Desenvolvimento e Reutilização de Testes Automatizados em Aplicações Web

Lucas Antunes Amaral
Universidade Federal de Santa Maria

12 de agosto de 2015

1 Identificação

Resumo:

A constante busca pela qualidade de uma solução em forma de software, fez com que empresas do ramo de desenvolvimento, aderissem e enxergassem a importância da realização de testes automatizados em seus sistemas. A partir deste cenário, surgiram inúmeras ferramentas e frameworks para suprir esta demanda, que se propõe a ampliar a otimização de tempo e eficácia das aplicações implementadas, visando uma garantia maior na qualidade das mesmas. Contudo, é sabido que criar um novo teste para cada nova funcionalidade ou demanda do sistema, se torna muito custoso, sendo necessário um grande desprendimento de recursos humanos. Assim, este trabalho objetiva apresentar scripts de testes automatizados para sistemas web, que possam, de maneira mais genérica e com poucas alterações, serem reutilizados, de forma escalável, para novos casos de testes, que sigam o mesmo escopo.

Período de execução: Agosto de 2015 a Dezembro de 2015

Unidades participantes:
Curso de Ciência da Computação

Área de conhecimento: Ciência da Computação

Linha de Pesquisa: Linguagens de Programação, Qualidade de Software, Testes de Software

Tipo de projeto: Trabalho de Conclusão de Curso

Participantes:

Profª Andrea Schwertner Charão – Orientadora
Lucas Antunes Amaral – Orientando

2 Introdução

Dada a atual conjectura do mercado de desenvolvimento de software, é fundamental para que uma aplicação se mantenha viva de forma competitiva, apresentar diferenciais ao seu público alvo. Desta maneira a área de qualidade de software ganha cada vez mais espaço dentro das empresas de TI e, em especial, as ferramentas e metodologias de teste de software ganham maior visibilidade.

Os testes de software podem ocorrer em todas as etapas do desenvolvimento e de diferentes formas, contudo, sempre objetivam atender na totalidade os requisitos do sistema e, simultaneamente, amplificar a qualidade da solução codificada. São inúmeras as vantagens de se utilizar testes automatizados ao invés dos testes manuais em uma aplicação. Apesar de, aparentemente, ser mais prático e rápido realizar um teste manual, a cada nova alteração em um módulo do sistema, o teste tem que ser todo refeito e a tendência que novos erros sejam gerados até mesmo em funcionalidades já testadas é enorme, problema este que não ocorre quando a abordagem escolhida é a automatização dos testes.

Mesmo apresentando grandes vantagens, os testes automatizados demandam um grande custo inicial em sua codificação e, com isso, aumenta o envolvimento da equipe de qualidade. Pensando nesse problema, podemos buscar formas alternativas para que se possa usufruir de todas estas virtudes dos testes automatizados, e, ao mesmo tempo, utilizar de forma eficiente os recursos disponíveis em uma instituição.

3 Objetivos

3.1 Objetivo Geral

Este trabalho tem como objetivo principal apresentar um conjunto de casos de teste e testes que possam ser reutilizados de forma otimizada em novas funcionalidades de uma aplicação ou em sistemas que sigam os mesmos padrões e comportamento dos softwares conhecidos.

3.2 Objetivos Específicos

- Apontar diferenças de casos de testes;

- Gerar testes automatizados reaplicáveis em novos casos;
- Gerar casos de testes genéricos para um escopo definido.

4 Justificativa

A qualidade de software é uma das variáveis essenciais para que um projeto de software tenha sucesso. Sendo assim, torna-se cada vez mais necessário a inserção de testes automatizados em projetos web, agregando aos mesmos uma maior confiabilidade e redução nos possíveis erros que o sistema possa apresentar. Para que haja a possibilidade de aumentar a qualidade dos sistemas, sem que seja necessário uma maior demanda de recursos humanos para a área de qualidade, podemos adotar práticas de reuso de códigos de testes, visando maximizar a produtividade e eficiência, além de, simultaneamente, obter um produto final com uma garantia de qualidade superior.

5 Revisão de Literatura

Abaixo, serão apresentados conceitos relativos os conteúdos abordados neste trabalho, descrevendo, qualidade de software, ferramentas de teste de software, assim como o reuso de testes.

5.1 Qualidade de Software

A Qualidade de software é uma subárea oriunda da engenharia de software, que tem com foco central apresentar metodologias, procedimentos e métricas que garantam a qualidade no processo de desenvolvimento de um sistema. Apesar de ocorrer no processo, a Qualidade de software objetiva obter qualidade no produto final, e, com isso, consiga contemplar na totalidade os requisitos tratados com o cliente ao longo do processo. de Vasconcelos et al. [6] afirma que a qualidade de software está diretamente relacionada a um gerenciamento rigoroso de requisitos, uma gerência efetiva de projetos e em um processo de desenvolvimento bem definido, gerenciado e em melhoria contínua. Afirma também, que atividades de verificação e uso de métricas para controle de projetos e processo também estão inseridas nesse contexto, contribuindo para tomadas de decisão e para antecipação de problemas.

