Aspectos sociais, humanos e econômicos da utilização de testes automatizados no desenvolvimento de sistemas

Rachel CHICANELLI
Institute of Computing, Federal University of Mato Grosso
Cuiabá, Mato Grosso 78060-900, Brazil

Daniel VECCHIATO
Institute of Computing, Federal University of Mato Grosso
Cuiabá, Mato Grosso 78060-900, Brazil

Thiago VENTURA
Institute of Computing, Federal University of Mato Grosso
Cuiabá, Mato Grosso 78060-900, Brazil

Raphael GOMES
Institute of Computing, Federal University of Mato Grosso
Cuiabá, Mato Grosso 78060-900, Brazil

Nilton TAKAGI
Institute of Computing, Federal University of Mato Grosso
Cuiabá, Mato Grosso 78060-900, Brazil

Luana MENDES
Institute of Computing, Federal University of Mato Grosso
Cuiabá, Mato Grosso 78060-900, Brazil

RESUMO

Este trabalho apresenta uma análise acerca de diferentes aspectos da aplicação de testes automatizados em comparação aos testes manuais. Como estudo de caso, foram observadas e comentadas as experiências no desenvolvimento de um sistema para a área da saúde, desenvolvido por uma equipe utilizando metodologia ágil. Neste estudo é apresentado a aplicação de técnicas de testes automatizados, utilizando as ferramentas Selenium e SpecFlow, no qual são analisadas as vantagens e desvantagens frente às questões sociais, humanas e econômicas. A análise final mostrou principalmente que: (i) no aspecto social há vantagem na impessoalidade ao apontar as falhas; (ii) no aspecto humano os testes automatizados não possuem as desvantagens com relação às características emocionais das pessoas e da possibilidade de rotatividade nas empresas; (iii) no aspecto econômico o tempo economizado em comparação ao teste manual é de mais de 3 vezes.

Palavras-chave: Selenium, SpecFlow, Ágil, Equipe, Saúde.

1. INTRODUÇÃO

No Brasil, a Vigilância Sanitária (VISA) é uma forma complexa de existência da saúde pública, pelo fato de ser multidisciplinar e ter uma atuação descentralizada, com direção única em cada esfera de governo [1]. Sistemas que atuam nesta área possuem uma complexidade elevada devido a suas inúmeras regras de negócio, tornando o teste de *software* fundamental para atingir a qualidade desejada em sistemas desta área.

Com a criação da Agência Nacional de Vigilância Sanitária (Anvisa) em 1999 no Brasil, houve o a expansão dos sistemas de serviços de saúde e na sua informatização. Neste contexto, surgiu o Sistema de Vigilância Sanitária (SVS), que tem como propósito a automação dos serviços deste setor, como a emissão de alvará sanitário e controle de inspeções sanitárias. Para isso é fundamental que se tenha a introdução de inovações em tecnologias de saúde principalmente porque a regulação da VISA possui diferentes fontes de dados e diferentes legislações, os quais devem estar atualizados e disponíveis sempre que necessário nas funcionalidades para o usuário.

Tendo em vista as especificidades da área de saúde pública, a automação de testes corrobora com a verificação das diversas regras de negócio de forma mais determinística, eficiente e executado em um maior número de vezes. Dentro desta ótica, problemas no código podem ser encontrados e corrigidos mais rapidamente, melhorando a qualidade geral do sistema.

Convém ressaltar que a automação dos testes precisa estar integrada com o processo de desenvolvimento de *software*. Os integrantes de um time de desenvolvimento precisam trabalhar de forma integrada e coesa, controlando a versão do sistema e executando os testes automatizados de maneira constante. Neste contexto surge a integração contínua. A integração contínua permite a entrega mais eficaz do *software* e com menos defeitos [2], incrementando funcionalidades do *software* a cada *build*.

Diante disso, este trabalho tem como objetivo compartilhar a experiência em diversas perspectivas, apresentando os resultados da aplicação de testes automatizados utilizando ferramentas de automação de integração contínua. É importante salientar que o projeto possui a necessidade de melhorar a

cobertura de testes e aumentar a detecção de defeitos sem um expressivo aumento de tempo e custo, visando garantir a qualidade geral *software*.

