UNIVERSIDADE EVANGÉLICA DE GOIÁS - UNIEVANGÉLICA
ENGENHARIA DE COMPUTAÇÃO/ENGENHARIA DE SOFTWARE

JONAS SANTOS CORRÊA E WELINGTON TIERRY VITOR SILVA

Elaboração de um Mínimo Processo Viável para Automação de Testes de Interface em Fábricas de *Software*

Anápolis

Setembro, 2021

UNIVERSIDADE EVANGÉLICA DE GOIÁS - UNIEVANGÉLICA
ENGENHARIA DE COMPUTAÇÃO/ENGENHARIA DE SOFTWARE

JONAS SANTOS CORRÊA E WELINGTON TIERRY VITOR SILVA

Elaboração de um mínimo processo viável para automação de testes de interface em fábricas de *software*

Trabalho apresentado ao Curso de Engenharia de Computação da Universidade Evangélica de Goiás – UniEVANGÉLICA, da cidade de Anápolis-GO como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Computação.

Orientador (a): Prof. Ms. Walquíria Fernandes Marins

Anápolis
Dezembro, 2021

UNIVERSIDADE EVANGÉLICA DE GOIÁS - UNIEVANGÉLICA
ENGENHARIA DE COMPUTAÇÃO/ENGENHARIA DE SOFTWARE

JONAS SANTOS CORRÊA E WELINGTON TIERRY VITOR SILVA

Elaboração de um mínimo processo viável para automação de testes de interface em fábricas de *software*

Monografia apresentada para Trabalho de Conclusão de Curso de Engenharia de _____ da Universidade Evangélica de Goiás - UniEVANGÉLICA, da cidade de Anápolis-GO como requisito parcial para obtenção do grau de Engenheiro(a) de _____.

Aprovado por:

_____
Nome completo do orientador, [Especialista | Mestre | Doutor], Sigla da universidade que atua
(ORIENTADOR)

_____
Nome completo do examinador, [Especialista | Mestre | Doutor], Sigla da universidade que atua
(AVALIADOR)

Anápolis, 7 de Dezembro de 2021.

FICHA CATALOGRÁFICA

[CORRÊA, Jonas Santos] [SILVA, Welington Tierry Vitor] **Elaboração de um mínimo processo viável para automação de testes de interface em fábricas de software** [Anápolis] 2021.
(Universidade Evangélica de Goiás – UniEVANGÉLICA, Engenheiros (as) de Computação, 2021).
Monografia. Universidade Evangélica de Goiás, Curso de Engenharia de Computação, da cidade de Anápolis-GO.

1. Testes Automatizados. Processo. Fábricas de Software.

REFERÊNCIA BIBLIOGRÁFICA

CORRÊA;SILVA, Jonas Santos; Welington Tierry. Elaboração de um mínimo processo viável para automação de testes de interface em fábricas de software. Anápolis, 2021. 41 p. Monografia - Curso de Engenharia de Computação Universidade Evangélica de Goiás - UniEVANGÉLICA.

CESSÃO DE DIREITOS

AGRADECIMENTOS

Agradecemos primeiramente a Deus, por essa conquista e a todos que tornaram possível a realização deste trabalho: nossos familiares, amigos, professores e em especial à Walquíria, nossa orientadora.

RESUMO

O desenvolvimento de software é dividido em etapas, sendo elas: definir, desenvolver, testar e manter um software (BENITTI; ALBANO, 2012). Especificamente a etapa de teste é responsável por verificar e validar se o que foi proposto no início do desenvolvimento foi atingido ao longo do processo, ou seja, delimitar se o que foi construído possui qualidade. Para esse fim são realizados testes funcionais de interface para que sejam analisadas as telas e validado se o que foi realizado está de acordo com o que foi proposto. No entanto para otimizar o tempo perdido nos testes de interface foram criados os testes de automação que também possuem como vantagem a elevação do potencial dos testes no quesito qualidade. Neste contexto, o objetivo deste trabalho será criar um processo de automação de teste de interfaces das funcionalidades de software que seja objetivo, eficiente e aplicável para fábricas de software de qualquer porte. Para isso foram realizadas pesquisas para atestar a sua eficiência e adesão que serão demonstradas nas sessões posteriores.

Palavras-chave: Qualidade de Software. Teste Funcional. Testes Automatizados. Processo. Fábricas

LISTA DE ILUSTRAÇÕES E IMAGENS

LISTA DE ABREVIATURAS E SIGLAS

Siglas	**Descrição**
BDD	*Behavior Driven Development*
FTT	Fábrica de Tecnologias *Turing*
HTML	*HyperText Markup Language*

SUMÁRIO

INTRODUÇÃO

A construção de softwares pode ser vista como um conjunto de atividades organizadas de maneira que sejam usadas para definir, desenvolver, testar e manter um software. No entanto, quando uma dessas atividades é executada de maneira falha todo o processo de construção do software é impactado. A prevenção desse impacto é norteada através dos testes de software, sendo a parte responsável por apontar possíveis defeitos e aumentar a confiança sobre o que está sendo desenvolvido, sendo assim um elemento crítico para a garantia de qualidade (BENITTI; ALBANO, 2012).

Nessa busca pela garantia da qualidade de um software é imposto o desafio de se fomentar uma cultura de não tolerância a erros, por intermédio de práticas que tem por objetivo inibir ou impedir falhas. Dentre essas práticas existe o teste funcional do software, que ocorre de uma forma mais eficiente e rápida, possibilitando que sejam encontradas inconformidades no software (SILVA; ALVES; BRUNO, 2011). Pensando em otimizar o tempo empregado nos testes funcionais sem perder qualidade, foram desenvolvidos os testes automatizados, os quais têm por finalidade exercitar o sistema, validando as funcionalidades e verificando se o que foi desenvolvido está de acordo com as especificações dos requisitos do sistema (WU; SHUQIN; JINGJING, 2009).

Os testes automatizados possuem diversas vantagens, dentre elas destacam-se: menor tempo com relação a execução dos testes; verificação do sistema durante o processo de desenvolvimento; casos de teste mais elaborados; aumento na qualidade dos testes; e a vantagem principal é a possibilidade de execução do teste sempre que necessário. Ou seja, sua implantação garante que não ocorrerão falhas humanas nos testes, além de reduzir os esforços empregados e o tempo gasto nos testes (COLLINS; KON, 2013). No entanto, a sua utilização ainda não é adotada por parte das fábricas e empresas, entre os motivos dessa não utilização pode-se destacar a falta de habilidade e de conhecimento dos analistas de teste com relação ao teste automatizado (AGUIAR, 2017).

A falta de scripts bem escritos e padronizados que descrevem bem as funcionalidades e qualquer mudança que possa ter ocorrido, ausência da extração de métricas a serem usadas no teste e a falta de investimento em ferramentas de automação e em capacitação dos analistas de teste (COLLINS; KON, 2013). A fim de criar um subprocesso a partir de um processo existente e aumentar a eficiência do processo de testes, esse trabalho terá como foco formalizar as etapas da automação e apresentar um processo minimalista que possa ser implementado na fábrica utilizada como experimento, sendo ela, a Fábrica de Tecnologias Turing (FTT). Um ambiente acadêmico de desenvolvimento de software que ao decorrer dos anos deixou de aplicar os testes de interface automatizados devido, entre alguns aspectos, a curva de aprendizado e a não replicação do conhecimento, a alta rotatividade de recursos humanos e a ausência de formalização do processo específico de automação (TEIXEIRA; CAIXÊTA, 2020).

Partindo deste ponto, o presente trabalho se justifica pelo crescimento da complexidade de softwares e necessidade de entregas com mais qualidade. Em virtude dessa necessidade, atualmente são realizadas a verificação e validação de softwares, para determinar se o que foi

desenvolvido está de acordo com a necessidade levantada pelos stakeholders e também para dizer se o software não apresenta defeitos. No entanto, mesmo com essa necessidade, a maioria das empresas tende a adotar apenas testes funcionais manuais, que demandam um tempo maior para a sua execução e também possuem um risco de falha humana elevada (AUTOMTECH, 2018).

Para suprir essa exaustão de testes funcionais de interfaces foram criados os testes automatizados, que por sua vez não possuem as limitações dos testes manuais (no que se refere a tempo e qualidade) e que podem ser aplicados sempre que houver necessidade e rapidez sem que seja perdida a qualidade. No entanto, os processos de testes automatizados atuais possuem alta complexidade para serem implantados e escassez de profissionais que passem adiante esse conhecimento (AUTOMTECH, 2018). Dessa forma o analista de teste atualmente precisa entender um pouco mais sobre programação, para assim criar os chamados scripts de automação (SILVA; ALVES; BRUNO, 2011).

Diante desse conhecimento mais técnico é evidenciada a necessidade de uma gestão e propagação do conhecimento adquirido. No entanto, essa área de gestão de conhecimento ainda está em ascensão, e mesmo com os esforços que têm sido empregados, como a criação de sistemas para que seja mantida essa experiência em locais acessíveis, no entanto ainda é um setor que carece de muitos investimentos e inovações (JANNUZZI et al., 2016). Dessa forma, Almeida e Batista (2020) exemplificam formas de propagar conhecimento dentro de uma fábrica de software, com o objetivo de reduzir a curva de aprendizagem e disseminar a cultura dos processos e ferramentas utilizadas.

