Universidade de Brasília

FGA

Engenharia de Software

Framework para Geração de Testes Unitários

Autores: Tomáz Felipe Rodrigues Martins e

Thaiane Ferreira Braga

Orientadora: Profª. Dra. Milene Serrano

Coorientador: Prof. Dr. Maurício Serrano

Brasília

Setembro de 2015

Sumário

1 Contextualização

A produção de software é uma atividade que, devido à sua natureza abstrata, vem acompanhada da eventual inserção de defeitos no código (TRODO, 2009). A partir da década de 1990, os usuários de software passaram a exigir maior atenção por parte das empresas desenvolvedoras em relação à redução de falhas de seus produtos e serviços (SOMMERVILLE, 2007). Além disso, a demanda por software no mercado está crescendo (PHILIPSON, 2004) e, por conseguinte, a exigência de maior qualidade no resultado final do processo de desenvolvimento aumenta (BARBOSA et al., 2009).

Nesse contexto, a identificação de defeitos que casualmente estejam presentes no código é uma preocupação constante no processo de produção de software. Para essa finalidade, surge a prática de testes do produto. Em essência, essa etapa visa a execução do software alvo com dados de teste, de forma a verificar se os resultados observados correspondem à expectativa. Isso permite ao desenvolvedor demonstrar aos seus clientes que o software está de acordo com as suas especificações, além de viabilizar ao programador a busca e descoberta de defeitos no software (SOMMERVILLE, 2007).

A prática de testes de software tornou-se parte importante do processo de desenvolvimento (BARBOSA et al., 2009). Diversas categorias de testes, com diferentes intenções de análise sobre o produto, foram surgindo, tais como: testes de unidade, testes de integração e testes de aceitação. Testes de integração são aqueles que permitem a observação de que os componentes que formam o software funcionam corretamente em conjunto. Testes de aceitação garantem que as interfaces do sistema estão de acordo com o especificado, bem como o comportamento esperado por elas. Os testes de unidade tem como proposta a garantia de que uma determinada parte (unidade) do código esteja respondendo como o esperado (SOMMERVILLE, 2007).

Outra questão que surge na produção de software, e que é um fator importante a ser considerado, é o custo que *bugs* geram no desenvolvimento e na fase de manutenção. Estimativas apontam que os gastos com a correção de *bugs* chegam a mais de 59,5 bilhões de dólares anuais nos Estados Unidos (JANTTI, 2008). No Brasil, gasta-se 70% do tempo de desenvolvimento corringindo-se erros (JANONES, 2010). Esse cenário aumenta o preço do produto final.

A produção de testes afeta positivamente o desenvolvimento de software não apenas no nível estratégico. Segundo Burke e Coyner (2003), há diversas razões para que se escreva testes unitários, dentre elas:

- Testes reduzem defeitos em funcionalidades novas e já existentes.
- Testes auxiliam na documentação do código.
- Testes permitem refatoração com maior qualidade.
- Testes reduzem o receio de alterar o código.
- Testes defendem o código contra alterações indesejáveis de outros programadores.

2 Questão de Pesquisa

O intuito deste TCC é auxiliar os desenvolvedores de software em relação à seguinte questão: o quanto é possível auxiliar o desenvolvedor na geração de testes unitários sobre o código de um produto de software?

3 Justificativa

Testes são cruciais no desenvolvimento de software, como evidenciado na Contextualização. Contudo, de maneira geral, desenvolvedores não escrevem testes para os seus programas (BURKE; COYNER, 2003). As razões são variadas, argumentando que não sabem escrever testes ou que não têm tempo para fazê-los (BURKE; COYNER, 2003). Tendo em vista este cenário, onde o mercado de software torna-se cada vez mais exigente com relação à qualidade dos produtos e serviços, bem como os altos custos relacionados à falta de empenho e previdência sobre a qualidade dos sistemas produzidos, é contraditório observar que a prática de fazer testes não seja comum, ou mesmo prioritária por parte dos programadores.

No artigo de Burke e Coyner (2003), intitulado *Top 12 Reasons to Write Unit Tests*, os autores revelam que algumas das colocações mais frequentes que vivenciaram em suas carreiras para os programadores não fazerem testes de unidade são:

- *"Eu não sei escrever testes."*
- *"Escrever testes é muito difícil."*
- *"Não tenho tempo suficiente para fazer testes."*
- *"Testes não são o meu trabalho."*

Essa lista evidência um fato já conhecido no desenvolvimento de software: a produção de testes é uma atividade onerosa (BARBOSA et al., 2009). Esse cenário é intricado, pois há evidências dos benefícios da produção de testes e, no entanto, há um distanciamento dos desenvolvedores em relação aos testes unitários. Considerando esse panorama, acredita-se que a elaboração de um suporte capaz de apoiar o programador na tarefa de gerar os testes unitários, procurando reduzir o esforço e os custos associados, colaborará tanto no âmbito técnico quanto estratégico no processo de desenvolvimento de software.

