UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENADORIA DO CURSO DE ENGENHARIA DE SOFTWARE

PAULO HENRIQUE BORDIGNON

FERRAMENTAS VISUAIS DE TESTE DE INTERFACES GRÁFICAS PARA CHECAR CONFORMIDADE DE REQUISITOS NÃO FUNCIONAIS

TRABALHO DE CONCLUSÃO DE CURSO

DOIS VIZINHOS

2019

PAULO HENRIQUE BORDIGNON

FERRAMENTAS VISUAIS DE TESTE DE INTERFACES GRÁFICAS PARA CHECAR CONFORMIDADE DE REQUISITOS NÃO FUNCIONAIS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Software, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Rafael A. P. de Oliveira

DOIS VIZINHOS

2019

TERMO DE APROVAÇÃO

Ferramentas Visuais de Teste de Interfaces Gráficas para Checar Conformidade de Requisitos não Funcionais

por

Paulo Henrique Bordignon

Este Trabalho de Conclusão de Curso foi apresentado em 05 de Julho de 2019 como requisito parcial para a obtenção do título de Bacharel em Engenharia de Software. O(a) candidato(a) foi arguido(a) pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Rafael Alves Paes de Oliveira
Presidente da Banca

Gustavo Jansen de Souza Santos
Membro Titular

Yuri Kaszubowski Lopes
Membro Titular

* A Folha de Aprovação assinada encontra-se na Coordenação do Curso

Dedico este trabalho aos meus pais, que estiveram ao meu lado durante todo o curso.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, por terem me auxiliado e motivado durante todo o curso, aos professores que me proporcionaram aprendizado e experiência e, aos amigos que foram facilitadores nessa caminhada.

Agradeço em especial meu orientador, por ter prestado todo auxílio necessário para que este trabalho fosse concluído e, também a empresa que aceitou participar do estudo e que disponibilizou os recursos necessários.

RESUMO

BORDIGNON, Paulo Henrique. FERRAMENTAS VISUAIS DE TESTE DE INTERFACES GRÁFICAS PARA CHECAR CONFORMIDADE DE REQUISITOS NÃO FUNCIONAIS. 77 f. Trabalho de Conclusão de Curso – Coordenadoria do Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2019.

Ferramentas VGT (Visual GUI Testing) podem ser utilizadas para automatizar testes, que são realizados manualmente por meio da interface gráfica do usuário. Tais ferramentas combinam o uso de *scripts* com o reconhecimento de imagem, e seu uso pode gerar economia de tempo e diminuição de ocorrência de erros relacionados a testes. Neste estudo, são avaliados por meio de um estudo de caso executado dentro da indústria, os benefícios encontrados ao usar uma ferramenta VGT para checar requisitos não funcionais, durante a implantação de um software. Os resultados indicam que o uso de tal ferramenta, trouxe benefícios para a checagem de requisitos não funcionais, entretanto, devido a algumas falhas apresentadas conclui-se que a tecnologia de reconhecimento de imagem, utilizada pela ferramenta, deve ser aprimorada.

Palavras-chave: Teste de Software, Requisitos Não Funcionais, Automatização de Testes, Teste Visual GUI.

ABSTRACT

BORDIGNON, Paulo Henrique. VISUAL GUI TESTING TOOLS TO CHECK COMPLIANCE OF NON-FUNCTIONAL REQUIREMENTS. 77 f. Trabalho de Conclusão de Curso – Coordenadoria do Curso de Engenharia de Software, Universidade Tecnológica Federal do Paraná. Dois Vizinhos, 2019.

VGT (Visual GUI Testing) tools can be used to automate tests that are performed manually through the graphical user interface. Such tools combine the use of scripts with image recognition, and their use can save time and reduce the occurrence of test-related errors. In this study, the benefits of using a VGT tool to check non-functional requirements during software deployment are evaluated through a case study performed within the industry. The results indicate that the use of such tool has brought benefits to the check of non-functional requirements, however, due to some flaws presented, it is concluded that the image recognition technology used by the tool should be improved.

Keywords: Software Testing, Non-functional Requirements, Test Automation, Visual GUI Testing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ES	Engenharia de Software
TI	Tecnologia da Informação
NFR	*Non-Functional Requirements*
ERP	Enterprise Resource Planning
GUI	*Graphical User Interface*
VGT	*Visual GUI Testing*
VV&T	Validação, Verificação e Teste
GQM	*Goal Question Metric*
CMD	Windows Command Prompt

SUMÁRIO

1 INTRODUÇÃO

Com o passar dos anos, as pessoas estão se tornando cada vez mais dependentes do uso de sistemas computacionais e sistemas de software em seus cotidianos. Nos dias atuais, sistemas de software são usados para controlar infraestruturas, serviços, produtos elétricos, manufaturas, sistemas financeiros e, outras diversas áreas da sociedade e do setor produtivo/econômico (SOMMERVILLE, 2011).

Diante desse cenário, a Engenharia de Software (ES) surgiu com o objetivo, de aumentar a qualidade desses sistemas de software, desenvolvendo-os de forma ágil sobre um processo preestabelecido e bem definido (SOMMERVILLE, 2011). A ES é oriunda da engenharia tradicional e visa a utilizar recursos de medições, análises, projetos, testes e automatizações para a implementação de software com qualidade (PRESSMAN, 2011).

Adicionalmente, a ES também contribui para que o setor de Tecnologia da Informação (TI) atenda às crescentes demandas por sistemas de software. A ES é necessária devido à complexidade dos requisitos dos sistemas contemporâneos que vêm se tornando cada vez mais complexos (PRESSMAN, 2011).

A qualidade que é um dos objetivos que a ES busca garantir nos sistemas, é também um assunto bastante antigo. Existem relatos que há mais de quatro mil anos, foi estabelecida uma medida de comprimento, que deveria ser usada pelos egípcios em suas construções, o uso da medida era obrigatório e a não conformidade de medidas em construções acarretava em punições aos responsáveis (KOSCIANSKI; SOARES, 2007).

Na indústria de software, além do ajuste a normas, a qualidade de um produto é melhorada por meio da execução de atividades de testes, dessa forma a execução de testes é imprescindível para que a qualidade de um software seja assegurada.

O teste de software é um conjunto de atividades dinâmicas que consistem na execução do programa com algumas entradas específicas, visando verificar se o comportamento do programa é condizente com sua especificação (BERTOLINO, 2007; DELAMARO et al., 2016). O teste quando feito de forma manual é reconhecidamente mais demorado, tedioso e propenso

a erros (BERTOLINO, 2007).

A automatização de atividades de testes e verificações é uma opção para diminuir a ocorrência de problemas. O emprego de testes automatizados contribui significativamente para a redução de custos e tempo de projeto durante o processo de desenvolvimento (MYERS, 2004). Segundo Rafi et al. (2012), aumento da qualidade do produto final, alto índice de cobertura, redução do tempo de teste, aumento da confiança, diminuição de esforços humanos e redução de custos são alguns dos benefícios do teste automatizado relatados na literatura.

Por isso, e pela aceitação entre pesquisadores e engenheiros de software, da ideia de que qualidade é um fator essencial no desenvolvimento de software, muito se tem investido em pesquisas na área de teste de software (DELAMARO et al., 2007).

Um problema latente do teste automatizado, é a dificuldade de encontrar estratégias produtivas que verifiquem Requisitos Não Funcionais (NFR do inglês, *Non-Functional Requirements*). No contexto de Engenharia de Software, NFRs especificam critérios que devem ser utilizados para julgar a operação de um software, em detrimento ao seu comportamento específico (Requisitos Funcionais) (BAJPAI; GORTHI, 2012).

No processo de desenvolvimento de software, existe a preocupação em atender aos requisitos não funcionais, pois eles levam benefícios adicionais ao usuário. O não atendimento desses requisitos pode acarretar na inoperabilidade do software, na perca de informações e, outros problemas que podem causar prejuízos ao usuário (ULLAH et al., 2011).

Uma necessidade latente da averiguação de NFRs na indústria ocorre no momento da implantação do software. Em tais cerimônias, o implantador deve avaliar o *hardware* disponível para a instalação do software e, essa avaliação muitas vezes é feita de forma manual e sem seguir um padrão criterioso de especificações. Quando esse processo de verificação ocorre de forma manual, são abertas possibilidades de problemas futuros relacionados a requisitos não funcionais.

A margem para problemas, deixada pela execução de testes manuais, pode ser minimizada por meio da execução de testes automatizados no momento da implantação. Uma possibilidade de automatização do processo de verificação de requisitos não funcionais é por meio do uso de ferramentas visuais de teste de interface gráficas como, por exemplo, EyeAutomate[1], JAutomate[2] e SikuliX[3].

Por meio de recursos de reconhecimento de imagem, essas ferramentas possibilitam a

[1] veja: http://eyeautomate.com/
[2] veja: http://jautomate.com/
[3] veja: http://sikulix.com/

reprodução de um processo, os testes automatizados interagem diretamente com a interface do software simulando o processo de um usuário. Tempo de resposta, questões de usabilidade, performance e outros, podem ser medidos por meio da implementação de *scripts* que sistematizem sua verificação.

Diante do problema descrito, o presente estudo visa facilitar, por meio da automatização de testes, a avaliação do correto funcionamento de NFRs, assim evitando futuros problemas por meio de uma análise precisa e sistemática. Portanto, o presente trabalho insere-se no contexto de estudos que visam garantir a qualidade de sistemas de software durante seus ciclos de vida.

1.1 PROBLEMA E JUSTIFICATIVA

O contexto do problema a ser abordado no presente estudo emana da indústria, em particular, de *software houses* de pequeno e médio porte que fazem implantação de seus produtos utilizando recursos tecnológicos (máquinas e internet) de clientes que muitas vezes não estão dispostos a investir em infraestrutura. Diante disso, em alguns casos, no ambiente de homologação, o software demonstra um correto atendimento aos NFRs. Porém, no ambiente de produção, após feita a instalação do software, o comportamento apresentado é outro.

Dessa forma, o profissional responsável pela implantação leva um tempo precioso para manualmente fazer averiguações de NFRs para conseguir diagnosticar o problema. A automatização de atividades de testes e verificações é uma opção para diminuir a ocorrência de problemas. Essa automatização objetiva minimizar de forma rápida e eficaz, os problemas ocasionados pelo não atendimento dos NFRs.

Diante do cenário apresentado, são necessários esforços de pesquisas que visem a automatizar a verificação de NFRs em ambientes de produção, desonerando e sistematizando o trabalho do profissional que faz a implantação do software. Perante o exposto, o presente estudo se justifica como uma necessidade de automatização de atividades de teste, visando a sistematização da averiguação de requisitos não funcionais.

1.2 OBJETIVOS

O estudo objetiva coletar problemas ocasionados pelo não funcionamento correto de NFRs, durante a fase de implantação de um determinado software ERP (do inglês, *Enterprise Resource Planning*), por conseguinte analisar ferramentas de automatização com intuito de

escolher a que melhor se encaixar para solucionar os problemas encontrados, destacando seus benefícios e identificando suas limitações. Por fim, concebendo um estudo detalhado com o objetivo de solucionar os problemas coletados.

Uma possível alternativa para a averiguação automatizada de NFRs em processos de implantação de software, é o uso de alguma ferramenta de automatização de testes funcionais. Por exemplo, ferramentas que automatizam o teste visual de interfaces gráficas.

1.2.1 OBJETIVO GERAL

Este trabalho de conclusão de curso está inserido no contexto da Engenharia de Software automatizada, propondo uma investigação da possibilidade prática do uso de ferramentas visuais de teste de interfaces gráficas, para o apoio automatizado à checagem de conformidade de requisitos não funcionais em ambientes de produção.

Diante do apresentado, o objetivo geral do trabalho é propor e avaliar o uso de ferramentas visuais de teste de interfaces gráficas para automatizar a validação de requisitos não funcionais, relacionados a desempenho e portabilidade. Por consequência, minimizando prejuízos e problemas a equipe responsável pela implantação do software.

1.2.2 OBJETIVOS ESPECÍFICOS

A partir do objetivo geral, apresentado na Seção anterior, foram levantados os seguintes objetivos específicos, apresentados abaixo:

- Realizar uma revisão bibliográfica sobre a validação de requisitos não funcionais;
- Realizar um levantamento de estudos acerca das ferramentas de teste visual de interfaces gráficas disponíveis e gratuitas para uso acadêmico;
- Estudar a viabilidade do uso de ferramentas de teste visuais para automatização de validações em atividades de implantação;
- Definir os requisitos não funcionais a serem testados;
- Proposição de um processo de teste que envolva ferramentas de teste visual, para a averiguação de conformidades de requisitos não funcionais;
- Definição e execução de validações empíricas para a proposta; e
- Realização de validações empíricas e coleta de dados.

1.3 METODOLOGIA

O presente estudo visa aproveitar conceitos de teste de interfaces gráficas para automatizar uma etapa da ES que é a averiguação de conformidade de requisitos não funcionais em ambiente de produção.

O teste automatizado de interfaces gráficas (GUI do inglês, *Graphical User Interface*) consiste no exercício de componentes da GUI de um sistema em teste por meio da simulação de interações com o usuário (ex: *clicks* do mouse em botões, preenchimentos de campos de textos, etc) (ALÉGROTH et al., 2015b).

Considerando o suporte à automatização, há uma classificação com três gerações de ferramentas:

- *(i)* a primeira geração consiste de automatizações baseadas em coordenadas de tela, no modelo record/playback;
- *(ii)* a segunda geração consiste em simulações de comportamentos de componentes; e
- *(iii)* por fim, a terceira geração consiste de ferramentas que combinam reconhecimento de imagem e *scripts* – as ferramentas visuais de teste GUI (VGT do inglês, *Visual GUI Testing*) (OLIVEIRA et al., 2015).

Diante desse contexto, o presente trabalho de conclusão de curso promove uma contribuição a partir de uma investigação sobre o uso de ferramentas VGT, para a automatização da averiguação de conformidade de requisitos não funcionais em ambientes de produção. Estudos empíricos foram propostos e realizados com esse intuito.

1.4 ORGANIZAÇÃO DA MONOGRAFIA

Este Capítulo introdutório, apresenta o tema a ser abordado no estudo, contextualizando o mesmo com a ES. Adicionalmente, o presente documento conta com 6 outras seções a saber:

- No Capítulo 2 são apresentados os aspectos conceituais;
- No Capítulo 3 é apresentado a metodologia do trabalho;
- No Capítulo 4 é apresentada a avaliação empírica;

- No Capítulo 5 são apresentados os resultados, junto com as ameaças a validade;
- No Capítulo 6 são apresentados os trabalhos relacionados; e
- No Capítulo 7 as conclusões do estudo.

2 ASPECTOS CONCEITUAIS

Neste Capítulo serão definidos alguns conceitos importantes, que estão relacionados a teste de software, ferramentas VGT e requisitos não funcionais. Serão apresentados termos técnicos referentes as áreas relacionadas ao estudo, com o objetivo de delimitar e elucidar o tema abordado no trabalho.

2.1 FUNDAMENTOS DO TESTE DE SOFTWARE

Segundo Sommerville (2011) o processo de teste possui dois objetivos, *(i)* demonstrar que o software atende a seus requisitos e, *(ii)* descobrir cenários nos quais o software se comporte de maneira incorreta. Ao testar um software são descobertas falhas para que sejam posteriormente corrigidas. De acordo com Myers (2004), por meio dessa atividade de encontrar e corrigir erros eleva-se a qualidade e a confiabilidade do software.

Na presente seção serão apresentados os conceitos e definições relacionados ao teste de software. Adicionalmente, também serão abordadas as técnicas, os critérios e as fases de teste. A Seção enfatiza a técnica de teste funcional, pois foi a técnica utilizada durante o desenvolvimento do trabalho.

2.1.1 DEFINIÇÕES E CONCEITOS

A construção de um software é uma atividade complexa, que objetiva atender aos requisitos coletados juntamente ao cliente por meio de um produto de software (PRESSMAN, 2011). Diversos problemas, relacionados a atividades de testes conduzidas de forma errônea, podem acarretar em um resultado diferente do esperado, ao concluir a construção do software. Desta forma, o teste é uma atividade crucial no desenvolvimento de um software, podendo determinar o sucesso ou o fracasso de um projeto (BERTOLINO, 2007).

Para serem descobertos problemas, antes e após o software ser liberado para sua utilização (ambiente de produção), existe uma série de atividades, coletivamente chamadas de

"Validação, Verificação e Teste" ou "VV&T", com a finalidade de garantir que tanto o modo pelo qual o software está sendo construído, quanto o produto em si estejam em conformidade com o especificado (DELAMARO et al., 2016).

A atividade de Validação deve assegurar que o produto sendo desenvolvido corresponda ao produto solicitado, conforme os requisitos do usuário. Por outro lado, a atividade de Verificação tende a assegurar consistência, completude e corretude do produto de software em cada fase e entre as fases consecutivas do ciclo de vida (MYERS, 2004).

Dessa forma, a VV&T é aplicada para garantir, que o software esteja em conformidade com os requisitos funcionais e não funcionais, solicitados pelo cliente. Ou seja, o objetivo é assegurar que o software desenvolvido satisfaça às necessidades do cliente e não possua problemas.

O termo "problemas" ou "inconsistência", genericamente utilizado em trabalhos sobre teste de software, refere-se a erros, defeitos, falhas e enganos. A literatura diferencia cada um desses quatro termos que podem ser apontados como os principais jargões da pesquisa aplicada na área de teste de software. O termo "defeito" é definido como sendo um passo, processo ou definição de dados incorretos. "Engano" é referido como sendo a ação humana que produz um defeito (DELAMARO et al., 2016).

A existência de um defeito pode ocasionar a ocorrência de um "erro" durante a execução do programa, que se caracteriza por um estado inconsistente ou inesperado. Tal estado pode levar a uma "falha", ou seja, pode fazer com que o resultado produzido pela execução seja diferente do resultado esperado (DELAMARO et al., 2016).

