Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Ciência da Computação

Um estudo sobre as práticas de testes funcionais de GUI na indústria brasileira de software

Nailton Soares de Almeida Junior

Feira de Santana

2023

Universidade Estadual de Feira de Santana

Programa de Pós-Graduação em Ciência da Computação

Nailton Soares de Almeida Junior

Um estudo sobre as práticas de testes funcionais de GUI na indústria brasileira de software

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Larissa Rocha Soares Bastos

Coorientador: Heitor Augustus Xavier Costa

Feira de Santana

2023

Ficha Catalográfica - Biblioteca Central Julieta Carteado - UEFS

A448 Almeida Júnior, Nailton Soares de
Um estudo sobre as práticas de testes funcionais de GUI na indústria brasileira de software / Nailton Soares de Almeida Júnior. – 2023.
117 f. : il.

Orientadora: Larissa Rocha Soares Bastos.
Coorientador: Heitor Augustus Xavier Costa.
Dissertação (mestrado) – Universidade Estadual de Feira de Santana, Programa de Pós-Graduação em Ciência da Computação, Feira de Santana, 2023.

1. Interface homem – computador. 2. Desenvolvimento de software. I. Universidade Estadual de Feira de Santana. II. Bastos, Larissa Rocha Soares, orient. III. Costa, Heitor Augustus Xavier, coorient. IV. Título.

CDU 004.5

Renata Aline Souza Silva - Bibliotecária - CRB-5/1702

Nailton Soares de Almeida Junior

Um estudo sobre as práticas de testes funcionais de GUI na indústria brasileira de software

Dissertação apresentada à Universidade Estadual de Feira de Santana como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Feira de Santana, 16 de fevereiro de 2023

BANCA EXAMINADORA

Larissa Rocha Soares Bastos (Orientadora)
Universidade Estadual de Feira de Santana

Victor Travassos Sarinho
Universidade Estadual de Feira de Santana

Alcemir Rodrigues Santos
Universidade Estadual do Piauí

Abstract

Software testing is an important activity in the software development cycle, which seeks to ensure the quality of software created. When software does not meet its users' implicit and explicit needs, one can say that it lacks quality. The lack of quality in software can generate several problems for the software development project, such as friction between supplier and clients/users, financial losses, legal problems due to breach of contracts, and even risk to the life of its users in critical contexts.

Along these lines, different levels of testing have been developed to ensure quality in different aspects of the software. A popular approach is functional interface testing, which evaluates software by executing events on the software screen, such as filling in a text field or clicking in a button. GUI (Graphical User Interface) functional testing can be executed either manually, with the tester interacting directly with the software, or automatically, by running the software through a programmed *script*.

In recent years, one can find several solutions and techniques proposed to deal with GUI testing in the literature. However, they usually do not analyze how industry professionals perform this type of testing, leaving a gap in the area. Therefore, this study investigates how testers perform interface testing on their projects. First, an online survey was conducted with 222 testing professionals with different backgrounds, job titles, and roles. The results indicated that many professionals perform tests exclusively in a manual way. Additionally, most of them agree that tools used in test automation meet business needs, although they have technical limitations.

Then, we conducted a semi-structured interview with 20 testing professionals to complement the results obtained with the questionnaire. The results indicated that testers who only perform manual GUI tests do not create automated GUI tests due to the context of the project in which they work, their professional preferences and also for educational reasons, such as difficulty in programming. In addition, distinct limitations were observed for manual and automated GUI tests that negatively impact the activities performed by professionals, such as difficulties related to the company's culture and technical problems related to the software under test. The combined results provide a general overview of GUI testing available in the Brazilian software industry.

Keywords: Software testing, GUI testing, software testers, surveys, interviews

Resumo

Teste de software é uma importante atividade no ciclo de desenvolvimento de software, a qual busca garantir a qualidade dos software criados. Quando um software não supre as necessidades implícitas e explicitas de seus usuários, pode-se dizer que ele carece de qualidade. A falta de qualidade de um software pode gerar vários problemas para o projeto de desenvolvimento de software, como atritos entre fornecedor e clientes/usuários, perdas financeiras, problemas legais por descumprimento de contratos e até risco à vida dos seus usuários em contextos críticos.

Nesse sentido, distintos níveis de testes foram desenvolvidos para garantir a qualidade em diferentes aspectos do software. Uma abordagem popular é o teste funcional de interface, que avalia o software através da execução de eventos na tela do software, como o preenchimento de um campo de texto ou o clique de um botão. Os testes funcionais de interface ou GUI (*Graphical User Interface*), podem ser executados tanto de forma manual, com a interação direta do testador no software, quanto automatizada, com a execução do software através de um *script* programado.

