1 Introdução

Uma parcela significativa do tempo e do custo utilizados na construção dos produtos de software é consumida pelos testes. Com a crescente complexidade e tamanho do software e a necessidade de entregas rápidas, a expectativa é que a quantidade de testes necessários vá continuar a aumentar. Além disso, são necessárias grandes melhorias na tecnologia de testes e o teste de software automatizado é uma das respostas mais promissoras (Dustin *et al.* 2009).

Para se desenvolver um software podemos fazer diferentes tipos de teste, com diversos objetivos - performance, carga, funcionalidade, estresse, entre outros (Karner *et al.* 2000).

Os testes funcionais são testes derivados da especificação do software ou componente, ou aqueles que o testador se preocupa somente com a funcionalidade do sistema e não com sua implementação, são os mais utilizados devido à necessidade que os softwares produzidos façam o que foi acordado com o cliente na fase de análise de requisitos (Sommerville, 2003).

Já a execução de testes de software pode ser feita tanto de forma manual quanto automatizada. A execução manual consiste na reprodução por uma pessoa do teste previamente definido e documentado. Já a execução automática consiste na automação do processo de teste manual atualmente em uso (Zambelich, 2006). Scripts de teste devem ser construídos para reproduzir os testes que serão executados automaticamente.

Os testes funcionais devem ser automatizados quando são muito repetitivos e demandam um esforço considerável de tempo quando realizados manualmente (Zambelich, 2006). A realização de testes automatizados, além de possibilitar a redução do ciclo de testes, permite um aumento indireto da cobertura do software e, consequentemente, da sua qualidade, porque permite que os testadores foquem seus esforços em outros tipos de teste ou em testes que não possam ser automatizados.

Algumas das características dos processos de desenvolvimento de software que seguem o paradigma de metodologias ágeis, como o "*extreme programming*" (Beck e Andres, 2004), são a entrega frequente de novas funcionalidades, ciclos de desenvolvimento curtos e rapidez de resposta a mudanças nos requisitos ao longo do projeto e foco em manter o software funcionado ao invés de documentação abrangente. Uma das consequências do desenvolvimento iterativo com ciclos curtos e da tolerância a mudanças é a necessidade de testes frequentes, em particular, torna-se necessária uma elevada frequência de execução de testes de regressão, a fim de determinar se novas funcionalidades não comprometeram o correto comportamento de outros requisitos anteriormente implementados e já validados.

Uma das grandes diferenças entre processos tradicionais e ágeis, é que os processos tradicionais geralmente são bastante prescritivos e temos que documentar tudo que estiver definido no processo, o que geralmente é muita coisa. Em métodos ágeis não há prescrição de documentação, mas isso não significa que não há necessidade de documentar desde que seja necessário de fato. Muitas pessoas confundem isso e dizem que nunca se deve documentar em projetos ágeis, o que é um grande engano. Em projetos ágeis podemos documentar, desde que seja necessário de fato. A ideia é não perder tempo com nada que não seja requerido de verdade para o projeto.

Para (Highsmith, 2009), a atividade de documentar tem que ser rápida, não pode dar trabalho. Usar ferramentas como wikis e geradores de documentação podem ajudar. Se for fácil documentar, as chances de fazê-lo serão maiores. Se a documentação for fácil de ser acessada (e tiver busca) ela será mais útil. Além disso, prefira usar uma tecnologia fácil e conhecida para que todos os membros do time possam documentar. Por exemplo, se você escolher usar *Latex*, podemos reduzir as chances de designers documentarem.

Como integrante da mesma equipe de desenvolvimento de software há cinco anos, e que nos últimos três anos migrou da metodologia de desenvolvimento tradicional para uma metodologia de desenvolvimento ágil, observamos no âmbito do nosso ambiente de desenvolvimento ágil, alguns problemas relacionados a testes e documentação, tais como:

- documentação escassa e muitas vezes inexistente;
- documentos sem padronização;

- documentos descentralizados;
- documentos difíceis de localizar;
- dificuldades técnicas para automatizar os testes funcionais;
- necessidade de executar manualmente um grande número de testes funcionais a cada nova release do software;
- testes funcionais executados de forma desordenada, uma vez que existem poucos documentos de requisitos para se basear. Um ponto forte é que a cada execução novos caminhos são testados, um ponto fraco é a dificuldade em realizar os testes de regressão;
- pouca ou nenhuma contribuição dos envolvidos não técnicos na geração e execução de testes funcionais, como é o caso do cliente;
- dentro outros.

