8-10-2020

Automação de Testes: Um Estudo de Caso

Jorair Alberto
Universidade Federal de Mato Grosso, jorair.alberto@gmail.com

Joyce Oliveira
Universidade Federal de Mato Grosso, joyceoliveira@ufmt.br

Saulo Reis
Universidade Federal de Mato Grosso, saulorsreis@gmail.com

Vanice Cunha
Universidade Federal de Mato Grosso, vanice@ic.ufmt.br

Simone Pelegrini
Tribunal de Contas do Estado de Mato Grosso, simonea@tce.mt.gov.br

See next page for additional authors

Automação de Testes: Um Estudo de Caso

Artigo Completo

Jorair Alberto
Aluno de Graduação
Universidade Federal de Mato Grosso
jorair.alberto@gmail.com

Joyce Oliveira
Professor
Universidade Federal de Mato Grosso
joyceoliveira@ufmt.br

Saulo Reis
Professor
Universidade Federal de Mato Grosso
saulorsreis@gmail.com

Vanice Cunha
Professor
Universidade Federal de Mato Grosso
vanice@ic.ufmt.br

Simone Pelegrini
Secretaria de Controle Externo
Tribunal de Contas do Estado de Mato Grosso
simonea@tce.mt.gov.br

Eunice dos Santos Nunes
Professor
Universidade Federal de Mato Grosso
eunice.ufmt@gmail.com

Abstract

This paper presents a case study about the use of automation of regression tests with continuous integration to make the development process faster, more efficient and to reduce costs. A set of tests was carried out manually in a system of the Court of Accounts of the State of Mato Grosso, after they were recreated in an automated way and with continuous integration with free tools. Finally, the quality of the methodology used was analyzed by comparing the results of manual tests with automated tests. The results showed that the use of automated tests with continuous integration potentializes resources, dynamizes the testing process and increases the quality of the software.

Key-words

Regression tests, continuous integration, software quality

Resumo

Este artigo apresenta um estudo de caso sobre o uso de automação de testes de regressão com integração contínua para tornar o processo de desenvolvimento mais célere, eficiente e reduzir custos. Um conjunto de testes foi realizado manualmente em um sistema do Tribunal de Contas do Estado de Mato Grosso, depois foram recriados de forma automatizada e com integração contínua com ferramentas gratuitas. Por fim, foi analisada a qualidade da metodologia utilizada comparando os resultados de testes manuais com testes automatizados. Os resultados mostraram que uso de testes automatizados com integração contínua potencializa recursos, dinamiza o processo de testes e aumenta a qualidade do software.

Palavras-chave

Testes de regressão, integração contínua, qualidade de software

Introdução

A inserção de novas tecnologias no mercado tem possibilitado a celeridade e melhor qualidade na entrega das soluções computacionais desenvolvidas por organizações especializadas na área. Desde a década de 90, Porter (1986), já considerava "crucial a utilização efetiva da TI para a sobrevivência e a estratégia competitiva das organizações". Existem várias fases no desenvolvimento de software, dentre as quais se destaca a fase de testes. De acordo com Mann et al (2019), teste de software existe com o objetivo principal de aferir a qualidade do que está sendo desenvolvido, como também, comparar os resultados dos testes com os esperados nos requisitos elencados nas fases iniciais do projeto. Candea e Godefroid (2019) afirmam que testes quando realizados manualmente consiste em um processo excruciante e quase inviável do ponto de vista financeiro e operacional. Felderer e Garousi (2020) indicam que muitas organizações não testam seus softwares de forma adequada ou simplesmente não realizam os testes em seus produtos, fato que tem reduzido drasticamente a qualidade dos sistemas, além de atrasar o processo de desenvolvimento, pois as manutenções corretivas são constantes. Segundo dos Santos (2016), com investimento, o teste de software reduz em 70% o índice de retrabalho para a correção de falhas com o produto já no ambiente de produção, além de obter uma melhora no deploy de novas versões do produto, com uma grande probabilidade de que as falhas sejam detectadas antes da homologação do produto. Logo, encontrar alternativas que facilitem o trabalho da equipe de testes pode tornar o processo mais atraente, principalmente do ponto de vista operacional e financeiro, reduzindo custos e potencializando recursos (Mann et al, 2019).

Este trabalho apresenta um estudo de caso realizado no Tribunal de Contas do Estado de Mato Grosso para tornar o processo de testes de software mais célere, eficiente e com custos reduzidos. O foco da solução é a área de Qualidade de Software, precisamente, Automação de Testes de Regressão. O teste de regressão é aplicado quando uma nova versão do sistema é colocada em produção ou ainda quando há manutenção em algum módulo do software, onde os desenvolvedores alteram ou introduzem um novo código (Savoldi e Oliveira). Para aferir se houve compatibilidade com o código existente o teste de regressão é executado, rodando toda massa de teste criada para a aplicação (Sommerville, 2019).

