UNIVERSIDADE EVANGÉLICA DE ANÁPOLIS – UniEVANGÉLICA

BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

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

Welington Tierry Vitor Silva

Jonas Santos Corrêa

Anápolis

2021-01

Welington Tierry Vitor Silva

Jonas Santos Corrêa

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

Trabalho de Conclusão de Curso I apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso I do curso de Bacharelado em Engenharia de Computação da Universidade Evangélica de Anápolis – UniEVANGÉLICA.

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

Anápolis

2021-01

Welington Tierry Vitor Silva

Jonas Santos Corrêa

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

Trabalho de Conclusão de Curso I apresentado como requisito parcial para a obtenção de grau do curso de Bacharelado em Engenharia de Computação da Universidade Evangélica de Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em 9 de Junho de 2021, composta por:

_____

Prof. Walquíria Fernandes Marins
Orientador

_____

Prof. Millys Fabrielle Caravalhaes

Resumo

O desenvolvimento de software é dividido em etapas, sendo elas: definir, desenvolver, testar e manter um *software*. Especificamente a etapa de teste é responsável por verificar e validar se o que foi proposto no início do desenvolvimento foi atingido no resultado final, ou seja, delimitar se o que foi construído possui qualidade. Por diversos motivos, técnicas disponíveis para que essa etapa seja efetiva para o produto final nem sempre são incorporadas ao processo de desenvolvimento. Um exemplo destas técnicas é a automação dos testes que foi pensada para otimizar tempo e elevar o potencial das validações sem perder qualidade das mesmas. Neste contexto, o objetivo deste trabalho será criar um processo de automação de teste funcional de software que seja objetivo, eficiente e aplicável para fábricas de software de qualquer porte. Espera-se fornecer dados quantitativos da eficiência, da incorporação e manutenção de um processo desse segmento em fábricas de *software*. Até o presente momento foi modelado um processo inicial, no qual a ferramenta cypress foi definida para ser utilizada no ambiente de experimento.

[Lista de Ilustrações]

Resumo **4**

1. Problema **7**

2. Objetivos **9**
2.1. Objetivo Geral 9
2.2. Objetivos Específicos 9

3. Justificativa **10**

4. Fundamentação Teórica **11**
4.1. Processo de Desenvolvimento de Software 11
4.2. Qualidade de Software 11
4.3. Plano de Teste 12
4.4. Casos de Teste 12
4.5. Tipos de Teste 13
4.5. Automação de Testes 14

5. Metodologia **15**

6. Cronograma **16**

7. Resultados alcançados **18**

8. Resultados esperados **21**

9. Referências Bibliográficas **22**

1. Problema

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. 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 aumentar a eficiência desse processo da automação de testes, esse trabalho terá como foco formalizar as etapas da automação e apresentar um processo minimalista que pode ser implementado em fábricas de *software*, como objeto de testes e validação da proposta a mesma será aplicada na Fábrica de Tecnologias Turing (FTT). Um ambiente acadêmico de desenvolvimento de

software que ao decorrer dos anos deixou de aplicar os testes funcionais 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).

Diante deste cenário, como poderia ser um processo para criação de testes funcionais automatizados? Como tornar o esforço de automação viável a longo prazo? Com a implementação desse mínimo processo viável quais serão os benefícios e as dificuldades encontradas dentro do ambiente de uma fábrica de software?

2. Objetivos

2.1. Objetivo Geral

Validar um processo mínimo viável para a execução de testes automatizados em fábricas de software para facilitar a criação, execução e manutenção dos mesmos.

2.2. Objetivos Específicos

- Pesquisar abordagens relacionadas;
- Propor um processo de testes automatizados de interface genérico que seja utilizado por qualquer fábrica de software
- Analisar o ambiente atual da fábrica de software que foi escolhida como experimento antes da intervenção e colher dados;
- Analisar o ambiente após a intervenção do novo processo;
- Validar se com a implantação de testes automatizados a qualidade nas entregas será melhorada.
- Validar se o processo terá sucesso se implantado em qualquer fábrica de software.

3. Justificativa

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 (IZABEL, 2014).

Para suprir essa exaustão de testes funcionais 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 (BERNARDO; KON, 2008). 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 esse experiência em locais acessíveis, no entanto ainda é um setor que carece de muitos investimentos e inovações (JANNUZZI et al., 2016). Dessa forma, em um TCC apresentado na Unievangélica no ano de 2020 são exemplificadas formas de propagar esse conhecimento dentro do ambiente da FTT, a fim de reduzir a curva de aprendizagem e de espalhar a cultura dos processos e das ferramentas utilizadas (ALMEIDA; BATISTA, 2020).

Com isso, a fim de complementar esse estudo citado, este trabalho tem como foco aplicar um processo de testes automatizados que seja minimalista e de fácil compreensão para aqueles que o usarão, fazendo com que esse não seja esquecido no decorrer do tempo e ainda garantir mais qualidade aos produtos que serão entregues. A implantação deste ainda visa quantificar como os testes automatizados podem aumentar a qualidade do produto entregue e a redução de tempo das entregas. Ainda com a utilização dos testes automatizados pode-se viabilizar a execução de outros testes, como o de integração e o de regressão.