4 Objetivos

Com esse trabalho, busca-se alcançar os objetivos geral e específicos acordados nas subseções 4.1 e 4.2 a seguir apresentados.

4.1 Objetivo Geral

Desenvolver um *framework* capaz de auxiliar o desenvolvedor na geração de testes unitários.

4.2 Objetivos Específicos

Os seguintes itens são considerados importantes, devido à sua relevância para o entendimento de teste de software e aplicação dos conhecimentos sobre Engenharia de Software e, portanto, fazem parte dos objetivos específicos desse TCC. São eles:

1. Aprofundar o conhecimento na área de testes de software.

2. Investigar abordagens associadas ao tema foco desse TCC, via revisão bibliográfica e provas de conceito, no intuito de compilar soluções candidatas ao desenvolvimento do *framework* de geração de testes de unidade.

3. Aplicar métodos, técnicas e boas práticas de Engenharia de Software no processo de desenvolvimento do *framework*.

4. Gerar testes unitários por meio do *framework* que cubram métodos simples, como criar, editar, excluir e buscar objetos.

5. Coletar primeiras impressões dos testes gerados pelo *framework* e documentá-las, afim de facilitar a evolução do suporte no futuro.

5 Metodologia

Tendo em vista o objetivo geral do trabalho, o desenvolvimento de um *framework* para a geração de testes unitários, observa-se que o modelo de pesquisa que mais se enquadra para o contexto é um misto de pesquisa exploratória (FONSECA, 2002), na medida em que busca-se criar maior familiaridade com a temática, e pesquisa experimental (FONSECA, 2002), pois pretende-se criar condições em ambiente controlado para averiguar determinados casos, com a finalidade de testar o *framework*. Não necessariamente será utilizado o rigor de um experimento. Serão produzidos cenários de uso, visando a realização dos testes e a coleta das primeiras impressões do *framework*.

Devido à necessidade da constante ealização de testes, com a participação dos interessados, cabe ressaltar o uso da modalidade de pesquisa-ação (FONSECA, 2002). Isso conferirá ciclos de coleta e análise de dados e desenvolvimento para alteração do objeto de estudo, conforme as análises do ciclo anterior.

No que se refere ao desenvolvimento do software, pretende-se utilizar uma adaptação do Scrum (SUTHERLAND; SCHWABER, 2014), com *Sprints* de quinze dias e algumas de suas práticas ágeis, tais como: estimativas relativas, *timebox*, *backlog*, definição de pronto e quadro de tarefas. Procurar-se-á prover o desenvolvimento também com algumas práticas do XP (WELLS, 2009), como *Planning Poker*, Padronização do Código, Integração Contínua e Programação em Pares.