Este trabalho possui por objetivo apresentar o relato da adoção de testes automatizados por meio de um estudo de caso em uma organização pública e analisar os resultados considerando-se a prática de testes automatizados em detrimento de testes manuais em termos de rapidez, eficiência e custo.

Teste de regressão

Bernardo e Kon (2008) consideram os testes automatizados como essenciais no processo de teste de software, pois exercitam funcionalidades de sistemas através de programas ou scripts que ao testar a aplicação fazem verificações automáticas nos efeitos colaterais obtidos. Com a vantagem de executar uma grande massa de testes com agilidade e pouco esforço. Bernardo (2011) afirma que os testes automatizados são de grande ajuda para garantir uma melhor qualidade na produção de softwares. Eles podem ser eficazes, viáveis do ponto de vista econômico e de fácil manutenção. O mesmo apresenta em sua pesquisa as principais práticas, padrões e técnicas para guiar o processo da criação, manutenção e gerenciamento dos casos de testes automatizados, tudo de forma crítica e sistemática. Barbosa et .al (2011) concluíram que na busca pela melhor solução quando o assunto é qualidade de software, as empresas devem investir em estratégia de automação de testes, todavia, elas não podem estar focadas na técnica ou estratégia em si, mas nas soluções que realmente cobrirão as situações de risco do sistema.

Segundo Vaz (2015), implementar um programa que testa seu programa, parece redundante, mas é a maneira mais viável do ponto de vista financeiro e que varre todo o sistema, buscando garantir que a saída será sempre a esperada. O teste automatizado, se comparado com o manual, é muito rápido, logo pode ser rodado repetidamente. Isso faz com que os eventuais problemas sejam detectados e corrigidos rapidamente, gerando redução de custos.

A complexidade e o tamanho dos códigos que compõem um software tornam praticamente impossível determinar os impactos se houver a necessidade de uma pequena correção de bugs, o que realmente se sabe é que ao longo de décadas, defeitos de software têm causado prejuízos incalculáveis, sejam eles de natureza financeira ou não (Alves, 2018). Por esse motivo que os testes de regressão são tão essenciais no processo de desenvolvimento. Conforme Salomão (2016) testes regressivos asseguram que

mudanças feitas no software não alterem comportamentos de componentes que permaneceram inalterados. O teste irá reduzir os impactos causados por uma eventual correção de bugs, garantindo também a seguridade da entrega de uma aplicação que condiz com o esperado. Segundo Pressman (2016), o teste de regressão consiste em reexecutar o subconjunto de testes rodados antes da inserção de uma nova versão do software ou integração de um novo módulo do sistema, pois novos caminhos de fluxos de dados podem ser estabelecidos quando o novo código é inserido. Logo, como estratégia de teste de integração, o teste de regressão é executado visando assegurar que as alterações não tenham propagado efeitos colaterais indesejados (Pressman, 2016).

Conforme representado na figura 1, ao se realizar testes de regressão, um conjunto de testes é criado e inserido no software de automação, feito a partir de um roteiro de testes, que é um artefato do software em desenvolvimento. Após sua execução, caso seja diagnosticada alguma falha, o analista avalia as possíveis causas. Se se as falhas forem em decorrência de bug no software de automação, os scripts são mantidos, o fluxo é encerrado e aplicado novamente. Caso a falha seja de origem da nova implementação, a falha é diagnosticada e posteriormente o relatório de erro é encaminhado para o setor responsável para as devidas correções. Para aferir se há compatibilidade com o código existente, o teste de regressão é executado, rodando toda a massa de testes criada para a aplicação. Para efeito de aumento de produtividade e de viabilidade dos testes, recomenda-se a utilização de ferramentas de automação de testes, dessa forma todos os testes podem ser reexecutados com maior agilidade (Finkler apud Papo, 2010).

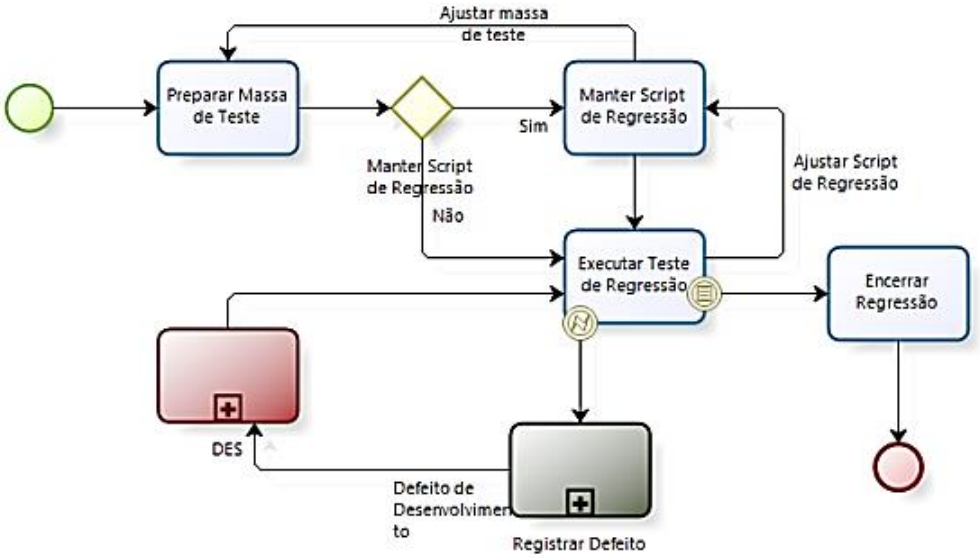


Figura 1- Esquema para o teste de regressão (UFG, 2011)