Nos últimos anos, é possível encontrar na literatura diversas soluções e técnicas propostas para lidar com os testes de GUI. No entanto, as pesquisas não se aprofundam em analisar como os profissionais da indústria vem executando esse tipo de teste, deixando assim uma lacuna na área. Por isso, este estudo propõe investigar como os testadores estão realizando testes de interface em seus projetos. Assim, primeiramente foi realizado um questionário online com 222 profissionais de teste com distintas experiências, cargos e funções. O resultado indicou que ainda há muitos profissionais que realizam testes exclusivamente de forma manual. Além disso, a maioria dos profissionais concordam que as ferramentas utilizadas na automação de testes atendem as necessidades de negócio, embora tenham limitações técnicas.

Em seguida, realizamos uma entrevista semiestruturada com 20 profissionais de testes para complementar os resultados obtidos com o questionário. Os resultados indicaram que testadores que realizam apenas testes de GUI manuais não criam testes de GUI automatizados devido ao contexto do projeto em que trabalham, suas preferências profissionais e também por questões educacionais, como dificuldade com programação. Além disso, foram observadas distintas limitações para testes de GUI manuais e automatizados que impactam negativamente as atividades desempenhadas pelos profissionais, como dificuldades relacionadas à cultura da empresa e problemas

técnicos relacionados ao software sob teste. Os resultados combinados fornecem uma visão geral dos testes de GUI na indústria de software brasileira.

Palavras-chave: Teste de software; Teste de GUI; Testadores de software; Questionário; Entrevistas.

Prefácio

Esta dissertação de mestrado foi submetida à Universidade Estadual de Feira de Santana (UEFS) como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

A dissertação foi desenvolvida no Programa de Pós-Graduação em Ciência da Computação (PGCC), tendo como orientadora a Profª. Dra. **Larissa Rocha Soares Bastos** e como coorientador o Prof. Dr. **Heitor Augustus Xavier Costa**.

Agradecimentos

Escrever um texto de agradecimento é uma atividade muita complexa quando várias pessoas contribuíram positivamente em sua vida.

Antes de tudo, dedico esse trabalho aos meus pais Nailton e Ednolia, que me mostraram desde sempre, que o melhor caminho a ser seguido é o da educação. Sem as longas conversas e incentivo recebido de vocês, eu não estaria onde cheguei.

À minha irmã Miriã, obrigado por todo carinho e afeto compartilhado em todos esses anos. Um dos dias mais felizes da minha vida foi quando eu soube que minha mãe estava grávida, e desde então, me sinto privilegiado de ser o irmão mais velho.

Agradeço imensamente a minha companheira Alana, que esteve firme ao meu lado em vários momentos difíceis, até quando eu não acreditava em mim mesmo. Todas as minhas vitórias são provenientes do seu amor, apoio e dedicação. Obrigado por me inspirar a ser um homem melhor.

Aos meus sogros Antônio e Eldeni que me acolheram em sua família e sempre estiveram por mim na hora da necessidade. Ao meu cunhado Ariel pelas conversas agradáveis sobre cultura pop.

À minha professora-orientadora Larissa Rocha, que abriu as portas do mundo acadêmico para mim. Sou grato pela confiança em me orientar, por todas as horas dedicadas em reuniões em auxílio a minha pesquisa e pelo conhecimento compartilhado. Aos professores Heitor Costa e Ivan Machado, que sempre estiveram disponíveis para compartilhar boas ideias, além de ajudar na revisão de textos e codificação de dados de estudos.

Aos professores da Pós-Graduação em Ciência da Computação da UEFS, que me fizeram acreditar ainda mais na Ciência. Aos meus colegas mestrandos que mostraram para mim que o ato de estudar não precisa ser uma trajetória solitária, e que colaborativamente podemos crescer juntos.

Aos meus companheiros Celso Tribuno, Rodrigo Borba, Calvin Pacheco e Walter Junior, que mesmo seguindo rumos diferentes, sempre estão a postos para uma cerveja gelada e boas risadas.

Sou grato pelo universo dos quadrinhos que me apresentaram histórias fantásticas, as quais me ensinaram a ser perseverante nos desafios, a ter esperança nos momentos ruins e sonhar com um futuro melhor.