Com isso, nas sessões seguintes serão levantadas as hipóteses acerca da construção desse processo de testes automatizados de interfaces, para que o mesmo seja aplicado no ambiente selecionado, visando uma resposta positiva. Trazendo assim respostas acerca de como a inclusão desse tipo de teste pode auxiliar a qualidade de software de uma fábrica deste.

1. FUNDAMENTAÇÃO TEÓRICA

1.1 Processo de Desenvolvimento de Software

O Processo de Software é um conjunto de atividades que são realizadas com foco em produzir sistemas computacionais, desde seu início ao fim (ALMEIDA et al., 2017).

Segundo AMORIM et al. (2017), o Processo de Software é um conjunto de atividades, ligadas por padrões de relacionamento entre elas, nas quais quando operadas de forma correta e em conformidade com os padrões é produzido o resultado final esperado. Esse resultado desejado é um software com baixo custo e elevada qualidade. Dessa forma chega-se à conclusão que um processo de qualidade é aquele que produz software com a mais alta qualidade (ALMEIDA et al., 2017).

Atualmente no mercado se popularizaram os processos de desenvolvimento de software ágeis que tem como proposta serem mais enxutos e menos burocratizados como o modelo cascata que era seguido até então. As metodologias ágeis promovem respostas rápidas aos ambientes de negócios que sofrem mudanças constantes, pois possuem práticas mais leves e que são mais fáceis de aplicar se consideradas ao modelo cascata (ONTIVERO, 2020).

A produção de softwares utilizando modelos ágeis parte da premissa que à medida que o usuário interage com o sistema que é criado entende suas necessidades e faz com que as prioridades e imprescindibilidades do sistema sejam avaliadas. Isso se incorpora ao feedback do cliente fazendo com que este seja mais conciso e que traga mais valor ao software que está sendo desenvolvido para ele mesmo (ALMEIDA et al, 2017).

Alguns exemplos de metodologias ágeis que podemos citar podem ser o XP e o Scrum. O Scrum é um framework baseado em três pilares fundamentais: Transparência, Adaptação e Inspeção. A utilização desses métodos ágeis é mais indicada no desenvolvimento de sistemas aos quais os requisitos passam por mudanças repentinas no decorrer do desenvolvimento (ALMEIDA et al, 2017).

1.2 Engenharia de Requisitos

A engenharia de requisitos pode ser descrita como a elaboração das metas a serem alcançadas pelo sistema, bem como a instrumentalização dessas metas em restrições e serviços. Ela também pode ser vista de uma forma mais técnica como sendo a ciência da análise e documentação dos requisitos. Mesmo em meio a tantas definições, o aspecto de intersecção entre os autores é o que uma obtenção de requisitos mal feita pode ocasionar em um projeto falho (FIGUEIRA, 2012).

Requisito é a descrição dos serviços que são fornecidos pelo sistema além de suas restrições. O requisito será sempre uma característica do software que fará com que o usuário final busque respostas para seus problemas. Contudo fica evidente a necessidade que os requisitos de software sejam bem claros e objetivos para que eles transmitam suas ideias, dado que esses serão responsáveis por apresentar as características do sistema de acordo com as definições apresentadas (FIGUEIRA, 2012).

A fase de levantamento de requisitos é uma fase crítica para o negócio, visto que é nela que serão validadas e levantadas as funcionalidades que estarão presentes no sistema inicial para que seja feito toda a parte de estimativa de esforço e de prazos. Nesse momento são levantados quatro pontos: o entendimento de domínio da aplicação, do problema, do

negócio e das necessidades e restrições do sistema. Analisando esses quatro pontos é feito o entendimento e levantamento com os stakeholders acerca de todo o escopo do sistema (FIGUEIRA, 2012).

1.3 Qualidade de Software

Um dos focos centrais do processo de desenvolvimento de software é melhorar a qualidade dos produtos que serão desenvolvidos. Para atingir essa qualidade faz-se necessário que os requisitos estejam em conformidade com o que o usuário definiu. O que acarreta em altos níveis de eficiência para o uso. Dessa forma muitos autores acreditam que a qualidade do produto está diretamente ligada a eficiência do processo que foi utilizado para construir o sistema (VARGAS; DUARTE, 2013).

Nessa busca incessante pela qualidade dos produtos são adotadas normas e padrões regulados por organismos internacionais de normalização. As normas redigidas por esses organismos fornecem atividades e processos que são essenciais para que seja produzido um software com qualidade. Os principais organismos reguladores são: a ISO (International Organization for Standardization), CMMI (Capability Maturity Model Integration) e o MPS.BR um padrão criado no Brasil que engloba os melhores padrões internacionais (VARGAS; DUARTE, 2013).

Na norma ISO 9126 é descrito um modelo de qualidade de software que é composto por duas partes, sendo elas, interna e externa. São especificadas seis características de qualidade: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Funcionalidade define como as funções do produto irão satisfazer os desejos do usuários; Confiabilidade será quando o produto é capaz de manter seu desempenho no tempo estabelecido; Usabilidade é o esforço empregado na utilização do sistema; Eficiência para como os recursos e o tempo são compatíveis com relação ao desempenho do sistema; Manutenibilidade refere-se ao esforço para realizar alterações no software e; Portabilidade é a capacidade de o software ser transferido de um ambiente para outro (BALTHAZAR, 2017).

O modelo CMMI é um modelo de referência que define a capacidade organizacional em cinco níveis, sendo eles: Inicial, Gerenciado, Definido, Gerenciado Quantitativamente e Em Otimização. O nível inicial é conhecido como caótico, nele a empresa não tem nenhuma área de processo implementada. O nível Gerenciado é para empresas que trabalham de forma reativa, mas que tem processos caracterizados por projetos. Gerenciado quantitativamente ocorre quando a empresa consegue medir e gerir seus processos. Em otimização a empresa busca sempre melhorar seus processos levantando suas falhas. Ao ser classificada por meio de um processo formal de avaliação, uma organização tem seu nível de maturidade analisado e isso diz respeito ao produto que esta pode entregar no contexto que está inserida (BALTHAZAR, 2017).

O modelo MPS.BR (Melhoria de Processos do Software Brasileiro) foi criado em 2003 e é um modelo de qualidade brasileiro que foi elaborado com o intuito de melhorar a criação de softwares das empresas brasileiras. Para a sua elaboração foram levadas em consideração as normas ISO/IEC 12207 e ISO/IEC 15504 e também o modelo internacional (CMMI Capability Maturity Model Integration). Foram estabelecidos níveis de maturidade em que as empresas se encontram para que estas possam realizar determinados processos, sendo esses níveis: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado.

1.4 Plano de Teste

O plano de teste é um documento produzido na condução de um projeto. Ele tem a função de delimitar diversas atividades de testes no projeto, guiar a exceção e o controle de atividades de teste e um mecanismo de comunicação com os stakeholders. Dessa forma será uma atividade importante para que a organização seja inserida dentro de um dos níveis de maturidade mais elevados (CMMI/MPS.BR) se o fizer de maneira correta (SOUZA; GASPAROTTO, 2013).

Esse documento pode ser elaborado tanto pelo gerente de projeto, como também pelo gerente de testes, visando planejar as atividades a serem realizadas, definir os métodos a serem empregados e estabelecer métricas e formas de acompanhamento do processo (SOUZA; GASPAROTTO, 2013).

Sendo assim o documento deve conter:

- Introdução com identificação do projeto (definições, abreviações, referências), definição de escopo e objetivos;
- Conjunto de requisitos a serem testados;
- Tipos de testes a serem realizados e ferramentas utilizadas;
- Recursos utilizados nos testes;
- Cronograma de atividades (e definição de marcos de projeto).

1.5 Casos de Teste

Os casos de testes são a descrição do que deverá ser testado, sendo composto por entradas, restrições e os resultados que serão obtidos a partir dessa interação, o qual é considerado como comportamento esperado. Originalmente os cenários de teste são derivados dos casos de uso, sendo necessário à sua escrita para cada cenário (DIAS NETO, [s.d.]).

Existem também os casos de testes escritos em Behavior Driven Development (BDD) que é utilizado quando o cliente está em sintonia com o analista de negócios, dessa forma são escritos os cenários em conjunto com o analista de testes e o desenvolvedor para que assim, os três sejam capazes de entender, se complementando, em uma linguagem comum. O BDD é escrito de maneira que traduza os cenários possíveis de ocorrer com a funcionalidade para que ajudem o time na resolução de problemas do sistema e são a base para o desenvolvimento de critérios de aceitação (ROQUETTE, 2018).

1.6 Tipos de Teste

Para certificar que não haja surpresas desagradáveis no ciclo de vida de um software são empregadas várias maneiras de se testar o que está sendo produzido. Assim, é verificado se o que foi feito está de acordo com o planejado. Dentre os tipos de testes principais estão: Teste de Regressão, Usabilidade, Segurança, Integração, Performance, Manutenção, Funcional e Interface (EQUIPE MONITORA, 2019).