O teste automatizado, assim como o manual, é feito pelo analista de teste, contudo, utilizando ferramentas de automação. Isso faz com que os eventuais problemas sejam detectados e corrigidos rapidamente, gerando redução de custos (Salomão, 2016). Segundo Bernardo e Kon (2008), muitas empresas preferem os testes manuais, fato que pode gerar alguns problemas que implicam diretamente no desenvolvimento do software, tais como: atrasos e falta de confiabilidade. Bernardo e Kon (2008) ainda afirmam que os testes automatizados são de grande ajuda para garantir uma melhor qualidade na produção de softwares. Testes automatizados podem ser eficazes, viáveis do ponto de vista econômico e de fácil manutenção e são essenciais na realização de testes de regressão. Atualmente existem várias ferramentas para automação de testes. Algumas Open Source, como o SpecFlow - estrutura de teste que suporta práticas de BDD (Desenvolvimento Guiado por Comportamento). Outra opção Open Source é o aplicativo Watir,

ferramenta de testes no Ruby, usada especialmente para aplicações web. Ainda há o Katalon Studio, ferramenta utilizada neste projeto, que se destaca pela riqueza de recursos e facilidade de uso.

Segundo Chicanelli et al (2019), 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. Chicanelli et al (2019) ainda afirma que é neste contexto que surge a integração contínua que permite a entrega mais eficaz do software e com menos defeitos, incrementando funcionalidades do software a cada build.

Metodologia

A pesquisa foi conduzida por meio de um estudo de caso. Estudos de caso são fortemente recomendados por Pries-Heje et al (2008), como estratégia de análise por permitir aprofundar a avaliação do objeto de estudo considerando-se uma realidade concreta. O método de estudo de caso foi selecionado por possibilitar que fenômenos complexos sejam pesquisados dentro de seus contextos (Baxter e Jack, 2008).

Descrição do caso

O estudo de caso foi conduzido no Tribunal de Contas do Estado de Mato Grosso (TCE-MT) devido a um convênio existente entre esta organização e a universidade na qual os autores atuam. Atualmente, a equipe de desenvolvimento de software do Tribunal de Contas do Estado de Mato Grosso, utiliza um Processo de Desenvolvimento e Manutenção de Software (PDMS) que foi criado em 2010 a partir do framework RUP (Rational Unified Process), que é uma metodologia para gerenciar projetos de desenvolvimento de software orientados baseado em UML (Unified Modeling Language) como uma das ferramentas para especificação de sistemas.

O PDMS é representado e estruturado com base em elementos básicos, que representam “quem” faz “o quê”, “como” e “quando” é feito e “porque” são feitos, representado através de modelos de processos. Cada uma das 4 fases (Concepção, Elaboração, Construção, Transição) está representada em um modelo de processo separado com um objetivo específico e eventos iniciais e finais. Cada atividade possui papéis responsáveis por estas e os artefatos produzidos, assim como o uso de ferramentas e recursos de apoio a mesma.

Dentro da fase de Construção, os testes são elaborados e executados pela equipe de forma manual e exploratória (teste funcional – Caixa-preta). O teste funcional – ou Caixa Preta - é aplicado nas novas versões integradas do software em desenvolvimento. Contudo, sua característica é atuar no código executável, não acessando o código fonte do módulo ou unidade. O conhecimento prévio das saídas esperadas é essencial para comparar as respostas quando essa abordagem for utilizada. Sendo assim, o teste falha quando o resultado obtido for diferente do esperado (Everett et al, 2007).

O teste de regressão foi realizado no Sistema JusConex-e (Figura 2) desenvolvido pela própria equipe de Tecnologia da Informação do TCE-MT. Este sistema está em execução e os testes foram realizados na versão 1.7. Este sistema é instrumento de transparência e eficiência que permite à comunidade de usuários o acesso rápido. No JusConex-e podem ser realizadas pesquisas por Enunciados de Jurisprudência (decisões em caso concreto); Prejulgados de Tese (consultas e prejulgados); Súmulas; e Jurisprudência (todas as bases).