Mas como medir a qualidade de um sistema em questão? Para responder este questionamento, Garvin [7] propõe o conceito que ele chama de oito dimensões, que seriam em ordem, qualidade do desempenho, qualidade dos recursos, confiabilidade, conformidade, durabilidade, facilidade de manutenção, estética e percepção. Segundo Gar-

vin, atendendo a estes oito critérios, o sistema apresentará qualidade. Pressman [14] complementa a definição de Garvin mesclando-a com a norma ISO 9126, e aponta os fatores críticos para o sucesso neste caso, como sendo: Intuição, Eficiência, Robustez e Riqueza.

5.1.1 Qualidade do processo

A qualidade de software é largamente determinada pela qualidade dos processos utilizados para o desenvolvimento. Deste modo, a melhoria da qualidade de software é obtida pela melhoria da qualidade dos processos [9].

5.1.2 Qualidade do produto

Existe uma relação direta entre qualidade de produto e qualidade do processo, pois, para obtenção da qualidade do produto final, faz-se necessário adquirir primeiramente qualidade nos processos que compõem o desenvolvimento do mesmo. Avaliar a qualidade de um produto de software é verificar, através de técnicas e atividades operacionais o quanto os requisitos são atendidos. Tais requisitos, de uma maneira geral são a expressão das necessidades, explicitados em termos quantitativos ou qualitativos, e têm por objetivo definir as características de um software, a fim de permitir o exame de seu atendimento [9].

5.1.3 Testes de Software

É a atividade responsável por apresentar os erros existentes em um determinado programa, por isso, pode ser vista como uma atividade destrutível, pois visa expor os defeitos para depois corrigir os mesmos, e, de preferência, em um estágio inicial. Quanto mais tarde um defeito for identificado mais caro fica para corrigi-lo e mais, os custos de descobrir e corrigir o defeito no software aumentam exponencialmente na proporção em que o trabalho evolui através das fases do projeto de desenvolvimento [2]. O teste possibilita também, validar se os requisitos inicias do sistema, alinhados pelos *stakeholders*, estão contemplados em sua plenitude.

Apesar de não ser possível, através de testes, provar que um programa está correto, os testes, se conduzidos sistemática e criteriosamente, contribuem para aumentar a confiança de que o software desempenha as funções especificadas e evidenciar algumas características mínimas do ponto de vista da qualidade do produto [11]. Sendo assim, se faz essencial o mapeamento de um processo de testes para que se possa criar garantias e métricas que reduzam os erros, maximizando a qualidade, Crespo et al. [5] descreve o processo de teste, como sendo a composição de quatro macro etapas: Planejamento, projeto, execução e acompanhamento dos testes de unidade.

5.1.4 Estratégias de Testes

A Estratégia de testes se caracteriza pela definição da abordagem geral a ser aplicada nos testes, descrevendo como o software será testado, identificando os níveis de testes que serão aplicados, os métodos, técnicas e ferramentas a serem utilizada [15]. Existem muitas estratégias mas podemos destacar entre elas: Teste de unidade, integração, sistema, aceitação e regressão, funcional e carga. Se pode também, mesclar mais de uma estratégia com o intuito de reduzir os possíveis defeitos que o sistema venha a ter.

5.2 Ferramentas para teste de software

Existem muitas ferramentas desenvolvidas para realização de testes de software web, neste trabalho serão descritas duas(2) dentre elas, que serão as ferramentas utilizadas no desenvolvimento dos testes estudados.

5.2.1 Cucumber

Cucumber é um ferramenta de desenvolvimento de testes, voltado para sistemas web, que adota uma linguagem de alto nível bem próxima à uma linguagem natural e tem suas origens fixadas sobre a metodologia BDD (Behavior Driven Development). Cucumber é escrita em linguagem Ruby, mas pode ser utilizada para executar especificações de aplicações escritas em qualquer linguagem [12].

A escolha dessa ferramenta, baseou-se na fácil transcrição dos requisitos do sistema para a linguagem em questão, tornando possível conferir se os requisitos estão contemplados pelas funções e métodos descritos no sistema *web*. Lopes[10] compara Cucumber com o *software* Capybara e afirma que o primeiro apresentar um código mais legível e amigável. Uma observação é que o código com Capybara faz referência para vários detalhes de implementação, enquanto o código do Cucumber reserva isso apenas para os *Steps* e não para o arquivo de feature [10].