Testar um programa consiste da execução sistemática e controlada de um sistema de software visando a encontrar falhas para que seja possível revelar defeitos (BERTOLINO, 2007). Desse modo, é possível afirmar que o teste de software é uma atividade que envolve a execução sistemática de um sistema de software visando a identificar inconsistências (AMMANN; OFFUTT, 2016). A Figura 1 representa graficamente o conceito sobre teste de software.

Conforme apresentado pela Figura 1, os elementos essenciais para um ambiente automatizado de teste são:

- **P:** Programa em teste;
- **D(P):** Domínio de entrada de **P**;
- **T:** Dados de teste (entradas) selecionados para teste **P**;

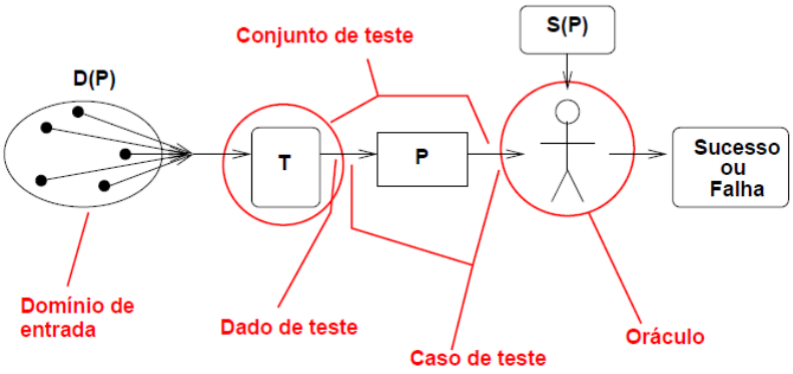

Figura 1: Representação gráfica do teste de software

- **S(P):** Domínio de saída de **P**;
- **Dado de teste:** entradas válidas ou inválidas para o domínio de entrada de **P**;
- **Caso de teste:** Par ordenado de entradas e saídas esperadas por **P**;
- **Conjunto de teste:** um conjunto de casos de teste para **P**; e
- **Oráculo:** mecanismo para definir se a saída obtida pela execução do dado de teste corresponde à saída esperada designada pelo caso de teste.

Diante do cenário exposto, é possível afirmar que o teste de um sistema com todos os valores do domínio de entrada tem sua prática inviável. Diante disso, técnicas e critérios de teste de software servem como estratégias para diminuir o domínio de entrada de um sistema de software durante atividades de teste. As seções abaixo apresentam detalhes conceituais acerca de técnicas e critérios de teste de software.

2.1.2 TÉCNICAS E CRITÉRIOS DE TESTE DE SOFTWARE

O teste realizado por meio da interface gráfica do sistema, pode detectar todos os defeitos, se o sistema for submetido a todas as possíveis entradas, entretanto, o domínio de entrada pode ser infinito ou muito grande. Essa limitação da atividade de teste, fez com que fossem definidas as três técnicas de teste e os diversos critérios pertencentes a cada uma delas (DELAMARO et al., 2016). Conceitualmente, técnicas estão associadas à fonte de informação utilizada para a criação de casos de teste.

Na técnica de teste funcional, também conhecida como teste de caixa preta, o sistema é avaliado segundo o ponto de vista do usuário, são fornecidas entradas e avaliadas as saídas

geradas para verificar se estão em conformidade com os objetivos especificados. Os critérios mais conhecidos do teste funcional são: Particionamento de Equivalência, Análise do Valor Limite, Grafo Causa-Efeito e *Error-Guessing*. (DELAMARO et al., 2016)

A técnica estrutural, ou caixa branca, estabelece os requisitos de teste com base em uma dada implementação, requerendo a execução de partes ou de componentes elementares do programa. Os caminhos lógicos do software são testados, fornecendo-se casos de teste que põem à prova tanto conjuntos específicos de condições e/ou laços bem como pares de definições e usos de variáveis. Os critérios desta técnica são geralmente classificados em: Critérios Baseados na Complexidade, Critérios Baseados em Fluxo de Controle, Critérios Baseados em Fluxo de Dados (DELAMARO et al., 2016).

Na técnica baseada em defeitos, são considerados defeitos típicos do processo de desenvolvimento de software, para que sejam derivados os requisitos de teste. Um dos critérios desta técnica é o teste de mutação, neste o programa que está sendo testado, é alterado várias vezes, criando-se vários novos programas ”mutantes”, assim simulando a inserção de defeitos no programa original. O trabalho do testador é criar casos de teste capazes de diferenciar o mutante do programa original. (DELAMARO et al., 2016)

A técnica de teste utilizada na realização deste trabalho, é a técnica funcional. Alguns requisitos não funcionais, como requisitos relacionados a portabilidade e ao desempenho, podem ser checados através da interface gráfica. Desta forma a técnica de teste funcional pode ser utilizada nestas checagens.

A técnica de teste funcional, não depende da utilização do código do software, nela é observado o comportamento da interface gráfica para a identificar problemas. Para realizar a checagem dos requisitos não funcionais, não foi utilizado nenhum dos critérios convencionais pertencentes a técnica funcional. Pelo fato de que nenhum dos critérios tenha demonstrado aplicabilidade na checagem de requisitos não funcionais.

2.1.3 FASES DO TESTE DE SOFTWARE

A atividade de teste é dividida em fases, devido a complexidade que um sistema pode possuir. Segundo (DELAMARO et al., 2016) existem quatro fases de testes, sendo elas: teste de unidade, teste de integração, teste de sistema e teste de regressão.

O teste de unidade tem o objetivo de verificar cada unidade que compõem o sistema, verificando se os dados de entrada e saída estão de acordo com a especificação da unidade. Após cada unidade ter sido testada, começa-se o teste de integração, onde são agrupadas as unidades

que se relacionam, com o objetivo de averiguar se as unidades estão integradas conforme a especificação (DELAMARO et al., 2016).

Após o sistema estar funcionando como um todo, são aplicados os testes de sistema, onde é verificado se o software possui um correto funcionamento quando combinado com um determinado tipo de *hardware*, banco de dados e SO (Sistema Operacional). Durante a manutenção do software, são realizados os testes de regressão, onde é avaliado se novas implementações feitas no sistema, não acarretaram em erros, nas partes que já haviam sido testadas (DELAMARO et al., 2016).

2.1.4 TESTE AUTOMATIZADO E TESTE MANUAL

Os testes automatizados são realizados através da utilização de ferramentas, que realizam entradas de dados com o objetivo de avaliar as saídas fornecidas pelo software. Desta forma a ferramenta checa os requisitos do software, e apresenta ao usuário se o software está ou não, funcionando como o esperado.

Empresas de software realizam testes automatizados por diversos motivos, como aumento da cobertura de testes, capacidade de reutilização de *scripts* de teste, redução de esforço, junto com a redução de falhas humanas, ganho de velocidade na execução dos testes e redução de custos com testes (RAFI et al., 2012).

Em testes manuais o testador fornece as entradas e avalia os resultados, enquanto no teste automatizado este trabalho é feito por ferramentas. Uma grande vantagem do teste manual em relação ao teste automatizado, é que a pessoa consegue perceber diversos comportamentos inesperados, e variar testes facilmente (HOFFMAN, 2001).

Com o objetivo de comparar testes manuais com testes automatizados, Oliveira (2013) apresenta um comparativo de vantagens e desvantagens entre essas duas abordagens, conforme demonstrado no Quadro 1.

Como pode ser visto no Quadro 1, Oliveira (2013) cita como vantagem do teste manual, a possibilidade que o humano possui de visualizar diversos comportamentos inesperados, e como desvantagens cita, que a eficiência em detectar erros pode ser prejudicada pela fadiga e, que determinados aspectos, podem não ter soluções manuais.

Em relação a testes automatizados Oliveira (2013) apresenta como vantagem, a obtenção de produtividade e eficácia, assim como a elaboração de casos de testes mais complexos e eficientes. As desvantagens citadas estão relacionadas a algumas dificuldades e complexidades, e também a não verificação de erros que não foram projetados para serem

Quadro 1: Teste Manual e Teste Automatizado

Teste Manual	
Vantagens	**Desvantagens**
Acompanhamento humano possibilita a notificação de diversos comportamentos inesperados.	A eficiência em detectar erros deve considerar a fadiga humana.
Notificação de comportamentos inesperados das execuções sob diferentes dados, estados, configurações e ambientes.	Determinados dados de teste, iterações e combinações podem requerer automatização.
	Determinados aspectos podem não ter soluções manuais (ex: análises de performance).
	A espera do testador por determinados erros, após algumas repetições, prejudica a detecção de defeitos.
Teste Automatizado	
Vantagens	**Desvantagens**
Casos de testes mais complexos, bem elaborados e eficientes.	Alguns domínios de teste proporcionam dificuldade extrema para automatizações.
Ferramentas aliam quantidade, produtividade e eficácia.	Verificação apenas das condições para as quais a automatização foi projetada para averiguar.
Possibilidade de transferência de tecnologia entre indústria e academia.	Dificuldades em definir quais as condições devem ser verificadas para classificar um teste como bem sucedido.
Eficiência para a detecção de tipos específicos de erros em determinados domínios.	Automatizações que geram casos de teste são de implementação complexa.

Fonte: (OLIVEIRA, 2013)

averiguados.

2.2 TESTE VISUAL DE INTERFACES GRÁFICAS

Ferramentas VGT são definidas como a terceira geração de ferramentas, que realizam testes utilizando-se da interface gráfica do usuário (OLIVEIRA et al., 2015). Um dos benefícios da utilização destas ferramentas é a flexibilidade de uso, as mesmas podem ser utilizadas para testar qualquer sistema que possua interface gráfica (GAROUSI et al., 2017).

Nesta Seção será discorrido a respeito da utilização de ferramentas VGT para a execução de testes. Serão abordadas as definições e os conceitos, que estão relacionados a este conteúdo.

2.2.1 DEFINIÇÕES E CONCEITOS

Ferramentas VGT combinam o reconhecimento de imagem com *scripts*, para executar testes automatizados, essas ferramentas permitem ao testador automatizar testes com um alto nível de abstração do sistema e, emular o comportamento do usuário final (ALÉGROTH et al., 2013).

Ferramentas VGT são utilizadas para testar interfaces de aplicativos, assim verificando suas funcionalidades. Utilizando essas ferramentas, os casos de teste criados pelo testador, podem ser traduzidos em forma de *scripts*, assim a ferramenta VGT executará o *script* e apresentará o resultado do teste para o testador.

Uma interface gráfica bem projetada e bem desenhada, é muitas vezes a chave para o sucesso do software. Devido a razões como essa, o teste visual de interface gráfica, atrai a atenção de profissionais e pesquisadores, que estão gastando uma quantidade significativa de tempo modelando, implementando e testando sistemas de software com tais ferramentas (OLIVEIRA et al., 2015).

2.2.2 FERRAMENTAS DE APOIO

Através de pesquisas, foram encontradas três ferramentas VGT que possuem licença gratuita, e que podem ser utilizadas durante o estudo, são elas: EyeAutomate, JAutomate e SikuliX.

Com intuito de conhecer melhor e poder comparar tais ferramentas, para cada uma delas, foi criado um *script* capaz de verificar se a versão do Microsoft .NET Framework, presente na máquina do usuário, é igual ou superior a versão 4. A seguir serão apresentadas as ferramentas juntamente com tais *scripts*.

EyeAutomate pode ser usada em qualquer sistema operacional que suporte Java, a ferramenta possui muitos comandos e é de fácil manuseio. A empresa que produz a ferramenta é chamada Auqtus e, é fundada por acadêmicos dedicados à transferência de conhecimento acadêmico para a indústria (AUQTUS, 2019). Na Figura 2, pode ser visto a interface gráfica da ferramenta, junto com o *script* que verifica a versão do Microsoft .NET Framework.

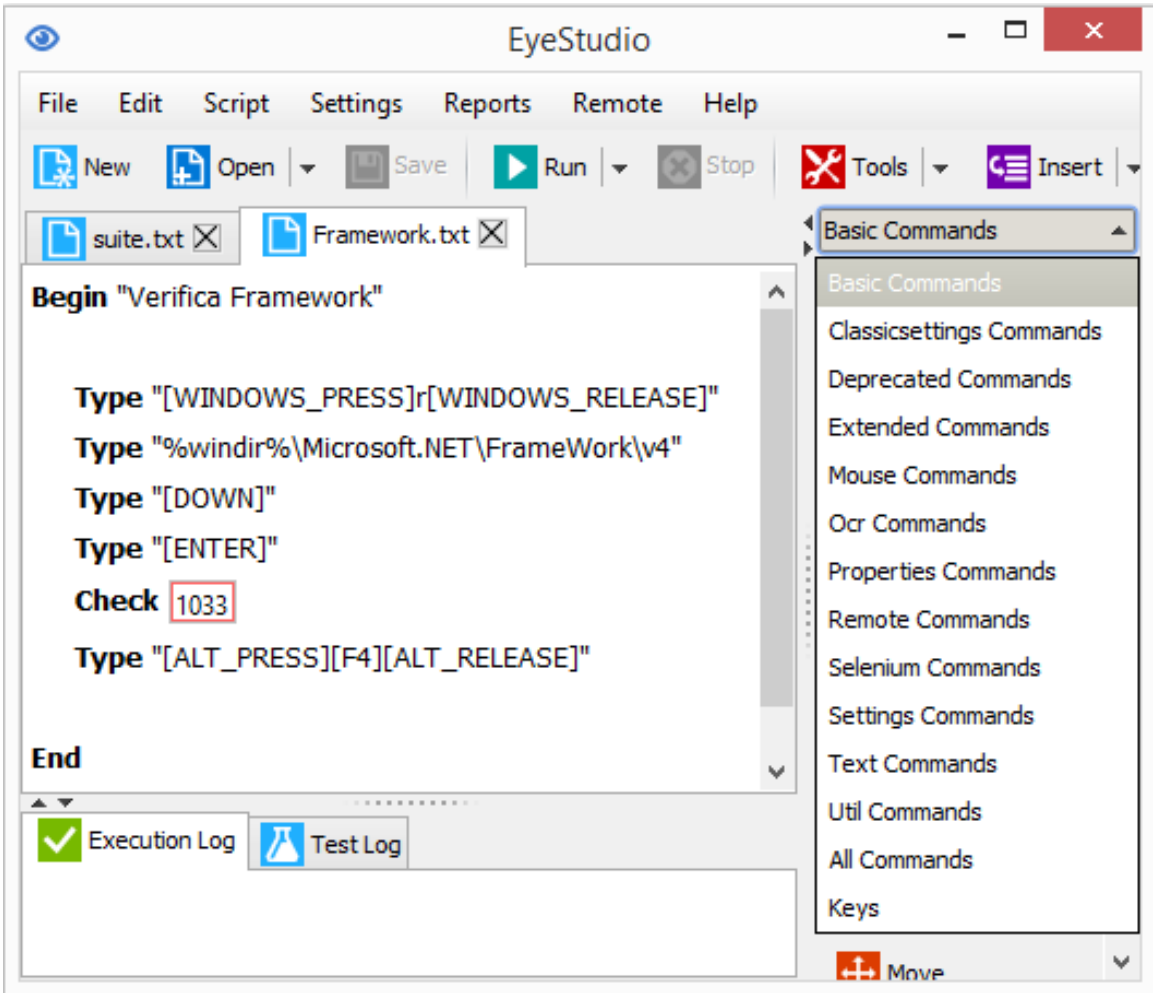

Figura 2: Ferramenta VGT EyeAutomate.

Fonte: Autoria própria.

Após executar um conjunto de testes, a ferramenta mostra ao usuário qual foi o resultado de cada teste, permitindo também visualizar um histórico de resultados, junto com o tempo consumido e, entre outras estatísticas de execução. A Figura 3 mostra os resultados após a execução de um conjunto de testes, em vermelho ficam os testes que falharam e, em verde os testes que obtiveram sucesso na verificação.

Command	Run Status
StopIfFailed No	Passed
ManualRecovery No	Passed
Call scripts/TCC/versaoWindows.txt	Passed
Call scripts/TCC/Framework.txt	Failed Screenshot
Call scripts/TCC/versaoIE.txt	Passed
Call scripts/TCC/aberturaSistema.txt	Passed
Call scripts/TCC/logaSistema.txt	Passed
Call scripts/TCC/listagemClientes.txt	Passed
Call scripts/TCC/cadastroCliente.txt	Passed
Call scripts/TCC/listagemVendas.txt	Passed
Call scripts/TCC/cadastrarVenda.txt	Passed
Call scripts/TCC/verificaCalendario.txt	Passed
Call scripts/TCC/fazLogoff.txt	Passed

Figura 3: Resultado de testes exibido pela EyeAutomate.

Fonte: Autoria própria.

A ferramenta JAutomate que também pode ser executada em qualquer sistema operacional que suporte Java, possui a interface gráfica muito parecida com a EyeAutomate, porém possui um menu com menor número de comandos. Na Figura 4, pode ser visto a interface gráfica da ferramenta, junto com o *script* que verifica a versão do Microsoft .NET Framework.