4. Fundamentação Teórica

4.1. 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 tradicional 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 as metodologias tradicionais (PRESNER; SANTOS JUNIOR, 2014).

4.2. 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).

4.3. 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 (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).

4.4. 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 o comportamento esperado. Originalmente os cenários de teste são derivados dos casos de uso, sendo necessário a sua escrita para cada cenário do caso de uso (DIAS NETO, [s.d.]).

4.5. 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. Desta forma 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 o 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).

4.5. 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). As ferramentas de teste automatizadas são capazes de executar testes, reportar resultados 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:*** Ela 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. E suas desvantagens são: 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 e 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?]).

5. Metodologia

A pesquisa a ser realizada neste trabalho pode ser classificada como explicativa, tendo em vista que o trabalho propõe um processo para realização de testes automatizados que visa auxiliar as equipes de teste a ter uma cultura de testes automatizados.

Para elaboração desse processo serão levantados trabalhos relacionados que contenham as seguintes palavras chaves: 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. Após o levantamento teórico será definida uma ferramenta de teste que terá o papel de auxiliar na escrita dos scripts de teste automatizado, a ferramenta escolhida deve atender os seguintes critérios: Ter uma linguagem de programação atual e consolidada no mercado, que seja de rápido aprendizado, e que a ferramenta esteja recebendo atualizações constantemente.

Neste trabalho serão coletados tanto dados quantitativos e qualitativos os quais serão utilizados para a validação do processo proposto: Para coleta dos dados quantitativos será proposto a criação de duas equipes de testadores uma fará os testes utilizado o processo que está sendo proposto nesse trabalho e a outra fará os testes de maneira tradicional (Teste manuais), as duas equipes farão os testes utilizado o mesmo requisito para que seja verificado quantos de bugs foram encontrados e tempo gasto de cada equipe. Depois de finalizado o primeiro requisito será entregue outro de mesma complexidade agora invertendo as equipes, para que todos tenham conhecimento sobre o processo proposto. Serão coletados dados qualitativos sobre a perspectiva da equipe de testadores sobre o processo, verificando assim sua aceitação. Esses dados serão coletados em primeiro momento na Fábrica de Tecnologias Turing (FTT).

6. Cronograma

ATIVIDADE	2021										2021									
	Fev		Mar		Abr		Maio		Jun		Jul		Ago		Setr		Out		Nov	
	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena	1ª Quinzena	2ª Quinzena
Levantar documentação teórica testes automatizados	X																			
Analisar o processo de testes da FTT			X																	
O escolher Linguagem e ferramenta para automação			X	X																
Realizar a instalação e configuração da ferramenta para validação. Elaboração do processo e Levantar dados quantitativos e qualitativos						X	X				X							X	X	
Apresentar resultados																				X
Levantar documentação teórica sobre a ferramenta			X	X	X															
Levantar documentação teórica sobre a linguagem			X	X	X															
Escrita do TCC		X X X		X		X												X		
Adequar melhorias da banca									X											
Implementar processo										X										

Realizar o levantamento das melhorias que foram observadas com a aplicação do processo																	X	X	X

7. Resultados alcançados

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. 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 1 - Processo

Fonte: 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*.

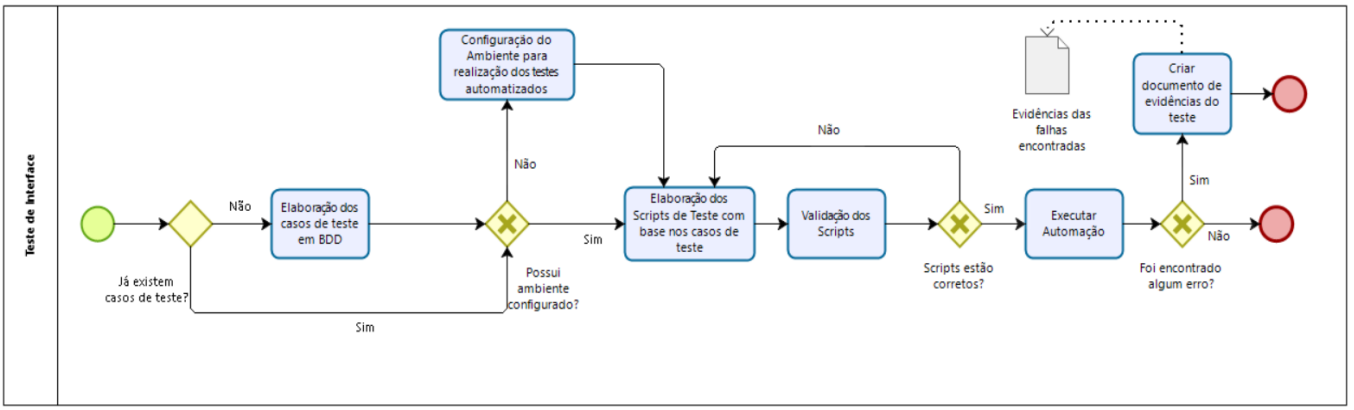
Após essa análise foi definida que será usada a ferramenta *Cypress.* Essa ferramenta foi citada na *Technology* Radar, uma revista que é publicada pela *ThoughtWorks,* uma empresa que busca atualizar a comunidade de software sobre as tendências que estão emergindo no mercado mundial através da publicação desta revista (THOUGHTWORKS, [2021?]).