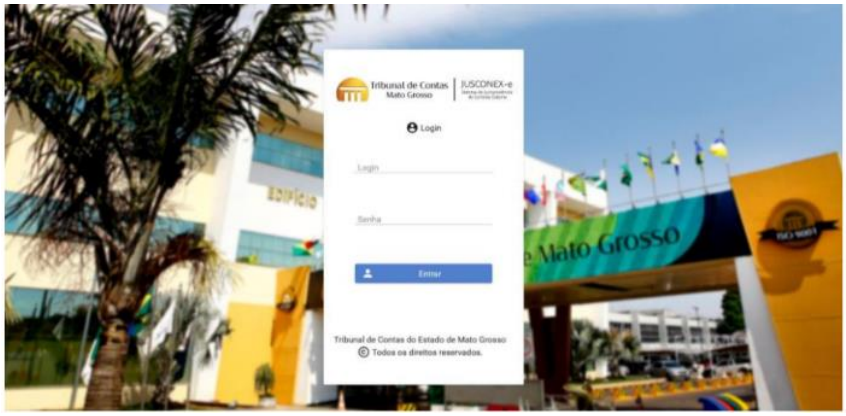


Figura 2. Tela de login do JusConex-e (Fonte: TCE –MT, 2018)

O teste focou no módulo de Cadastro, Áreas e Temas, no qual foram automatizados todos os casos de teste que já haviam sido testados manualmente. A bateria de testes foi criada segundo o Roteiro de Teste e Aceitação - RTA versão 1.0 gerado na Fase 3 do PDMS Processo de Construção.

Ferramentas utilizadas e processo de aplicação dos testes

A metodologia aplicada para a realização dos testes foi: Escolher um sistema que já está em desenvolvimento no Órgão; selecionar uma bateria de testes realizada manualmente pela equipe responsável; recriar os testes de forma automatizada utilizando as ferramentas previamente selecionadas; rodar os testes em nos ambientes selecionados; fazer integração contínua e comparar os resultados. Para os testes e aplicação do estudo de caso as ferramentas utilizadas foram o Katalon Studio, o JBoss, o SVN Subversion e o Jenkins.

O Katalon Studio é uma solução de automação simples e poderosa criada para testadores em todos os lugares. O Katalon Studio revoluciona o uso de estruturas de automação de teste de código aberto, como Selenium e Appium, eliminando suas complexidades técnicas para permitir que desenvolvedores e QAs configurem, criem, executem, relatem e gerenciem com eficiência seus testes automatizados. Ele também oferece uma alternativa viável para soluções de automação de testes comerciais que não são acessíveis para muitas equipes de pequeno e médio porte (Katalon, 2018). Essa ferramenta foi escolhida para fazer o processo de automação do teste de regressão no Sistema JusConex-e. Após análise de outras soluções existente no mercado, ela foi a que melhor se encaixou no perfil, do ponto de vista da economicidade, facilidade no manuseio e maior flexibilidade na interação com navegadores.

O JBoss AS (atualmente denominado WildFly) é um servidor de aplicações baseado em Java. A grande vantagem de um servidor de aplicações é que os desenvolvedores podem se concentrar nas necessidades de negócio. Aspectos como conexões a bancos de dados, autenticação e gerenciamento de recursos são gerenciados pelo servidor de aplicações. Além disso, o padrão Java EE define padrões abertos que aceleram o desenvolvimento com uso de API padronizada e pensada para computação distribuída (JBOSS, 2018). A ferramenta já é utilizada no ambiente de desenvolvimento do TCE-MT, logo, não houve necessidade de configurá-lo para realizar a bateria de testes. Isso também facilitou o processo de implantação da integração contínua dos testes automatizados. O Subversion existe para ser universalmente reconhecido e adotado como um sistema de controle de versão de código aberto e centralizado, caracterizado por sua confiabilidade como um refúgio seguro para dados valiosos; a simplicidade de seu modelo e uso; e sua capacidade de suportar as necessidades de uma ampla variedade de usuários e projetos, de indivíduos a operações corporativas de larga escala (Subversion, 2018).

Os gerenciadores de versão são excelentes e indispensáveis para que a equipe de desenvolvimento possa trabalhar de forma harmônica e com responsabilidade no versionamento do software. O SVN é utilizado em vários projetos do Tribunal, como também o Git (outro gerenciador de versão). Através do

Subversion foi possível centralizar, além das versões do Sistema, a massa de teste criada no Katalon, para posteriormente fazer a integração contínua.

O Jenkins é um servidor de automação independente e de código aberto que pode ser usado para automatizar todos os tipos de tarefas relacionadas à criação, teste e distribuição ou implementação de software (Jenkins, 2018). Essa ferramenta é extremamente útil e eleva o nível do desenvolvimento de Software. Uma poderosa solução que é capaz de integrar praticamente todo tipo de tarefa. Com ela é possível dinamizar o processo de teste de forma que os recursos pessoais possam ser otimizados em outras áreas da Qualidade. Com tantas possibilidades, seu uso seria inevitável no projeto.

A mecânica da Integração Contínua funcionou da seguinte forma: Todas as vezes que alguém realizava o upload do código no SVN, o Jenkins contruía a aplicação, colocava-a para rodar no JBoss e chamava a ferramenta Katalon que executava a bateria de testes no Sistema JusConex-e. Assim, eram realizados os Testes de Regressão, aferindo se houve ou não regressão nas novas releases do sistema em desenvolvimento. A Figura 3 mostra um processo de integração contínua sendo executado de forma metódica. Nesse exemplo, foi acrescentado em partes específicas (em vermelho) o local de interação de cada ferramenta proposta neste trabalho, com o objetivo facilitar ao leitor a compreensão sobre a dinâmica da integração contínua dentro do ciclo de desenvolvimento.

