ANÁLISE DE FERRAMENTAS DE TESTES BASEADAS EM BDD

ANALYSIS OF TEST TOOLS BASED ON BDD

Valter José do Monte Negreiros Filho
valternegreiros@gmail.com

RESUMO

Este artigo tem como tema principal de sua abordagem uma grande área da Ciência da Computação, Testes de Software, focado nas ferramentas de testes automatizados em que utiliza uma linguagem estruturada baseada na linguagem natural. Em estudos realizados foi percebido que ferramentas voltadas para Teste de Software ganham cada vez mais espaço no mercado. Em cima dessa observação, foram analisadas ferramentas com critérios baseado no SQFD onde a ferramenta Cucumber teve melhor aceitação e foi utilizada para o estudo de caso.

Palavras-chave: linguagem natural. Teste de software. Cucumber.

ABSTRACT

This article's main theme of his approach a large area of Computer Science, Software Testing, focused on automated testing tool that uses a structured language based on natural language. In studies it was realized that tools geared for Software Testing gain more space on the market. Upon this observation criteria with tools were analyzed based on SQFD where Cucumber tool had better acceptance and was used for the case study

Keywords: natural langurage. Software testing. Cucumber.

1 INTRODUÇÃO

De acordo com Veloso (2010), as ferramentas de geração de testes geralmente especificam os casos de teste gerados em uma linguagem não natural (comumente uma linguagem formal). Entretanto, essa linguagem pode ser não trivial para os engenheiros que executarão manualmente os casos de testes. De

acordo com o fato apresentado, seria de grande ajuda uma ferramenta que mapeasse os casos de testes gerados em uma linguagem natural.

Este artigo em seu escopo apresenta ferramentas para escrever estórias de usuário em linguagem natural estruturada, de acordo com as estórias escritas, serão implementados testes de validação, com o intuito de ajudar os *stakeholders* a executarem testes manuais.

As ferramentas serão avaliadas e executadas em um ambiente de desenvolvimento real, nesses aspectos foram escolhidas ferramentas *open-source* executadas em um ambiente de um Software Público do Portal Brasileiro (GGAS).

Foram escolhidas as ferramentas de construção de cenário de testes que utilizam linguagem natural estruturada, as ferramentas são: *Pyccuracy, FitNesse, Cucumber.*

Em seguida foi escolhido o ambiente do GGAS por ser um software disponível no Portal de Software Público Brasileiro (SPB), ter o código disponível em *open-source* e refletir um cenário real de desenvolvimento assim descrito em seu site (Sobre o Portal, 2015). O GGAS é um software livre e público, desenvolvido para apoiar a Gestão Comercial das empresas de distribuição de gás natural. A solução realiza o gerenciamento e automatização dos processos comerciais das companhias de gás, trazendo mais agilidade, transparência e segurança à gestão.

No site do Portal (2015), descreve que o Portal de Software Público Brasileiro é um ambiente de compartilhamento do Software Público Brasileiro, que é um tipo específico de software que adota um modelo de licença livre para o código-fonte. A proposta do portal, ao fornecer um espaço de compartilhamento de soluções, resulta em uma gestão de recursos e gastos de informática mais racionalizada, ampliação de parcerias e reforço da política de software livre no setor público brasileiro.

Assim como Veloso (2010), descreve em seu artigo, as métricas de avaliação que foram utilizadas como base surgiu do processo de SQFD que está mais detalhado em seu próprio trabalho.

Este artigo tem como objetivo apresentar as principais ferramentas que têm como base o BDD, avaliá-las e usar a superior em análise como estudo de caso, os fundamentos de análise que surgiram do processo de SQFD que está descrito e com detalhes no trabalho de Veloso (2010).

Este artigo possui a seguinte estrutura: A seção 1 apresenta a Contextualização do Problema e Objetivo, a seção 2 o Referencial Teórico onde são apresentados as ferramentas e os critérios de avaliação, na seção 3 o estudo de caso onde contém a avaliação das ferramentas e a criação de um cenário de teste com a ferramenta *Cucumber*, na seção 4 é apresentado a conclusão deste artigo, na seção 5 são apresentados os Agradecimentos a Instituição de ensino IFPE e ao orientador de pesquisa, na seção 6 deste artigo é apresentado a proposta de trabalhos futuros, na seção 7 são apresentadas as referências.