- Teste de Regressão: Esse tipo de teste é indicado para aplicações que possuem uma certa criticidade e consequentemente necessitam de testes frequentes, seja por funcionalidades constantemente modificadas ou por manutenções executadas. Desta forma esse teste será realizado a cada nova iteração que for gerado, ou seja, a cada nova alteração do sistema ou modificação feita (BAUMGARTNER, 2018).
- Teste de Usabilidade: O teste de usabilidade é utilizado para realização de validações acerca da usabilidade de um produto. São analisadas neste teste questões de navegação e de entendimento da interface (VOLPATO, 2015).
- Teste de Segurança: É o teste capaz de analisar, medir e resumir o nível de segurança de uma aplicação, sendo possível a identificação de vulnerabilidades ou ameaças que coloquem em risco a integridade do sistema que está sendo desenvolvido (TECNISYS, 2020).
- Teste de Integração: É utilizado para realizar os testes de grupos de unidades integradas que formam um sistema. Esse teste tem como objetivo garantir que essas unidades que se encontram integradas estejam em conformidade entre si (FIGUEIREDO, [s.d.]).
- Teste de Performance: Com o teste de performance é possível identificar e avaliar o nível de robustez, capacidade de resposta, confiabilidade e escalabilidade de uma aplicação. É avaliado como o sistema se comporta com uma carga de trabalho considerada alta para que se tenha as respostas para cada um dos requisitos desejados (SILVA, 2019).
- Teste de Manutenção: Os testes de manutenção são realizados para garantir que com o decorrer do tempo e com atualizações ou mudanças sofridas pelo sistema não afetem o mesmo, fazendo com que fique defasado ou até mesmo inoperante (EQUIPE MONITORA, 2019).
- Teste Funcional: O teste funcional busca encontrar falhas ou erros expressos pela programação olhando para a interface gráfica ao invés do código. Dessa forma esse tipo de teste é caracterizado como sendo um teste de caixa-preta. Utilizando-se desse método de testes é mais provável uma detecção adiantada de possíveis inconformidades o que implica em um custo menor para retrabalho (SOUZA; GASPAROTTO, 2013).
- Teste de Interface: O teste de interface é realizado para garantir que todos os elementos de uma tela estejam funcionando corretamente. Desta forma esse teste permite que sejam verificados cada elemento presente na tela a partir da interação realizada com ela (ARARUNA, 2017).

1.7 Automação de Testes

A automação de testes tem como objetivo a redução do fator humano em atividades manuais que impactam no tempo e no custo final da demanda. Dessa forma seu objetivo é utilizar softwares para que sejam controladas as execuções dos softwares por intermédio de aplicações e estratégias (LIMA, 2014).

A automação reduz a probabilidade de erros das tarefas de teste, libera tempo para que o resto do trabalho de testes seja realizado da melhor forma e provê uma rede de segurança ao

sistema. A automação de testes está diretamente relacionada à qualidade do produto final. (OLIVEIRA, 2018).

O trabalho desenvolvido por TEIXEIRA (2015) possui pontos similares com o que está sendo proposto no decorrer deste experimento. As diferenças se iniciam na ferramenta escolhida que foi o Selenium IDE, para realizar os testes de interface automatizados. Esse estudo também realizou questionários com profissionais para validar suas hipóteses e verificar se as respostas condizem com seu experimento.

Partindo da ideia desse trabalho, a próxima seção busca estruturar a forma como será feito o experimento e explicar cada passo que ele possuirá.

2. METODOLOGIA DA PESQUISA

O processo mínimo que foi elaborado neste trabalho pode ser classificado como explicativo, tendo em vista que ele propõe um processo para realização de testes automatizados que visa auxiliar as equipes de teste a ter uma cultura de automação.

Para elaboração desse processo foi realizada uma pesquisa bibliográfica que continha as seguintes palavras-chave: Teste de software, testes automatizados, ferramentas de testes automatizados, testes funcionais, testes interface, qualidade de teste, processo de software, processo de teste, caso de teste. Foram selecionados portais de periódicos e de artigos. Os trabalhos escolhidos foram os que mais tiveram palavras-chaves definidas na pesquisa destacada acima. E por fim, o que foi encontrado nos trabalhos lidos foram a base da elaboração do processo proposto.

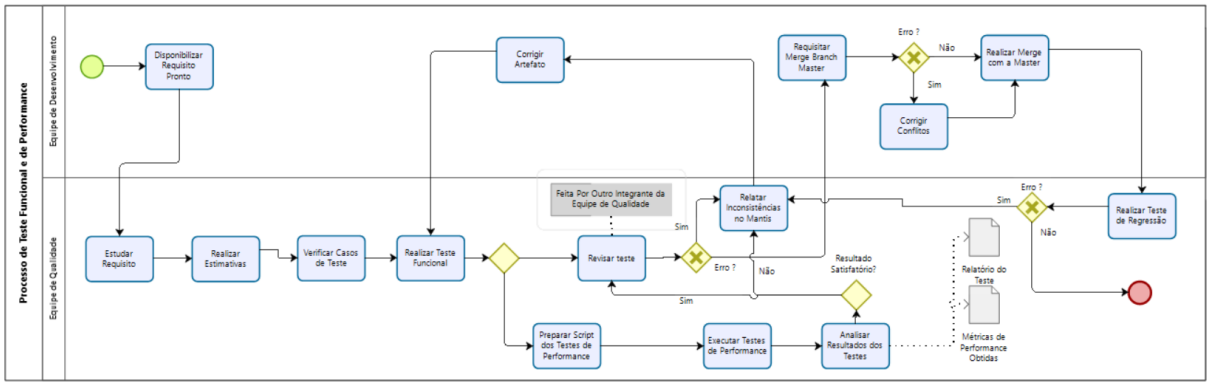
Após o levantamento teórico foi definida uma ferramenta de teste que tinha o papel de auxiliar na escrita dos scripts de teste automatizado, a ferramenta escolhida atendia os seguintes critérios: Ter uma linguagem de programação atual e consolidada no mercado, que seja de rápido aprendizado, fácil instalação e que a ferramenta esteja recebendo atualizações constantemente.

Neste trabalho serão coletados dados qualitativos para avaliar o processo proposto. Os dados qualitativos foram coletados por meio de questionários sobre a satisfação com a utilização do processo e da ferramenta e até mesmo se esse trabalho produziu um aumento em relação ao conhecimento dos membros que hoje atuam como analista tester.

3. RESULTADOS

De acordo com o que foi definido na metodologia, no tópico anterior, foi analisado o processo da Fábrica de Tecnologias Turing (FTT), ambiente que será utilizado como experimento do estudo. Com base nessa análise foi identificado o ponto onde o processo que será proposto se encaixa, levando em consideração que ele será um processo genérico para qualquer fábrica de software. Segue abaixo o processo atual da fábrica:

Imagem 1 – Processo antes da intervenção

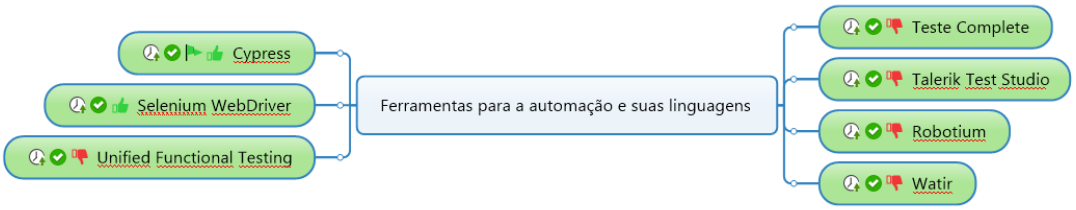


Fonte: Fábrica de Tecnologia Turing

O processo atual se inicia com a disponibilização da funcionalidade codificada pelo desenvolvedor para a equipe de teste, a partir disso o analista de testes responsável faz um estudo, estima e verifica se ela possui casos de teste, caso não o tenha, ele os escreve. Dessa forma se dá o início da etapa de teste funcional da interface a qual é feita uma revisão do teste por outro membro da equipe assim que é finalizado. Caso haja alguma inconsistência na interface o *tester* deverá realizar o registro das mesmas na ferramenta *Mantis* (ferramenta de relato de bugs adotada pela FTT) e a funcionalidade será enviada para o desenvolvedor realizar as correções. Caso não haja nenhum problema será efetuado um *merge* com a *branch master*. A próxima etapa a ser seguida será a execução do teste de regressão.

Após isso foi realizado um estudo acerca de ferramentas e linguagens para escrita de *scripts* automatizados para testes de interfaces. Um estudo de caso foi realizado então buscando por uma ferramenta que se adeque ao ambiente e as tecnologias usadas em uma fábrica de *software* normal. Segue abaixo um mapa exemplificando as principais ferramentas que foram estudadas como possíveis para serem utilizadas:

Imagem 2 - Ferramentas:



Os autores.