O restante deste artigo está estruturado da seguinte forma: A seção 2 trata do contexto relacionado aos testes automatizados; a seção 3 do ambiente de integração contínua, viabilizando o feedback ao time de desenvolvimento; a seção 4 refere-se ao Estudo de Caso, levando em consideração os aspectos sociais do teste de *software*, os aspectos humanos e os aspectos econômicos; por fim, na seção 5 estão as conclusões deste trabalho.

2. TESTES AUTOMATIZADOS

Esta seção apresenta estudos relacionados à área de testes de *software*, bem como as dificuldades e os desafios que testadores enfrentam com a finalidade de obter melhoria da qualidade no desenvolvimento de *software*. A automação de testes de *software* traz benefícios para a qualidade de *software* [3]. Tais benefícios incluem a reutilização do teste, economizando esforço na verificação do *software* e, por conseguinte, aumentando a quantidade de vezes que o teste é executado, impactando diretamente na confiabilidade percebida pelo cliente ao validar o produto final [4].

O alto custo da execução de testes manuais e o número cada vez maior de ambientes a serem testados dificultam manter a qualidade do produto. Dessa maneira, a comparação entre testes manuais, testes automatizados e semi-automatizados são mostrados em outros trabalhos [5, 6, 7], relatando ganhos obtidos como a redução no esforço, aumento na quantidade de ambientes, quantidade de repetições de testes e maior segurança.

A escrita de cenários em linguagem natural e o reaproveitamento de métodos, facilitam a manutenção dos testes, permitindo a colaboração entre os integrantes do time e aumentando a qualidade do *software*. Esta etapa pode ser auxiliada por ferramentas, como por exemplo o Selenium WebDriver e o Cucumber [8]. Por outro lado, há o custo em se criar esses testes incialmente, algo que não há nos testes manuais.

Testes elaborados utilizando as ferramentas automatizadas e que possuam integração contínua, permitem gerar relatórios em formato HTML, proporcionando um *feedback* por e-mail ou outra forma de comunicação, evitando riscos associados a atrasos. Essa integração pode ser idealizada através das ferramentas SpecFlow e Jenkins [9].

Em métodos ágeis é recomendado que todas as pessoas envolvidas em um projeto trabalhem controlando a qualidade do produto. Porém, acontece que essa responsabilidade às vezes é dada ao testador que deve possuir criatividade, boa comunicação, dispor de pensamento crítico e proatividade. Dentre as dificuldades encontradas por esse profissional, estão problemas na documentação e a forma como os testes são medidos em relação a produtividade e a eficiência [10]. Dessa forma, adotar métricas para facilitar o acompanhamento e melhoria, como a de coesão entre os testes, métodos pequenos e consequentemente de melhor entendimento, possibilitam que a equipe acompanhe, avalie e apoie a melhoria contínua através da utilização eficiente de ferramentas de teste de *software* [11].

No entanto, é normal as empresas de desenvolvimento de *software* não terem como prioridade o investimento na quantidade ideal de recursos para melhorar a qualidade de *software*. Apesar disso, possuem consciência de que a aplicação dos testes aumenta da confiança do produto fornecido ao cliente [12]. Diante desse contexto, a discussão proposta neste trabalho articula os elementos supracitados, analisando a implementação de testes de *software* com a finalidade de se obter um produto de melhor qualidade sem o aumento de custo, com maior rapidez e com uma cobertura maior de testes. Dessa maneira, é descrito o desenho e a implementação de uma solução para atingir esses objetivos utilizando testes automatizados com o Selenium e SpecFlow.

3. AMBIENTE DE INTEGRAÇÃO CONTÍNUA

O processo utilizado para o desenvolvimento é de extrema relevância para a garantia de qualidade de um *software*. Possibilitar a utilização de um processo no qual desenvolvedores e gerentes de desenvolvimento tenham um retorno ágil sobre novas alterações no código fonte é de grande valia, visto que estes conseguem perceber problemas mais rapidamente e, com isso, tomar decisões para agir corretamente sobre eles.