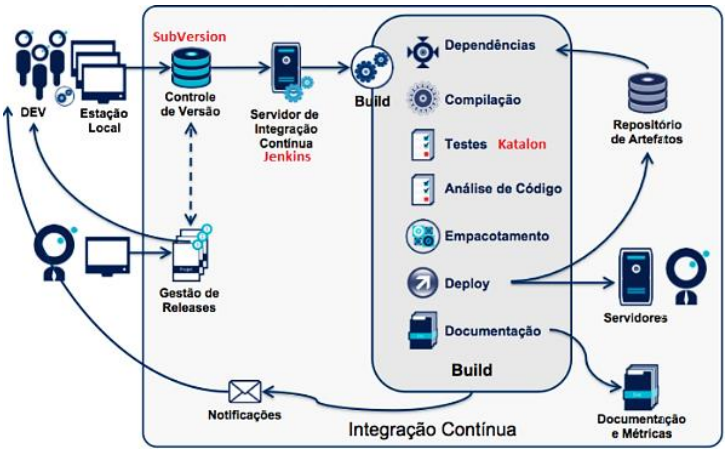


Figura 3: Processo de Integração Contínua em Ambiente de Desenvolvimento (Fonte: Fernandes, 2015 (Adaptado)).

Resultados

No Katalon, foram automatizados 45 testes referentes ao módulo de Cadastro, Áreas e Temas do Jusconex-e (Figura 4). Os casos de testes foram organizados através da criação de diretórios: CADASTRO AREAS E TEMAS – FLUXO PRINCIPAL – FLUXO ALTERNATIVO – FLUXO DE EXCEÇÃO – REQUISITOS ESPECIAIS. Após organizar os testes através dos diretórios foi simulada sua execução via browser headless (navegador sem interface gráfica), pois o servidor que deu suporte a Integração Contínua (IC) é Linux e não possui interface gráfica (via linha de comando).

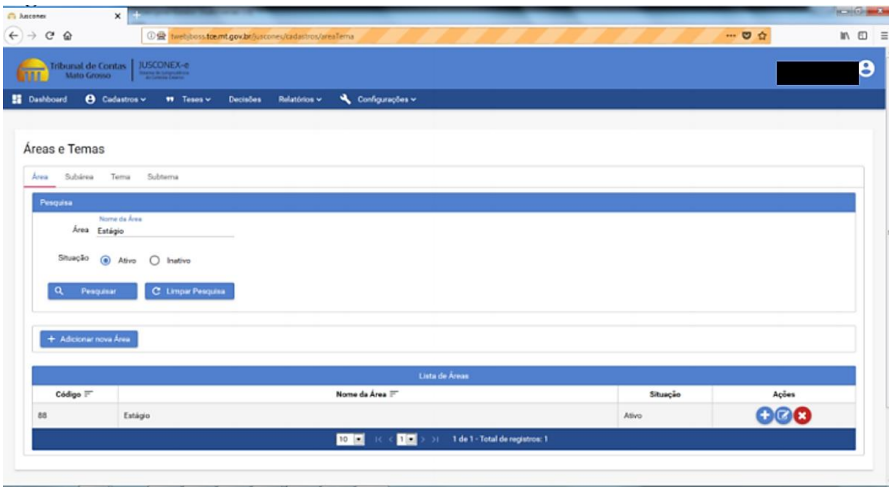
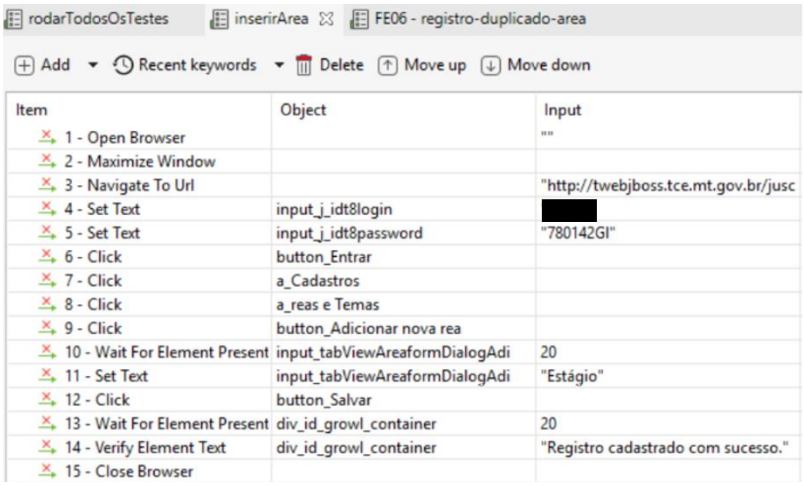


Figura 4. Tela do módulo de cadastro do JusConxex-E – Áreas e temas (Fonte: TCE-MT, 2018)

As falhas encontradas foram: Redundância na mensagem de cadastro, Erro de acentuação, Excesso de mensagens – Inserção, Excesso de mensagens e redundância – Edição, Registros inativados sem requisição do usuário e Mensagem não apresentada ao usuário.