Por fim, agradeço a minha pet Nina por todo apoio emocional dado inconscientemente. Olhar para você deitada nos meus pés durante as longas jornadas na frente do computador fizeram com que tudo parecesse mais fácil.

"Não devemos nos esquecer nunca do passado, mas precisamos sempre contemplar o futuro, porque é lá que vamos passar o resto de nossas vidas."

– Marv Wolfman

Sumário

Alinhamento com a Linha de Pesquisa

Linha de Pesquisa: Software e Sistemas Computacionais

A dissertação em questão se enquadra na linha de pesquisa, pois busca contribuir com avanço na área de Engenharia de Software, especificamente, no que diz respeito ao teste de software. O nível de teste abordado nesta pesquisa foram os testes de GUI, com intuito de compreender as práticas utilizadas na execução desta difundida maneira de avaliar software.

Espera-se que essa pesquisa possa aprofundar o entendimento para com a qualidade de software, através da análise de dados coletados na indústria com profissionais de testes, considerando a importância do tópico para com o ciclo de desenvolvimento de software.

Produções Bibliográficas, Produções Técnicas e Premiações

0.1 Produção do autor

Nailton Junior, Heitor Costa, Leila Karita, Ivan Machado, and Larissa Soares. 2021. Experiences and Practices in GUI Functional Testing: A Software Practitioners' View. Brazilian Symposium on Software Engineering. Association for Computing Machinery, New York, NY, USA, 195–204. DOI:https://doi.org/10.1145/3474624.3474640

0.2 Coprodução do autor

Lais Farias, Larissa Rocha , Claudia Pereira, Ivan Machado, Windson Viana, Nailton Junior. 2022. Estudantes com Deficiência Visual em Computação: participação, perspectivas e desafios enfrentados. Simpósio Brasileiro de Educação em Computação (EduComp).

Lista de Tabelas

Lista de Figuras

Lista de Abreviações VERIFICAR ABREVIAÇÕES

Abreviação	**Descrição**
APIs	Application Programming Interface
BDD	Behavior Driven Development
BSTQB	Brazilian Software Testing Qualifications Board
CBTS	Certificação Brasileira de Teste de Software
CISQ	Consortium for Information & Software Quality
CSS	Cascading Style Sheets
CTAL-AT	Certified Tester Advanced Level - Automation Test
CTAL-SEC	Certified Tester Advanced Level - Security Tester
CTAL-TA	Certified Tester Advanced Level - Test Analyst
CTAL-TAE	Certified Tester Advanced Level - Test Automation Engineer
CTAL-TM	Certified Tester Advanced Level - Test Manager
CTAL-TM	Certified Tester Advanced Level - Test Manager
CTAL-TTA	Certified Tester Advanced Level - Technical Test Analyst
CTFL	Certified Tester Foundation Level
CTFL- AT	Certified Tester Foundation Level - Agile Tester
CTFL-MBT	Certified Tester Foundation Level - Model Based Test
CTFL-PT	Certified Tester Foundation Level – Performance Testing
GUI	Graphical User Interface
HTML	HyperText Markup Language
ISTQB	International Software Testing Qualifications Board
TCLE	Termo de Consentimento Livre e Esclarecido
UFT	Unified Functional Testing
VGT	Visual GUI Testing

Capítulo 1

Introdução

1.1 Considerações Preliminares

Com a crescente dependência de software, seja em atividades triviais, como um aplicativo para troca de mensagens, seja em atividades complexas, como um software empregado em um processo industrial, os usuários finais têm exigido cada vez mais qualidade nos software utilizados por eles. A qualidade de software é definida pela ISO/IEC 25010 como o grau em que ele satisfaz as necessidades explícitas e implícitas de seus interessados (25000, 2011). Assim, quando um software não atende os critérios de aceitação dos usuários, diversos problemas podem ocorrer, por exemplo, a redução da confiança para a equipe de desenvolvimento de software, perdas financeiras, ações legais por não cumprimento de contratos e riscos à vida de pessoas em contextos críticos (Alferidah e Ahmed, 2020).

Segundo a ISTQB (*International Software Testing Qualifications Board*), uma das maneiras de evitar os problemas mencionados é realizar testes de software, visando reduzir os riscos de falhas (Klaus Olsen e Ulrich, 2019). Para garantir a qualidade do software, o processo de teste precisa ser planejado, por exemplo, identificar o momento mais adequado para iniciar os testes, pois, quanto mais cedo um software for avaliado, mais rápidos os defeitos existentes podem ser detectados e corrigidos, evitando aumento dos custos de manutenção (Glenford, 1979).