A ferramenta funciona basicamente através da leitura de arquivos com a extensão feature, os quais descrevem em linguagem natural uma funcionalidade e casos de teste, conhecidos como cenários. Como os testes estão escritos em uma linguagem natural, e não de programação, Cucumber precisa pesquisar pelo código associado aos passos que formam o cenário em arquivos auxiliares [16]. Cucumber executa seus arquivos *.feature*, e esses arquivos contêm especificações executáveis escritos em uma linguagem chamada Gherkin [1], que, possui um layout bem definido. Inicia pela descrição de uma funcionalidade, que por sua vez possui cenários, onde, um cenário é descrito da seguinte forma: *Dado* alguma condição *Quando* outra condição *E* terceira condição *Então* faça algo, conforme ilustra a figura 1.

```
Feature: Sign up

  Scenario: Successful sign up
    Given I am on the homepage
    When I follow the sign up link
    And I fill out the form with valid details
    Then I should receive a confirmation email
    And I should see a personalized greeting message
```

Figura 1: Um exemplo de código cucumber.

5.2.2 Selenium HQ

É um framework open source, utilizado para automatização de testes funcionais em aplicações web [4]. Segundo Pereira [13], se trata de uma ferramenta de fácil uso e eficiente para desenvolver casos de teste, permitindo os testes de aceitação ou funcional, regressão e de desempenho. Selenium trabalha como um plugin do navegador Firefox, o mesmo traz muita praticidade pois permite que se possa capturar cliques e valores digitados transformando-os em um caso de teste. Ele é composto por quatro ferramentas: Selenium IDE, Selenium Grid, Selenium RC e Selenium WebDrive.

A utilização de comandos no Selenium consiste em digitar o comando seguido de dois parâmetros tal como por exemplo *verifyText //div//a[2] Login*. Dependendo do comando, os parâmetros poderão ser opcionais, aliás, alguns comandos não necessitam de parâmetros para serem executados [17].

Selenium foi escolhida como umas das ferramentas no desenvolvimento deste projeto pelo fato de ser um *framework open source* que possui um vasto leque de ferramentas para testes de sistemas web. Outro fator importante para sua escolha, é a possibilidade de interação do Selenium HQ com o Cucumber. Ela se destaca entre as demais ferramentas gratuitas pelo fato de ser a mais completa, permitindo integração com várias linguagens, outros *frameworks*, além de suportar inúmeros navegadores e sistemas operacionais [13].

5.3 Reuso de testes

Visando melhor aproveitar os recursos existentes em uma instituição e ainda assim apresentar garantias na qualidade do produto/serviço entregues aos clientes, desenvolvedores do mundo todo, começaram a apresentar teses e modelos que buscam criar testes genéricos e padronizados. Um padrão é um pedaço de informação instrutiva e nomeada, que captura a estrutura essencial e *"insights"* de uma família bem sucedida de

soluções aprovadas para um determinado problema, o qual surge em um determinado contexto [3].

Guizzardi diz que por razões histórias a área de desenvolvimento de software não atingiu a maturidade que outras áreas da engenharia atingiram, complementa afirmando, que apesar disso, é inegável que algum avanço tenha sido alcançado, pois a forma de realização dessa atividade evoluiu de uma atividade realizada de forma quase artesanal, para um processo de desenvolvimento bem estruturado e que, nos melhores casos, contempla inclusive atividades de gerência e avaliação da qualidade [8]. Com isso, a reutilização dos testes já codificados, se mostra uma importante prática no desenvolvimento e que ainda apresenta muitas incógnitas e possibilidades para as equipes de TI, principalmente na geração de casos de testes que possam ser reutilizados em situações que apresentem um padrão parecido com os casos já conhecidos.

6 Metodologia

O projeto em questão, se trata de um trabalho exploratório, onde o mesmo será dividido em três(3) grandes fases, que seriam:

6.1 Planejamento dos testes necessários

Fase na qual serão elencados potenciais classes que necessitam de maiores cuidados nos sistemas escolhidos, será também, o momento onde ocorrerá o mapeamento dos casos de testes necessários que deverão ser desenvolvidos nas fases seguintes deste projeto.

6.2 Programação dos testes automatizados

Momento no qual serão escritos os códigos dos testes planejados na primeira fase do projeto, utilizando Selenium HQ e Cucumber como ferramentas para codificação dos mesmos.

6.3 Análise e geração de casos de teste reutilizáveis

Última etapa do projeto, responsável pela geração de *scripts* que contenham testes que possam ser reaproveitados em novas situações e funcionalidades para aplicações web que sigam o mesmo perfil das aplicações para os quais os mesmos foram criados.