Vimos então, a necessidade e a oportunidade de estudos visando desenvolver maneiras úteis para facilitar e incentivar a documentação de requisitos, bem como meios de minimizar o esforço e as dificuldades da equipe de desenvolvimento para automatizar testes funcionais. Pensamos em utilizar documentos de caso de uso como instrumento capaz de apoiar a equipe na documentação dos requisitos funcionais e na geração e execução automática de testes funcionais, com o intuito de verificar automaticamente se o resultado obtido nos testes gerados e executados está em conformidade com o especificado.

O trabalho consiste em pesquisar na literatura uma forma de utilizar uma abordagem de casos de uso direcionados por comportamento para documentar requisitos de software e a partir dessas informações automaticamente gerar e executar scripts de teste para verificar o comportamento funcional de aplicações web. As informações do caso de uso, em especial os fluxos de eventos, devem ser estruturados obedecendo a um "modelo de comportamento" para que seja possível armazenar os dados e utilizá-los como entrada na integração com o *framework* de testes. Levando em conta a disponibilidade e a experiência já adquirida, foi escolhida a ferramenta *Selenium* (Selenium, 2011), que será responsável por executar automaticamente os scripts de teste com a interface web.

Ao documentar um requisito utilizando a abordagem apoiada pela ferramenta o fluxo principal e cada fluxo alternativo do caso de uso será um cenário a ser coberto por testes automáticos.

1.1. Objetivo da Dissertação

Essa pesquisa tem como objetivo especificar, projetar e desenvolver uma ferramenta para viabilizar a documentação de requisitos funcionais na forma de casos de uso, bem como utilizar a documentação cadastrada para gerar e executar automaticamente scripts de testes em aplicações web. Os testes verificarão se o comportamento das aplicações está em conformidade com o que foi documentado. Utilizar a ferramenta dentro de um ambiente de desenvolvimento real e avaliar o esforço para redigir os casos de uso, o esforço para assegurar que tenham sido criados suficientes scripts de teste, fazer a comparação com testes produzidos manualmente e comparação com testes produzidos através de "*capture and replay*" e por fim, relatar as dificuldades encontradas.

A expectativa é:

- Facilitar a criação de documentos funcionais que geram testes;
- proporcionar confiança e dar mais agilidade, uma vez que os principais testes funcionais serão executados automaticamente e darão retorno rápido para que a equipe possa decidir sobre a transferência da release para o ambiente de produção;
- melhorar a coerência entre a documentação e o funcionamento do software, uma vez que o caso de uso será capaz de verificar o comportamento nele descrito.

1.2. Geração e execução automática dos testes

Como parte do processo, o cadastro dos casos de uso pode ser realizado por qualquer membro da equipe do projeto, com ou sem conhecimento técnico, e os dados serão armazenados pela ferramenta em um sistema gerenciador de banco de dados. Após realizar o cadastro do caso de uso, a ferramenta pode utilizar os passos informados no fluxo básico e nos fluxos alternativos para montar scripts de teste e integrar ao *framework* de testes *Selenium* (Selenium, 2011) para executar a interação com a interface web.

A ferramenta armazena o resultado de cada script executado e durante a execução é possível acompanhar o resultado dos testes e verificar rapidamente se

o comportamento da aplicação está em conformidade com o que foi descrito na forma de caso de uso.

1.3. Organização da Dissertação

O restante deste documento está organizado em cinco capítulos, da seguinte forma:

O **Capítulo 2** apresenta o estado da arte para geração de testes automáticos a partir de casos de uso e as ferramentas e *frameworks* existentes comparados a ferramenta proposta.

O **Capítulo 3** descreve uma visão macro do processo de documentação dos casos de uso e da geração e execução automática dos testes.

O **Capítulo 4** descreve a ferramenta desenvolvida para apoiar na aplicação da processo de geração automática dos testes, os requisitos funcionais e não funcionais necessários e desejados, diagramas, modelo de dados, persistência dos dados e como os testes são gerados e executados e como o resultado pode ser visualizado.

O **Capítulo 5** apresenta os resultados da utilização da abordagem proposta em um cenário real.

O **Capítulo 6** apresenta as conclusões e trabalhos futuros.