A adoção do Cypress para a aplicação do processo para o experimento foi motivada pela sua curva de aprendizado simplificada, 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 que sua capacidade de

ser aplicado a testes de qualquer tipo de estrutura *front-end.*

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

Imagem 2 - Processo



Fonte: Os autores.

O processo foi definido com foco em agilidade e buscando ser o mais simples e direto possível para que não haja uma curva grande de aprendizado. Da forma como foi elaborado, 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 (https://www.dtidigital.com.br/blog/bdd-como-metodologia-agil/). 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.

8. Resultados esperados

Seguindo corretamente o cronograma proposto espera-se que sejam levantados os seguintes resultados:

- Levantamento de abordagens relacionadas;
- Propor um processo para Fábrica de Tecnologias Turing (FTT) que englobe o processo de testes funcionais;
- Ter o ambiente da Fábrica mapeado para a aplicação do processo;
- Modelar um processo mínimo viável;
- Aplicar o processo no ambiente de experimento;
- Mapear o ambiente após a intervenção do novo processo para analisar as vantagens e desvantagens da intervenção;
- Analisar os dados de mapeamento e da intervenção.
- Gerar conhecimento nas equipes com relação a automação de testes, para que a prática não se perca com o fator de alta rotatividade presente na fábrica.

Dessa forma espera-se que ao aplicar esse processo as empresas ou fábricas possam diminuir o tempo em seus testes e melhorar a qualidade dos softwares desenvolvidos, dessa forma elas possuirão um processo definido que norteará as tomadas de decisões do projeto na fase de verificação e validação.

9. Referências Bibliográficas

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.

ARARUNA, João Guilherme Santana. **Estratégias para realizar testes funcionais de interface com o usuário – visão de uma equipe de testes**. [S. l.], 2017. Disponível em: https://bdm.unb.br/bitstream/10483/19853/1/2017_JoaoGuilhermeSantanaAraruna.pdf. Acesso em: 15 jun. 2021.

BAUMGARTNER, Cristiano. **Testes de Regressão! Como, onde e quando utilizar?**. [S. l.], 16 ago. 2018. Disponível em: https://testingcompany.com.br/blog/testes-de-regressao-como-onde-e-quando-utilizar. Acesso em: 15 jun. 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.

DIAS NETO, Arilo Cláudio. **Introdução a Teste de Softwar**e. [S. l.], [s.d.]. Disponível em: http://www.garcia.pro.br/EngenhariadeSW/artigos%20engsw/teste/teste%20de%20software%20-%20artigo%201%20-%20rev1%20-%20introducao%20a%20teste%20de%20sw.pdf. Acesso em: 15 jun. 2021.

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.

EQUIPE MONITORA. **Quais os tipos de testes de software e por que automatizá-los?**. [S. l.], 11 fev. 2019. Disponível em: https://www.monitoratec.com.br/blog/quais-os-tipos-de-testes-de-software-e-por-que-automatiza-los/. Acesso em: 15 jun. 2021.

FIGUEIREDO, Eduardo. **Teste de Integração e Teste de Sistema**. [S. l.], [s.d.]. Disponível em: https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/aulas/testes-de-release_v01.pdf. Acesso em: 15 jun. 2021.

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

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.

METODOLOGIA EXTREME PROGRAMMING EM UMA APLICAÇÃO WEB. [S. l.], 2014. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/16837/3/PG_COADS_2014_1_06.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.

PRESNER, Diego Henrique; SANTOS JUNIOR, Elson Luiz dos. **ESTUDO SOBRE METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO APLICANDO A**

QUALIDADEBR. **Conheça o Watir**. [S.I] [2011]. Disponível em:<https://qualidadebr.wordpress.com/2011/03/05/conheca-o-watir/> Acesso em: 12 maio 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, Nikollas Filgueiras da. **Teste de Performance e a sua importância**. [S. l.], 7 out. 2019. Disponível em: https://medium.com/@nikollasfs22/teste-de-performance-e-a-sua-import%C3%A2ncia-2a4271fe37a6. Acesso em: 15 jun. 2021.

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.

TECNISYS. **TESTE DE SEGURANÇA DE APLICATIVOS DA WEB: O QUÊ? COMO? POR QUÊ?**. [*S. l.*], 16 set. 2020. Disponível em: http://www.tecnisys.com.br/noticias/2020/teste-de-seguranca-de-aplicativos-da-web-o-que-como-por-que. Acesso em: 15 jun. 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.

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.

VOLPATO, Elisa. T**este de usabilidade: o que é e para que serve?**. [S. l.], 15 set. 2015. Disponível em: https://brasil.uxdesign.cc/teste-de-usabilidade-o-que-%C3%A9-e-para-que-serve-de3622e4298b. Acesso em: 15 jun. 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.