2 FERRAMENTAS

2.1 *Pyccuracy*

Ferramenta *open-source* de desenvolvimento de testes baseado em BDD (*Behavior Driver Development*), ou Desenvolvimento guiado por comportamentos, que visa tornar mais fácil a escrita de testes de aceitação automatizados. Foi escolhida por já ser utilizada comercialmente na Yahoo! Brasil. Atualmente suporta múltiplos idiomas, pois é baseada na estrutura original da linguagem *Gherkin*, sendo esta uma das maiores metas do *Pyccuracy* desde o primeiro lançamento, atualmente suporta apenas Inglês e Português. Segundo Laís (2013), e a própria comunidade do *Pycuraccy* a ferramenta segue uma estrutura lógica de uma estória de usuário, onde possui um título, uma narrativa e critérios de aceitação (cenários).

2.2 *FitNesse*

Ferramenta open-source, baseado em um conjunto de páginas interligadas, e cada uma delas pode ser visitada e editada por qualquer pessoa. Você pode editar esta página, clicando no separador no início da página (ou no link no fim da página, dependendo do template que estiver usando), mais conhecida como Wiki, nessas páginas criadas são criados casos de testes em Linguagem Natural em que as especificações podem ser executadas estaticamente contra a aplicação.

Têm várias vantagens sobre o teste funcional tradicional. Estas vantagens são boas tanto para o gerenciamento de requisitos e para verificação de recursos. Somente os testes de aceitação automatizados (tais como o previsto pelo FitNesse) permite medir uma meta de testes a ser coberto, que é uma métrica importante para medir o progresso de projetos de software.

FitNesse é um complemento natural para *xUnit* (baseado em testes de unidade). O *xUnit* é um framework ajuda a construir o código de teste de unidade, enquanto o *FitNesse* ajuda a construir o código a partir de casos de testes escritos pelos *stakeholders*. Considere o uso de ambas as ferramentas (Sobre o *FitNesse*, Acesso em 2016).

2.3 Cucumber

Ferramenta *open-source* conhecida popularmente no mercado, ele tem a função de unir estórias de usuário e casos de testes em um único documento totalmente estruturado e coeso. Como a estória de usuário é usada para testes, temos uma consequência ótima por isso, os documentos e especificações estarão sempre atualizados com o projeto.

O *Cucumber* utiliza o *Gherkin* uma DSL (*Domain Specific Language*, Linguagem de domínio específico). Podendo-se utilizar uma linguagem natural estruturada onde sua estrutura possui um título da funcionalidade, o título do cenário e as etapas (Sobre o *Gherkin*, Acesso em 2016).

De acordo com Nick Sieger (2015), As especificações executáveis do *Cucumber* incentivam uma colaboração mais estreita, ajudando as equipes a manter o objetivo de negócio em mente em todos os momentos. O *Cucumber* possui uma comunidade mais ativa com blogs, eventos, fóruns e treinamentos.

3 METODOLOGIA

Dan North (2012) descreveu em seu artigo que Linguagem natural

(língua humana, língua idiomática, ou somente língua ou idioma) é qualquer linguagem desenvolvida naturalmente pelo ser humano, de forma não premeditada, como resultado da facilidade inata para a linguagem possuída pelo intelecto humano. Vários exemplos podem ser dados como as línguas faladas e as línguas de sinais.

A linguagem natural é normalmente utilizada para a comunicação. As línguas naturais são diferentes das línguas construídas e das línguas formais, tais como a linguística computacional, a língua escrita, a linguagem animal e as linguagens usadas no estudo formal da lógica, especialmente da lógica matemática.

Ferramentas que tratam os cenários de teste com linguagem natural baseiam-se na abordagem do BDD (Desenvolvimento Guiado por Comportamento), a partir de uma Linguagem Natural Estruturada.

De acordo com Dan North (2010), *Behavior Driven Development* (Desenvolvimento Guiado por Comportamento ou BDD) é uma técnica de desenvolvimento ágil que encoraja a colaboração entre os desenvolvedores, setores de qualidade e pessoas não técnicas ou de pessoas de negócios em um projeto de software.

