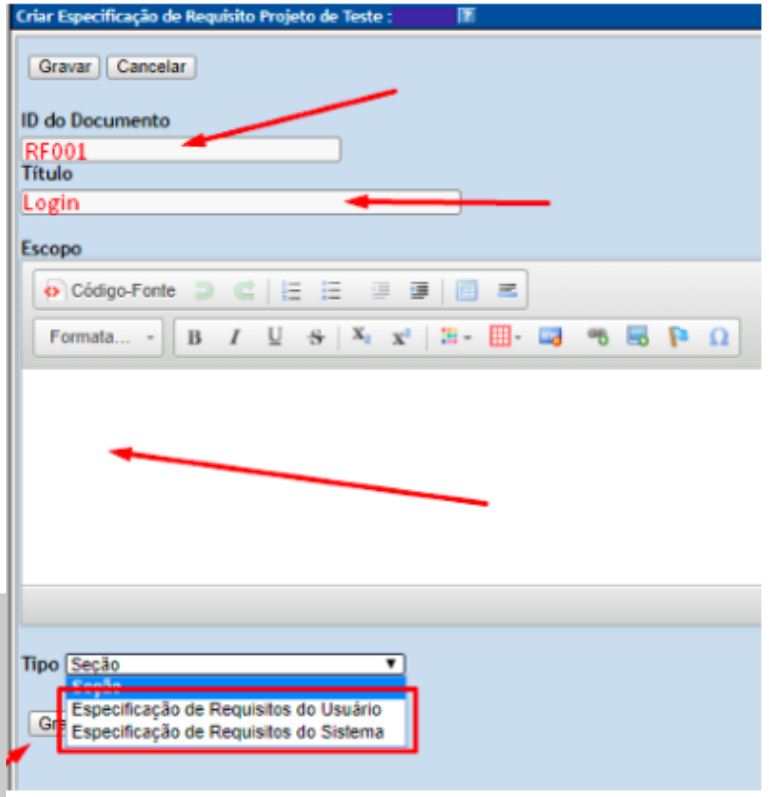
Após obter acesso à ferramenta, seu usuário poderá criar um Projeto de Teste clicando no link Gerenciar Projeto de Teste, conforme exibido na Figura 3.14.



A partir dessa escolha, o usuário terá acesso ao formulário de criação do Projeto de Teste, por meio do qual poderá informar um nome, a descrição do projeto e habilitar algumas funcionalidades, incluindo a possibilidade de automatizar o teste. A ferramenta deverá, então, habilitar o gerenciamento do plano de teste e, nesse contexto, uma baseline/release do projeto poderá criada. A Figura 3.15 exibe o resultado dessa criação e as orientações fornecidas pela ferramenta.



Seguindo esse mesmo padrão procedimental e visual, a ferramenta também permite o registro de requisitos e de casos de teste, sempre associados ao projetoatual. Na Figura 3.16, é possível visualizar o contexto de especificação de requisitos do software.



Embora a TestLink ofereça inúmeras outras funcionalidades, fica claro que se trata de um recurso extremamente valioso nos casos em que vários processos de testeestão sendo executados simultaneamente. À propósito, já que mencionamos algovalioso, que tal estudarmos os procedimentos e os benefícios de um teste automático? Siga conosco.

TESTES AUTOMATIZADOS DE SOFTWARE

Em momentos anteriores desta seção, esclarecemos, por meio de um exemplo, que testar uma unidade considerando vários cenários e por diversas vezes é umaação fundamental para a obtenção da desejada qualidade do produto. Emboraesta seja uma necessidade, não se pode atendê-la apenas contando com oempenho humano. É preciso automatizar o teste. Teles (2004) ensina que os testesde unidade são automatizados na forma de classes do sistema, as quais têm como objetivo testar outras classes. O autor complementa que, de modo geral, para cada classe do sistema deverá existir uma outra com o único objetivo de testá-la e queesse é o passo inicial para que seja possível automatizar os testes de unidade.

De fato, automatizar o processo reduz o custo e o tempo do teste, se é que podemos separar esses dois fatores. Aniche (2014) afirma que escrever um teste automatizado não é tarefa complexa, pois se assemelha a um teste manual. Para validarmos essa afirmação, voltemos à aplicação de comércio eletrônico. Dessa vez, o desenvolvedor precisa testar o carrinho da loja virtual e, para este exemplo, consideraremos que há dois produtos cadastrados na loja. Ao simular o comportamento de um cliente, ele seleciona os dois produtos, segue para o carrinho de compras e finalmente verifica a quantidade de itens existentes nele. Essa quantidade deve ser dois e o valor da compra deve ser a soma dos valores dos dois produtos.