Os profissionais de tecnologia que testam software (testadores de software) analisam, com o uso de técnicas de testes apropriadas, se o software atende ao seu propósito definido (Bach, 2022). Nesse contexto, existem técnicas de testes estáticas e dinâmicas. Nas técnicas estáticas, tenta-se evitar que falhas nos artefatos de projeto de software induzam o desenvolvedor a implementar o software de maneira errônea, geralmente por meio da revisão de requisitos, das histórias do usuário, dos fluxogramas e do código fonte desse software. Nos testes dinâmicos, realiza-se a execução do software para buscar imprecisões existentes. Um tipo de teste dinâmico é o teste GUI (*Graphical User Interface*) ou teste de interface, que avaliam software que possuem interfaces gráficas (Banerjee et al., 2013).

No teste de GUI, há preocupação em verificar se os eventos realizados na tela do software respondem corretamente as ações do usuário, como o clique do *mouse* em um botão. Com a interface gráfica, torna-se mais fácil utilizar as funções disponíveis; além disso, é estimado que 60% do código fonte dos software estejam associados às suas opções gráficas (Jones, 2015).

Com o teste de GUI, pode-se testar o software similar à forma como um usuário final faria no ambiente de produção. Assim, o teste de GUI é priorizado pelos testadores para ser executado nos software desenvolvidos em seu projeto, como mostrado por Santos et al. (2022), que evidenciou a adoção de testes realizados por meio de interface gráfica pelas empresas brasileiras de software. Além do mais, o teste de GUI é popular pela possibilidade de ser usado em domínios e plataformas diversas, tais como, dispositivos móveis e computadores pessoais (AltexSoft, 2020).

Os testes de GUI podem ser realizados de forma manual ou de forma automatizada (Amalfitano et al., 2012). A realização de testes manuais necessita que os testadores interajam diretamente com o software sob teste. Por verificar aspectos funcionais e não-funcionais, o teste manual é parte essencial do processo de teste de software (Ramler e Wolfmaier, 2006). Os testes de GUI automatizados foram criados para suprirem as fragilidades dos testes de GUI manuais, analisando o software sem a supervisão direta de um testador. Uma fragilidade é em relação ao tempo de execução, pois os testes exercitados por uma máquina tendem a consumir menos tempo do que os realizados por uma pessoa (Shruti Malve, 2017). Outra fragilidade está relacionada ao reuso dos testes, que permite a utilização do teste em diferentes versões do software, independentemente do dispositivo ou sistema operacional. Com a automação dos testes de GUI, pode-se analisar o tempo de carregamento e desempenho geral do software, atividade que, se for feita de forma manual, pode não trazer resultados confiáveis, pois é difícil contabilizar e gerenciar os dados de performance manualmente (Molyneaux, 2015).

Em virtude dos benefícios e do amplo uso pelos profissionais de tecnologia dos testes de GUI manuais e automatizados, pesquisas recentes investem esforços em desenvolver novas propostas e analisam as técnicas consolidadas para a execução dos testes de GUI (Cheng et al., 2019), como apresentado a seguir. Considerando a popularização dos *smartphones* e o uso de aplicativos, há diversas pesquisas que estudam sobre testes de GUI feitos em plataformas móveis. Por exemplo, (i) Mohammad et al. (2019) avaliaram ferramentas para geração de testes automatizados para celulares utilizando nove atributos de qualidade, (ii) Gunasekaran e Bargavi (2015) compararam o desempenho de soluções para automatizar os testes de GUI para *Android* e *IOS*, (iii) Alotaibi e Qureshi (2017) propuseram uma estrutura de testes automatizados utilizando a biblioteca *Appium*[1] para os testes feitos para interfaces móveis; e (iv) Banerjee et al. (2013) realizaram um estudo sistemático para compreender o estado da arte dos testes de GUI automatizados para aplicações *desktop* e móveis.

[1] `https://appium.io/`

1.2 Problema

Embora a literatura apresente evidências empíricas sobre os testes de GUI, parte dos trabalhos objetivam desenvolver novas técnicas para a automação dos testes. As últimas revisões sistemáticas feitas por pesquisadores da área evidenciam uma limitação de estudos relevantes no que se diz respeito ao conhecimento de como os testes de GUI são executados na indústria e a identificação do perfil dos profissionais que executam esse tipo de teste. Por exemplo, (i) Garousi et al. (2020b) focaram no aspecto educacional do teste de software, agrupando publicações sobre o aprendizado da área de testes, (ii) Arnatovich e Wang (2018) trouxeram, por meio de uma revisão sistemática, diversas técnicas automatizadas para testes de GUI em aplicativos móveis e (iii) Banerjee et al. (2013) identificaram diversos artigos sobre avaliações feitas na interface do software. No entanto, ao analisar esses estudos, pode-se identificar que parte deles não aprofundam sobre os testes de GUI na indústria ou não centralizam o foco da pesquisa a partir da perspectiva dos testadores.