O BDD foi originalmente concebido em 2003, por Dan North como uma resposta à *Test Driven Development* (Desenvolvimento Guiado por Testes), e tem se expandido bastante nos últimos anos.

De acordo com a sintaxe da linguagem *Gherkin* (Acesso em 2015), as ferramentas mostradas na seção a seguir, *Pycuraccy* e o *Cucumber* utilizam uma linguagem natural estruturada chamada *Gherkin*, ao qual a sintaxe é utilizada por usuários técnicos e não técnicos para descrever as funcionalidades. Uma funcionalidade é iniciada por uma palavra chave ao qual deve possuir uma estrutura lógica, podendo ser escrita em até 30 idiomas e utilizando o “#” no início da linha para comentários e a sua estrutura é formada por títulos e descrições no início do documento “Funcionalidade”, “Contexto”, “Cenário”, seguidos de passos “Dados”, “Quando”, “Então”, “E”, “Mas”.

4 CRITÉRIOS DE AVALIAÇÃO

Veloso (2009), usou em seu trabalho critérios de avaliação com base no SQFD (*Software Quality Function Development*), e propôs o seu método de avaliação para avaliações futuras de ferramentas de testes onde tem como objetivo melhorar o desenvolvimento de software aplicando técnicas de melhoria de qualidade, focado nas especificações de requisitos.

4.1 Análise das Ferramentas

Tabela 1, Quadro comparativo entre as ferramentas de testes de desempenho *Pyccuracy*, *FitNesse* e *Cucumber*. IFPE 2016.

Satisfação do Cliente	24,605	25,145	**25,575**

Fonte: O Autor (2016).

4.2 Estudo de Caso

Para validar a análise das ferramentas é preciso um cenário real de desenvolvimento, onde foi escolhido o ambiente do GGAS um por ser um software disponível no Portal de Software Público Brasileiro (SPB) e ter o código disponível em open-source e refletir um cenário real de desenvolvimento.

Dentre as ferramentas, foi escolhido a que foi superior segundo as análises feitas anteriormente, O *Cucumber*.

Para obter o *Cucumber* existem diversas formas, onde pode ser incluído em ferramentas de inclusão de dependências como o *Maven*, *Gradle* ou até mesmo o *Bundler*.

A ferramenta usada neste artigo foi o *Bundler*, disponibilizada no site da *Ruby On Rails*: <http://www.railsinstaller.org/pt-BR>, depois de baixa-lo e instalá-lo é necessário abrir o Terminal ou *Prompt* de Comando.

Criamos a pasta *Cucumber*, e dentro da pasta *Cucumber* criamos as pastas '*features'* para criação das funcionalidades do *Cucumber*, dentro da pasta '*features'* criamos as sub-pastas '*step_definitions'* para criação dos Scripts de Testes e a '*support'* dentro da sub-pasta '*support'* criamos o arquivo 'env.br' para o *Cucumber* automaticamente fazer o download das dependências utilizadas no script de teste, só então executamos o Terminal ou Prompt de Comando e dentro da pasta do *Cucumber* executamos os seguintes comandos:

gem install bundler / bundle init / bundle install

Esses comandos instalam o *Bundler*, ferramenta utilizada para baixar o *Cucumber*.

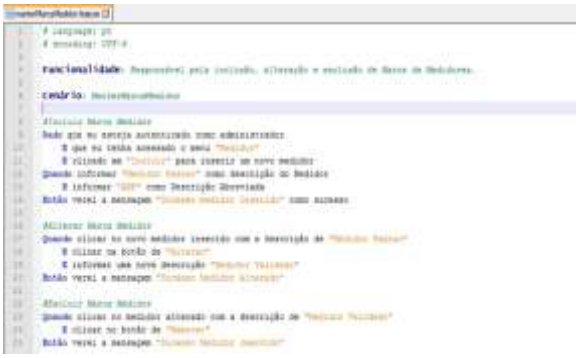
Depois disso criamos a estória de usuário 'manterMarcaMedidor.feature', que na prática deveria ser escrita por um usuário, através do Requisito:

Requisito Funcional 101: Manter Marca de Medidor.