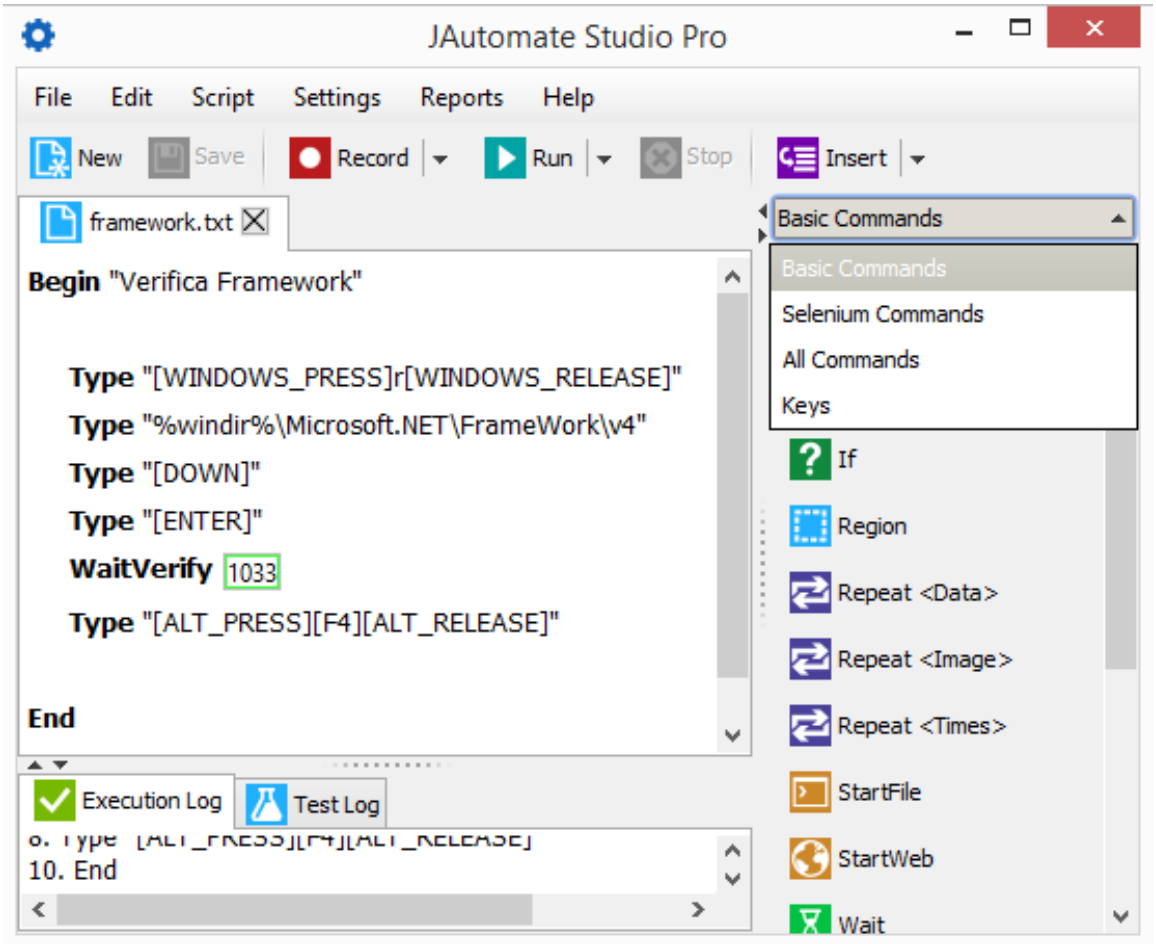

Figura 4: Ferramenta VGT JAutomate.

Fonte: Autoria própria.

Como pode ser visto as duas ferramentas são muito parecidas, apenas uma linha do *script* criado para checagem, apresenta diferença, a forma como os resultados são exibidos é o mesmo. A maior diferença entre essas duas ferramentas fica por conta da função ”Recorder”que é existente apenas em JAutomate, essa função cria *scripts* automaticamente, através da gravação de testes feitos manualmente pelo usuário.

A ferramenta SikuliX possui uma interface gráfica um pouco diferente das outras duas ferramentas e, também possui um menor número de comandos que as demais. A principal diferença desta para as outras duas, é que SikuliX possui código aberto, sendo possível acompanhar o desenvolvimento da mesma no GitHub. Na Figura 5, pode ser visto a interface gráfica da ferramenta, junto com o *script* que verifica a versão do Microsoft .NET Framework.

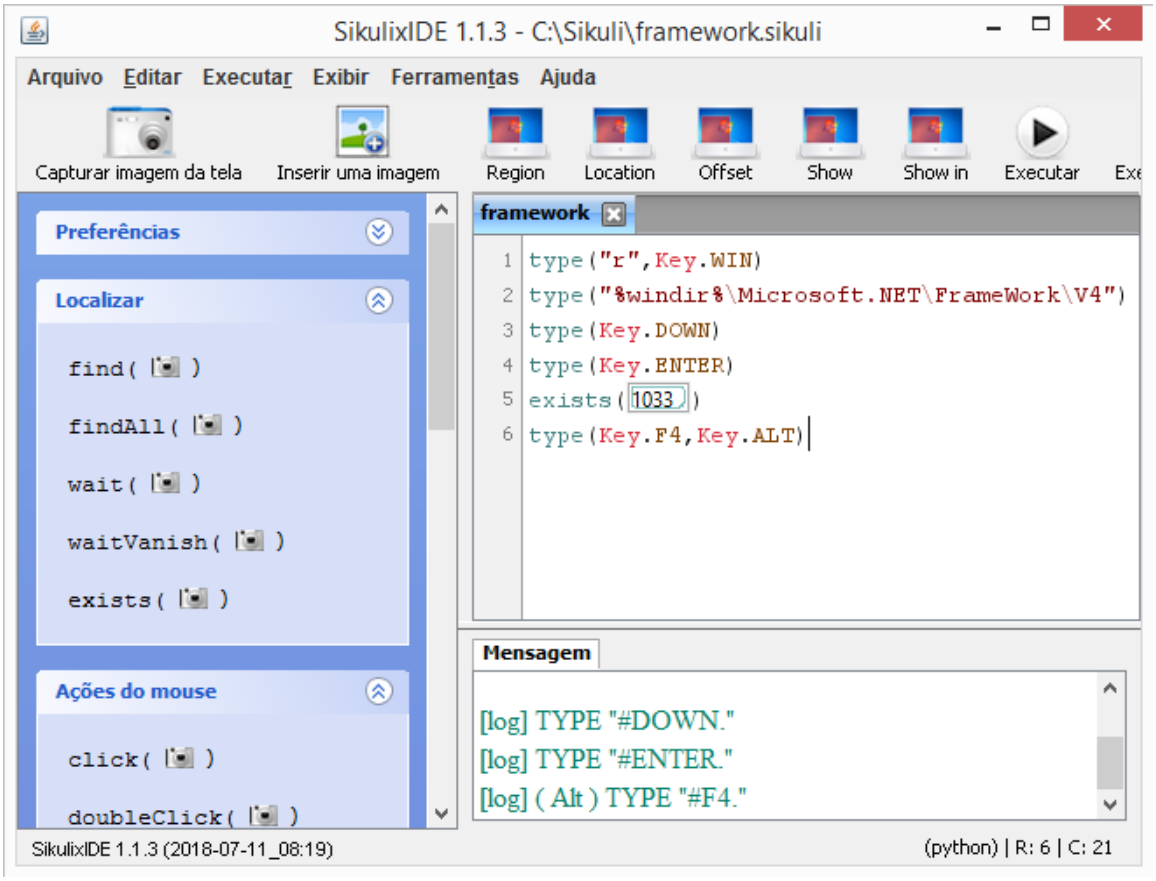

Figura 5: Ferramenta VGT SikuliX.

Fonte: Autoria Própria.

Conforme pode ser percebido, SikuliX é a ferramenta que mais se distingue das demais, porém a quantidade de linhas necessárias para realizar a checagem definida, foi a mesma para todas as ferramentas. Os comandos utilizados também são muitos parecidos, e a sintaxe possui poucas diferenças. A maior diferença fica por parte dos comandos, onde EyeAutomate possui um menu com muitas mais opções.

2.3 REQUISITOS NÃO FUNCIONAIS

Requisitos não funcionais são restrições aos serviços ou funções oferecidas pelo sistema. Incluem restrições de tempo de acesso, restrições no processo de desenvolvimento e restrições impostas por normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo (SOMMERVILLE, 2011).

Nesta Seção serão apresentados os conceitos e as definições relacionadas a requisitos

não funcionais, juntamente será apresentado uma categorização para NFRs e a forma como é averiguada a conformidade dos mesmos.

2.3.1 CONCEITOS E DEFINIÇÕES

Requisitos não funcionais, ao contrário dos requisitos funcionais, não estão relacionados a nenhuma função a ser realizada pelo software, estão relacionados aos comportamentos e restrições que o software deve satisfazer (CYSNEIROS L.M.; LEITE, 1997).

Requisitos não funcionais quando não atendidos, podem provocar prejuízos aos usuários, ocasionar erros críticos ao sistema e, até mesmo, causar o fracasso de um projeto. Os erros provocados pelo tratamento inadequado ou pela falta de tratamento de NFRs, são apontados entre os mais caros e difíceis de corrigir (CYSNEIROS; LEITE, 2004).

Deixar de atender a um requisito não funcional pode significar a inutilização de todo o sistema. Por exemplo, se um sistema de aeronaves não cumprir seus requisitos de confiabilidade, não será certificado como um sistema seguro para operar, e assim não poderá ser utilizado (SOMMERVILLE, 2011).

Portanto os requisitos funcionais estão relacionados as funções disponibilizadas pelo sistema, já os requisitos não funcionais estão relacionados ao uso do sistema, e abordam aspectos relacionados a qualidade do mesmo. Os requisitos não funcionais são divididos em algumas categorias, as quais serão explicadas na Subseção a seguir.

2.3.2 CATEGORIAS DE REQUISITOS NÃO FUNCIONAIS

Na Figura 6 é exibida a categorização de Sommerville (2011) para requisitos não funcionais. Conforme mostra a imagem, os NFRs podem ser provenientes de requisitos do produto, de requisitos organizacionais e de requisitos externos.

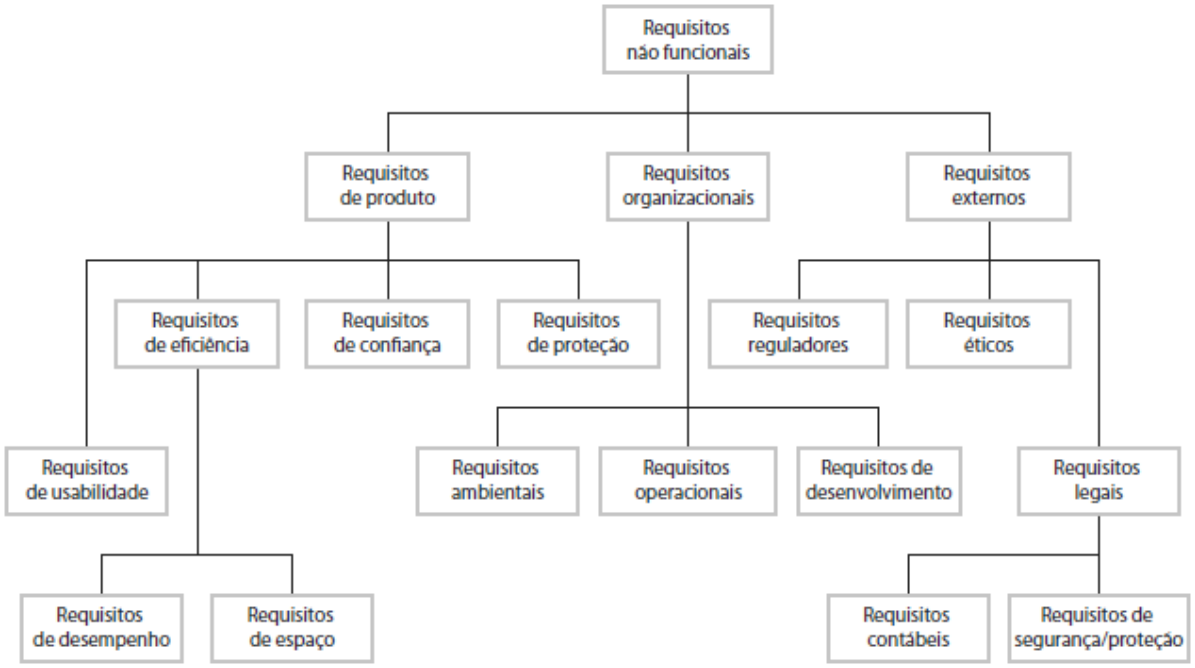

Figura 6: Tipos de requisitos não funcionais

Fonte: (SOMMERVILLE, 2011)

Os requisitos de produto especificam o comportamento do software, são exemplos requisitos relacionados a desempenho, confiabilidade, proteção e usabilidade. Requisitos organizacionais, estão relacionados à qualidade do projeto, um exemplo é o requisito do processo de desenvolvimento, que especifica a linguagem de programação, o ambiente e as normas de desenvolvimento. Requisitos externos estão relacionados a fatores externos, um exemplo é o requisito ético que deve assegurar que o sistema será aceitável para seus usuários e para o público em geral (SOMMERVILLE, 2011).

Conforme descrito anteriormente, os NFRs podem ser provenientes de três conjuntos de requisitos, entre eles está o conjunto de requisitos do produto, do qual derivam os tipos de NFRs que serão utilizados neste trabalho. São alguns exemplos de tipos de NFRs relacionados ao produto: usabilidade, confiabilidade, desempenho e portabilidade.

A usabilidade está ligada as questões de quão fácil é usar o sistema e, de quão bom é usar o sistema. Para Bezerra (2015) são exemplos de requisitos ligados a usabilidade, a facilidade de uso do sistema e a necessidade ou não de treinamento dos usuários.

A confiabilidade está relacionada a probabilidade de um sistema, não apresentar falhas durante um determinado tempo. Para Bezerra (2015) o tempo médio entre falhas, recuperação de falhas, e quantidade de erros por milhares de linhas de código, são exemplos de requisitos

relacionados a confiabilidade.

O desempenho está relacionado ao tempo levado pelo sistema, para realizar determinada operação. Bezerra (2015) diz que um NFR de desempenho, pode definir os tempos de resposta esperados para as funcionalidades do sistema.

A portabilidade está relacionada a facilidade do sistema ser executado em diferentes *hardwares*, softwares e sistemas operacionais. Para Bezerra (2015) tal requisito diz respeito sobre, as restrições relacionadas as plataformas de *hardware* e de software nas quais o sistema será implantado, e sobre o grau de facilidade para transportar o sistema para outras plataformas.

2.3.3 AVERIGUAÇÃO DE CONFORMIDADE

Segundo o que introduzem Ribeiro e Travassos (2016) a literatura não apresenta com clareza, quais testes de software devem ser realizados para uma correta checagem de NFRs. Ribeiro e Travassos (2016) concluem que a maioria das abordagens de testes de NFRs, não são avaliadas empiricamente, acarretando na falta de evidências sobre seus benefícios e riscos no uso prático.

Barmi e Ebrahimi (2011) concluem através de uma revisão sistemática, que existe a necessidade de que mais estudos sejam realizados, com o objetivo de oferecer propostas para solucionar problemas relacionados a testes de NFRs. Barmi e Ebrahimi (2011) citam também que estudos sobre testes de NFRs foram realizados principalmente na academia, e que são necessários mais estudos que avaliem a aplicabilidade de estudos na indústria.

3 METODOLOGIA

Este trabalho consiste na elaboração e execução de um estudo de caso, relacionado a automatização de checagem de requisitos não funcionais. A execução deste estudo foi realizada na indústria de software, utilizando um ERP para realização dos testes necessários para que o estudo fosse desenvolvido.

O objetivo geral deste estudo, é avaliar a eficácia do uso de ferramentas VGT para a automatização da verificação de requisitos não funcionais. Foram coletados problemas relacionados a NFRs durante a implantação do ERP utilizado, para que os mesmos pudessem ser utilizados no decorrer do estudo.

O trabalho apresentará de que forma as ferramentas VGT, podem realizar a verificação automatizada de NFRs e, como cada tipo de NFR pode ser verificado, através de *scripts* criados na ferramenta escolhida. O presente trabalho, apresenta uma estratégia para a averiguação de NFRs.

No decorrer deste Capítulo é discorrido a respeito da escolha da ferramenta utilizada, a respeito da escolha dos NFRs a serem checados e, também a respeito dos métodos utilizados no estudo.

3.1 ESCOLHA DA FERRAMENTA

Por meio de pesquisas relacionadas a automatização de testes de interface gráfica, realizadas em diversos repositórios, foram encontradas três ferramentas utilizadas em estudos desta área. As três ferramentas encontradas e que foram levadas em conta neste trabalho, foram: EyeAutomate, JAutomate e SikuliX.

Após realizada a busca pelas ferramentas, foi realizada uma análise exploratória afim de decidir qual ferramenta seria utilizada no trabalho. A primeira ferramenta a ser analisada foi a JAutomate, onde foi constatado que no site oficial da ferramenta, existe um anúncio dizendo que o desenvolvimento da mesma foi descontinuado, e um *link* referenciando o site da ferramenta

EyeAutomate. Desta forma, foi optado por descartar a utilização da ferramenta JAutomate.

Entretanto, foi enviado um e-mail ao suporte da ferramenta EyeAutomate, para saber qual era a relação existente entre JAutomate e EyeAutomate, o suporte rapidamente respondeu que EyeAutomate é uma continuação de JAutomate, e que a alteração de nome ocorreu quando a empresa Auqtus comprou os produtos da empresa Swifting.

Após descartar a utilização de JAutomate, foi iniciada uma comparação entre as ferramentas EyeAutomate e SikuliX, entretanto, não foram localizados estudos que realizassem comparação entre essas duas ferramentas. Porém foram encontrados estudos que comparavam JAutomate com SikuliX, nos dois estudos encontrados, JAutomate apresentou ligeira vantagem em relação a SikuliX.

Garousi et al. (2017) e Haar e Michaëlsson (2018) apresentam resultados que constatam uma ligeira superioridade da ferramenta JAutomate em relação a SikuliX, ambos estudos foram realizados dentro de empresas, com o objetivo de adotar o uso de uma delas para a realização de testes automatizados em projetos.

Por fim, sabendo que EyeAutomate é uma continuação de JAutomate, e que os resultados dos estudos encontrados apontaram uma superioridade de JAutomate em ralação a SikuliX, foi optado pela utilização da ferramenta EyeAutomate no decorrer deste trabalho.