Neste trabalho foi avaliado as vantagens e desvantagens dos testes automatizados em um ambiente de integração contínua e automação de testes ilustrado na Figura 1. Em síntese, os desenvolvedores e os testadores ao terminarem uma tarefa realizam o *commit* e o *push* do seu código conforme a plataforma GitHub (etapa 1). Neste momento, a ferramenta Jenkins (etapa 2) torna possível a integração entre o GitHub, SpecFlow/Selenium, Slack e o servidor *web*, explicados a seguir. No servidor, é realizada a construção do código (etapa 3) e a publicação do sistema no servidor HTTP (etapa 4). Após a publicação, os testes são iniciados por meio da utilização do Selenium e SpecFlow (etapa 5). Ao terminar os testes, um relatório é gerado. Este relatório contém a qualidade de testes executados e o resultado de cada um deles. Caso algum teste tenha falhado, ou a própria construção do código esteja quebrada, é gerada uma mensagem na ferramenta de comunicação Slack (etapa 6). Uma vez que a mensagem está no Slack, ela é enviada para todos os membros do time (etapa 7).

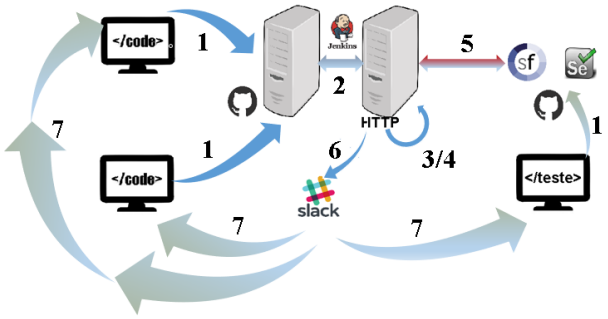

Figura 1: Integração contínua com as ferramentas GitHub, Jenkins, SpecFlow, Selenium, Slack e servidor web.

Com este ambiente, os desenvolvedores, testadores e gestores ficam ciente de algum problema que tenha ocorrido ao se realizar a integração do código, podendo tomar contramedidas de maneira mais eficiente, impactando diretamente no tempo de resposta aos erros encontrados e melhorando a qualidade do sistema.

4. ESTUDO DE CASO

O estudo de caso contempla a parte de teste automatizado de *software* no contexto de um sistema da área da saúde pública, servindo como uma referência ideal para análise de testes de *software*. O desenvolvimento do sistema citado utilizou o ambiente de desenvolvimento Visual Studio com a linguagem C#. Para os testes, optou-se pela utilização de ferramentas de código aberto que possuem integração com o *framework* .NET, como o Selenium WebDriver e SpecFlow.

Após a definição das ferramentas e do seu estudo, foi elaborado o plano de teste, incluindo os casos de testes, *stakeholders*, objetivos e metas. Neles são descritos os resultados previstos, as entradas não permitidas e quais as saídas esperadas para entradas erradas. Um exemplo simples seria o do campo "Data de nascimento", que deveria ser preenchido somente com números e não com caracteres. Caso seja preenchido com caracteres, uma mensagem deverá reportar o erro.

Dentre os casos de testes, alguns são reutilizados em outros cenários, como é o caso de autenticação no sistema. Esse tipo de recurso acelera a criação dos testes automatizados e foi aplicado em outras ações do sistema comumente utilizadas, como o filtro de dados em uma listagem ou a confirmação de uma informação em uma mensagem.

Esses cenários foram escritos por meio de uma linguagem natural chamada Gherkin, representada na Figura 2, que proporciona fácil entendimento tanto para os desenvolvedores quanto para os Stakeholders, possibilitando o entendimento entre os envolvidos no projeto e reduzindo problemas de comunicação entre o time.

```
@SolicitarAcessoUsuario
Scenario: TA02 - Solicitar acesso de usuário com sucesso
    Given eu acesso a url /Login
    When eu informo os dados de autenticação
    | cpf            | senha     |
    | 475.023.761-27| 123456    |
    Then eu clico no botão entrar no sistema
    Given eu clico na opção Acesso do menu principal
    And eu clico na opção Solicitacão de Acesso do submenu
    And eu preencho o formulário com dados válidos
    | tipoVinculo | ProfissionalSaude | CBOId              | OrgaoId                      | SetorId             | Matricula |
    | Estatutário | Sim               | 223505 - Enfermeiro | SECRETARIA DE ESTADO DE SAÚDE | Vigilância Sanitária | 010203    |
    Then eu clico no botao salvar solicitação de acesso
    And eu clico no botao sair do sistema
    Then eu sou redirecionado para a url /?LogOff=True
```

Figura 2: Cenário de teste utilizando a linguagem Gherkin.