- **Redundância na mensagem de cadastro:** Quando um registro é inserido ou excluído no banco de dados uma mensagem é mostrada ao usuário: "Registro Cadastrado com Sucesso. ou Registro excluído com Sucesso. "Todavia, existe um título na mensagem: "Sucesso", logo, as mensagens se tornam: "Sucesso Registro cadastrado com sucesso. e Sucesso Registro excluído com sucesso."
- **Erro de acentuação:** Segundo o RTA, as mensagens exibidas ao usuário, no momento da inserção ou exclusão de registros, devem ser apresentadas seguidas do ponto de exclamação (Registro Cadastrado com Sucesso! ou Registro excluído com sucesso!), o que não ocorre. As mesmas são exibidas com ponto final (Registro Cadastrado com Sucesso. Registro excluído com sucesso.).
- **Excesso de mensagens – Inserção:** Ao fazer a inserção da Área, Subárea, Tema e Subtema são exibidas além da mensagem: "Sucesso Registro cadastrado com sucesso." também a mensagem: "Aviso Pesquisa sem resultado.", sendo que esta última não deveria ser apresentada.
- **Excesso de mensagens e redundância – Edição:** Ao tentar editar um registro, uma mensagem redundante é apresentada: "Sucesso Registro editado com sucesso." como também uma mensagem de pesquisa é apresentada: "Aviso Pesquisa sem resultado.".
- **Registros inativados sem requisição do usuário:** Ao tentar excluir um registro de forma regressiva - Subtema - Tema - Subárea - Área - (mesmo não contendo vínculo à teses) o Sistema ao invés de fazer a exclusão física no banco, apenas edita o registro, colocando o mesmo no modo "inativo", isso para as telas - Subtema - Tema - Subárea. Todavia, ao fazê-lo, uma mensagem de exclusão é apresentada ao usuário como se de fato houvera a exclusão física.
- **Mensagem não apresentada ao usuário:** Quando o usuário requisita a exclusão do registro pai "Área", estando o mesmo com alguma tese vinculada, o Sistema não permite sua exclusão (previsto nas regras de negócio do Sistema JusConex-e – anexo 1). Todavia, não apresenta nenhuma mensagem ao usuário (a mensagem, segundo RTA, seria: "Registro não poderá ser excluído pois está vinculado a outro Registro") e as telas - Subárea - Tema e Subtema são automaticamente inativadas.

Na Figura 5 foi mostrada a execução do caso de teste Tela de Cadastro do Sistema JusConex-e na ferramenta Katalon. Na linha 2 foi executado um comando chamado, Maximize Window. O mesmo faz com que o navegador, no momento em que é aberto, tenha sua resolução de tela alterada para o tamanho da resolução da tela da máquina na qual está sendo executados os testes. Isso faz com que alguns problemas de responsividade sejam solucionados. Dos 45 testes rodados no Katalon, 13 falharam. É importante ressaltar que o Sistema JusConex-e na data de realização dos testes estava na versão 1.7. Portanto, as falhas aqui destacadas passaram despercebidas nos testes manuais exploratórios em todas as versões anteriores. Isso contraria a ideia de que o que efetivamente encontra erros é o teste manual, sendo que 70% dos testes no mercado são manuais. Os testes manuais são essenciais, pois a automação, no modo de gravação de passos é construído através das habilidades do profissional tester. Todavia, quando esses testes se tonam automatizados, a incidência de não captarem as possíveis falhas ou erros são drasticamente reduzidas. Uma dificuldade encontrada durante a realização dos testes se referiu ao tempo de execução dos testes realizados pela ferramenta Katalon. Em sua configuração padrão, o Katalon não possui um tempo de espera entre a execução dos comandos programados. Dependendo do navegador utilizado alguns testes falhavam, não pelo motivo do software estar com defeito, mas devido aos elementos HTML ainda não estarem presentes no momento em que o teste era executado. Para sanar esse problema, foi atribuído um delay geral na ferramenta Katalon de 1 segundo, como também foi atribuído uma função chamada "Wait For Element Present" que permite a ferramenta esperar o elemento buscado estar presente na página para dar prosseguimento na execução dos comandos seguintes.