Como exibido na imagem foram levantadas 7 ferramentas para automação de testes de interface, sendo elas: Cypress, Selenium WebDriver, Unified Functional Testing, Teste Complete, Talerik Test Studio, Robotium e Watir. Após essa análise foi definida que será usada a ferramenta Cypress. Essa ferramenta foi citada na Technology Radar, uma revista que é publicada pela ThoughtWorks, uma empresa que busca atualizar a comunidade de software sobre as tendências que estão emergindo no mercado mundial através da publicação desta revista (THOUGHTWORKS, [2021?]). Segue abaixo uma tabela contendo um comparativo entre as ferramentas levantadas:

Tabela 1 - Comparativo entre as ferramentas

Nome da ferramenta	**Linguagens compatíveis**	**Vantagem**	**Desvantagem**	**Aquisição**
Watir	Ruby	-Facilidade do Ruby para a escrita dos scripts.	-Pequena comunidade.	Open source
Robotium	Java	-Capacidade de gerenciar as várias atividades do android automaticamente e sua execução rápida dos casos de teste.	-A incapacidade de lidar com aplicações que usam flash ou componentes web. -É preciso muito tempo e esforço para criar testes, pois é necessário trabalhar com o código-fonte do programa para automatizar os testes.	Open source

Telerik Test Studio	Angular, ASP-NET, HTML5, JavaScript, AJAX, WPF, Silverlight, MVC, Ruby, IOS, Android e PHP	-A capacidade de criar testes poderosos usando lógica condicional, chamando comandos de JavaScript e desktop.	-Não é uma solução baseada na Web.	Pago, com versão de avaliação.
TestComplete	JavaScript, Python, VBScript, JScript, Delphi Script, C++Script e C#Script	-Facilidade de uso, a customização do seu ambiente e as suas atualizações constantes.	-Existe uma forte dependência da interface gráfica. Se a interface gráfica mudar, os testes falham.	Pago, com versão de avaliação.
Unified Functional Testing	PowerBuilder, Desktop, ActiveX, SAP, Delphi, Net, Flex, Java, Oracle, Siebel, Mobile, PeopleSoft, Stingray e Visual Basic	-Suporte ao Record e Play Suporte para Oracle, SAP, WebForms entre outros Suporte a automação Data Driven Possui uma IDE amigável Permite automação de Smoke testing e Regressão.	-O preço elevado, sua capacidade de escrita de scripts apenas em VBScript e suporte apenas para Windows. - Necessidade de hardware robusto para executar a ferramenta.	Pago, com 30 dias de avaliação.
Selenium WebDriver	Java, Perl, Python, C #, Ruby, Groovy, Java Script e VB Script	-Suporte aos navegadores Internet Explorer, Chrome, Firefox, Opera, Safari, entre outros. -Suporte aos sistemas operacionais como Windows, Macintosh, Linux, Unix entre outros.	-Pode ter um comportamento irregular quando se testam páginas com ajax, onde é necessário um controle maior do tempo. Este comportamento é dependente do engine JavaScript do browser. Dependendo do caso, às vezes pode apresentar	Open source

			falsos erros em função da priorização das atividades, já que, tanto quem testa, quanto quem é testado, estão sendo executados sob o mesmo engine JavaScript.	
Cypress	JavaScript	-Seu recurso de dashboard. -Possui uma curva de aprendizado pequena e ser de fácil instalação e configuração. -Sua função de Debug dos testes. -Consegue interagir com o tráfego da rede, podendo ler e manipular os dados recebidos e enviados. -Segue à risca o conceito de "*All in one*". -O cypress executa no mesmo loop de execução do navegador, permitindo então acesso a comandos nativos.	-Não pode testar várias guias ou várias janelas do navegador ao mesmo tempo. -Os scripts de teste precisam ser escritos em JavaScript.	Open source

Fonte: Os autores

As ferramentas de teste automatizadas são capazes de executar testes, reportar e comparar resultados, com testes anteriores. Testes realizados com essas ferramentas podem ser executados repetidamente, a qualquer hora do dia (LIMA, 2014).

Abaixo serão descritas algumas das ferramentas utilizadas no mercado para a realização de automação de teste:

- Watir: É uma ferramenta muito leve que é usada para automação de testes em sistemas web. A linguagem utilizada por ele nos testes é o Ruby. Sua vantagem é a facilidade do Ruby para a escrita dos scripts e como desvantagem destaca-se uma pequena comunidade. É difícil

encontrar desenvolvedores e documentação relacionados a essa ferramenta, dessa forma todo o conteúdo limita-se apenas ao próprio site da ferramenta (QUALIDADEBR, 2011).

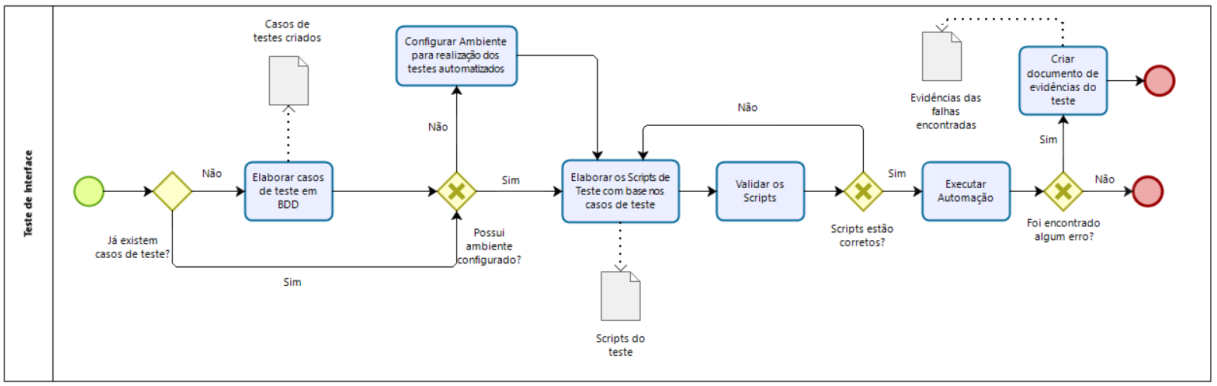
- Robotium: É comumente usado em testes de aplicações Android, sendo compatível com aplicações nativas e híbridas. Dentre suas vantagens se destacam sua capacidade de gerenciar as várias atividades do android automaticamente e sua execução rápida dos casos de teste. Mas apresentam desvantagens como: a incapacidade de lidar com aplicações que usam flash ou componentes web (COSTA, 2016).
- Telerik Test Studio: Uma ferramenta capaz de testar aplicações em várias tecnologias, sendo elas: Angular, ASP-NET, HTML5, JavaScript, AJAX, WPF, Silverlight, MVC, Ruby, IOS, Android e PHP. Pode ser usado para efetuar diversos testes, como performance, regressão, exploratórios e Mobile. No entanto, têm pouco material sobre a execução de seus testes e a ferramenta tem uma comunidade bem pequena, o que torna mais difícil o compartilhamento de informações acerca dos testes e de possíveis erros encontrados em sua execução (TELERIK, [2020?]).
- Teste Complete: Essa ferramenta pode utilizar testes em sistemas Desktop, web e android. As linguagens utilizadas nos testes dessa ferramenta são: JavaScript, Python, VBScript, JScript, Delphi Script, C++Script e C#Script. Suas vantagens principais são: a facilidade de uso, a customização do seu ambiente e as suas atualizações constantes, por ser um produto comercial. E como desvantagem o Teste Complete não possui suporte para Mac (OLIVEIRA, 2019).
- Unified Functional Testing: Ferramenta para automação de testes funcionais e de regressão de aplicações web em PowerBuilder, Desktop, ActiveX, SAP, Delphi, Net, Flex, Java, Oracle, Siebel, Mobile, PeopleSoft, Stingray e Visual Basic. Dentre suas vantagens se destacam a possibilidade de conversão de relatórios de testes manuais em casos de teste de automação. E suas desvantagens se destacam o preço elevado, sua capacidade de escrita de scripts apenas em VBScript e suporte apenas para Windows (MICROFOCUS, [2021?]).
- Selenium WebDriver: O Selenium é a ferramenta mais popularizada no ramo da automação de testes, ela suporta as seguintes linguagens: Java, C#, Python, Ruby, Perl, PHP e JavaScript. Essa ferramenta tem como vantagens o constante feedback para os usuários, seu suporte para metodologias ágeis, a documentação disciplinada para a elaboração dos casos de teste e um relatório personalizado para os defeitos. No entanto, para sua utilização é necessário que seja feita a instalação remota no servidor, além de ser limitada para testes mais complexos e também não funcionar corretamente quando são testadas páginas web que usam AJAX (SELENIUM, [entre 2013 e 2021]).
- Cypress: O Cypress é um framework utilizado para automação de testes end to end. Atualmente, utiliza JavaScript como linguagem de programação. Sua principal vantagem é a curva de aprendizado pequena, sua fácil instalação e configuração. Seu principal defeito era a sua utilização apenas em um navegador, no entanto com sua última atualização ele passou a poder ser executado em qualquer navegador (CYPRESS, [2021?])

A adoção do Cypress para a aplicação do processo para o experimento foi motivada pela sua curva de aprendizado simplificada, a fácil configuração, pela utilização da linguagem

JavaScript para a escrita dos scripts e pela sua completude com relação a testes end to end. E o mais importante é a sua capacidade de ser aplicado a testes de qualquer tipo de estrutura front-end (ATECH, 2021).