O sistema deve disponibilizar a tela no Menu de Medidor, com opções de Pesquisar, Incluir, Alterar e Excluir. O registro deve possuir dois campos, que são respectivamente Descrição e Descrição Abreviada. O campo Descrição deverá ser obrigatório.

Citado o requisito, criamos a estória de usuário:

Figura 1, Estória de Usuário criada utilizando a sintaxe *Gherkin* para o Cucumber. IFPE 2016.



Fonte: O Autor (2016).

4.2 Resultados e discussão

Escrito a estória de usuário, o desenvolvedor pode utiliza-la para escrever códigos de testes em diferentes linguagens, a mais comum é a *Ruby*, mas pode ser também

implementado em C, C++, Java entre outras. Cada linha da estória de usuário, é representada como um método ou função na escrita do teste unitário.

Cada funcionalidade pode ser escrita por um usuário, então essa estória de usuário escrita, poderá ser utilizada como caso de teste para um teste de validação automatizado.

5 CONSIDERAÇÕES FINAIS/ CONCLUSÕES

O uso de ferramentas de construção de cenário de testes utilizando uma linguagem estruturada poderá incentivar ao cliente/usuário escrever estórias de usuários para que possam validar devidas funcionalidades do software, fazendo com que o próprio cenário que ele criou possa ser usado para fins de testes.

Consequentemente, o esforço para fim de testes no início do desenvolvimento será reduzido, pois as estórias de usuários escritas serão usadas como cenários de testes, onde o cliente/usuário poderá ser responsável pela manutenibilidade e será incluída na documentação do software e constantemente atualizada.

6 REFERENCIAS

BERLATTO, Laís. **Aplicação de Técnicas de Processamento de linguagem natural para ferramenta *Pyccuracy***. 2013. Disponível em <https://pt.slideshare.net/LaisBerlatto2/aplicao-de-tcnicas-de-processamento-de-linguagem-natural-para-ferramenta-pyccuracy> Acesso em: 7 jun. 2016

ELIZA, Renata. **Ferramentas de Suporte ao Teste de Software**, Editora *DevMedia*, 2021. Disponível em: http://www.devmedia.com.br/ferramentas-de-suporte-ao-teste-de-software/28642 Acesso em: 7 jun. 2016.

GAMA, Dante Torres. **SpecNL : Uma Ferramenta para Gerar Descrições em Linguagem Natural a partir de Especificações de Casos de Teste.** 2016. Disponível em: https://repositorio.ufpe.br/handle/123456789/2585 Acesso em: 7 jun. 2016

J. DE SOUZA, VELOSO. **Avaliação de Ferramentas de Apoio ao Teste de Sistemas de Informação**. 2010.

NORTH, Dan. ***Introducing BDD* | *Dan North & Associates***, 2015,
Disponível em: <http://dannorth.net/introducing-bdd/>
Acesso em: 22 fev. 2015

PRESSMAN, R. S. Engenharia de Software. São Paulo: Makron Books, 2000.

***Reference-Cucumber*,** Página de Informações sobre o *Gherkin*.
Disponível em: <https://cucumber.io/docs/reference>
Acesso em: 10 abr. 2016

SIEGER, Nick. **Automação de Testes de Aceitação com *Cucumber* e *JRuby***, 2015.

FITNESSE. **Sobre o *Fitnesse*.** Disponível em: <http://www.fitnesse.org/>
Acesso em: 24 fev. 2016

PROCENGE**. Sobre o GGAS.** Página de Informações Sobre o Software GGAS na Procenge. Disponível em:
<http://www.procenge.com.br/site/solucoes/softwares/ggas/>
Acesso em: 05 abr. 2016

SOFTWARE LIVRE. **Sobre o *open-source***, Página Inicial da comunidade *Open Source* no Brasil. Disponível em: <http://softwarelivre.org/open-source-codigo-aberto>

Acesso em: 22 fev. 2016

SOFTWARE PÚBLICO**. Sobre o Portal**, Disponível em: <https://softwarepublico.gov.br/social/spb/sobre-o-portal>.

Acesso em: 05 out. 2015.

BEHAT. ***Writing Features – Gherkin Languagem***,

Disponível em: <http://docs.behat.org/en/v2.5/guides/1.gherkin.html>

Acesso em 10 out. 2015