3.2 ESCOLHA DOS REQUISITOS NÃO FUNCIONAIS

Após conversas realizadas junto aos implantadores, foram analisados os requisitos não funcionais de um sistema ERP em específico, para que fosse realizada a escolha de quais NFRs seriam checados pela ferramenta. Os requisitos não funcionais escolhidos para serem checados, estão dispostos no Quadro 2 abaixo.

Quadro 2: Requisitos não funcionais escolhidos.

Evento	RQ NFR	Tipo NFR
Emitir Doc Fiscal	S.O Windows 7 ou superior	Portabilidade
Emitir Doc Fiscal	Framework 4.0 ou superior	Portabilidade
Importar Doc Fiscal	IE11 instalado	Portabilidade
Abrir tela de login	Até 10 segundos	Desempenho
Realizar login	Até 10 segundos	Desempenho
Listar Clientes	Até 10 segundos	Desempenho
Cadastrar Clientes	Até 10 segundos	Desempenho
Listar Vendas	Até 10 segundos	Desempenho
Cadastrar Venda	Até 10 segundos	Desempenho
Abertura do Calendário	Até 5 segundos	Desempenho
Realizar logoff	Até 5 segundos	Desempenho

Fonte: Autoria própria.

Os requisitos não funcionais apresentados no Quadro 2, caso não estejam em conformidade, não possibilitarão ao usuário a melhor experiência possível de uso do sistema. Os NFRs escolhidos em teoria sempre devem ser checados na instalação do software, porém por se tratar de um processo realizado manualmente, é aberto margem para que seja esquecido de analisar um ou outro requisito.

3.3 MÉTODOS

No decorrer do estudo de caso foram utilizados dois métodos de pesquisa, questionários e pesquisa de campo. Para introduzir o estudo, foi aplicado um questionário, logo em seguida foi realizada a pesquisa de campo, e para concluir foi aplicado um segundo questionário.

O primeiro questionários foi realizado com objetivo de mensurar o grau de conhecimento acerca de NFRs e ferramentas VGT, por parte de profissionais atuantes nas áreas de teste, implantação ou desenvolvimento. O segundo questionário foi aplicado apenas aos implantadores, que participaram da execução dos *scripts* criados para checar os NFRs, o objetivo desse segundo questionário foi coletar conclusões relacionadas ao uso da ferramenta.

A pesquisa de campo foi realizada junto aos implantadores, com o objetivo de verificar como a ferramenta VGT se comportaria na verificação dos NFRs. Através da execução dos

scripts puderam ser avaliados alguns pontos como, quantidade de acertos e falhas, tempo levado para realizar a checagem automatizada, entre outros pontos.

4 AVALIAÇÕES EMPÍRICAS

4.1 QUESTÕES DE PESQUISA

As questões de pesquisa foram elaboradas utilizando a abordagem GQM (*Goal Question Metric*), nesta abordagem são definidas metas, questões de pesquisa e métricas. Com a meta sendo o objetivo geral deste trabalho, foram definidas as questões de pesquisa e as métricas assim sendo possível obter respostas para a meta.

O objetivo geral do estudo é verificar a eficácia/viabilidade do uso de ferramentas VGT, para checar NFRs no processo de implantação de software, no ambiente de produção de uma fábrica de software. Através do objetivo geral foram definidas as seguintes questões de pesquisa:

- **Q1**: Identificar e especificar NFRs passíveis de serem verificados através de ferramentas VGT;
- **Q2**: Produzir/Prototipar *scripts* para verificação dos NFRs;
- **Q3**: Verificar a robustez dos *scripts* VGT para verificação dos NFRs;
- **Q4**: Determinar a viabilidade da execução dos *scripts* no ambiente de implantação;
- **Q5**: Averiguar a opinião de profissionais dentro da indústria a respeito do uso de VGT;
- **Q6**: Fazer uma análise de *trade-offs* entre o uso de VGT para checagem versus checagem manual.

As questões de pesquisa foram elaboradas, com a finalidade de alcançar os objetivos definidos para este trabalho, estas questões serão respondidas por meio da realização do estudo de caso. No decorrer do trabalho, serão apresentados os resultados obtidos para cada questão de pesquisa.

4.2 ROTEIRO DO ESTUDO DE CASO

Após definida a ferramenta a ser utilizada e definidos os requisitos não funcionais que seriam checados, foi definido um roteiro para a realização do estudo de caso. Tal roteiro é demonstrado através de um fluxograma, o qual está presente na Figura 7.

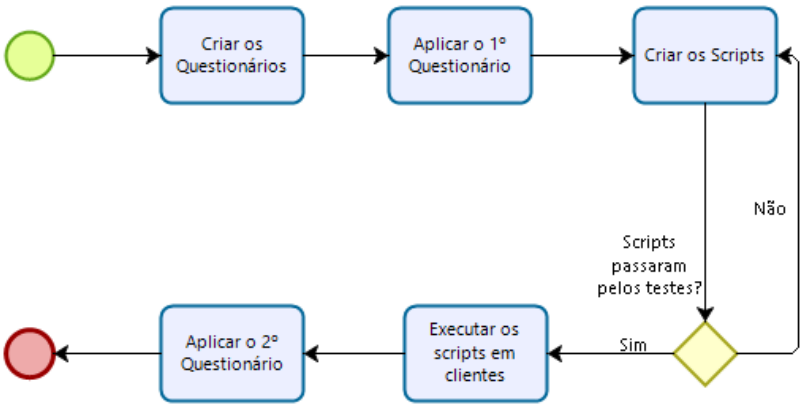

Figura 7: Roteiro do estudo de caso

Fonte: Autoria Própria

Como representado através do fluxograma, a primeira atividade definida para ser realizada foi a criação dos questionários, seguinte a isso deveria ser aplicado o primeiro questionário, o objetivo era medir alguns parâmetros entre os profissionais da área. A criação dos *scripts* veio logo em seguida, após a criação dos mesmos eles passaram por vários testes, e só foram executados em clientes após nenhuma falha ser detectada nestes testes. Após os *scripts* terem sido executados em clientes, o segundo questionário foi aplicado para coletar conclusões a respeito do estudo realizado.

4.2.1 QUESTIONÁRIOS

A primeira atividade realizada durante o estudo de caso foi a criação dos questionários. Foram criados dois questionários, o primeiro com intuito de mensurar o conhecimento de profissionais acerca de NFRs e ferramentas VGT, já o segundo com o objetivo de concluir o estudo, e coletar os resultados percebidos pelos implantadores que participaram do estudo.

O primeiro questionário foi criado com nove perguntas, de uma forma geral, foram criadas perguntas para saber a função que cada participante desempenhava, e o seu período de experiência com TI. Também foram criadas perguntas para verificar se o participante conhecia

e já havia utilizado ferramentas VGT, e se conhecia ou já havia trabalhado com NFRs, outra pergunta foi elaborada para saber se na opinião do participante, uma ferramenta VGT poderia ser utilizada para automatizar a checagem de NFRs.

O segundo questionário foi criado com quatorze perguntas, neste questionário, foram criadas perguntas com o objetivo de responder as questões de pesquisa e assim concluir o estudo. Para cada questão de pesquisa foram criadas duas perguntas relacionadas, foi criada também uma pergunta aberta para que o participante pudesse responder como a experiência poderia ser melhorada.

4.2.2 SCRIPTS CRIADOS

Para executar a realização das checagens dos requisitos não funcionais, foram desenvolvidos doze *scripts*, sendo um deles, uma suíte de testes, que realiza a chamada de cada *script* e define a ordem que cada um será executado. Cada um dos outros onze *scripts*, é responsável por realizar a checagem de um requisito não funcional definido na Seção 2 do Capítulo 3.

Dentro da suíte de testes, são realizadas as chamadas dos demais *scripts* por meio do comando ”Call”. O comando ”StopIfFailed”foi definido com valor ”não”para que caso algum *script* falhe, a execução dos demais não seja interrompida. Outro comando utilizado foi o ”ManualRecovery”onde foi atribuído o valor ”não”, para que não fosse solicitada intervenção humana caso ocorresse alguma falha na checagem. Na Figura 8 é apresentado o *script* referente a suíte de testes.

```
StopIfFailed "No"
ManualRecovery "No"

Call "scripts/TCC/versaoWindows.txt"
Call "scripts/TCC/Framework.txt"
Call "scripts/TCC/versaoIE.txt"

Call "scripts/TCC/aberturaSistema.txt"
Call "scripts/TCC/logaSistema.txt"

Call "scripts/TCC/listagemClientes.txt"
Call "scripts/TCC/cadastroCliente.txt"
Call "scripts/TCC/listagemVendas.txt"
Call "scripts/TCC/cadastrarVenda.txt"
Call "scripts/TCC/verificaCalendario.txt"

Call "scripts/TCC/fazLogoff.txt"
```

Figura 8: Suíte de Testes

Fonte: Autoria Própria

Seguindo a ordem de chamadas da suíte de testes, o primeiro requisito não funcional checado foi o que verifica a versão do Windows presente no computador. O *script* criado para realizar tal checagem está presente na Figura 9.

```
Begin "Verifica Versao Windows"

   Type "[WINDOWS_PRESS]r[WINDOWS_RELEASE]"
   Type "winver[ENTER]"
   Timeout "5"
   Check 
   Type "[ESCAPE]"
Catch
   Timeout "5"
   Check 
   Type "[ESCAPE]"
Catch
   Timeout "5"
   Check 
   Check
   Type "[ESCAPE]"

End
```


Figura 9: Script - Verifica a Versão do Windows

Fonte: Autoria Própria

Caso a versão do Windows presente no computador, não fosse igual a uma das versões checadas pelo *script* o teste falharia, e assim poderia ser concluído que um NFR não havia sido atendido. O comando ”timeout”é utilizado no *script* com o objetivo de delimitar o máximo de 5 segundos para que a imagem seja reconhecida, o principal objetivo da utilização deste comando, é evitar que seja desperdiçado tempo pela execução do *script*.

O segundo *script* chamado pela suíte de testes, é o responsável por verificar a versão do Microsoft .NET Framework, que deve ser superior a versão 4.0. O *script* utiliza apenas os comandos ”type”responsável por executar ações do teclado, e ”check”responsável por checar a existência de determinada imagem na tela, conforme pode ser visto na Figura 10.

```
Begin "Verifica Framework"

   Type "[WINDOWS_PRESS]r[WINDOWS_RELEASE]"
   Type "%windir%\Microsoft.NET\FrameWork\v4"
   Type "[DOWN]"
   Type "[ENTER]"
   Check 1033
   Type "[ALT_PRESS][F4][ALT_RELEASE]"

End
```

Figura 10: Script - Verifica a Versão do Framework

Fonte: Autoria Própria

Dentre os *scripts* que restam, a maior variação de comandos em relação aos *scripts* já apresentados, está no *script* que realiza o *login* no sistema, com o objetivo de medir o tempo levado. Tal *script* está presente na Figura 11.

```
Begin "Loga no Sistema"

   Check Confirmar
   IfNot Código
            1
      Type "1[ENTER]"
   EndIf
   StepDelay "1000"
   Type "visual[ENTER]"
   StepDelay "1000"
   Type "[CTRL_PRESS][F12][CTRL_RELEASE]"
   Type "vssql@!2007[ENTER]"
   Timeout "10"
   Type "[ALT_PRESS]n[ALT_RELEASE]"
   Type "[ENTER]"
   Check Servidor:
   Type "[ESCAPE][ESCAPE][ESCAPE]"
   Timeout "5"
   Type "[ESCAPE][ESCAPE][ESCAPE]"

End
```

Figura 11: Script - Verifica o tempo de login

Fonte: Autoria Própria

Neste *script* é utilizado o comando ”IfNot”para verificar a não existência de uma determinada imagem, e em seguida executar uma ação com o teclado. O comando ”StepDelay”é utilizado com o objetivo de forçar uma espera antes do próximo comando ser executado. Os outros *scripts* utilizam basicamente os mesmos comandos, foram realizadas poucas variações.

Após criado cada um dos doze *scripts*, foram realizados diversos testes para que não fossem enviados *scripts* com erro para serem executados em clientes. Para cada *script* foi verificado seu comportamento em cenários de falha e cenários de sucesso, o objetivo foi assegurar que quando tivesse que falhar o mesmo falharia, e quando não tivesse que falhar o mesmo não falharia.

Desta forma, nenhum *script* apresentou falsos negativos ou falsos positivos durante os testes. Tais testes foram realizados em mais de uma máquina, e seus resultados foram documentados em um quadro, o qual é apresentado abaixo.

Quadro 3: Testes realizados nos scripts.

NFRs	Categoria	Falha	Sucesso
Versão Windows	Portabilidade	Ok	Ok
Versão IE	Portabilidade	Ok	Ok
Framework	Portabilidade	Ok	Ok
Abertura Sistema	Desempenho	Ok	Ok
Logar Sistema	Desempenho	Ok	Ok
Listagem Clientes	Desempenho	Ok	Ok
Cadastro Clientes	Desempenho	Ok	Ok
Listagem Vendas	Desempenho	Ok	Ok
Cadastro Vendas	Desempenho	Ok	Ok
Calendário	Desempenho	Ok	Ok
Logoff	Desempenho	Ok	Ok

Fonte: Autoria própria.

4.2.3 APLICAÇÃO EM IMPLANTAÇÃO

Ao concluir a criação dos *scripts* e a realização dos testes, deu-se início a pesquisa de campo que foi realizada junto aos implantadores. O primeiro passo foi definir quais computadores poderiam ser utilizados no estudo. Foi definido que seriam utilizados computadores de testadores, implantadores e clientes.

No total foram selecionados dezesseis computadores, sendo três de testadores, outros

três de implantadores e dez de clientes. Desta forma os cenários onde os *scripts* deveriam ser executados foram bem variados.

A execução dos *scripts* foi realizada por três implantadores, sob a supervisão do autor deste estudo, a cada execução realizada eram marcados os resultados obtidos. Os pontos que foram averiguados durante a execução, estão listados abaixo:

- Tempo consumido pela ferramenta para a execução dos *scripts*;
- Tempo consumido para realizar manualmente a mesma checagem;
- Resultado do teste, verificar se a checagem automatizada apresentou falhas.

Os resultados obtidos junto com todas as informações relevantes, que foram percebidas durante tal execução, foram anotados em uma planilha, para que esses dados pudessem ser utilizados ao fim do estudo. O Quadro 4 apresenta a planilha mencionada.

Quadro 4: Planilha de execução dos scripts.

		Windows	Framework	IE	Abertura	Login	L Clientes	C Clientes	L Vendas	C Vendas	Calendário	Logoff	Tempo Scripts	Tempo Manual
	C1	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:24	00:02:37
	C2	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:29	00:02:03
	C3	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:03:04	00:02:20
	C1	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:45	00:01:56
	C2	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:53	00:01:45
	C3	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:35	00:02:06
	C1	----------	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:41	00:02:13
	C2	Passou	Passou	Passou	----------	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:03	00:01:31
	C3	Passou	Passou	Passou	Passou	Passou	----------	Passou	Passou	Passou	Passou	Passou	00:03:08	00:02:09
	C4	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:18	00:02:04
	C5	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:31	00:01:34
	C6	Passou	Passou	----------	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:25	00:01:51
	C7	----------	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:56	00:01:48
	C8	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:45	00:02:05
	C9	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	00:02:58	00:01:40
	C10	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	Passou	--------	00:03:05	00:02:12

Fonte: Autoria própria.

5 RESULTADOS

5.1 NÚMEROS E DADOS

O primeiro questionário foi aplicado com o intuito de mensurar o quão familiarizados com o assunto deste estudo, estão os profissionais ligados a teste, implantação ou desenvolvimento. Nesta Seção serão apresentados os resultados deste primeiro questionário.

Como pode ser visto na Figura 12, um total de treze desenvolvedores, nove testadores e nove implantadores, participaram do questionário. No total foram atingidos pelo questionário trinta e um profissionais de cinco empresas diferentes.

Figura 12: Questionário 1 - Pergunta 1

Fonte: Autoria Própria

O questionário atingiu desde profissionais com pouca experiência, até profissionais com experiência consideravelmente grande, e em proporções quase iguais, como pode ser visto na Figura 13 abaixo.

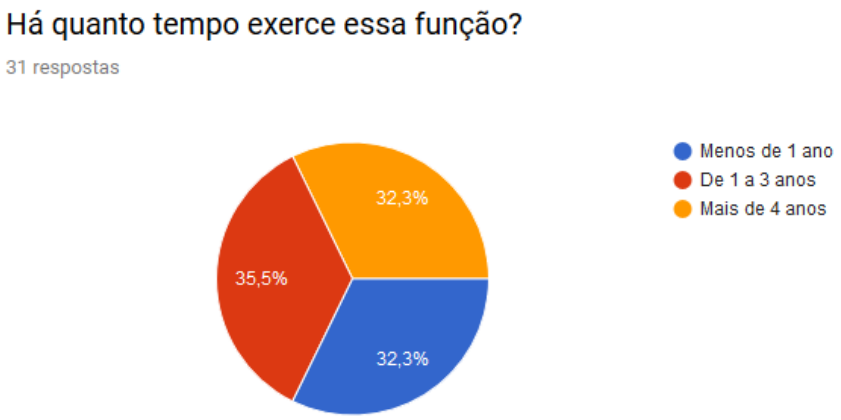

Figura 13: Questionário 1 - Pergunta 2

Fonte: Autoria Própria

Outro ponto avaliado foi se o profissional questionado possuía experiência com validação e/ou teste de software, a maioria dos questionados disse possuir experiência com estas atividades.