O que esse desenvolvedor fez foi imaginar um cenário (a compra de doisprodutos), executar uma ação (colocá-los no carrinho) e verificar o resultado(checar a quantidade e o valor dos itens comprados). Em um teste automatizadode software, a sequência a ser cumprida deverá ser exatamente essa e a intervenção humana fica restrita, neste caso específico, à conferência do valor obtido com o valor esperado. Com um breve retorno ao Código 3.8 , veremos que aquela classe monta um cenário (um carrinho de compras com três produtos),executa uma ação (chama o método encontra()), e valida a saída, que significa imprimir o maior e o menor produto. A simples execução da classe monta o cenário e executa a ação, sem intervenção do desenvolvedor (ANICHE, 2014).

Embora já tenhamos automatizado duas das três etapas da sequência, ainda seránecessário tornar a conferência da saída independente da intervenção humana. Aplena automatização do teste virá com a utilização do JUnit, o framework de testesde unidade do Java, que funciona em conjunto com o IDE Eclipse. A adaptação do nosso código ao JUnit será simples e atingirá a parte da validação, na qual os métodos do framework serão invocados para que comparem o resultado esperado com o obtido. O método Assert.assertEquals() fará esse trabalho. O Código,3.9, mostra a classe TestaMaiorEMenor
ajustada para funcionamento do JUnit.Como sabemos, esse teste falhará, pois há um defeito na classe MaiorEMenor, instanciada pela TestaMaiorEMenor. A presença do else naquele código não permite que o segundo if seja executado e, portanto, o maior elemento nunca é verificado. Com esse defeito corrigido, o teste passará.

Anhanguera

```
1  import org.junit.Assert;
2  import org.junit.Test;
3  public class TestaMaiorEMenor {
4     @Test
5     public void ordemDecrescente() {
6          CarrinhoDeCompras carrinho = new CarrinhoDeCompras();
7          carrinho.adiciona(new Produto("Geladeira", 450.0));
8          carrinho.adiciona(new Produto("Liquidificador", 250.0));
9          carrinho.adiciona(new Produto("Jogo de pratos", 70.0));
10
11         MaiorEMenor algoritmo = new MaiorEMenor();
12         algoritmo.encontra(carrinho);
13         Assert.assertEquals("Jogo de pratos",
   algoritmo.getMenor().getNome());
14         Assert.assertEquals("Geladeira",
   algoritmo.getMaior().getNome());
15     }
16 }
```

Dessa forma é que um teste automatizado se efetiva. Naturalmente que a complexidade, a quantidade e o tamanho das classes tornarão os testes mais ou menos complexos, mas a forma geral não se altera significativamente. A agilidade no procedimento, a redução de custos e a capacidade de o teste ser repetido quantas vezes forem necessárias são apenas algumas poucas vantagens da automação.

Antes de terminarmos esta seção, cabe ainda uma questão: com tantas vantagens e recursos interessantes, só temos uma ferramenta de automação de testesdisponível?

FERRAMENTA PARA TESTES DE SOFTWARE

Por meio do exemplo de teste automatizado que acabamos de desenvolver, você teve contato com uma das mais conhecidas e utilizadas ferramentas de teste que existem: a JUnit. No entanto, ela está longe de ser a única e, dada a importante função técnica exercida por essas ferramentas, dedicaremos as próximas linhas à explanação de uma delas.