Após a definição da ferramenta foi modelado o processo genérico que será inserido na FTT para que ele seja validado, evidenciando se o mesmo será útil ou não para qualquer fábrica de software. Segue abaixo a modelagem do processo mínimo de automação de testes de interface:

Imagem 3 - Processo Novo:



Os autores.

O processo será inserido como um subprocesso dentro da etapa que hoje é denominada como teste funcional do processo da equipe. Com isso ele foi definido com foco em agilidade e buscando ser o mais simples e direto possível para que não haja uma curva grande de aprendizado. Os usuários precisarão apenas de conhecimento prévio, de comandos da ferramenta de automação e sobre *tags HTML*. Para auxiliar na adaptação serão ofertados tutoriais nessa nova fase do projeto para que o público alvo tenha uma interação com o *cypress* para colocar em prática alguns dos conceitos de automação, a caráter de aprendizado nesse primeiro momento.

Analisando mais a fundo a estrutura do processo, foi definido inicialmente que serão feitos os casos de testes com *Behavior Driven Development* (BDD). O BDD é uma técnica para a criação de softwares ágeis, sendo realizado a escrita de cenários que visam aumentar a eficácia dos testes e do desenvolvimento do produto (FABIO, 2021). Após a escrita dos cenários de teste deverá ser realizada a configuração do ambiente de testes e da ferramenta de automação. Depois disso será realizada a automação, onde caso sejam encontrados erros será gerado um documento contendo as não conformidades e assim finalizado o processo. O processo foi modelado para ser genérico, mas completo, para que fábricas possam iniciar nesse mundo de automação, validando suas interfaces com mais qualidade e com menor interferência humana nessa ação. A FTT será utilizada como ambiente de estudos para que esse processo seja validado, levantando dados de seu sucesso ou insucesso para que se tenha um veredito sobre a possível expansão de sua utilização em outros ambientes.

Em um segundo momento foi realizado um questionário no âmbito da FTT para colher dados antes da intervenção para que os mesmos sejam analisados em relação à quando já

estiver instaurado o processo na equipe de testes. O questionário foi realizado no dia 26 de setembro de 2021, por meio de um formulário do *google forms* que foi aplicado e explicado para a equipe pela ferramenta de comunicação oficial da Fábrica, o Discord. Ao todo foram entrevistados 6 analistas de testes. Esse questionário foi formulado com perguntas chave para formular o estudo, bem como demonstrar um comparativo antes e pós intervenção. Dessas perguntas algumas foram norteadas por um estudo semelhante aplicado em uma empresa de tecnologia e usado em um trabalho de conclusão de curso (TEIXEIRA, 2015).

Do total dos entrevistados, 50% afirmaram possuir conhecimento moderado acerca de testes de software, 33,3% de conhecimento avançado e 17,7% de conhecimento básico.

A segunda pergunta levantou o questionamento sobre o tempo de experiência dos entrevistados com relação a testes de software, e foram colhidos os seguintes resultados: 66,7% dos entrevistados responderam possuir experiência entre um mês a um ano, com 16,7% menos de um mês de experiência e com essa mesma porcentagem aqueles que possuem experiência de um ano a 3 anos.

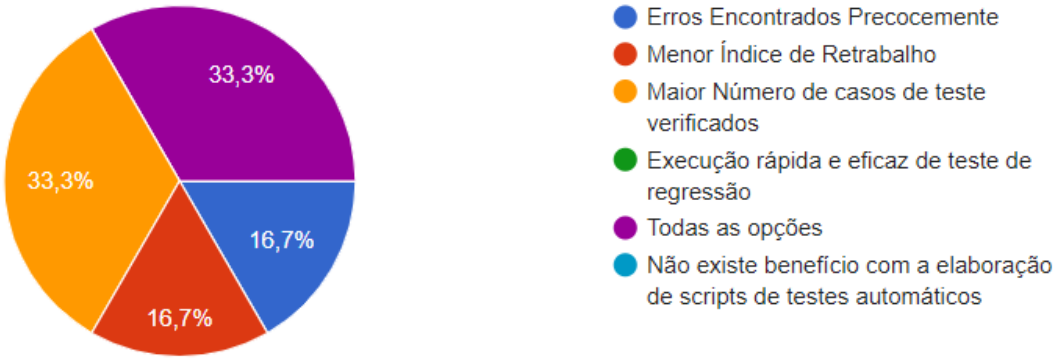
Com relação a relevância da aplicação de testes no desenvolvimento de software as respostas foram unânimes, onde os entrevistados colocaram como de extrema importância na escala definida a realização dos testes.

Quando perguntados sobre qual o processo de teste utilizado na empresa que trabalham atualmente, 83,3% responderam que utilizam testes manuais e 16,7% responderam que utilizam testes automatizados.

Em relação ao cenário que mais se encaixam os testes manuais para a validação e verificação de sistemas, 50% responderam que para sistemas dinâmicos, que necessitam de avaliação crítica e reflexiva e empatados com a mesma porcentagem de 16,7% responderam que se enquadram mais para sistemas com previsão de alterações constantes, Todas as opções e que não acreditam que os testes manuais sejam os mais adequados em nenhum cenário.

Acerca do maior benefício com a elaboração e execução de scripts de testes automáticos, 33,3% por cento acreditam que o maior benefício é que serão verificados um maior número de casos de teste, essa mesma porcentagem acredita em todas as opções que foram apresentadas também no estudo, 16,7% dos entrevistados disseram não haver benefício com a elaboração de scripts automatizados e essa mesma porcentagem acredita que um menor índice de retrabalho será obtido.

Imagem 4 – Maior benefício com a elaboração e execução de scripts automáticos



Os autores

Sobre o maior desafio na adesão do processo de testes automatizados, 50% responderam que manter a mesma produtividade, embora possua mais atividades, para elaborar scripts, executar e validar, 33,3% acreditam que seja a equipe de testes se enquadrar no novo processo e 16,7% em como criar scripts de testes.

Imagem 5 – Maior desafio na adesão do processo de testes automatizados

Os autores

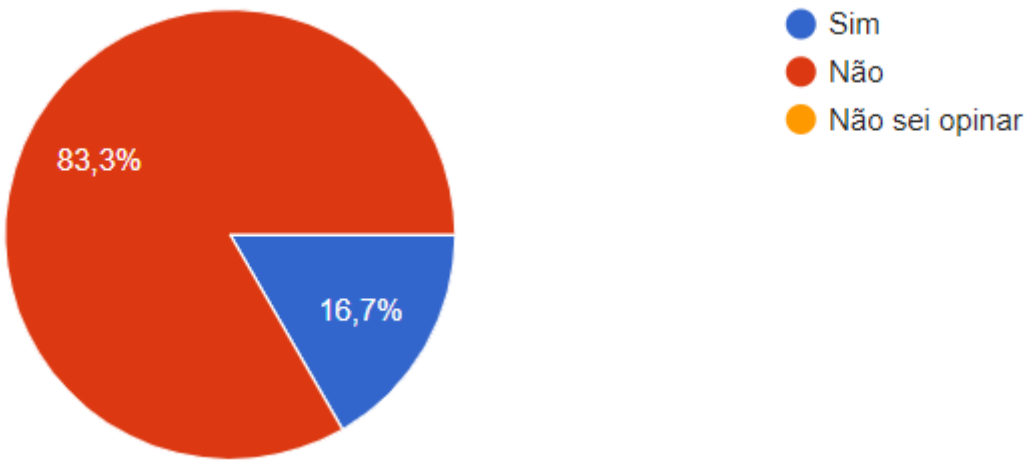
Foi realizada também uma pergunta quanto ao conhecimento dos entrevistados acerca de ferramentas de testes automatizados que necessitem de scripts, 83,3% responderam que possuem conhecimento sobre Selenium IDE/Webdriver e 16,7% que utilizam tanto Selenium IDE/Webdriver e Cypress.

Quando questionados no estudo se a execução dos testes de forma automática reduziria os custos a longo prazo, pois após os scripts serem criados, se tornaria mais rápida a execução dos testes, 83,3% responderam que concordam plenamente e 16,7% concordam parcialmente.

Na penúltima pergunta foram questionados se concordavam que com a realização dos testes de forma automática, os sistemas testados terão um maior nível de qualidade, o resultado obtido foi 100% concordando que sim.

E por fim, a última pergunta era se seria possível substituir totalmente os testes manuais por testes automatizados, 83,3% afirmaram que não seria possível e 16,7% acreditam nessa possibilidade.