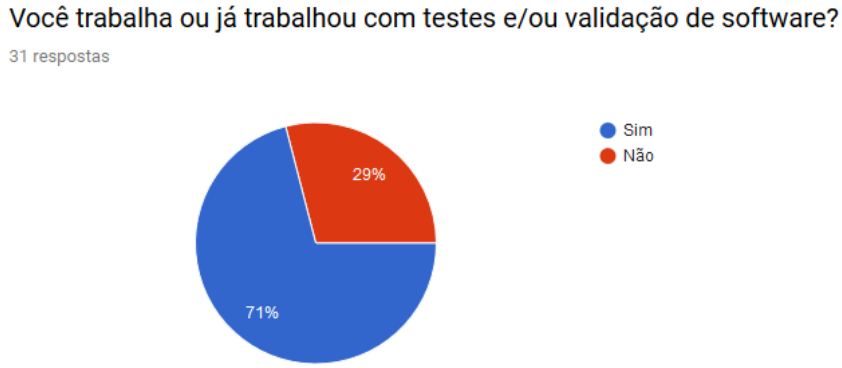

Figura 14: Questionário 1 - Pergunta 3

Fonte: Autoria Própria

A Figura 15 apresenta uma quarta pergunta, onde os participantes foram questionados a respeito de seu conhecimento sobre o termo ”Ferramenta VGT”. Por meio dessas duas últimas questões, pode ser percebido que setenta e um por cento dos questionados dizem já ter trabalho com validação e/ou teste de software, porém aproximadamente quarenta e dois por cento dizem conhecer o termo ”Ferramenta VGT”.

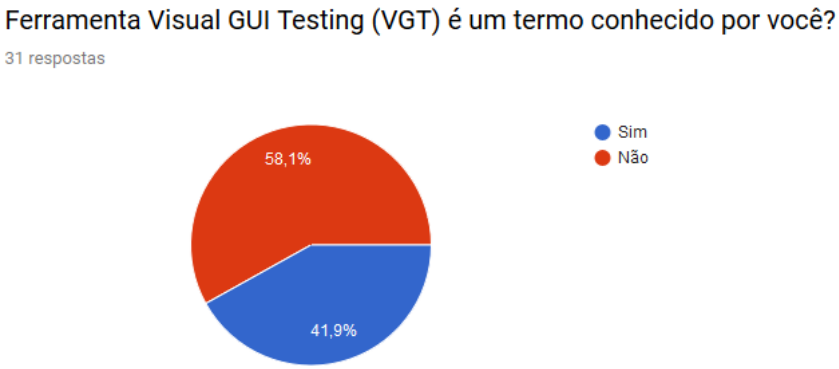

Figura 15: Questionário 1 - Pergunta 4

Fonte: Autoria Própria

Após solicitado a respeito do conhecimento sobre o termo ”Ferramenta VGT”, foi solicitado quais ferramentas VGT o questionado já havia utilizado. A Figura 16 apresenta tal questão, a pergunta era aberta e alguns participantes incluíram algumas ferramentas. Vale ressaltar que trinta e cinco por cento dos questionados, responderam nunca ter utilizado nenhuma das ferramentas listadas.

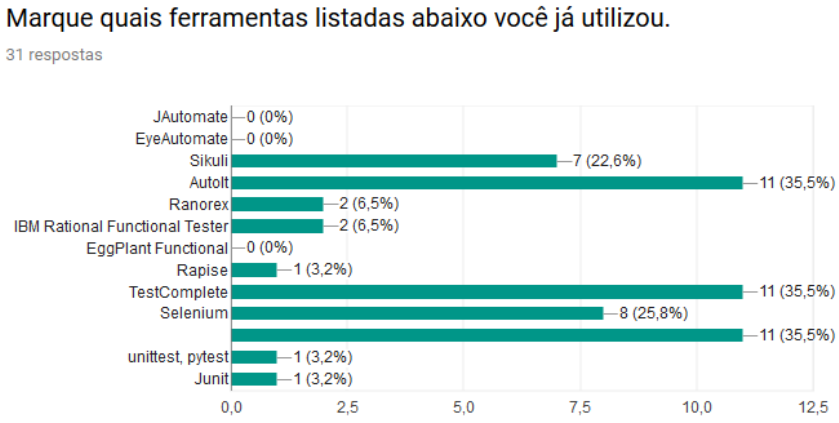

Figura 16: Questionário 1 - Pergunta 5

Fonte: Autoria Própria

A Figura 16 apresenta outro ponto relevante, dos trinta e um questionados, nenhum respondeu já ter utilizado a ferramenta EyeAutomate, o que deixa claro que a ferramenta é pouco conhecida na região onde o estudo foi realizado.

Os participantes também foram questionados a respeito do seu conhecimento sobre requisitos não funcionais, onde cerca de setenta e sete por cento diz já ter trabalhado ou pelo menos conhecer NFRs. Tal questão é exibida na Figura 17.

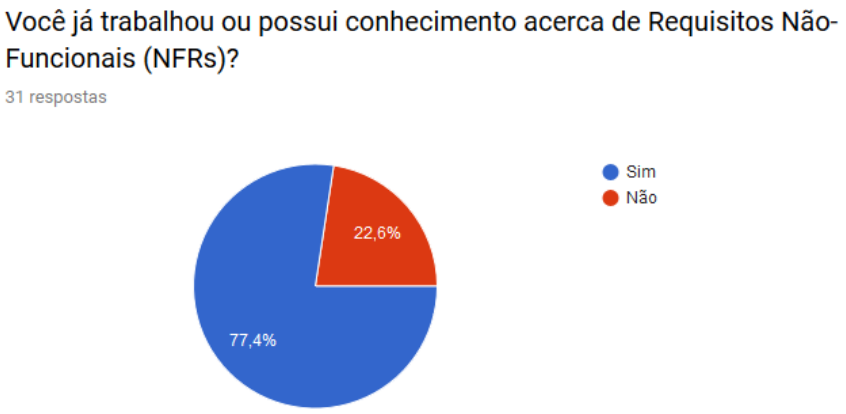

Figura 17: Questionário 1 - Pergunta 6

Fonte: Autoria Própria

Outra pergunta foi elaborada para saber se na opinião dos participantes, uma ferramenta VGT poderia auxiliar na verificação de NFRs. Como pode ser visto na Figura 18, cerca de setenta e um por cento dos questionados, concordam com o que foi perguntado.

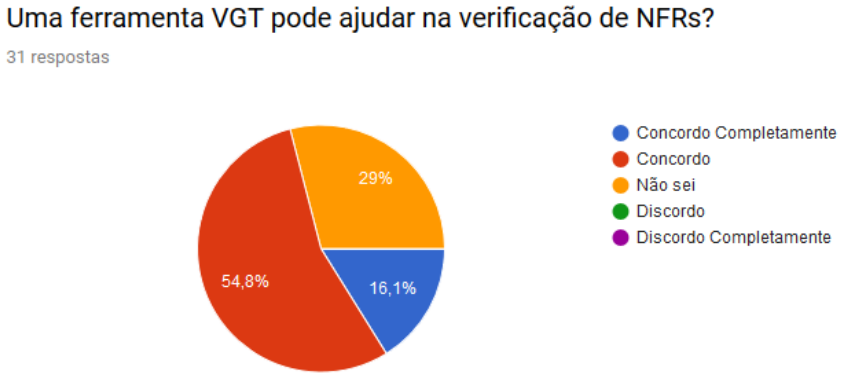

Figura 18: Questionário 1 - Pergunta 7

Fonte: Autoria Própria

Os participantes também foram questionados sobre como eles definem a verificação manual de NFRs. A Figura 19 mostra que para a maioria a verificação manual é propensa a erros e desgastante.

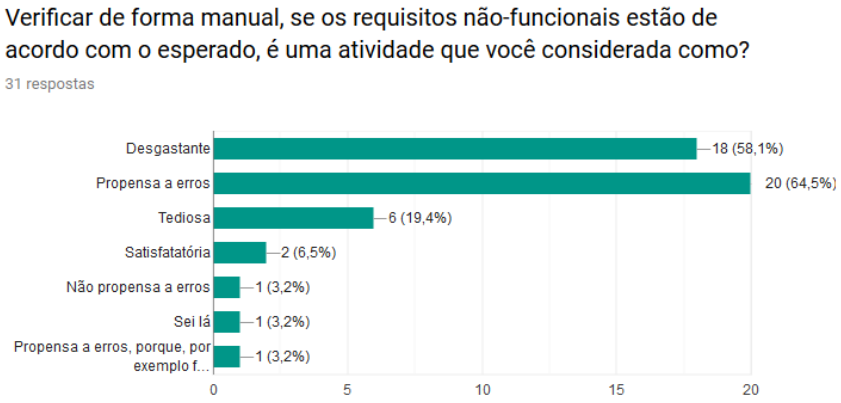

Figura 19: Questionário 1 - Pergunta 8

Fonte: Autoria Própria

A nona pergunta foi criada com o objetivo de saber quais qualidades de uma ferramenta VGT, os participantes consideram mais importantes. A Figura 20 apresenta que a qualidade mais esperada é confiabilidade. Interface intuitiva junto com histórico de resultados, aparecem logo em seguida com aproximadamente sessenta e quatro por cento. Na terceira posição aparecem portabilidade e rapidez com aproximadamente cinquenta e oito por cento.

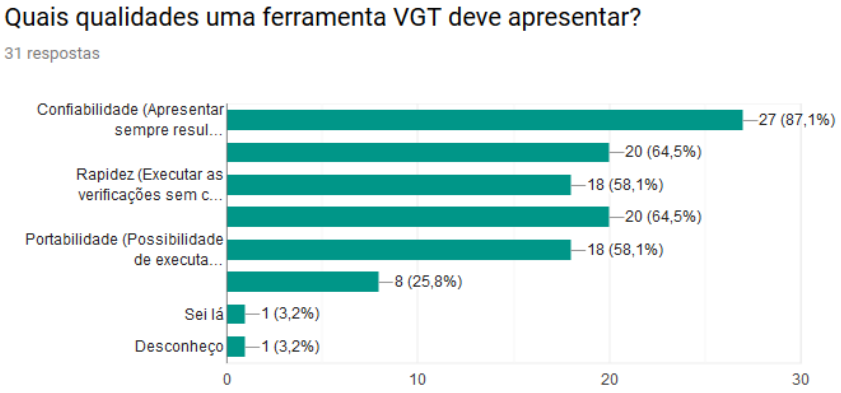

Figura 20: Questionário 1 - Pergunta 9

Fonte: Autoria Própria

De uma forma geral, este questionário foi criado com objetivo de mensurar, o conhecimento de profissionais acerca do tema deste estudo. Esse questionário apresentou resultados que demonstram que dentro da indústria de software, a maioria dos profissionais acreditam no potencial de ferramentas VGT, na realização de checagens automatizadas de NFRs.

5.2 RESPOSTAS ÀS QUESTÕES DE PESQUISA

Nesta Seção serão respondidas as questões de pesquisa levantadas no começo do estudo, as respostas foram obtidas durante a realização do estudo de caso, por meio de coletas de dados, os quais serão explanados a seguir.

Q1: Identificar e especificar NFRs passíveis de serem verificados através de ferramentas VGT

Conforme já introduzido no decorrer do estudo, ferramentas VGT utilizam o reconhecimento de imagem combinado com *scripts*, para realizar checagens na interface gráfica de um sistema. Desta forma alguns NFRs não são passíveis de verificação, através do uso de tais ferramentas.

Sabendo dessa limitação foram identificados e especificados os requisitos não funcionais que a ferramenta deveria checar. Levando em conta a demanda dos implantadores e as limitações da ferramenta, os NFRs escolhidos estavam relacionados a portabilidade e desempenho.

Entretanto, para averiguar se as limitações da ferramenta não prejudicaram na identificação e especificação dos NFRs, foi solicitado aos implantadores se os NFRs verificados atendiam a atual demanda e, se as limitações que impedem que alguns NFRs sejam checados foi prejudicial para as verificações.

As duas questões foram respondidas e por unanimidade foi constatado que para o cenário onde a ferramenta foi aplicada, tais limitações não apresentaram prejuízo algum à equipe de implantação e que a atual demanda foi atendida. Assim a ferramenta conseguiu atender a demanda existente de verificações e a identificação dos NFRs foi realizada sem problemas.

Para esta questão de pesquisa pode ser concluído que não são todos os NFRs que podem ser verificados por tais ferramentas e, que isso deve ser levado em conta durante a identificação dos NFRs. Não foram encontradas maneiras de checar alguns NFRs como os ligados a usabilidade, entretanto, isso não causou prejuízos para as checagens.

Q2: Produzir/Prototipar *scripts* para verificação dos NFRs

A criação dos *scripts* foi realizada pelo autor do estudo, o mesmo não possuía experiência de uso com a ferramenta escolhida. Para serem criados os *scripts*, foi necessário ler a documentação da ferramenta, que possui manuais e tutoriais explicando o uso de cada comando. A documentação da ferramenta está disponível em seu site oficial.

Essa questão de pesquisa buscou validar se a produção dos *scripts*, da forma como foi feita, foi eficaz e conseguiu checar os NFRs selecionados. Para que essa questão fosse respondida foram incluídas duas perguntas no segundo questionário.

A primeira pergunta questiona se o tempo de execução dos *scripts* é curto, ou se deveria ser reduzido. Dois dos três implantadores questionados responderam que o tempo levado deveria ser reduzido.

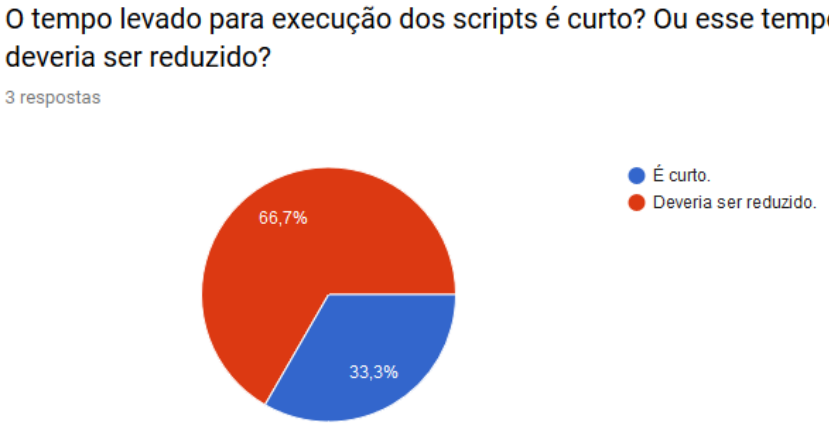

Figura 21: QP2 - Pergunta 1

Fonte: Autoria Própria

A segunda pergunta está relacionada a simplicidade de execução dos *scripts* e, se os mesmos apresentaram o resultado de forma clara. Todos os participantes responderam que a execução é simples e que os resultados são apresentados de forma clara.

Por tanto, conclui-se que o tempo de execução dos *scripts* criados foi pouco aceitável, e que a execução destes é simples assim como a apresentação dos resultados é feita de forma clara. A produção dos *scripts* demonstrou não ter sido tão eficaz, devido ao tempo levado para a execução dos mesmos.

Q3: Verificar a robustez dos *scripts* VGT para verificação dos NFRs

Como já mencionado anteriormente, os *scripts* foram testados exaustivamente para que não ocorresse nenhum erro ao executá-los no ambiente de produção. Para medir a robustez e responder essa questão de pesquisa, foram incluídas duas perguntas no segundo questionário e também foram levantados dados durante a execução dos *scripts*.

A primeira pergunta questionou aos implantadores, se durante a execução dos *scripts* foram encontrados problemas. Conforme é apresentado na Figura 22, dois dos três implantadores respondem que a ferramenta se saiu bem, mas que alguns pontos esperados não

foram atendidos.

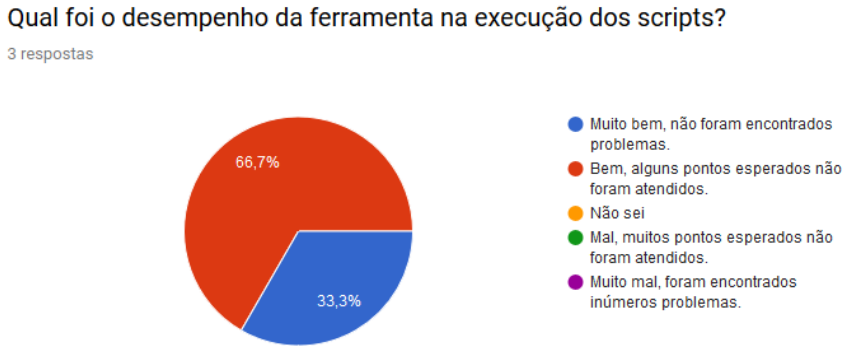

Figura 22: QP3 - Pergunta 1

Fonte: Autoria Própria

A segunda pergunta questiona se o uso da ferramenta é seguro, ou se a ferramenta passou alguma insegurança. A Figura 23 apresenta que dois dos três participantes responderam que a ferramenta é segura, e que um participante respondeu que a ferramenta é parcialmente segura.

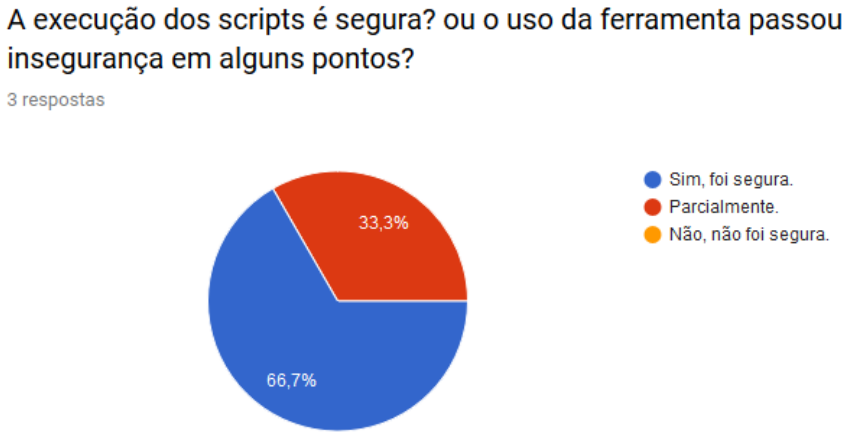

Figura 23: QP3 - Pergunta 2

Fonte: Autoria Própria

A robustez foi levada em consideração durante a execução dos onze *scripts* nos dezesseis computadores, durante a execução todos os resultados foram conferidos, a fim de encontrar falhas neste processo. Desta forma foram encontradas seis falhas em verificações, dessas seis falhas quatro foram originadas pela ferramenta, e duas ocorreram por existir algum defeito na implementação do *script*.