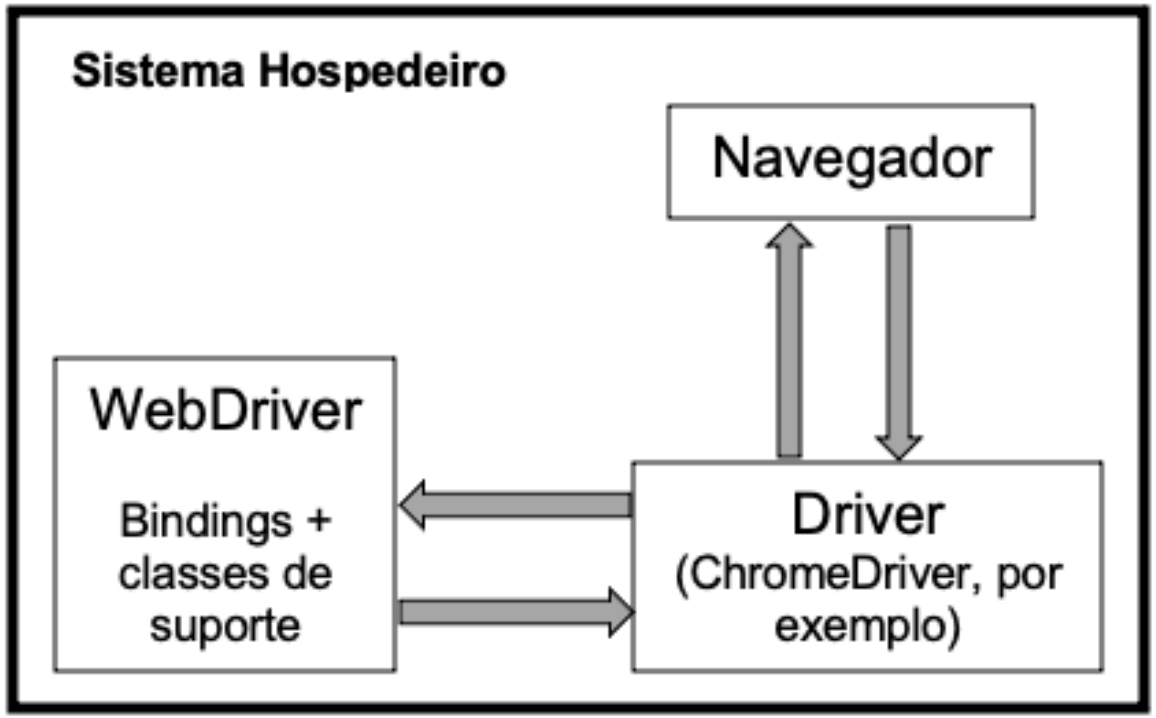
SELENIUM

Trata-se de uma ferramenta de testes open source, multiplataforma e com várias frentes de utilização, especialmente em testes de aplicações web. O site oficial da ferramenta (SELENIUM, [s.d.]) apresenta três projetos e indica o mais apropriado para cada aplicação.

Selenium WebDriver: indicado para quem deseja criar testes automatizados emaplicações baseadas em navegador. O Selenium WebDriver é capaz de automatizarinterações com a maioria dos navegadores, tanto local quanto remotamente erealizar a simulação da operação de um usuário real é o seu objetivo.

Observe a Figura 3.17. Por meio dela é possível observar que o WebDriver secomunica com um navegador através de um

Driver de modo bidirecional: o WebDriver passa os comandos para o navegador pelo driver e recebe as informações pela mesma rota. O driver é específico para o navegador, como o ChromeDriver é para o Chrome, como o GeckoDriver é para o Mozilla Firefox, entreoutros. O driver é executado no mesmo sistema do navegador.


Sistema Hospedeiro
Navegador
WebDriver
Bindings + classes de suporte
Driver
(ChromeDriver, por exemplo)

Por meio do WebDriver, o Selenium oferece suporte a todos os principais navegadores do mercado, como o Chrome, o Firefox, o Internet Explorer, o Operae o Safari. Sempre que possível, o WebDriver dirige o navegador usando o suporte integrado dele para a automação, mesmo que nem todos os navegadores tenhamsuporte oficial para controle remoto. Embora todos os drivers compartilhem uma única interface voltada ao usuário para controlar o navegador, eles têm maneiras ligeiramente diferentes de configurar suas sessões. Como muitas das implementações de driver são fornecidas por terceiros, eles não estão incluídos na distribuição padrão do Selenium (SELENIUM, [s.d.]).

Selenium IDE: trata-se de uma extensão para os navegadores Chrome e Firefoxque permite a gravação e a reprodução dos testes neles.

Selenium Grid: esta modalidade do Selenium é capaz de executar testes em váriasmáquinas ao mesmo tempo, reduzindo, assim, o tempo necessário para realizartestes em vários navegadores ou Sistemas Operacionais.

Este foi, portanto, o conteúdo que queríamos compartilhar com você nesta seção. Longe de ser uma mera opção, automatizar testes é uma providência absolutamente necessária em ambientes profissionais de Engenharia de Software. Como tivemos oportunidade de discutir, as reduções nos custos e no tempo de execução excedem qualquer tempo que se leve para que se possa conseguirfamiliaridade com as ferramentas de teste. A respeito do custo, inclusive, tivemos a oportunidade de conhecer duas ferramentas com custo zero de aquisição e implantação.

Observe o esquema expresso na figura abaixo.

Em relação a ele, avalie as afirmações que seguem:

I.Trata-se das etapas de um teste executado em uma unidade e que são passíveis de serem automatizadas por uma ferramenta.

II.Representa a criação de um cenário de teste, ou seja, uma ação que seria executada pelo usuário e a checagem do resultado.

III.A sequência pode ser alterada sem que o resultado seja prejudicado, já que ela representa o funcionamento de uma ferramenta automatizada.

É correto o que se afirma em:

a. II e III apenas.

b. II apenas.

c. I apenas.

d. I e II apenas.

e. I, II e III.

Imagine o seguinte cenário: ao iniciar a oficialização do TDD como metodologia de testes da empresa de desenvolvimento que gerencia, você se deparou com algumas dificuldades que, apesar de todo seu cuidado de planejamento, não haviam sido previstas. Após conseguir delineá-las, você as expressou textualmente para que pudesse discuti-las com a equipe, ação que resultou no seguinte:

I .As barreiras técnicas para a implantação do TDD justificam o atraso no atingimento desse objetivo.

II. Os desenvolvedores consideram que o TDD aumentará o trabalho deles e o tempo investido não será compensado a longo prazo.

III. Os proprietários da empresa entendem que a programação em par nada mais é do que um meio de atingir com dois desenvolvedores o objetivo que poderia ser atingido com apenas um.

Com base nesse cenário, assinale a alternativa cujos itens, de fato, representamum fator que dificulta a implantação do TDD.

a. II apenas.

b. I apenas.

c. II e III apenas.

d. I, II e III.

e. I e III apenas.