Imagem 6 – Seria possível substituir totalmente os testes manuais por automatizados



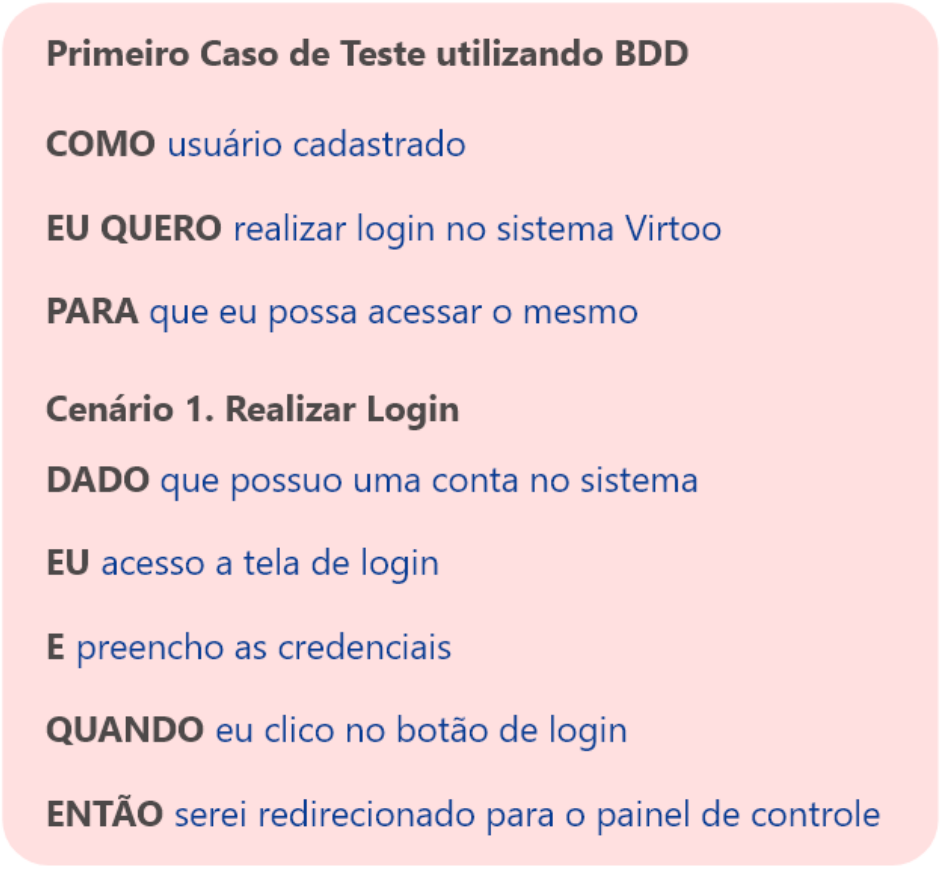
Os autores

Dessa forma foram encerrados os questionários com esses dados que podem ser vistos com mais detalhes na seção de anexos deste trabalho. Conclui-se com esse levantamento inicial que a equipe é um pouco inexperiente, sendo sua maioria inferior a um ano com testes automatizados com base em suas respostas. Sendo necessário um processo de capacitação inicial antes de iniciar a intervenção na Fábrica que será usada como ambiente de estudo de caso. No entanto, com isso poderemos colher dados mais precisos com relação a intervenção, já que o time não possui esse contato com a tecnologia e nem o processo de testes automatizados, fazendo com que esses tenham uma imersão completa nesse meio no decorrer do experimento.

3.1 Aplicação do Processo

Foi realizada uma explicação acerca do processo para a equipe de testes no dia 13 de outubro na ferramenta oficial de comunicação da fábrica, o Discord. Os membros tiraram suas dúvidas com relação ao que seria modificado no que já existe hoje, e foi explicado a eles que seria apenas inserido um subprocesso onde eles fariam o teste funcional de interface deles, modificando apenas esse ponto para ser totalmente automatizado. Ainda foi demonstrado a eles um vídeo gravado para esse fim, sobre como instalar a ferramenta Cypress a ferramenta escolhida para ser usada no estudo, nesse vídeo foram apresentados também a instalação de pacotes necessários para a ferramenta ser usada e também alguns comandos básicos. A equipe escolheu então em um outro momento duas funcionalidades para realizar o experimento, as funcionalidades escolhidas foram: Realizar Login e Cadastrar Pessoa.

Imagem 7 – Cenário com BDD exemplo



Os autores

No dia 18 de outubro houve um treinamento momentâneo com a equipe, para explicar o processo, a ferramenta Cypress e sanar dúvidas. A equipe foi conduzida no processo de escrita dos scripts do teste. Mais alguns dias foram dedicados a esta tarefa até a sua finalização e validação pelo líder da equipe. Após isso, foi feita a execução da automação do teste de interface onde conseguiram testar ambas as funcionalidades. Para colher os resultados do experimento foi aplicado novamente o questionário, com alguns ajustes para ser mais específico, incluindo perguntas sobre a satisfação da aplicação do processo na busca pelos resultados após sua aplicação. Com isso 5 membros da equipe de testes foram submetidos a responder esse novo questionário no dia 28 de outubro, o que gerou os seguintes resultados:

3.2 Relatos e Resultados Coletados

Do total de entrevistados com relação ao questionamento sobre o conhecimento atual sobre testes de software, 80% responderam possuir conhecimento moderado e 20% acreditam estar com nível avançado. Em uma análise com essa mesma pergunta feita anteriormente houve um aumento no nível do conhecimento no qual os membros acreditam estar, esse aumento pode estar baseado na apresentação de métodos de teste que antes não eram conhecidos por eles, no caso os testes automatizados de interfaces.

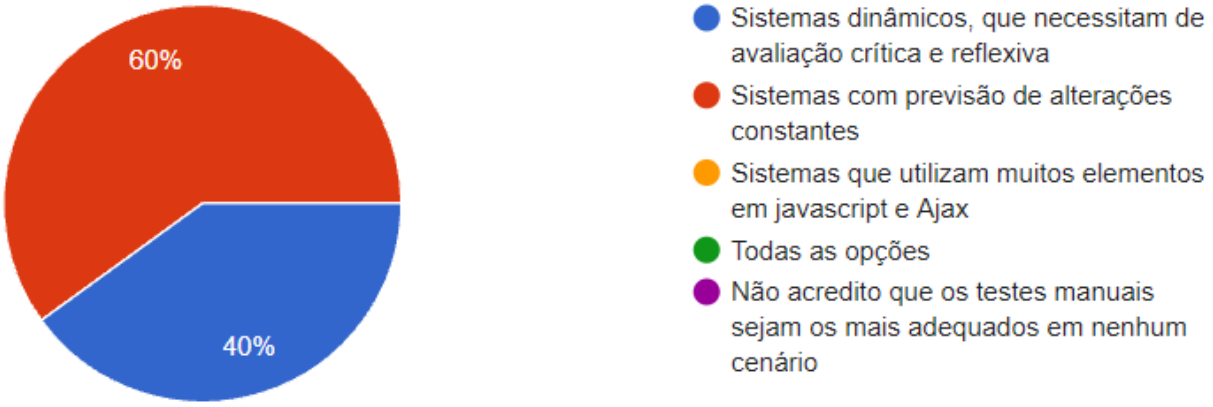
Com relação ao tempo de experiência com testes, 80% dos membros responderam ter de um mês a um ano de experiência e 20% possuem de um a três anos.

A próxima pergunta foi sobre a relevância da aplicação de testes no cenário de desenvolvimento de software, nessa pergunta os membros atribuíram pontos de um a cinco para a relevância dos testes e o resultado obtido foi de 80% marcando cinco e 20% marcando quatro.

Quando questionados sobre qual o processo de teste adotado pela empresa que trabalham atualmente 80% responderam que manual e 20% responderam híbrido ou misto.

Ao serem questionados sobre em qual cenário os testes manuais seriam os mais adequados para a validação e verificação de um sistema 60% responderam que para sistemas com previsão de alterações constantes e 40% acreditam que para sistemas dinâmicos, que necessitam de avaliação crítica e reflexiva. Nesse segundo questionário os membros se mostraram mais centralizados em suas respostas com relação ao primeiro com o que diz respeito a essa pergunta, demonstrando estarem mais em sintonia com seu aprendizado no decorrer da aplicação do estudo.

Imagem 8 – Cenário mais adequado para testes manuais



Os autores

Quando a pergunta foi sobre o maior benefício com a elaboração e execução dos scripts de testes automáticos 20% responderam que seria pelo maior número de casos de testes verificados, outros 20% que seria pelo menor índice de retrabalho, outra parcela de 20% dos entrevistados acredita que o maior benefício é a descoberta de erros precocemente e os 40% que restaram responderam todas as opções. Traçando um paralelo entre essa pergunta no primeiro questionário e agora no segundo os membros ainda se mantiveram divididos em suas respostas, no entanto antes do estudo eles apenas responderam essa pergunta com noções teóricas e após a aplicação o feedback vem da prática da escrita desses scripts.

Imagem 9 – Maior benefício com a elaboração e execução dos scripts de testes automáticos

Os autores

Sobre o maior desafio na adesão do processo de testes automatizados 80% dos entrevistados responderam que manter a produtividade, embora possua mais atividades, para elaborar scripts, executar e validar e 20% acreditam que manter os scripts validados e funcionais seja o maior desafio dessa adesão.

Os membros foram questionados sobre quais ferramentas eles possuíam maior conhecimento ou qual eles achavam que tinha maior aplicação no mercado para a escrita de scripts de testes automatizados. 80% dos entrevistados responderam Selenium IDE/ Webdriver e 20% têm mais conhecimento da ferramenta Cypress.