As duas falhas originadas por defeitos existentes na implementação dos *scripts*, estão ligadas a particularidades presentes nos computadores onde os *scripts* foram executados, tais particularidades não foram detectadas no ambiente de teste. A primeira falha ocorreu devido as versões do Windows Server, não estarem sendo verificadas pelo *script*. E a outra falha ocorreu devido ao idioma do computador checado estar definido como inglês, por este motivo as teclas de atalho do navegador Internet Explorer foram alteradas, o que ocasionou falha na verificação.

Já a primeira falha originada pela ferramenta está relacionada com o *script* que checa a tela de abertura do software, aparentemente devido aos comandos terem sido executados fora de tempo, a ferramenta acabou fechando indevidamente a tela de abertura o que acarretou em falha ao checar tal tela. Porém ao executar a suíte de testes novamente a falha não voltou a se manifestar.

A segunda falha ocasionada pela ferramenta está relacionada ao *script* que checa a listagem de clientes, por algum motivo o *script* não conseguiu reconhecer o cabeçalho da tela, entretanto, neste caso novamente ao rodar o *script* pela segunda vez a falha não voltou a ocorrer.

A terceira falha originada pela ferramenta ocorreu com o *script* que verifica a versão do Windows, neste caso se tratava de um computador com Windows 7, onde o *script* não conseguia reconhecer a versão aparentemente devido a uma resolução diferente estar configurada no computador. Neste caso o *script* foi executado várias vezes e a versão não foi reconhecida nenhuma vez.

A quarta falha ocasionada pela ferramenta, ocorreu com o *script* responsável por realizar o *logoff* do sistema, a falha ocorreu devido a ferramenta não ter fechado todas as telas necessárias para a realização do *logoff*. Certamente neste caso também o fato ocorreu devido aos comandos terém sido executados fora do tempo estabelecido, esta falha não ocorreu novamente após o *script* ser executado pela segunda vez.

No que diz respeito as falhas originadas pela própria ferramenta, na maioria das vezes ao executar a suíte de testes novamente a falha não voltava a ocorrer. Isso acaba passando uma certa insegurança a quem presencia o ocorrido, entretanto, também deixa claro que os testes realizados pela ferramenta devem ser validados manualmente, caso a equipe pretenda que nenhuma falha passe despercebida.

Percebeu-se também que durante a confecção dos *scripts*, que em alguns poucos casos algumas imagens eram reconhecidas apenas na primeira execução, o que forçava o desenvolvedor do *script* a utilizar um recorte diferente para imagem. Outro ponto percebido durante esta fase, foi ao trabalhar com textos, onde a ferramenta não conseguia reconhecer

corretamente a maioria dos trechos de textos, neste caso a ferramenta apresentava resultado positivo ao encontrar um trecho de texto parecido e/ou com a mesma fonte.

Levando em consideração que onze *scripts* foram executados em cada computador para realizar todas as checagens, ao total temos cento e setenta e seis execuções. Dessas 176 execuções em apenas quatro ocorreram falhas originadas por defeitos da ferramenta, o que representa aproximadamente dois por cento (2,27%) que pode ser considerado um número baixo de falhas.

Através dos dados dispostos acima, pode ser concluído que os *scripts* não são robustos a ponto de não ocorrer nenhuma falha durante as execuções, mas o número de falhas encontradas ficou próximo de zero. Após a criação dos *scripts* em meio a inúmeros testes, a quantidade de falhas apresentadas durante as execuções no ambiente de produção foi de 2,27%.

Q4: Determinar a viabilidade da execução dos *scripts* no ambiente de implantação

A quarta questão de pesquisa buscou avaliar se a execução de *scripts* através do uso de ferramentas VGT, era uma pratica viável para os implantadores. Os implantadores responderam duas perguntas relacionadas a essa questão de pesquisa.

A primeira pergunta questiona se os *scripts* podem ser utilizados durante implantações, no dia a dia do implantador. Dois dos três implantadores, responderam que sim, conforme pode ser visto na Figura 24.

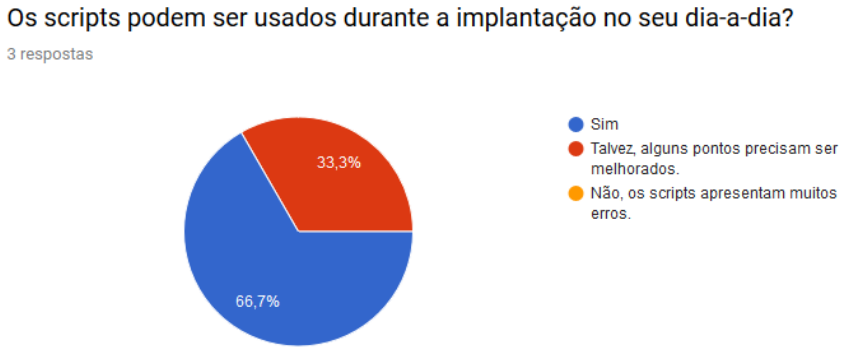

Figura 24: QP4 - Pergunta 1

Fonte: Autoria Própria

A segunda pergunta questionou a respeito da possibilidade de substituir as checagens manuais, realizadas diariamente, pela checagem automatizada. Todos os implantadores responderam que a checagem automatizada pode substituir a checagem manual.

Após realizada a criação dos *scripts* e realizado todos os testes, foi visto com os

implantadores que a execução dos *scripts* inicialmente demandava muito trabalho, pois era necessário instalar a ferramenta na máquina para posteriormente importar os *scripts* e em seguida conseguir realizar as checagens.

Conseguinte a isso foi procurada uma solução para este problema. Através de consultas realizadas no manual da ferramenta, foi verificado que os *scripts* poderiam ser executados através do *Windows Command Prompt* (CMD).

Desta forma, foi gerado um único arquivo instalador que instalava a ferramenta com os *scripts* e gerava um atalho na área de trabalho do usuário, ao executar este atalho, através de um arquivo com extensão .bat eram executados todos os *scripts* por meio do CMD. Assim a execução dos *scripts* deixou de ser trabalhosa e foi simplificada.

Para esta questão de pesquisa pode ser concluído que a simplificação encontrada para a execução dos *scripts*, ajudou a determinar a viabilidade de execução dos mesmos, tornando a execução possível de ser utilizada no dia a dia dos implantadores, conforme asseguram as respostas obtidas através das perguntas.

Q5: Averiguar a opinião de profissionais dentro da indústria a respeito do uso de VGT

A quinta questão de pesquisa objetivou coletar as impressões que os implantadores tiveram após terem utilizado a ferramenta VGT. Essa questão de pesquisa também foi respondida por meio de duas perguntas realizadas aos implantadores.

A primeira pergunta questionou se os implantadores recomendariam a ferramenta para realizar a verificação de NFRs, todos os implantadores responderam que recomendariam.

A outra pergunta realizada questionou se o uso da ferramenta satisfez as necessidades relacionadas a verificação de NFRs no dia a dia, todos os implantadores responderam também que sim.

Desta forma, é concluído através dessa questão de pesquisa, que mesmo com algumas limitações apresentadas pela ferramenta, o uso da mesma é benéfico e pode ser recomendado. É concluído também que a opinião dos profissionais de dentro da indústria é positiva em relação ao uso de ferramentas VGT.

Q6: Fazer uma análise de *trade-offs* entre o uso de VGT para checagem versus checagem manual

Para realizar a comparação entre a checagem manual e a checagem automatizada, os implantadores foram questionados a respeito de suas opiniões e também foram levantados dados

durante a execução dos *scripts*.

A primeira pergunta questionada aos implantadores, foi se de um modo geral a checagem automatizada se saiu melhor que a checagem manual, todos os participantes responderam que sim, que a checagem automatizada se saiu melhor.

A segunda pergunta questionou se as checagens manuais são mais desgastantes e propensas a erros do que as checagens automatizadas. Dois dos três implantadores responderam que sim, conforme pode ser visto na Figura 25.

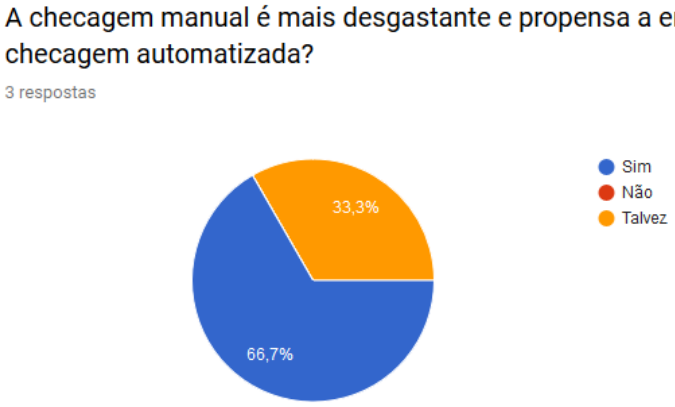

Figura 25: QP6 - Pergunta 2

Fonte: Autoria Própria

Como pode ser percebido, de um modo geral, os implantadores concordam que a checagem automatizada é preferível em relação a checagem manual. Entretanto, os dados coletados durante a execução dos *scripts*, revelaram que a checagem automatizada levou mais tempo para ser executada do que a checagem manual.

Quadro 5: Tempo entre checagem manual e checagem automatizada.

Tempo Scripts	Tempo Manual
00:02:24	00:02:37
00:02:29	00:02:03
00:03:04	00:02:20
00:02:45	00:01:56
00:02:53	00:01:45
00:02:35	00:02:06
00:02:41	00:02:13
00:02:03	00:01:31
00:03:08	00:02:09
00:02:18	00:02:04
00:02:31	00:01:34
00:02:25	00:01:51
00:02:56	00:01:48
00:02:45	00:02:05
00:02:58	00:01:40
00:03:05	00:02:12
00:43:00	00:31:54
00:02:41	00:02:00

Fonte: Autoria própria.

O Quadro 5 mostra que somando o tempo levado nos dezesseis computadores, a média de tempo de execução dos *scripts* é de dois minutos e quarenta e um segundos, já o tempo médio levado para realizar a checagem manual foi de dois minutos.

O tempo de checagem automatizada ficou superior devido ao *script* de verificação da versão do Internet Explorer, para realizar esta verificação foram encontradas algumas dificuldades, e não foi conseguido reduzir o seu tempo de verificação. Este *script* consumiu um tempo médio de aproximadamente cinquenta segundos em cada execução.

Para fazer a checagem automatizada da versão do navegador Internet Explorer, era necessário aguardar o mesmo carregar por completo, porque em alguns casos se não fosse esperado e em seguida fosse executado comandos, o mesmo travava e fechava automaticamente. Então foi devido a este problema que o *script* ficou lento.

Entretanto o tempo consumido para a execução do teste, não é o fator mais levado em conta em ferramentas VGT, como foi percebido na resposta da nona pergunta do primeiro questionário (Figura 20), onde os participantes responderam que o fator mais importante é a confiabilidade passada pela ferramenta.

5.3 AMEAÇAS À VALIDADE

A validade de um experimento está ligada ao quão confiável são as conclusões apresentadas no mesmo, por este motivo, é fundamental realizar uma avaliação para responder o quão válidos são os resultados do experimento. Por se tratar de uma pesquisa quantitativa, relacionada a engenharia de software, será considerada a validade interna, externa, de construção e de conclusão (TRAVASSOS et al., 2002; FELDT; MAGAZINIUS, 2010).

As ameaças à validade apresentadas abaixo, são as mesmas presentes no estudo de caso incluído como apêndice. São apresentadas as mesmas ameaças a validade devido a estas estarem relacionadas a um mesmo experimento, que é abordado nos dois documentos.

A validade interna define se o relacionamento observado entre o tratamento e o resultado é causal, e não é o resultado da influência de outro fato que não é controlado ou não foi medido (TRAVASSOS et al., 2002). Todos os participantes do segundo questionário participaram também do estudo, o que evitou desentendimentos relacionados a termos ou conceitos, desta forma foram minimizadas qualquer ameaça à validade interna. Já no primeiro questionário, que participaram profissionais não ligados diretamente ao estudo, foi introduzido os conceitos de cada termo utilizado no questionário, para evitar também ameaças à validade interna. Para os dois questionários o número de perguntas não foi elevado, aproximadamente 15 para cada, assim evitando também que os participantes ficassem cansados ou desanimados. Todos os participantes são profissionais que trabalham dentro do ciclo de produção de software.

A validade externa define as condições que limitam a habilidade de generalizar os resultados de um experimento para a prática industrial (TRAVASSOS et al., 2002). Durante a execução dos *scripts* e a aplicação dos questionários, foi levado em conta a disponibilidade dos participantes, para que não fosse realizada nenhuma atividade em dias que os participantes estivessem atarefados. Desta forma evitando que fatores externos prejudicassem os resultados.

A validade de construção considera os relacionamentos entre a teoria e a observação, ou seja, se o tratamento reflete a causa bem e o resultado reflete o efeito bem (TRAVASSOS et al., 2002). Durante o estudo procurou-se ser o mais o abrangente possível, o fato de que apenas duas categorias de NFRs terem sido utilizadas, pode apresentar determinada ameaça a validade de construção, porém os NFRs utilizados estão ligados a demanda da equipe que participou do estudo.

A validade de conclusão está relacionada a obtenção de uma conclusão correta a respeito dos relacionamentos entre o tratamento e o resultado do experimento (TRAVASSOS et al., 2002). O segundo questionário, foi aplicado a um número baixo de participantes, isso

se deve ao fato de que, o estudo foi realizado dentro de apenas uma empresa. O baixo número de participantes neste questionário, apresenta uma certa ameaça a validade para o experimento, porém o estudo já apresenta avaliações iniciais da aplicabilidade da ferramenta dentro de uma empresa de software.

5.4 DISCUSSÕES

O primeiro questionário apresentou alguns resultados importantes, como o fato de que nenhum dos entrevistados disse já ter utilizado a ferramenta VGT utilizada neste trabalho. O que havia sido percebido através de pesquisas, onde foram encontrados poucos estudos que utilizassem a ferramenta, foi percebido também dentro da indústria.

Outro resultado que pôde ser percebido, foi que 71% dos participantes disse já ter trabalhado com testes e/ou validação de software, e 41,9% dos participantes disse já conhecer o termo ”Ferramentas VGT”. O que demonstra que nem todos os participantes com experiência em teste e/ou validações, possui conhecimento sobre ferramentas VGT, entretanto, a quantidade de participantes que conhecem tal ferramenta não pode ser considerado baixa.

O primeiro questionário também mostrou que as ferramentas AutoIt e TestComplete, são as ferramentas mais utilizadas pelos participantes. Alguns participantes também relataram nunca ter utilizado nenhuma ferramenta VGT, a quantidade destes corresponde a 35,5% dos participantes.

Outro resultado levantado é que 71% dos participantes, diz concordar que uma ferramenta VGT pode ajudar na verificação de NFRs, e nenhum dos participantes diz discordar. Isso demonstra que o tema deste estudo não é visto como algo utópico dentro da indústria.

Ao questionar os participantes a respeito de quais qualidades uma ferramenta VGT deve possuir, a maioria 87,1% dos participantes disse ser confiabilidade. Para os participantes, resultados precisos são mais importantes que o tempo utilizado pela ferramenta e outros pontos.

A execução dos *scripts* e as respostas do segundo questionário, trouxeram mais resultados ao estudo. Pôde ser visto que ocorreu na prática um caso onde a checagem manual foi mais eficaz que a checagem automatizada, tal fato ocorreu com a verificação da versão do Internet Explorer. A checagem automatizada poderia ter sido mais eficaz, porém não foi encontrado um método para isso, o que deixa perceptível também, que a forma como o *script* é implementado, pode influenciar nos resultados da execução.

Ou seja, outro ponto percebido durante o estudo, é que a forma como o *script* é

implementado, afetará no tempo de execução e nos resultados que serão obtidos durante o teste. Isso demonstra que a implementação de *scripts*, deve ser realizado por alguém que domine, de certa forma, a ferramenta.

A ferramenta VGT utilizada também demonstrou precisar evoluir seu algoritmo de reconhecimento de imagem, por mais que apenas dois por cento dos testes executados, tenham apresentados falhas decorrentes da ferramenta, este é o fator mais levado em conta pelos profissionais, a ferramenta não deve apresentar falhas.

Por fim, foram coletados *feedbacks* com os implantadores, através do segundo questionário, um dos implantadores respondeu que a ferramenta poderia exibir na tela, qual teste está sendo realizado, desta forma o implantador conseguiria mensurar a quantidade de tempo faltante para o término da execução, e conseguiria entender com clareza o que estaria sendo feito naquele momento. Este é um ponto que também pode ser levado em consideração pelas empresas que desenvolvem tais ferramentas.

6 TRABALHOS RELACIONADOS

Com intuito de obter embasamento teórico, durante a realização deste trabalho, foram realizadas diversas buscas por trabalhos relacionados a este. Foi possível notar que existem poucos estudos relacionados a ferramenta EyeAutomate, e que a maioria destes estudos são de universidades presentes na região escandinava. Através destas buscas, não foi localizado nenhum trabalho realizado por brasileiros, que fosse relacionado especificamente a ferramenta EyeAutomate.

As buscas foram realizadas em diversos repositórios, a procura foi por trabalhos que fizessem o uso de ferramentas VGT, para a realização de automatização de testes. Os trabalhos relacionados, que foram encontrados durantes as buscas, serão apresentados a seguir.