7 Plano de Atividades e Cronograma

O cronograma de atividades será composto de 5 etapas, são elas:

1. **Configuração e estudo sobre *Selenium HQ* e *Cucumber***: Nesta etapa, ocorrerá os devidos estudos sobre as ferramentas escolhidas para desenvolvimento deste trabalho, assim como a instalação das mesmas.

2. **Desenvolvimento de testes 1° software**: Trata-se de elencar alguns testes necessários no primeiro software selecionado para o desenvolvimento, e após, codificá-los.

3. **Desenvolvimento de testes 2° software**: Desenvolvimento de testes para o segundo software web deste projeto.

4. **Análise das similaridades dos testes desenvolvidos**: Análise das diferenças e igualdades dos testes desenvolvidos sobre os dois sistemas web selecionados, elencando um padrão que possa ser reutilizado.

5. **Confecção de rotinas e testes genéricos para aplicações web**: Codificação de testes padronizados, que possam ser utilizados em grande parte e com poucas alterações, em outros projetos que sigam o mesmo padrão que os estudados neste trabalho.

Etapa	Agosto	Setembro	Outubro	Novembro	Dezembro
1	✓	✓			
2		✓	✓		
3		✓	✓		
4			✓	✓	
5					✓

Tabela 1: Cronograma de Atividades

8 Recursos

Como recursos físicos, serão utilizados neste trabalho, apenas o computador pessoal do pesquisador juntamente com ferramentas open source para teste de software web, que serão instaladas no mesmo.

9 Resultados Esperados

Ao término deste projeto, espera-se obter como resultado um conjunto de testes e casos de testes genéricos, que possam ser aplicados em um software web que possua características similares aos softwares utilizados nesta pesquisa.

Referências

[1] Reference cucumber.io. `https://cucumber.io/docs/reference`. Accessed: 2015-08-05.

[2] B. W. Boehm, J. R. Brown, and M. Lipow. Quantitative evaluation of software quality. In *Proceedings of the 2nd international conference on Software engineering*, pages 592–605. IEEE Computer Society Press, 1976.

[3] M. I. Cagnin, J. C. Maldonado, A. Chan, R. Penteado, and F. Germano. Reuso na atividade de teste para reduzir custo e esforço de vv&t no desenvolvimento e na reengenharia de software. *XVIII Simpósio Brasileiro de Engenharia de Software (SBES)*, pages 71–84, 2004.

[4] R. B. Chiavegatto[1], L. V. da Silva, A. Vieira, and W. R. Malvezzi[1]. Desenvolvimento orientado a comportamento com testes automatizados utilizando jbehave e selenium.

[5] A. N. Crespo, O. J. Silva, C. A. Borges, C. F. Salviano, M. Teive, A. Junior, and M. Jino. Uma metodologia para teste de software no contexto da melhoria de processo. *Simpósio Brasileiro de Qualidade de Software*, pages 271–285, 2004.

[6] A. M. L. de Vasconcelos, A. C. Rouiller, C. Â. F. Machado, and T. M. M. de Medeiros. Introdução à engenharia de software e à qualidade de software. 2006.

[7] D. A. Garvin. Competing on the 8 dimensions of quality. *Harvard business review*, 65(6):101–109, 1987.

[8] G. Guizzardi. *Desenvolvimento para e com reuso: Um estudo de caso no domínio de Vídeo sob Demanda.* PhD thesis, Universidade Federal do Espírito Santo, 2000.

[9] A. Koscianski and M. d. S. Soares. Qualidade de software, 2007.

[10] D. Lopes. O cucumber ainda tem o seu valor. `http://www.infoq.com/br/articles/valor-cucumber`. Accessed: 2015-08-10.

[11] J. C. Maldonado, E. F. Barbosa, A. M. R. Vincenzi, M. E. Delamaro, S. Souza, and M. Jino. Introduçao ao teste de software. *São Carlos*, 2004.

[12] D. Nunes. Automação de testes de aceitação com cucumber e jruby.

[13] D. V. Pereira. Estudo da ferramenta selenium ide para testes automatizados de aplicações web.

[14] R. S. Pressman. *Engenharia de software*. McGraw Hill Brasil, 2011.

[15] E. Rios. *Teste de software*. Alta Books Editora, 2006.

[16] F. H. Schmitz, H. Becker, and L. d. F. Berlatto. Cucumber - um breve review. `http://pt.slideshare.net/LaisBerlatto2/cucumber-um-breve-review`. Accessed: 2015-08-06.

[17] E. Sixpence, P. Adão, and C. Smith. Automatização de casos de teste como processo de melhoria da qualidade do software: O caso da aplicação e-learning isupac3 no isutc.