Dessa mesma maneira, para se obter mensagens de erros e resultados dos testes mais claros, foram utilizadas "*Fluent Assertions*", que permitem uma sintaxe intuitiva e simples, em vez de ler a linguagem de programação. Outras características do desenvolvimento deste sistema são que: o projeto foi desenvolvido com metodologia ágil, no caso o Scrum; dez pessoas fazem parte da equipe de desenvolvimento; uma pessoa era responsável pelos testes manuais de exploração; em média 9,3 *builds* eram realizados por dia de desenvolvimento.

Ademais, com este estudo de caso foi possível o relato de experiências utilizando testes manuais e automatizados com a finalidade de comparação entre as abordagens, nas perspectiva sociais, humanas e econômicas.

Aspectos sociais

Em algumas organizações que não compreendem a importância das atividades de teste em conjunto com as atividades de desenvolvimento, são contratados desenvolvedores com várias habilidades, com a finalidade de que façam os testes além de suas funções, o que pode ser ruim, pois, via de regra, o desenvolvedor intrinsecamente busca testar o cenário de sucesso, ou seja, se o que ele desenvolveu está correto. Em testes manuais, foi observado que esse caso acontece com uma maior intensidade. Nos testes automatizados, é mais fácil notar se está sendo testado apenas o caminho de sucesso ou se os pontos de falha estão sendo considerados.

Por outro lado, quando há uma posição específica de testador, é comum haver uma percepção de que o desenvolvedor e o testador estão em lados opostos. Isso ocorre porque os desenvolvedores não gostam que outras pessoas mostrem seus erros. Além disso, a automação de um teste pode exigir mudanças no código que não refletem em novas funcionalidades, como por exemplo a adição de atributos "id" em elementos HTML para facilitar a identificação dos componentes pela ferramenta. Para os desenvolvedores, este é um trabalho adicional que não refletirá em adição de valor ao sistema.

Nesses casos, os integrantes do time devem trabalhar no sentido de minimizar atritos, buscando benefícios para todos os envolvidos e o sucesso do sistema em desenvolvimento. Os testes, além de ter o objetivo de encontrar o maior número de defeitos, devem também ajudar os desenvolvedores a entregar um produto com melhor qualidade, desenvolvido não somente pelo desenvolvedor, mas pela equipe como um todo. Dessa maneira, os defeitos devem ser reportados de forma mais objetiva possível, e não causar nenhum tipo de constrangimento a quem receber para executar o trabalho de correção do *software*.

Em geral, a utilização de testes automatizados diminui os atritos entre a equipe, pois os desenvolvedores não terão mais a percepção direta de que uma pessoa encontrou um problema em seu código, e sim de que um mecanismo implementado para tal

foi o responsável por isso. Dessa forma, divergências pessoais podem ser diminuídas melhorando a interação social entre a equipe.

Aspectos humanos

O teste manual de *software* está sujeito a erros, pois o testador, ao percorrer muitas vezes pelo mesmo fluxo, pela mesma tela e pelos mesmos componentes, não possui a mesma cautela com o teste como na primeira vez que o executou. O ser humano está sujeito a erros, a variações de humor, experiência profissional, cansaço, entre outros fatores que podem acarretar prejuízo na qualidade do teste de *software*.

A rotatividade da equipe também pode prejudicar os testes manuais. Quando um membro experiente da equipe de testes sai do projeto e um novo membro o substitui, há uma latência para obter a mesma qualidade nos testes que no momento anterior à saída do membro experiente. Estes dois pontos não acontecem nos testes automatizados. Na automatização, é suprido essas características humanas que podem levar a não identificar alguns defeitos no sistema.