Haar e Michaëlsson (2018) realizaram uma tese de mestrado, onde comparam duas ferramentas VGT, especificamente Selenium e EyeAutomate, com o objetivo de conhecer o custo de manutenção destas ferramentas. Este estudo concluiu que o uso de ferramentas VGT, é benéfico desde que o produto não esteja passando por constantes alterações gráficas. O resultado da comparação entre as ferramentas, foi que os *scripts* criados através da ferramenta Selenium, demoraram cerca de 91% mais tempo, mas tiveram um custo de manutenção 32% menor do que EyeAutomate.

Borjesson e Feldt (2012) fazem um estudo que compara duas ferramentas VGT, sendo utilizadas dentro da indústria. O estudo concluiu que ambas as ferramentas funcionaram igualmente bem no contexto industrial e, também foi concluído que o tempo consumido pelo teste automatizado, foi 78% menor que o tempo consumido pelo teste manual.

Alégroth et al. (2013) desenvolveram uma pesquisa com profissionais da indústria, para identificar quais problemas relacionados a teste, poderiam ser resolvidos através do uso de ferramentas VGT. Foi concluído que a indústria possui interesse na utilização de ferramentas VGT, e que estas podem resolver muitos problemas existentes.

Alégroth et al. (2018) realizaram um estudo dentro da indústria, focado na avaliação da aplicabilidade, de ferramentas VGT para testes automatizados, em um ambiente de integração

contínua. O estudo concluiu que tais ferramentas são aplicáveis na indústria, e fornecem benefícios semelhantes a outras técnicas de automatização. No entanto, o estudo também identifica que a técnica possui vários desafios, como tempo de execução dos *scripts* ser longo, alto custo de manutenção e, alto custo para análise de resultados falsos positivos.

Alégroth et al. (2015a) através de um estudo realizado com dois projetos industriais, onde profissionais usaram ferramentas VGT, para automatizar cerca de 300 casos de teste, concluem que os profissionais consideraram eficaz, o uso de ferramentas VGT para encontrar defeitos, apesar dos desafios identificados.

Alégroth et al. (2013) realizaram um estudo de caso dentro da indústria, onde objetivavam fazer a transição de testes manuais, para testes automatizados utilizando VGT. O estudo foi realizado dentro da empresa Saab AB, e a transição foi bem sucedida, foram encontradas limitações e problemas com o uso de VGT, mas seu uso demonstrou-se benéfico e viável dentro da indústria.

Liebel et al. (2013) apresentam um estudo de caso, com objetivo de investigar o estado da prática de utilização de ferramentas VGT, para testes de aceitação em seis empresas de desenvolvimento de software. A principal descoberta é que o uso de VGT para automatizar testes de aceitação, existe apenas em pequena escala. Além disso, o estudo identifica os principais problemas com VGT e os testes de aceitação, tais como as limitações da ferramenta, os altos custos dos testes e o envolvimento do cliente nos testes.

Sjöblom e Strandberg (2015) através de uma tese de mestrado, investigam a capacidade de ferramentas VGT na realização de testes de regressão, dentro da empresa Saab AB. Nesta tese também foi realizado uma comparação entre duas ferramentas VGT, onde ambas apresentaram uma boa precisão e desempenho geral satisfatório. O estudo concluiu que a utilização de tais ferramentas é muito promissora, mas que possuem limitações e problemas relacionados ao reconhecimento de imagem.

Garousi et al. (2017) motivados por uma necessidade real de uma grande empresa turca, avaliam o uso de duas ferramentas VGT para decidir qual utilizar em um projeto de testes. Neste estudo perceberam que ambas ferramentas, apresentaram incapacidade de encontrar imagens em tamanho menor do que o especificado. O estudo comparou as ferramentas JAutomate e Sikuli, e foi concluído que a ferramenta JAutomate demonstrou-se um pouco melhor para as necessidades da empresa.

A grande maioria dos trabalhos citados acima, estão relacionados ao uso de ferramentas VGT dentro da indústria, e possuem o objetivo de comparar o desempenho de duas

ou mais ferramentas. Durante as pesquisas por trabalhos relacionados, notou-se também, que a Universidade Chalmers possui muitos trabalhos relacionados ao tema, e que Emil Alégroth e Robert Feldt estão relacionados a muitos destes estudos.

Os trabalhos relacionados que foram coletados e apresentados neste Capítulo, estão relacionados ao uso de ferramentas VGT para automatizar testes. Não foram encontrados trabalhos, onde ferramentas VGT tenham sido utilizadas especificamente, com o objetivo de automatizar testes de requisitos não funcionais.

7 CONCLUSÕES

A realização deste trabalho foi motivada por uma necessidade real da indústria, neste mostramos que ferramentas VGT podem ser utilizadas para automatizar checagens de requisitos não funcionais. Entretanto, foram encontrados alguns problemas durante a experiência com uso da ferramenta.

O estudo analisou a possibilidade da aplicação de ferramentas VGT, para realizar checagens de NFRs durante a implantação de um software. Os resultados foram obtidos por meio da execução de atividades dentro da indústria.

O estudo constata que ocorreram falhas durante a execução dos *scripts*, que foram originadas por defeitos existente na ferramenta, também foram percebidas falhas com reconhecimento de imagem e com tempo de execução dos comandos. Entretanto, o trabalho constata também que é baixo o número de falhas ocorridas, desta forma, este não é um motivo para descartar o uso de tais ferramentas.

A avaliação de robustez mostrou que apenas 2,27% das execuções realizadas, terminou com falha ocasionada por defeitos na ferramenta, um número que pode ser considerado baixo, se levado em conta que testes manuais também estão suscetíveis a falhas. A comparação entre o desempenho de checagem manual e automatizada, mostrou que a forma como o *script* é implementado, pode aumentar o tempo consumido pelo mesmo durante a sua execução e, pode influenciar no resultado da checagem.

Levando em consideração que a maioria dos sistemas possuem interface gráfica, e que ela pode ser a chave para o sucesso do mesmo, o teste através de reconhecimento de imagem, é algo que tende a continuar crescendo, e deste modo as ferramentas VGT possuem potencial para solucionar todos os atuais problemas encontrados com seu uso.

A ferramenta também proporcionou benefícios durante a execução das verificações, um deles é que a checagem automatizada dispensou o uso de *checklists* e outras anotações que seriam realizadas pelo implantador. Outro ponto benéfico é que as checagens podem ser realizadas por qualquer pessoa, inclusive pelo cliente, não sendo necessário conhecimento a

respeito do que está sendo checado.

A ferramenta utilizada não apresentou 100% de acerto nos testes, o estudo conclui que o algoritmo de reconhecimento de imagem deve ser aprimorado, e que o uso de tal ferramenta deve ser supervisionado durante as execuções, caso o usuário pretenda que nenhuma falha passe despercebida.

REFERÊNCIAS

ALÉGROTH, E.; FELDT, R.; OLSSON, H. H. Transitioning manual system test suites to automated testing: An industrial case study. In: **2013 IEEE Sixth International Conference on Software Testing, Verification and Validation**. [S.l.: s.n.], 2013. p. 56–65. ISSN 2159-4848.

ALÉGROTH, E.; FELDT, R.; RYRHOLM, L. Visual gui testing in practice: Challenges, problems and limitations. **Empirical Softw. Engg.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 20, n. 3, p. 694–744, jun. 2015. ISSN 1382-3256. Disponível em: <http://dx.doi.org/10.1007/s10664-013-9293-5>.

ALÉGROTH, E. et al. Conceptualization and evaluation of component-based testing unified with visual gui testing: An empirical study. In: **2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)**. [S.l.: s.n.], 2015. p. 1–10. ISSN 2159-4848.

ALÉGROTH, E.; KARLSSON, A.; RADWAY, A. Continuous integration and visual gui testing: Benefits and drawbacks in industrial practice. In: **2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)**. [S.l.: s.n.], 2018. p. 172–181.

ALÉGROTH, E.; NASS, M.; OLSSON, H. H. Jautomate: A tool for system- and acceptance-test automation. In: **2013 IEEE Sixth International Conference on Software Testing, Verification and Validation**. [S.l.: s.n.], 2013. p. 439–446. ISSN 2159-4848.

AMMANN, P.; OFFUTT, J. **Introduction to Software Testing**. 2. ed. New York, NY, USA: Cambridge University Press, 2016. ISBN 9781107172012.

AUQTUS. **The AUQTUS Team**. 2019. Disponível em: <http://eyeautomate.com/auqtus/\team>. Acesso em: 27 de maio de 2019.

BAJPAI, V.; GORTHI, R. P. On non-functional requirements: A survey. In: **Electrical, Electronics and Computer Science (SCEECS), 2012 IEEE Students' Conference on**. [S.l.: s.n.], 2012. p. 1–4.

BARMI, Z. A.; EBRAHIMI, A. H. **Automated testing of non-functional requirements based on behavioural scripts**. 63-65 p. Dissertação (Mestrado) — Department of Computer Science and Engineering - Chalmers University of Technology, Goteborg, Sweden, 2011.

BERTOLINO, A. Software testing research: Achievements, challenges, dreams. In: **2007 Future of Software Engineering**. [S.l.: s.n.], 2007. p. 85–103.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. 3. ed. [S.l.]: Elsevier, 2015. ISBN 978-85-352-2626-3.

BORJESSON, E.; FELDT, R. Automated system testing using visual gui testing tools: A comparative study in industry. In: **2012 IEEE Fifth International Conference on Software Testing, Verification and Validation**. [S.l.: s.n.], 2012. p. 350–359. ISSN 2159-4848.

CYSNEIROS, L. M.; LEITE, J. C. S. do P. Nonfunctional requirements: from elicitation to conceptual models. **IEEE Transactions on Software Engineering**, v. 30, n. 5, p. 328–350, May 2004. ISSN 0098-5589.

CYSNEIROS L.M.; LEITE, J. Definindo requisitos não funcionais. **XI Simpósio Brasileiro de Engenharia de Software**, p. 49 – 54, 10 1997.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao teste de software**. 1. ed. [S.l.]: Elsevier, 2007.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. **Introdução ao teste de software**. 2. ed. [S.l.]: Elsevier, 2016.

FELDT, R.; MAGAZINIUS, A. Validity threats in empirical software engineering research - an initial survey. In: **Proceedings of the 22nd International Conference on Software Engineering and Knowledge Engineering SEKE**. Redwood City, USA: [s.n.], 2010. p. 374–379.

GAROUSI, V. et al. Comparing automated visual gui testing tools: an industrial case study. In: **Proceedings of the 8th ACM SIGSOFT International Workshop on Automated Software Testing**. New York, NY, USA: ACM, 2017. p. 21–28. ISBN 978-1-4503-5155-3.

HAAR, P.; MICHAëLSSON, D. **Automated GUI Testing: A Comparison Study With A Maintenance Focus**. 14-15 p. Dissertação (Mestrado) — Department of Computer Science and Engineering - Chalmers University of Technology, Goteborg, Sweden, 2018.

HOFFMAN, D. Using oracles in test automation. **Proceedings of the Nineteenth Annual Pacific Northwest Software Quality Conference (PNSQC)**, Portland, OR, USA, p. 90–117, 2001.

KOSCIANSKI, A.; SOARES, M. dos S. **Qualidade de Software Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. São Paulo: Novatec Editora, 2007.

LIEBEL, G.; ALÉGROTH, E.; FELDT, R. State-of-practice in gui-based system and acceptance testing: An industrial multiple-case study. In: **2013 39th Euromicro Conference on Software Engineering and Advanced Applications**. [S.l.: s.n.], 2013. p. 17–24. ISSN 1089-6503.

MYERS, G. J. **The Art of Software Testing**. 2. ed. Hoboken, New Jersey: John Wiley & Sons, Inc., 2004. ISBN 0-471-46912-2.

OLIVEIRA, R. A. P.

Automatização de oráculos de teste para sistemas com saídas complexas — Instituto de Ciências Matemáticas e de Computação (ICMC) – Universidade de São Paulo (USP), São Carlos/SP, 2013.

OLIVEIRA, R. A. P. et al. Definition and evaluation of mutation operators for gui-level mutation analysis. In: **2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. [S.l.: s.n.], 2015. p. 1–10.

PRESSMAN, R. S. **Engenharia de Software Uma Abordagem Profissional**. Porto Alegre: AMGH, 2011.

RAFI, D. et al. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In: **2012 7th International Workshop on Automation of Software Test (AST)**. [S.l.: s.n.], 2012. p. 36–42.

RIBEIRO, V. V.; TRAVASSOS, G. H. Testing non-functional requirements: Lacking of technologies or researching opportunities? In: **XV Simpósio Brasileiro de Qualidade de Software**. [S.l.: s.n.], 2016. p. 226–240.

SJÖBLOM, J.; STRANDBERG, C. **Automatic Regression Testing using Visual GUI Tools**. Dissertação (Mestrado), 2015. 84.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. **Introdução à Engenharia de Software Experimental**. Rio de Janeiro: COPPE / UFRJ, 2002.

ULLAH, S.; IQBAL, M.; KHAN, A. M. A survey on issues in non-functional requirements elicitation. In: **International Conference on Computer Networks and Information Technology**. [S.l.: s.n.], 2011. p. 333–340. ISSN 2223-6317.

APÊNDICE A – ESTUDO DE CASO

Este apêndice apresenta o resultado do estudo de caso realizado dentro da indústria, o mesmo foi realizado com o objetivo de investigar a viabilidade da utilização de ferramenta VGT para checar requisitos não funcionais. Este estudo de caso faz parte do trabalho de conclusão de curso.

Checagem de Requisitos Não Funcionais através de Ferramenta Visual GUI Testing

Paulo H. Bordignon[1], **Rafael A. P. Oliveira**[1]

[1]Universidade Tecnológica Federal do Paraná – Curso de Engenharia de Software
Estrada para Boa Esperança, Km. 04, Dois Vizinhos-PR

paulobordignon@alunos.utfpr.edu.br, raoliveira@utfpr.edu.br

Abstract. *Manual checking of nonfunctional requirements (NFRs) of a software is an activity that can be considered as time consuming and repetitive. To facilitate this activity, an alternative is the automation of these checks, which can be performed using Visual Tools GUI Testing (VGT). These tools use scripting and image recognition to automate testing processes. However little is known about the applicability of VGT tools in the automation of NFRs checking. In this work we present a study carried out within the software industry, with the objective of verifying the applicability of VGT tools for checking NFRs.*

Resumo. *A checagem manual de requisitos não funcionais (RNFs) de um software, é uma atividade que pode ser considerada desgastante e repetitiva. Para facilitar essa atividade, uma alternativa é a automatização dessas checagens, que pode ser realizada por meio de Ferramentas Visual GUI Testing (VGT). Essas ferramentas utilizam de scripts e reconhecimento de imagem para automatizar processos de testes. Entretanto pouco se sabe a respeito da aplicabilidade de ferramentas VGT na automatização de checagem de RNFs. Neste trabalho apresentamos um estudo realizado dentro da indústria de software, com o objetivo de verificar a aplicabilidade de ferramentas VGT para checagem de RNFs.*

1. Introdução

Os requisitos não funcionais (NFR do inglês, Non-Functional Requirements) tipicamente especificam restrições globais que devem ser satisfeitas pelo software, por exemplo, performance, tolerância a falhas, disponibilidade, segurança, usabilidade e assim por diante, ou seja, NFRs definem como o software opera e como as funcionalidades são exibidas [Rosa et al. 2004].

Dentro das atividades de implantação de um software, os NFRs devem ser avaliados, a fim de garantir uma melhor experiência de uso ao usuário. [Mockus et al. 2005] propõem que a qualidade da implantação de software é um fator-chave que contribui para a percepção do cliente sobre a qualidade do software.

Sabe-se que a implantação do software, é uma atividade fundamental para garantir que o cliente esteja satisfeito com o mesmo. Neste documento, implantação de software é entendida como o processo que engloba desde a instalação do software, até o ponto em que os usuários estão aptos a usar o sistema.

As ferramentas visuais de teste de interface gráfica (VGT do inglês, Visual GUI Testing), podem contribuir para uma melhor checagem de NFRs durante o processo de

implantação, pois o teste quando feito de forma manual é reconhecidamente mais demorado, tedioso e propenso a erros [Bertolino 2007].

Ferramentas VGT combinam o reconhecimento de imagem com *scripts*, para executar testes automatizados, essas ferramentas permitem ao testador automatizar testes com um alto nível de abstração do sistema e, emular o comportamento do usuário final [Alégroth et al. 2013].

O foco deste trabalho é analisar a aplicabilidade do uso de ferramentas VGT, para validar requisitos não funcionais, no momento da implantação de um software. Para este estudo foi realizado uma pesquisa *in vivo* dentro de uma fábrica de software, utilizando a ferramenta VGT EyeAutomate. Todos os NFRs escolhidos para o estudo, estão relacionados ao software instalado pelos implantadores.

Este artigo está organizado da seguinte maneira, Seção II apresenta o referencial teórico. A Seção III apresenta os materiais e métodos utilizados durante a realização do estudo. A Seção IV apresenta os resultados obtidos durante o estudo. A Seção V apresenta as ameaças à validade e, a Seção VI apresenta as conclusões que foram obtidas após o término do estudo.

2. Referencial Teórico

Este estudo propõe o uso de ferramentas VGT para automatizar os testes de requisitos não funcionais. As subseções a seguir explanam a respeito de todos os conceitos e definições que estão envolvidos neste estudo.

2.1. Teste de Software

A atividade de teste é considerada uma atividade de validação e verificação, que tem como objetivo executar um programa com a intenção de encontrar erros. Atividades de verificação procuram garantir que a implementação e/ou especificação do sistema estejam corretas. Validação refere-se ao conjunto de atividades que procuram garantir que o sistema esteja de acordo com os requisitos do usuário [Pressman 2010].

O objetivo do teste de NFRs, no contexto em que o estudo foi aplicado, é verificar se as máquinas dos usuários, estão aptas a receber a instalação do software. O não atendimento de um NFR, pode impedir que o sistema funcione corretamente em alguns pontos.