Item	Object	Input
1 - Open Browser		""
2 - Maximize Window		
3 - Navigate To Url		"http://twebjboss.tce.mt.gov.br/jusc
4 - Set Text	input_j_idt8login	[redacted]
5 - Set Text	input_j_idt8password	"780142GI"
6 - Click	button_Entrar	
7 - Click	a_Cadastros	
8 - Click	a_reas e Temas	
9 - Click	button_Adicionar nova rea	
10 - Wait For Element Present	input_tabViewAreaformDialogAdi	20
11 - Set Text	input_tabViewAreaformDialogAdi	"Estágio"
12 - Click	button_Salvar	
13 - Wait For Element Present	div_id_growl_container	20
14 - Verify Element Text	div_id_growl_container	"Registro cadastrado com sucesso."
15 - Close Browser		

Figura 5: Teste da tela de cadastro - Inserir Área no Katalon (Fonte: o autor)

Em alguns casos foi preciso analisar o texto de alerta ou mensagens exibidas pelo sistema ao usuário, fato que demandou muito tempo, pois nesse caso era necessário localizar o elemento em que a mensagem foi inserida e fazer uma leitura do seu conteúdo. Localizar esses elementos e tratar essas informações foi um desafio, pois os identificadores dos elementos são dinâmicos e criados através de frameworks. A solução foi utilizar o XPath para localizar os elementos na página e em alguns casos fazer comparações utilizando classes em vez de Strings.

Conclusão

Chicanelli et al (2019), realizou um trabalho semelhante sobre a aplicação de testes automatizados em comparação aos testes manuais também por meio de um estudo de caso na área de saúde com base em uma metodologia ágil. Chicanelli et al (2019) utilizaram as ferramentas Selenium e SpecFlow, analisaram a

automação de testes sob aspectos sociais, humanos e econômicos e chegaram as seguintes conclusões: (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 foi de mais de 3 vezes.

No presente artigo foram utilizadas as ferramentas Katalon, Jenkins e SVN e foi relatada a adoção de uma nova solução tecnológica para auxiliar no processo de testes de software em uma organização do setor público (TCE-MT). O foco foi a Área de Qualidade por meio da aplicação de testes de regressão. Utilizando ferramentas gratuitas e integração contínua, foi possível propor uma forma ágil de realizar os testes de regressão automatizados. O objetivo foi alcançado, pois a equipe presenciou e testou uma solução viável, eficaz, eficiente e que possibilita a redução de custos de produção, uma vez que os recursos pessoais utilizados nos testes manuais exploratórios, podem ser remanejados para outras áreas deixando a ferramenta automatizada fazer o trabalho exaustivo e ainda contemplando resultados que sem o uso delas seriam quase impossíveis alcançar.

Após a realização do estudo de caso, foi percebido que o uso de testes automatizados agrega na qualidade e produção do software. Entretanto, durante o processo de adesão aos testes automatizados foram identificados os seguintes desafios:

1. Dificuldade de escolher as ferramentas certas considerando aspectos de viabilidade técnica e econômica: foi realizado um vasto estudo com comparação entre várias ferramentas. Este estudo foi apresentado para a gestão que aprovou a seleção proposta dado os benefícios apresentados em detrimento de outras ferramentas pesquisadas.
2. Necessidade de rápido aprendizado sobre as ferramentas e as tecnologias envolvidas devido a demanda de adotar a prática de testes automatizados não só no Sistema JusConex-e, mas em todos os sistemas do TCE-MT. Isto foi facilitado porque há uma grande variedade de material disponível para consulta na Web sobre o assunto. As ferramentas utilizadas são simples de instalar e utilizar.
3. Dificuldade de mudar a cultura organizacional do uso de testes manuais. Apesar da necessidade de o TCE-MT adotar a automação de testes e de os benefícios desta prática terem sido visíveis durante a execução deste projeto, percebeu-se uma certa resistência por parte dos desenvolvedores de aderir a este formato. Isto se deu tanto pela cultura arraigada do uso de testes manuais, quanto pelo receio de atrasar a entrega de funcionalidades devido ao tempo a ser dedicado para aprender sobre automação de testes. Percebeu-se que uma forma de agilizar a curva de aprendizado seria disponibilizar cursos sobre o tema ministrados por especialistas. Entretanto, a burocracia envolvida na elaboração, aprovação e disponibilização do curso, que seria realizado por uma organização conveniada ao TCE-MT, não traria tanta agilidade quanto o auto aprendizado.

Em temos teóricos, este artigo contribui para ampliação do corpo de conhecimento sobre o tema. Em termos práticos, os resultados são úteis para desenvolvedores e analistas que buscam compreender como testes automatizados funcionam, quais ferramentas podem ser utilizadas para realiza-los, além dos benefícios e desafios associados a automação de testes. Como trabalho futuro pretende-se realizar o mesmo processo de testes automatizados em outros sistemas para investigar a efetividade do uso destes em detrimento de testes manuais considerando-se outra bateria de testes. Além disso, pretende-se desenvolver um guia que visará a padronização da elaboração e execução de testes na organização. O guia poderá ser útil para que outras organizações públicas possam adotar a prática de automação de testes em sua rotina de desenvolvimento e manutenção de software.

Referências

Alves, L., S. 2018. Automatizando a Formação de Oráculos de Testes de Regressão Constituídos por Dados Reais. Trabalho de conclusão do curso de Bacharelado em Informática - Instituto Federal de Educação, Ciência e Tecnologia de Goiás - Campus Inhumas.