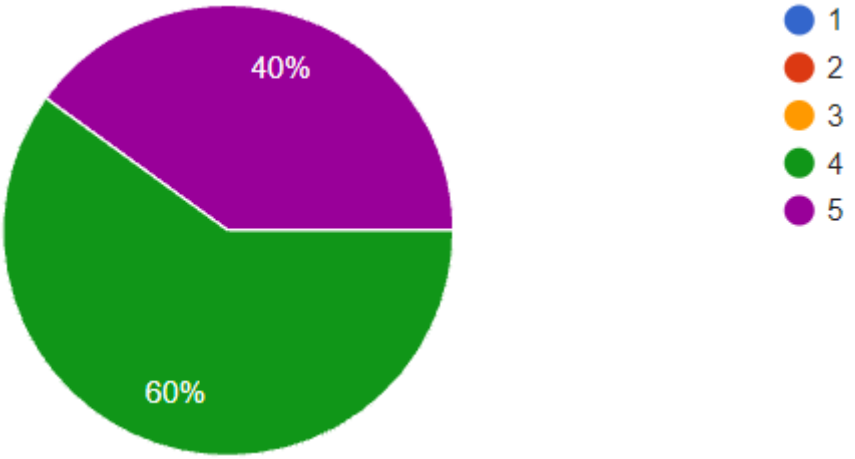
Dos analistas 100% concorda totalmente com a afirmação que a execução dos testes de forma automática reduz os custos a longo prazo, pois após os scripts serem criados, se torna mais rápido de serem executados.

Em unanimidade os analistas concordam que os sistemas testados de forma automática terão um maior nível de qualidade. Isso ocorreu pelo fato do aprofundamento no assunto dos testes automatizados proporcionado pelo estudo, que fez com que os mesmo buscassem por cursos e aprendizados complementares e autores que os fizeram chegar a essa conclusão.

Quando perguntados se era possível substituir totalmente os testes manuais por testes automatizados, os membros responderam em unanimidade que não.

Foi inserida uma pergunta sobre como a aplicação do experimento contribuiu para a melhoria do conhecimento dos membros com relação a testes de software, utilizando uma escala de pontos de um a cinco, nessa pergunta sessenta por cento responderam o nível quatro de contribuição e quarenta por cento o nível cinco de contribuição.

Imagem 10 – Contribuição do experimento para a melhoria do conhecimento em testes



Os autores

A última pergunta levantou o questionamento sobre a melhoria da qualidade dos testes após a aplicação do experimento, onde os membros deveriam atribuir uma escala de um a cinco para o nível de melhoria da qualidade, sessenta por cento atribuíram valor quatro e quarenta por cento colocaram o valor cinco.

De acordo com os dados coletados houve uma melhoria com relação a percepção dos analistas de testes que se comprometeram a participar do experimento no que diz respeito ao entendimento do que seriam os testes automatizados de interfaces. Também foram boas as respostas para o experimento, as melhorias que os membros avaliaram nas duas últimas perguntas desse último questionário onde falaram que houve uma contribuição para o seu conhecimento e para a qualidade dos testes.

O estudo apontou um processo para a inserção dos testes automatizados em ambientes que não os usam ainda, e se mostrou simples para adoção até mesmo daquelas que possuem processos iniciais que não tem maturidade. O processo busca trazer os conceitos básicos de um teste de interface simples e também conceitos como o de casos de teste para que a criação dos scripts e sua execução seja feita. No entanto, o estudo não se aprofundou tanto em relação aos casos de testes, deixando apenas algumas dicas de como os analistas poderiam manter os mesmos atualizados, no caso utilizando o BDD.

4. CONCLUSÃO E CONSIDERAÇÕES FINAIS

Este trabalho foi elaborado para colaborar com a resolução de um problema enfrentado em diversas empresas que atuam no desenvolvimento de software, especificamente na área de testes, um ponto crítico que delimita a qualidade dos sistemas que são produzidos.

O processo elaborado foi possível de ser executado no ambiente de experimento e se provou favorável para ser usado em outros ambientes de fábrica, por sua simplicidade. Dessa forma os membros tiveram mais dificuldade em aprender a utilizar a ferramenta de criação dos scripts e execução dos testes automatizados do que seguir o próprio processo. Esse relato foi o mais comum levantado por eles no momento em que estiveram em contato com o experimento. Foi verificado também que o processo ajudou na melhoria dos testes na visão de mais casos de testes verificados dado que os membros se preocuparam mais em validar mais cenários com os seus scripts.

A inexperiência da equipe foi um dos agravantes do estudo além da alta rotatividade da mesma, dado que esta passou por uma reformulação no meio da aplicação do processo, algo comum para o ambiente. Aplicar um processo assim em um ambiente em constante mudança se torna um desafio e se faz necessário mais tempo para que seja incorporado o conhecimento para transmitir aos novos membros que se renovam com o passar dos meses.

No entanto, isso não impediu o andamento do experimento. Nesse contexto foram coletados dados da aplicação com os membros da equipe de teste atual, que se submeteram a alguns questionários além da execução do processo propriamente dito. A equipe de analistas possui uma média de experiência inferior a 3 anos na área o que gerou respostas favoráveis no quesito de geração de conhecimento para os membros acerca dos testes automatizados, que era um tema ainda obscuro para uma parcela significativa dos integrantes da equipe.

Portanto, analisando os dados coletados a equipe conseguiu melhorar a qualidade de suas entregas e também avançar em relação ao conhecimento em testes de software, aprendendo mais sobre uma nova vertente, que são os testes automatizados de interfaces, e a equipe se propôs a manter a execução do processo para manter esse conhecimento. Ao analisar a forma como foram feitos os casos de testes com o BDD foi levantada uma possível hipótese de melhoria futura para esse processo, onde os membros possam utilizar da ferramenta que estão estudando, o Cypress, para elaborar os cenários para testar nela mesmo.

Um trabalho que aborda mais profundamente essa questão de cenários em BDD é o do FERNANDES; FONSECA, (2019) com uma ferramenta diferente para sua aplicação sendo um trabalho que foi realizado na empresa *Softplan,* eles utilizaram as ferramentas Junit e o Cucumber em seu experimento. Outro trabalho interessante que segue a aplicação de processos de automação, foi o de IZABEL (2014) desenvolvido na Universidade do Rio de Janeiro que buscava formular um processo aplicando-o com a ferramenta Selenium Webdriver, para testes automatizados de interfaces, delimitando os momentos que os testes automatizados seriam melhores de serem utilizados, o que segundo o autor em testes

repetitivos que os humanos poderiam cometer erros por sua exaustão interferir em seu julgamento.

Esses trabalhos se diferenciam do que foi apresentado nesse experimento por levarem conceitos mais complexos e mais detalhados, o que não foi o foco dado que o objetivo maior do trabalho era apresentar um processo mais simplificado para nortear um início de uma aplicação de testes automatizados.

REFERÊNCIAS

AGUIAR, Lucas Tagliani. **Por que alguns times não tem testes automatizados pra desenvolver software?.** [S. l.], 12 ago. 2017. Disponível em: https://medium.com/@lucastagliani/por-qu%C3%AA-algumas-empresas-de-tecnologia-n%C3%A3o-tem-testes-automatizados-pra-desenvolver-software-b5e961784a8c. Acesso em: 10 dez. 2021.

ALMEIDA, P. H. P. DE; BATISTA, R. A. **PROCESSO DE GESTÃO DO CONHECIMENTO PARA AMBIENTE ACADÊMICO DE DESENVOLVIMENTO DE SOFTWARE**. 2020.

ALMEIDA, Samyra L. F.; RODRIGUES, Nadja N.; LIRA, Heremita B.; LIMA, Carlos D. Q.;

AMORIM, Bianca Pinto de; SANTOS, Daniele Ribeiro; MUNIZ, Diego Antunes; COSTA, Janaina Pedrina de Moraes; ANTUNES, Leandro Alaor; AMARAL, Eliane Cristina; SABINO, Eliney; ABE, Narumi. **GESTÃO DE QUALIDADE NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE**. [*S. l.*], 2017. Disponível em: https://portal.unisepe.com.br/unifia/wp-content/uploads/sites/10001/2018/06/060_gestao.pdf. Acesso em: 13 maio 2021.

ATECH**. CYPRESS: o novo conceito em testes automatizados**. [*S. l.*], 8 fev. 2021. Disponível em: https://atech.com.br/cypress-o-novo-conceito-em-testes-automatizados/. Acesso em: 13 nov. 2021.

AUTOMTECH. **Desafios da automação de testes.** [*S. l.*], 8 out. 2018. Disponível em: https://www.automtech.com.br/2018/10/08/desafios-da-automacao-de-testes/. Acesso em: 10 dez. 2021.

BALTHAZAR , Glauber da Rocha. **Visão Geral da Qualidade de Software**. [S. l.], 2015. Disponível em: http://re.granbery.edu.br/artigos/MjUw.pdf. Acesso em: 24 set. 2021.

BENITTI, Fabiane Barreto Vavassori; ALBANO, Edson Lucas. **Teste de Software: o que e como é ensinado?**. [S. l.], 2012. Disponível em: http://www2.sbc.org.br/csbc2012/anais_csbc/eventos/wei/artigos/Teste%20de%20Software%20o%20que%20e%20como%20e%20ensinado.pdf. Acesso em: 13 maio 2021.