2.2. Teste Automatizado

As atividades de teste são essenciais e inevitáveis para garantir produtos de software de alta qualidade. A automação de testes é uma abordagem amplamente utilizada para reduzir custo do teste manual, diminuir falhas humanas nas verificações e, diminuir esforço [Amannejad et al. 2014].

Em testes automatizados, são desenvolvidos *scripts* de código de teste, com o objetivo de realizar testes de forma autônoma, sem que haja necessidade de intervenção humana.

2.3. Requisitos não funcionais

Requisitos não funcionais são restrições aos serviços ou funções oferecidas pelo sistema. Incluem restrições de tempo de acesso, restrições no processo de desenvolvimento e

restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo [Sommerville 2011].

Pode ser concluído que requisitos não funcionais, não estão relacionados a funcionalidades de um sistema, mas sim relacionados a comportamentos e características que o software como um todo deverá possuir. Os NFRs são divididos em algumas categorias como, usabilidade, desempenho, portabilidade, segurança, entre outras.

2.4. Ferramentas VGT

Ferramentas VGT são utilizadas para testar interfaces de aplicativos e verificar suas funcionalidades. Utilizando essas ferramentas, os casos de teste criados pelo testador, são traduzidos em forma de *scripts*, assim a ferramenta VGT executará o *script* e apresentará o resultado do teste para o testador.

3. Materiais e Métodos

Para o presente estudo, os métodos de pesquisa escolhidos para a coleta de dados, foram questionários e pesquisa em campo. Estes métodos permitirão coletar vantagens e desvantagens, na aplicação de uma ferramenta VGT dentro da indústria, para a verificação de NFRs no processo de implantação do software.

3.1. Objetivos

Para que seja possível responder o objetivo geral deste trabalho, foram elaboradas as questões de pesquisa apresentadas abaixo:

Q1: Identificar e especificar NFRs passíveis de serem verificados com ferramenta VGT;

Q2: Produzir/Prototipar *scripts* para verificação dos NFRs;

Q3: Verificar a robustez dos *scripts* VGT para verificação dos NFRs;

Q4: Determinar a viabilidade da execução dos *scripts* no ambiente de implantação;

Q5: Averiguar a opinião de profissionais dentro da indústria a respeito do uso de VGT;

Q6: Fazer uma análise de *trade-offs* entre o uso de VGT para checagem versus checagem manual.

3.2. Estrutura da Pesquisa

O estudo foi realizado em três etapas. Na primeira etapa, foi aplicado um questionário direcionado a profissionais da área, para dimensionar o conhecimento a respeito de NFRs e ferramentas VGT. Na segunda etapa, foi realizado a pesquisa in vivo, onde junto a implantadores foram coletados dados a respeito do uso da ferramenta VGT. A terceira etapa, consistiu em aplicar novamente um questionário, com o objetivo de responder algumas das questões de pesquisa.

3.3. Questionários

A primeira parte do estudo, consistiu em aplicar um questionário para profissionais das áreas de desenvolvimento, teste e implantação. Este questionário foi respondido por 31 participantes.

As perguntas contidas neste questionário, foram feitas com o objetivo de dimensionar o quão conhecida as ferramentas VGT são, e verificar qual a opinião dos participantes a respeito do uso das mesmas na automatização de checagem de NFRs. Neste mesmo questionário foram avaliadas outras questões as quais serão explanadas no decorrer deste trabalho.

Um segundo questionário foi aplicado, este direcionado somente para implantadores que participaram do estudo de caso, usando a ferramenta no seu dia a dia para checar NFRs. Este questionário foi aplicado ao fim do estudo de caso com o objetivo de responder algumas das questões de pesquisa, o mesmo foi respondido por três participantes.

3.4. Pesquisa de campo

Para a realização da pesquisa *in vivo*, foi necessário coletar os requisitos não funcionais e, em seguida criar os *scripts*. A pesquisa teve como principais objetivos, avaliar a robustez dos *scripts* e da ferramenta e, comparar o tempo levado para a realização da checagem manual e checagem automatizada.

Esta pesquisa foi realizada em computadores de clientes, implantadores e testadores. No total foram utilizados 16 computadores para aplicar a pesquisa, a mesma foi realizada junto a implantadores.

4. Resultados

Os resultados serão apresentados de forma separada, levando em conta cada questão de pesquisa apresentada na Seção 2.1, junto ao resultado também será detalhado como o mesmo foi obtido. Questionários, pesquisas, NFRs escolhidos, *scripts* criados, serão apresentados nesta Seção.

Q1: Identificar e especificar NFRs passíveis de serem verificados com ferramenta VGT.

Devido às limitações das ferramentas VGT, nem todos os requisitos não funcionais podem ser checados. NFRs relacionados a usabilidade e reusabilidade, são exemplos que não podem ser checados por ferramentas VGT, a primeira devido a sua complexidade e, a segunda devido envolver o código do projeto.

Levando em consideração este ponto e, também a demanda dos implantadores, foram levantados quais NFRs seriam utilizados para a checagem. Os NFRs escolhidos estão dispostos abaixo, na Figura 01.

Os NFRs escolhidos estão relacionados a portabilidade e desempenho do software. Para um correto funcionamento do software implantado, o computador precisa possuir o sistema operacional Windows acima de determinada versão. O computador deve possuir também, o navegador Internet Explorer na última versão e o Windows .NET Framework acima de determinada versão.

NFRs	Categoria
Versão Windows	Portabilidade
Versão IE	Portabilidade
Framework	Portabilidade
Abertura Sistema	Desempenho
Logar Sistema	Desempenho
Listagem Clientes	Desempenho
Cadastro Clientes	Desempenho
Listagem Vendas	Desempenho
Cadastro Vendas	Desempenho
Calendário	Desempenho
Logoff	Desempenho

Figura 1. NFRs escolhidos para checagem

Os NFRs relacionados ao desempenho, são verificados com o objetivo, de garantir que o usuário não tenha uma experiência desagradável, ocasionada por lentidão do sistema. Então são checados os processos mais utilizados, como, abertura do sistema, cadastro de clientes, cadastro de vendas e saída do sistema.

A primeira questão de pesquisa, foi respondida através do segundo questionário aplicado aos implantadores. Foi perguntado então, se os NFRs checados atendiam a atual demanda de checagem durante a implantação, e também perguntado se as limitações apresentadas pela ferramenta, prejudicaram na checagem dos NFRs.

Todos os participantes responderam que os NFRs checados, atendiam a atual demanda. Responderam também que as limitações apresentadas pela ferramenta, não prejudicaram na realização da checagem.

Q2: Produzir/Prototipar *scripts* para verificação dos NFRs.

Depois de levantados os NFRs que seriam checados, teve que ser realizado a produção dos *scripts*. A produção não foi realizada pelos implantadores, os mesmos apenas executaram-os. Na Figura 2 presente abaixo, é apresentado um *script* criado com a ferramenta EyeAutomate.

O *script* presente na Figura 2, é um dos menos extensos, e serve para checar se a listagem de clientes será carregada dentro de 10 segundos. Para cada NFR a ser checado, foi criado um *script*, todos os *scripts* são chamados por uma suíte de testes. A ferramenta permite, que seja feito várias chamadas de um mesmo *script*, assim é possível reaproveitar o código.

Para validar se a produção dos *scripts* foi eficaz e resolveu problemas dos implantadores, foi perguntado através do segundo questionário, se o tempo de execução dos *scripts* é aceitável, se a execução dos *scripts* é fácil e, se os resultados apresentados são claros.

Dois dos três participantes disseram que o tempo de execução deveria ser reduzido. Todos os participantes responderam que a execução dos *scripts* é fácil e que os

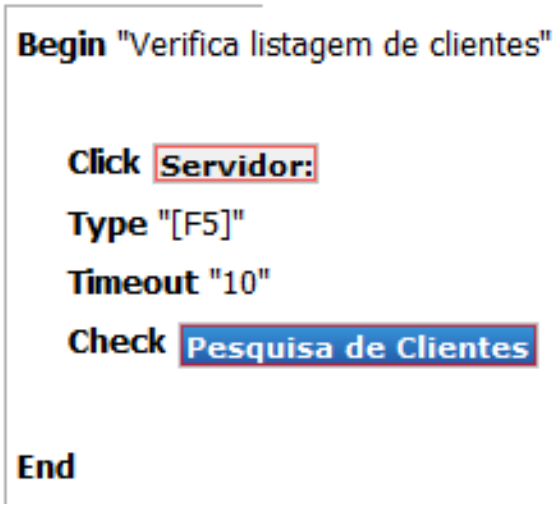

Figura 2. Script criado

resultados apresentados são claros.

Q3: Verificar a robustez dos *scripts* VGT para verificação dos NFRs.

A robustez dos *scripts* foi analisada em dois momentos, ao produzir os *scripts* e, ao executar os *scripts* em computadores de clientes. A versão com os *scripts*, só foi enviada para rodar em clientes, após todos os *scripts* apresentarem resultado verdadeiro positivo.

Após todos os *scripts* terem passados pelos testes, ao executá-los nos computadores dos clientes, foi analisado se o resultado mostrado pelo *script*, era o mesmo resultado coletado através da checagem manual. Desta forma a robustez dos *scripts* pôde ser avaliada em dois momentos.

Os onze *scripts* criados, foram executados dezesseis vezes cada, desta forma 176 execuções de *scripts* foram realizadas. Dessas 176 execuções, apenas 4 delas apresentaram resultados diferentes do que era correto.

Q4: Determinar a viabilidade da execução dos *scripts* no ambiente de implantação.

Após concluída a produção dos *scripts*, os mesmos foram apresentados a um implantador, para que fosse coletada a sua opinião a respeito da viabilidade de executar tais *scripts* em clientes. Neste momento foi percebido que a instalação da ferramenta, seguida da importação dos *scripts*, e da execução dos mesmos, era um processo manual muito demorado.

Pensando nisso, uma nova forma de executar os *scripts* teve que ser pensada. Analisando a documentação da ferramenta EyeAutomate, foi percebido que os *scripts* poderiam ser executados através do CMD. Com isso em mente, foi elaborado um único instalador, com a ferramenta, os *scripts*, e um arquivo .bat, capaz de executar os *scripts* automaticamente. Desta forma a execução dos *scripts* tornou-se rápida e de fácil uso.

A viabilidade da execução dos *scripts*, foi questionada através do segundo questionário. Foram realizadas perguntas para saber se os *scripts* poderiam ser usados no

dia a dia do implantador, e para saber se a checagem automatizada poderia substituir a checagem manual.

Dois dos três implantadores responderam que os *scripts* poderiam ser utilizados no dia a dia, apenas um disse que alguns pontos precisam ser melhorados. Todos os participantes disseram que a checagem automatizada poderia substituir a checagem manual.

Q5: Averiguar a opinião de profissionais dentro da indústria a respeito do uso de VGT.

Para saber mais a respeito, do que pensam os profissionais de dentro da indústria, sobre o uso de ferramentas VGT, foram realizadas perguntas com este propósito, no primeiro e no segundo questionário.

No primeiro questionário, foi perguntado se o termo ferramenta VGT, já era conhecido pelo participante. Dentre todos 58,1% responderam que já conheciam este termo e 41,9% disseram não conhecer o termo antes do questionário.

Também foi solicitado aos mesmos participantes, se ferramentas VGT poderiam ser utilizadas para checar NFRs. Para esta pergunta, 71% disseram concordam, e 29% disseram não saber opinar a respeito.

No segundo questionário foi solicitado aos participantes, se o uso de ferramentas VGT para checagem de NFRs seria recomendado por eles, todos os participantes responderam que recomendariam. Também foi perguntado se o uso da ferramenta facilitou o trabalho que era realizado por eles, todos os participantes responderam que sim.

Q6: Fazer uma análise de trade-offs entre o uso de VGT para checagem versus checagem manual.

Esta questão de pesquisa foi respondida através da pesquisa *in vivo*, ao realizar a execução dos *scripts* com os implantadores, foi levantado o tempo levado pelo *script* e o tempo levado na checagem manual.

Após realizada a execução dos *scripts*, foi levantado que o tempo médio da execução dos mesmos foi de dois minutos e quarenta e um segundos. Já o tempo médio levado para a checagem manual foi de dois minutos. Desta forma a execução manual se saiu mais rápida que a execução automatizada.

A diferença de tempo mencionada, se deve ao fato de que o *script* que verifica a versão do navegador, levava em média, aproximadamente um terço do tempo total de execução. Portanto dos 02:41, cerca de 00:53 era utilizado apenas por um dos dez scripts. Deste modo, a diferença de tempo entre as execuções é um problema relacionado a implementação dos *scripts*, e não da ferramenta em si.

Também foi comparado o resultado de cada verificação realizada pelo *script*, com o resultado obtido através da verificação manual, neste caso de 176 verificações realizadas pelo *script*, apenas 4 delas tiveram resultado incorreto.

Estes quatro resultados incorretos, foram ocasionados por problemas com a ferramenta e não com os scripts. Destes 4 resultados, 2 deles ocorreram devido a ferramenta executar comandos fora do tempo estabelecido, nesses casos, sempre que o *script* fosse rodado novamente, o problema não ocorria outra vez. Os outros 2 resultados incorretos,

ocorreram devido a reconhecimento de imagem, neste caso um dos *scripts* não conseguiu reconhecer a versão do S.O e o outro não conseguiu reconhecer o cabeçalho de uma tela do sistema.

5. Ameaças à Validade

A validade de um experimento, diz respeito ao quão certas estão as conclusões apresentadas no estudo. É uma questão fundamental avaliar o quão válidos são os resultados de um experimento. Por se tratar de uma pesquisa quantitativa, relacionada a engenharia de software, será considerada a validade interna, externa, de construção e de conclusão [Travassos et al. 2002, Feldt and Magazinius 2010].

A validade interna define se o relacionamento observado entre o tratamento e o resultado é causal, e não é o resultado da influência de outro fato que não é controlado ou não foi medido [Travassos et al. 2002]. Todos os participantes do segundo questionário participaram também do estudo, o que evitou desentendimentos relacionados a termos ou conceitos, desta forma foram minimizadas qualquer ameaça à validade interna. Já no primeiro questionário, que participaram profissionais não ligados diretamente ao estudo, foi introduzido os conceitos de cada termo utilizado no questionário, para evitar também ameaças à validade interna. Para os dois questionários o número de perguntas não foi elevado, aproximadamente 15 para cada, assim evitando também que os participantes ficassem cansados ou desanimados. Todos os participantes são profissionais que trabalham dentro do ciclo de produção de software.

A validade externa define as condições que limitam a habilidade de generalizar os resultados de um experimento para a prática industrial [Travassos et al. 2002]. Durante a execução dos *scripts* e a aplicação dos questionários, foi levado em conta a disponibilidade dos participantes, para que não fosse realizada nenhuma atividade em dias que os participantes estivessem atarefados. Desta forma evitando que fatores externos prejudicassem os resultados.

A validade de construção considera os relacionamento entre a teoria e a observação, ou seja, se o tratamento reflete a causa bem e o resultado reflete o efeito bem [Travassos et al. 2002]. Durante o estudo procurou-se ser o mais o abrangente possível, o fato de que apenas duas categorias de NFRs foram utilizadas, pode apresentar determinada ameaça a validade de construção, os NFRs utilizados estão ligados a demanda da equipe que participou do estudo.

A validade de conclusão está relacionada a obtenção de uma conclusão correta a respeito dos relacionamentos entre o tratamento e o resultado do experimento [Travassos et al. 2002]. O segundo questionário, foi aplicado a um número baixo de participantes, isso se deve ao fato de que, o estudo foi realizado dentro de apenas uma empresa. O baixo número de participantes neste questionário, apresenta uma certa ameaça a validade para o experimento, porém o estudo já apresenta avaliações iniciais da aplicabilidade da ferramenta dentro de uma empresa de software.

6. Considerações Finais

O estudo objetivou averiguar a aplicabilidade de ferramentas VGT dentro da indústria, para checar requisitos não funcionais. Foi escolhido uma ferramenta para a execução do estudo.

A ferramenta escolhida passou por observações, durante a execução dos *scripts* em computadores de clientes. Por meio das execuções foi observado que o algoritmo de reconhecimento de imagem da ferramenta, precisa ser melhorado.

Foram encontradas falhas relacionadas ao reconhecimento de imagem, e falhas relacionadas ao tempo de execução dos comandos, a quantidade de falhas é muito baixa, o que não impede que a ferramenta seja utilizada para a realização de checagens automatizadas.

O estudo conclui que a ferramenta VGT pode ser utilizada para realizar a checagem de requisitos não funcionais, dentro da indústria no ambiente de produção. Entretanto, é recomendado que a execução dos *scripts* seja supervisionada por um humano.

Referências

Alégroth, E., Nass, M., and Olsson, H. H. (2013). Jautomate: A tool for system- and acceptance-test automation. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, pages 439–446.

Amannejad, Y., Garousi, V., Irving, R., and Sahaf, Z. (2014). A search-based approach for cost-effective software test automation decision support and an industrial case study. In *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, pages 302–311.

Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In *Future of Software Engineering (FOSE '07)*, pages 85–103.

Feldt, R. and Magazinius, A. (2010). Validity threats in empirical software engineering research - an initial survey. pages 374–379.

Mockus, A., Zhang, P., and Li, P. L. (2005). Predictors of customer perceived software quality. In *Proceedings of the 27th International Conference on Software Engineering*, ICSE '05, pages 225–233, New York, NY, USA. ACM.

Pressman, R. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, USA, 7 edition.

Rosa, N. S., Cunha, P. R. F., and Justo, G. R. R. (2004). An approach for reasoning and refining non-functional requirements. *Journal of the Brazilian Computer Society*, 10:59 – 81.

Sommerville, I. (2011). *Engenharia de Software*. Pearson Prentice Hall, São Paulo, 9 edition.

Travassos, G. H., Gurov, D., and Amaral, E. A. G. (2002). *Introdução à Engenharia de Software Experimental*. COPPE / UFRJ, Rio de Janeiro.