1.3 Objetivos

Considerando que a realização de teste de GUI é importante para a qualidade de software (Tanaka, 2019) e ainda há uma lacuna de conhecimento em como esse teste é feito por profissionais da indústria, esse trabalho objetiva investigar como os testes de GUI são executados na indústria brasileira a partir da perspectiva dos testadores de software. Como objetivos específicos, tem-se:

- **Verificar qual é o perfil e a experiência dos profissionais da área de teste**. Para alcançar esse objetivo, foi realizado um questionário *online* com 222 participantes para descobrir os perfis dos testadores e as práticas mais frequentes adotadas pelos profissionais na indústria brasileira de software;
- **Identificar o contexto de projeto de software em que os testadores atuam e as suas motivações para executar testes de GUI manual ou automatizado**. Foram realizadas entrevistas com um recorte dos profissionais de testes, buscando entender melhor a motivação para automatizar ou não os testes de GUI, além de identificar aspectos referentes ao projeto em que os testadores estão alocados nas empresas em que prestam serviços.

1.4 Questões de Pesquisa

Este estudo busca responder a seguinte questão de pesquisa: **Como os testes de GUI são realizados na indústria?** A questão principal foi dividida em questões de pesquisa específicas:

- **QP1. Como os profissionais executam os testes de GUI manuais?** O objetivo é revelar as práticas habitualmente utilizadas para o planejamento e a execução de testes de GUI manuais;

- **QP2. Como os profissionais executam os testes de GUI automatizados?** O objetivo é identificar as linguagens de programação e as ferramentas de suporte utilizadas para automatizar os testes de GUI. Além disso, verificar em que momento os profissionais de testes realizam a execução dos testes de GUI automatizados em seus software;
- **QP3. Quais soluções os profissionais comumente utilizam para registrar os erros e as falhas encontrados no projeto de software?** O objetivo é identificar como os profissionais relatam os *bugs* encontrados.
- **QP4. Quais são as motivações para testar GUI manualmente ou de forma automatizada?** O objetivo é identificar os motivos que levam um testador a executar somente testes de GUI manuais, bem como as razões que levam o testador a automatizar testes de GUI.
- **QP5. Como os testes de GUI manuais são planejados?** O objetivo é aprofundar o entendimento sobre o planejamento realizado pelos testadores para a execução manual dos testes de GUI, procurando identificar os objetivos do testes, quais documentações de apoio são utilizadas para a criação dos testes, o formato em que os testes são escritos e o reuso dos cenários de testes criados anteriormente.
- **QP6. Quais são as principais limitações no processo de testes de GUI manuais?** O objetivo é identificar quais são as maiores limitações que impactam negativamente a atividade de criação e execução dos testes de GUI manuais.
- **QP7. Como é o processo automatizado dos testes de GUI?** O objetivo é entender sobre o processo de automação de teste de GUI realizado pelos testadores, como validar as ferramentas utilizadas para as atividades de automação, os insumos de suporte utilizados para automatizar testes de GUI, os gatilhos para a realização de manutenção nos *scripts* automatizados e as boas práticas de projeto para a criação de testes de GUI automatizados.
- **QP8. Quais são as principais limitações no processo de testes de GUI automatizados?** O objetivo é identificar quais limitações dificultam a atividade de automação de testes de GUI nos projetos de desenvolvimento de software.

1.5 Relevância do Tema

Segundo o CISQ (*Consortium for Information & Software Quality*), estima-se que, em 2022 nos Estados Unidos, mais de 2 trilhões de dólares foram perdidos por causa da baixa qualidade de software (CISQ, 2022). Por conta dos riscos relacionados às falhas no desenvolvimento de software, mobilizações por parte da indústria vêm ocorrendo para melhorar a realização de testes de software.