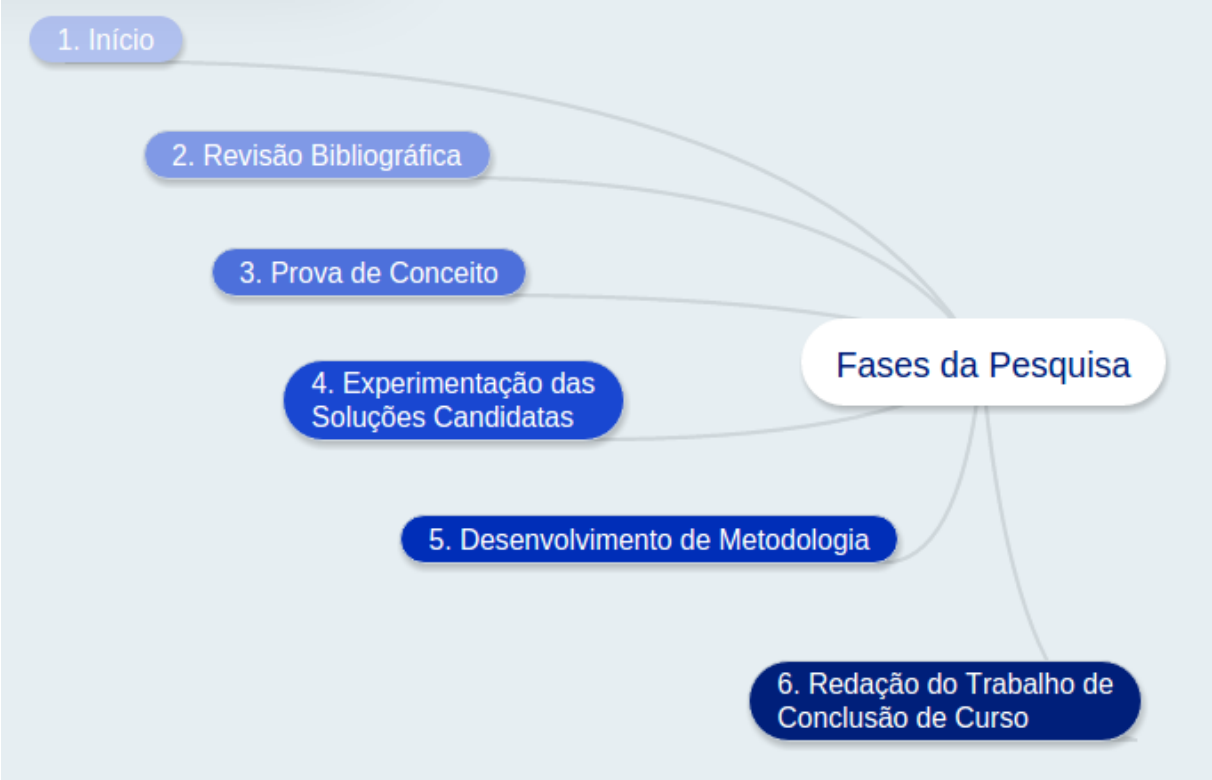

Figura 1: Fases da Pesquisa

Considerando-se o escopo do trabalho e a metodologia que se pretende utilizar, as etapas, ilustradas na Figura 1, serão guias do processo de pesquisa.

6 Cronograma

Consta, na Tabela 1, o cronograma que será adotado para o andamento dos trabalhos.

Tabela 1: Cronograma

Atividade	Ago	Set	Out	Nov	Dez
Pesquisar Referencial Teórico	X	X			
Definir Metodologia de Pesquisa		X			
Implementar Prova de Conceito		X	X		
Refinamento da Proposta			X	X	
Elaborar Documento Escrito do TCC 1				X	
Apresentar TCC 1				X	
Coletar Sugestões de Melhorias da Banca					X
Realizar Correções					X

Referências

BARBOSA, E. F. et al. *Introdução ao Teste de Software*. [S.l.], 2009. 49 p. Disponível em: <http://www.duguay.com.br/uploads/arquivos/apostilaUSP_Teste_de_Software.pdf>.

BURKE, E. M.; COYNER, B. M. *Top 12 Reasons to Write Unit Tests*. abr. 2003. Disponível em: <http://www.onjava.com/pub/a/onjava/2003/04/02/javaxpckbk.html>.

FONSECA, J. J. S. d. *Metodologia da Pesquisa Científica*. maio 2002.

JANONES, R. d. S. *Qualidade de Software: Uma questão de eficiência.* ago. 2010. Disponível em: <http://www.devmedia.com.br/qualidade-de-software-uma-questao-de-eficiencia/17803>.

JANTTI, M. *Difficulties in Managing Software Problems and Defects*. Tese (Dissertação de Doutorado) — University of Kuopio, Kuopio, Finlândia, jan. 2008. Disponível em: <http://epublications.uef.fi/pub/urn_isbn_978-951-27-0109-4/urn_isbn_978-951-27-0109-4.pdf>.

PHILIPSON, G. *A Short History of Software*. 2004. Disponível em: <http://www.thecorememory.com/SHOS.pdf>.

SOMMERVILLE, I. *Engenharia de Software*. 8ª. ed. São Paulo: Pearson Addison-Wesley, 2007.

SUTHERLAND, J.; SCHWABER, K. *The Scrum Guides*. 2014. Disponível em: <http://www.scrumguides.org/scrum-guide.html>.

TRODO, L. D. *Uso de Métricas nos Testes de Software*. Tese (TCC) — Universidade Federal do Rio Grande do Sul, Porto Alegre, nov. 2009. Disponível em: <http://www.lume.ufrgs.br/bitstream/handle/10183/18574/000730980.pdf?...1>.

WELLS, D. *The Rules of Extreme Programming*. 2009. Disponível em: <http://www.extremeprogramming.org/rules.html>.