Barbosa, F., Bernardes; T., Vasconcelos I. Teste de Software no Mercado de Trabalho. Revista Tecnologias em Projeção; v.2; n. 1; p. 49-52; jun. 2011.

Baxter, P. e Jack, S. 2008. Qualitative case study methodology: Study design and implementation for novice researchers. The qualitative report, 13(4), 544-559

Bernardo, P., C. 2011. Padrões de testes automatizados. Dissertação (Mestrado em Ciências da Computação) - Instituto de Matemática e Estatística da Universidade de São Paulo. São Paulo. 2011.

Bernardo, P C; Kon, F. 2008. A Importância dos Testes Automatizados. Engenharia de Software Magazine, 1(3), pp. 54-57.

Candea, G., e Godefroid, P. 2019. Automated software test generation: some challenges, solutions, and recent advances. In Computing and Software Science, pp. 505-531. Springer, Cham.

Chicanelli, R., Vechiato, D., Ventura, T., Gomes, R., Takato, N., e Mendes, L. 2019. Aspectos sociais, humanos e econômicos da utilização de testes automatizados no desenvolvimento de sistemas. Décima Oitava Conferência Ibero Americana de Sistemas Cibernética e Informática.

Dos santos, Daniel Botter. 2016. Implantação de teste de software em empresa de pequeno porte: um estudo de caso; orientador: Prof. Ms. Allan Cesar Moreira de Oliveira. Marília, SP: [s.n.], 2016.

Everett, Gerald D.; Mcleod JR, Raymond. 2007. Software testing: testing across the entire software development life cycle. John Wiley & Sons.

Felderer, M., e Garousi, V. 2020. Together We Are Stronger: Evidence-Based Reflections on Industry-Academia Collaboration in Software Testing. In International Conference on Software Quality (pp. 3-12). Springer, Cham.

Fernandes, S., L. 2015. Maturidade da integração contínua. Disponível em: https://blog.octo.com/pt-br/maturidade-da-integracao-continua/. Acesso em 27 de junho de 2020.

Finkler, F., E. 2017. Análise comparativa de testes estruturados e testes exploratórios de especialistas. Trabalho de Conclusão do Curso Sistemas de Informação – Universidade do Vale do Taquari - Univates, Lajeado.

Jboss as; disponível em: https://www.4linux.com.b; acessado em 10 de junho de 2018 às 18h.

Jenkins; Site oficial da ferramenta JENKINS disponível em: https://jenkins.io; acessado em 18 de julho de 2018 às 14h.

Jusconex-e. Versão 1.7. 2018. Sistema WEB para automatizar o trabalho da equipe do TCE-MT referente à Jurisprudência.

Katalon studio; Site oficial da ferramenta KATALON; disponível em: http:// www.katalon.com; acessado em 15 de julho de 2018 às 10h.

Mann, M., Tomar, P., & Sangwan, O. P. (2019). Automated software test optimization using test language processing. Int. Arab J. Inf. Technol., 16(3), 348-356.

Porter, Michael E. Estratégia Competitiva – Técnicas para análise de indústrias e da concorrência. 18ª Edição. São Paulo-SP: Campus, 1986.

Pressman, Roger S. 2016. Engenharia de software: uma abordagem profissional. 7. ed. – Dados eletrônicos. – Porto Alegre : AMGH, pg 357-421.

Pries-heje, J., Baskerville, R., e Venable, J. 2008. Strategies for design science research evaluation. ECIS 2008 proceedings, 1-12.

Salomão, R., Gonzaga. 2016. Análise da Relevância de Testes de Regressão para o Mercado de Desenvolvimento de Software do Triângulo Mineiro. Trabalho de Conclusão de Curso Sistemas de Informação – Universidade Federal de Uberlândia, Minas Gerais.

Savoldi, G., e Oliveira, R. Estudo e aplicação da ferramenta sikuli para auxílio em testes funcionais e de usabilidade por meio da automação por imagem. In Anais da II Escola Regional de Engenharia de Software (pp. 163-168). SBC.

Sommerville, Ian. Engenharia de Software 9. ed. São Paulo: Pearson Prentice Hall. pg 144-159. 2019.

SVN - Subversion; Site oficial da ferramenta Subversion; disponível em: https://subversion.apache.org/; acessado em 10 de junho de 2019 às 11h.

TCE-MT; Site oficial do Tribunal de Contas de Mato Grosso disponível em: http://www.tce.mt.gov.br; acessado em 17-8-2018 às 16h.

Vaz, L., M. Teste de software: importância e aplicação no desenvolvimento. 2015. Trabalho de Conclusão de Curso Sistemas de Informação – Faculdade Pitágoras, Divinópolis.

UFG. Processo de Teste de Software para Micro e Pequenas Empresas. Universidade Federal de Goiás. 2011. http://apoema.inf.ufg.br/pts-mpe/pts-mpe/873ed559-8aeb-4ede-96f9-2b029e185c48.htm. Acesso em 30 de abril de 2020.