BERNARDO, Paulo Cheque; KON, Fabio. **A Importância dos Testes Automatizados**. [S. l.], 2008. Disponível em: https://antigo.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf. Acesso em: 13 maio 2021.

COLLINS, E. F.; KON, F. **A Importância dos Testes Automatizados**. p. 1–100, 2013.

COSTA, Leonardo. **Ferramentas e técnicas de automação de testes**. [S. l.], 16 jul. 2016. Disponível em: https://medium.com/@leonardoteck/ferramentas-e-t%C3%A9cnicas-de-automa%C3%A7%C3%A3o-de-testes-16abc579943a. Acesso em: 15 maio 2021.

CYPRESS. **Why Cypress?**. [S.I] [2021?]. Disponível em: <https://docs.cypress.io/guides/overview/why-cypress#API> Acesso em: 12 jun. 2018.

DUNNING, Ana Karina de M. P.; FREITAS, Rayssa Mª S. de. **Aplicação e análise de processo de desenvolvimento de software: um estudo de caso no GPES-IFPB**. [*S. l.*], 2017. Disponível em: https://periodicos.ifpb.edu.br/index.php/principia/article/download/1798/996. Acesso em: 13 maio 2021.

FÁBIO, Thiago. **BDD como metodologia ágil.** [S. l.], 2021. Disponível em: https://www.dtidigital.com.br/blog/bdd-como-metodologia-agil/. Acesso em: 24 set. 2021.

FERNANDES, MATHEUS; FONSECA, SAMUEL TOMKELSKI. **AUTOMAÇÃO DE TESTES DE SOFTWARE: ESTUDO DE CASO DA EMPRESA SOFTPLAN**. Florianópolis, 2019. Disponível em: https://repositorio.animaeducacao.com.br/bitstream/ANIMA/10927/1/AUTOMA%C3%87%C3%83O%20DE%20TESTES%20DE%20SOFTWARE-ESTUDO%20DE%20CASO%20DA%20EMPRESA%20SOFTPLAN-TCC_MATHEUS%20FERNANDES-SAMUEL%20TOMKELSKI%20FONSECA.pdf. Acesso em: 13 nov. 2021.

FIGUEIRA, ANDERSON MARQUES DA SILVA. **ANÁLISE DAS TÉCNICAS DE LEVANTAMENTO DE REQUISITOS PARA DESENVOLVIMENTO DE SOFTWARE NAS EMPRESAS DE VITÓRIA DA CONQUISTA – BA.** [S. l.], 2012. Disponível em: http://www2.uesb.br/computacao/wp-content/uploads/2014/09/AN%C3%81LISE-DAS-T%C3%89CNICAS-DE-LEVANTAMENTO-DE-REQUISITOS-PARA-DESENVOLVIMENTO-DE-SOFTWARE-NAS-EMPRESAS-DE-VIT%C3%93RIA-DA-CONQUISTA-%E2%80%93-BA.pdf. Acesso em: 24 set. 2021.

IZABEL, L. R. P. **Testes Automatizados No Processo De Desenvolvimento De Softwares**. 2014.

IZABEL, Leonardo Roxo Pessanha. **TESTES AUTOMATIZADOS NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARES.** Rio de Janeiro, 2014. Disponível em: http://repositorio.poli.ufrj.br/monografias/monopoli10012548.pdf. Acesso em: 13 nov. 2021.

JANNUZZI , Celeste Sirotheau Corrêa; FALSARELLA , Orandi Mina; SUGAHARA , Cibele Roberta. **Gestão do conhecimento: um estudo de modelos e sua relação com a inovação nas organizações**. [*S. l.*], 2016. Disponível em: https://www.scielo.br/pdf/pci/v21n1/1413-9936-pci-21-01-00097.pdf. Acesso em: 13 maio 2021.

LIMA, Antonio Mendes da Silva. **AUTOMAÇÃO DE TESTES FUNCIONAIS EM AMBIENTES WEB: UM ESTUDO DE CASO NO LABORATÓRIO DE SISTEMA E BANCOS DE DADOS DA UNIVERSIDADE FEDERAL DO CEARÁ**. [*S. l.*], 2014. Disponível em: http://www.repositorio.ufc.br/bitstream/riufc/25003/1/2014_tcc_agdaslima.pdf. Acesso em: 13 maio 2021.

MICROFOCUS. **UFT One**. [S.I] [2021?]. Disponível em: <https://www.microfocus.com/pt-br/products/uft-one/overview> Acesso em: 12 jun. 2018.

OLIVEIRA, Ricardo. **Teste automatizado com o Test Complete**. [*S. l.*], 14 mar. 2019. Disponível em: https://blog.alterdata.com.br/teste-automatizado-com-o-test-complete/. Acesso em: 13 maio 2021.

OLIVEIRA, Thaís. **Automação de testes: o que é, quando e por que automatizar.** [S. l.], 13 abr. 2018. Disponível em: https://medium.com/venturus/quais-as-raz%C3%B5es-para-automa%C3%A7%C3%A3o-de-testes-c177cbd9397. Acesso em: 10 dez. 2021.

ONTIVERO, Pablo Exequiel Leyes. **Comparação das metodologia Cascata vs metodologias Ágil no mercado atual.** [S. l.], 2020. Disponível em: http://www.facom.ufu.br/~backes/publi_peq/tcc_cascata_agil.pdf. Acesso em: 10 dez. 2021.

QUALIDADEBR. **Conheça o Watir**. [S.I] [2011]. Disponível em: <https://qualidadebr.wordpress.com/2011/03/05/conheca-o-watir/> Acesso em: 12 maio 2021.

ROQUETTE, JOSÉ HENRIQUE. **UMA ABORDAGEM UTILIZANDO BEHAVIOR DRIVEN DEVELOPMENT PARA GERAÇÃO DE CASOS DE TESTE:: UM ESTUDO DE CASO NA ÁREA AUTOMOTIVA.** [S. l.], 2018. Disponível em: http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/10417/1/PG_COCIC_2018_2_06.pdf. Acesso em: 24 set. 2021.

SELENIUM. **The Selenium Browser Automation Project**. [S.I] [entre 2013 e 2021]. Disponível em: <https://www.selenium.dev/documentation/en/> Acesso em: 12 jun. 2018.

SILVA, P. C. B. DA; ALVES, T. S.; BRUNO, E. A. **AUTOMAÇÃO DE TESTES FUNCIONAIS** Testes funcionais automatizados de software. p. 95–112, 2011.

SOUZA, Karla Pires de; GASPAROTTO, Angelita Moutin Segoria. **A IMPORTÂNCIA DA ATIVIDADE DE TESTE NO DESENVOLVIMENTO DE SOFTWARE**. [*S. l.*], 2013. Disponível em: http://www.abepro.org.br/biblioteca/enegep2013_TN_STO_177_007_23030.pdf. Acesso em: 13 maio 2021.

TEIXEIRA, J. H. M.; CAIXÊTA, K. K. M. **DEFINIÇÃO DE UM ROTEIRO DE TESTE DE INSPEÇÃO COM MÉTODOS DE VERIFICAÇÃO PARA A FÁBRICA DE TECNOLOGIAS TURING (FTT)**. 2020.

TEIXEIRA, Lucas. **TCC - UM ESTUDO DE CASO SOBRE APLICACAO E BENEFICIOS DA AUTOMACAO DE TESTES FUNCIONAIS EM UMA FABRICA DE SOFTWARE**. FORTALEZA, 2015. Disponível em: https://www.academia.edu/36694834/TCC_-_UM_ESTUDO_DE_CASO_SOBRE_APLICACAO_E_BENEFICIOS_DA_AUTOMACAO_DE_TESTES_FUNCIONAIS_EM_UMA_FABRICA_DE_SOFTWARE. Acesso em: 13 nov. 2021.

TELERIK. **Telerik Test Studio Dev Edition** . [S.I] [2020?]. Disponível em: <https://docs.telerik.com/devtools/teststudiodev/introduction> Acesso em: 12 jun. 2018.

THOUGHTWORKS. **Technology Radar**. [S.I] [2021?]. Disponível em: <https://www.thoughtworks.com/pt/radar/faq> Acesso em: 12 jun. 2018.

VARGAS, Cristiane Machado de; DUARTE, Ana Marcia Debiasi. **APLICAÇÃO DE BOAS PRÁTICAS DE QUALIDADE DE SOFTWARE NO DESENVOLVIMENTO DE UM PROTÓTIPO DE SISTEMA DE REGISTRO ELETRÔNICO EM SÁUDE ASSISTENCIAL**. [S. l.], 2013. Disponível em: http://www.uniedu.sed.sc.gov.br/wp-content/uploads/2013/10/Cristina-Machado-de-Vargas.pdf. Acesso em: 13 maio 2021.

WU, D.; SHUQIN, L.; JINGJING, L. Automation testing process modeling method of SOA-based isomerous software. **2009 International Conference on Industrial Mechatronics and Automation, ICIMA 2009**, p. 129–132, 2009.

ANEXO (S)

Anexo A – https://forms.gle/BRMWNQfn2W43NwnZA