Outra diferença nesta perspectiva é que, após o testador manualmente encontrar o erro, deverá apresentá-lo ao desenvolvedor que deverá procurar pelo problema nas várias linhas de código para encontrar a inconsistência e, por fim, corrigi-la. Por sua vez, no teste automatizado, é gerado um relatório detalhado do teste, contendo: os itens testados, o tempo de execução do teste, a data e hora de execução, o resultado, a quantidade de testes, o *Error Summary*, que demonstra os erros e em que parte do código estão localizados, e o *Execute Details*, que demonstra o detalhamento de cada *Step*, bem como o tempo de execução de cada uma delas. Todas essas informações facilita para os desenvolvedores corrigirem o erro mais rapidamente.

Aspectos econômicos

Nesta perspectiva, deve ser analisado o tempo gasto com testes manuais em comparação aos testes automatizados, representando custos para o projeto. Foi utilizado um cenário de teste, considerando o tempo de teste nos dois métodos e as quantidades de *Builds*/Testes feitas por dia pelos desenvolvedores para verificar a possibilidade de um testador manual alcançar o mesmo objetivo do teste automatizado.

Para o teste manual, foi indicada somente uma pessoa, com familiaridade no sistema em desenvolvimento, com amplo conhecimento das funcionalidades do sistema e responsável por executar o teste manual de todas as tarefas desenvolvidas pelos programadores.

O processo utilizado para realização dos testes automatizados era composto por treze atividades, com entradas não permitidas (valores nulos, números ao invés de letras e quantidade de caracteres superiores ao estipulado). Tanto no teste manual quanto no teste automatizado, foram cronometrados o tempo de execução de cada *Step*. A Figura 3 mostra o comparativo de tempo entre o teste manual e o automatizado.

Conforme a Figura 3, o experimento possibilitou observar que o teste manual foi mais longo, demorando 860s contra 238s do teste automatizado, executando os mesmos 13 *Steps*. A média do tempo em segundos do teste manual foi de 66s. No teste automatizado foi de 18s. Portanto, o teste automatizado é pouco mais que 3 vezes mais rápido em relação ao teste manual, em termos de tempo de execução.

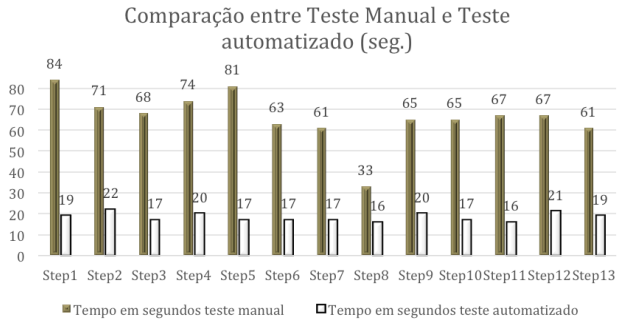

Figura 3: Comparação entre o teste manual e automatizado.

A média de *builds* no projeto deste estudo de caso foi de 9,3 por dia de desenvolvimento. Diante disso, a pessoa responsável pelos testes manuais necessitaria de aproximadamente 2 horas e 13 minutos (7.998s) somente para executar um cenário. Em uma aplicação complexa, com a existência de diversos cenários, seria inviável a execução de testes manuais a cada *build*. De tal forma, os defeitos seriam encontrados com menor agilidade, impactando diretamente o custo e a qualidade do sistema. Percebe-se, portanto, uma maior vantagem na execução de testes automatizados na garantia e controle de qualidade do processo de desenvolvimento de *software*.

5. CONCLUSÃO

A automação dos testes é uma prática imprescindível, principalmente no desenvolvimento ágil em que os ciclos de entrega são constantes e curtos. Isso ganha ainda mais importância dependendo da complexidade do sistema a ser desenvolvido, necessitando repetir os mesmos casos de teste, os mesmos caminhos e as mesmas operações em cada alteração de funcionalidade.

Em comparação aos testes manuais, os testes automatizados requerem custos superiores na sua fase inicial. Em contraste, as execuções dos testes manuais representam um custo muito mais elevado do que os testes automatizados, pois necessita que os casos de testes sejam executados desde a primeira etapa do desenvolvimento do *software* em todos os ciclos de entrega. No teste automatizado, há a necessidade do esforço somente uma vez para que depois seja executado os testes à vontade e de forma rápida permitindo encurtar o ciclo de comunicação com o cliente e perda de tempo e esforço.

Além disso, o teste automatizado é determinístico, isto é, se rodado diversas vezes, no mesmo sistema deve-se gerar o mesmo resultado. Por outro lado, o teste manual pode ter divergências de resultados, pois está sujeito a influências dos aspectos humanos do testador.

Devido a esses fatores, os testes automatizados tornam-se fundamentais para a obtenção de um sistema de melhor qualidade. Outrossim, elaborar cenários de testes por meio de ferramentas de fácil entendimento para desenvolvedores e clientes, resultam na redução do tempo gasto com a identificação e correção de erros, além de tender a ser encontrados mais cedo. Isso é mais vantajoso, pois à medida

que o tempo passa, os programadores esquecem dos detalhes da realização daquele erro, aumentando o custo da correção.

Os resultados deste trabalho sugerem que a execução automática de testes pode alcançar cobertura semelhante ao teste manual, mesmo em sistemas complexos que devem seguir as regras da VISA. Esse resultado pode ser alcançado em uma fração do tempo reduzida e que o desenvolvimento de *software* exige algumas premissas como conhecimento técnico, criatividade, organização, comunicação entre a equipe e atenção. Diante disso, é esperado que em alguns dos vários códigos essas premissas falhem, tornando necessária uma maneira fácil e ágil de executar os testes seja qual for o momento, o que é inviável em um ambiente que utiliza testes manuais.

6. REFERÊNCIAS

[1] S. Rozenfeld, Fundamentos da vigilância sanitária, Fiocruz, 2000.

[2] J. Humble; D. Farley, Continuous Delivery: Reliable Software Release Through Build, Test and Deployment Automation, Addison-Wesley Professional, 2010.

[3] D.M. Rafi; K.R.K. Moses; K. Petersen; M.V. Mäntylä, Benefits and limitations of automated software testing: Systematic literature review and practitioner survey, 7th International Workshop on Automation of Software Test, 2012.

[4] E.F. Collins; L.M.A. Lobão, Experiência em Automação do Processo de Testes em Ambiente Ágil com SCRUM e ferramentas Open Source, IX Simpósio Brasileiro de Qualidade de Software, 2010.

[5] A.M. Castro; G.A. Macedo; E.F. Collins; A.C.D. Neto, Extension of Selenium RC tool to perform automated testing with databases in web applications, 8th International Workshop on Automation of Software Test, 2013.

[6] V.S.F. Pinheiro; N.M.C. Valentim; A.M.R. Vincenzi, Um Comparativo na Execução de Testes Manuais e Testes de Aceitação Automatizados em uma Aplicação Web, XIV Simpósio Brasileiro de Qualidade de Software, 2015.

[7] J.L. Moura; A.S. Charão; J.C.D. Lima; B.O. Stein, Test case generation from BPMN models for automated testing of Web-based BPM applications, 17th International Conference on Computational Science and Its Applications, 2017.

[8] V.S.F. Pinheiro; R. Chiavegatto; A.F. Vieira; E.H. Oliveira; E. Barroso; A. Amorim; T. Conte, Especificação e Automação Colaborativas de Testes utilizando a técnica BDD, XII Simpósio Brasileiro de Qualidade de Software, 2013.

[9] S.A.M. Pinheiro, Estudo e implementação de testes de software em desenvolvimento ágil. Dissertação de mestrado da Universidade do Minho, 2015.

[10] M.B. Pontes, Introdução a testes de software, Engenharia de Software Magazine, 2009.

[11] A.A. Vicente, Definição e gerenciamento de métricas de teste no contexto de métodos ágeis, Tese de Doutorado da Universidade de São Paulo, 2010.

[12] Y. Cheon; G.T. Leavens, A simple and practical approach to unit testing: The JML and JUnit way, 16th European Conference on Object-Oriented Programming, 2002.