De acordo com o ISTQB, desde 2002, foram emitidas mais de 750 mil certificações relacionadas à área de Testes para profissionais de 129 países, mostrando que os

testadores buscam capacitar-se na área de qualidade de software (ISTQB, 2020). O ISTQB também evidenciou que as companhias de tecnologia estão investindo para terem profissionais de testes mais preparados para as suas atividades. Em uma pesquisa realizada em 2018 com mais de 2.000 testadores de diferentes países do mundo, aproximadamente 70% dos respondentes afirmaram que, em seus locais de trabalho, há empenho para melhorar a competência dos funcionários sobre testes por meio de treinamentos internos na empresa (ISTQB, 2018).

Para Garousi et al. (2020b), além da capacitação dos profissionais, é necessária a melhoria dos processos de testes. Os software estão cada vez mais complexos, exigindo testes mais sofisticados e eficientes para mitigar ao máximo a possibilidade de riscos e erros nesses software (Etheredge, 2018). A complexidade dos testes de GUI está presente na quantidade de elementos distribuídos nos fluxos de navegação que os software possuem nos dias de hoje, fazendo com que os testes precisem ser planejados com mais assertividade para que o software seja entregue ao cliente com baixa probabilidade de ocorrência de defeitos.

Buscando sobrepor a complexidade mencionada, soluções foram desenvolvidas para tentar facilitar as atividades dos testadores (Coutinho et al., 2022). Sabendo que há impasse em escolher o que deve ser testado em um software com várias funções, pesquisas sugerem regras de priorização para o que deve ser avaliado nos testes de GUI (Huang et al., 2010; Busjaeger e Xie, 2016; Schaefer, 2005). Adicionalmente, várias ferramentas foram criadas para auxiliar no processo de automação, pois testar um software de maneira manual em sua totalidade é uma atividade exaustiva (Moreira et al., 2017). *Selenium*[2], *Cypress*[3], *Puppeteer*[4], *Appium* e *Sikuli*[5] são exemplos de ferramentas de código aberto conhecidas e utilizadas pelos testadores em suas verificações (Sneha e Malle, 2017).

No entanto, a partir da realização de uma pesquisa com 222 profissionais de testes, foi observado que 49% dos participantes avaliam os seus software de maneira exclusivamente manual (Junior et al., 2021). Além disso, aproximadamente 60% dos testadores informaram que as ferramentas utilizadas no processo automatizado possuem limitações. Diante desses fatos, é relevante identificar qual o contexto da execução dos testes de GUI na indústria, a fim de descobrir, por exemplo, quais são as dores dos profissionais em suas funções, quais técnicas são utilizadas nos testes de GUI e as motivações das escolhas feitas em seus projetos.

Por conta do tema em questão, esse trabalho pode ser o primeiro passo para incentivar o desenvolvimento de novas soluções direcionadas às limitações que os testadores enfrentam com testes de GUI manuais ou automatizados, bem como identificar pontos de melhorias nos processos de testes.

[2] `https://www.selenium.dev/`
[3] `https://www.cypress.io/`
[4] `https://pptr.dev/`
[5] `http://sikulix.com/`

1.6 Organização do Trabalho

Os demais capítulos deste trabalho estão estruturados da seguinte maneira:

- No **Capítulo 2** é apresentada a metodologia definida para a produção da pesquisa;
- No **Capítulo 3** é apresentado a revisão bibliográfica do trabalho, a qual informa conceitos e fundamentos importantes da área de teste de software;
- No **Capítulo 4** são informadas as fases referentes ao questionário online. Primeiramente é abordado sobre o planejamento do questionário, como as questões de pesquisa levantadas, o público alvo escolhido e o design do questionário. Em seguida são informadas as atividades de teste piloto e distribuição do formulário. Por fim os resultados obtidos são apresentados junto com as ameaças a validade da atividade.
- No **Capítulo 5** são exibidos os detalhes a respeito da realização das entrevistas, como o público alvo definido, o critério para escolha dos entrevistados, a criação do termo de consentimento livre e esclarecido, a criação do roteiro de entrevistas e a participação piloto dos profissionais. Por fim os resultados são abordados junto com as ameaças a validade dessa atividade da pesquisa;
- No **Capítulo 6** os resultados obtidos através do questionário *online* e as entrevistas são correlacionadas e discutidos;
- No **Capítulo 7** são apresentadas as conclusões da pesquisa, as suas respectivas contribuições e os trabalhos futuros.

Capítulo 2

Metodologia de Pesquisa

Neste capítulo, são apresentadas a classificação e as etapas de desenvolvimento da pesquisa realizada. Essa pesquisa pode ser classificada segundo os seguintes aspectos (Do Nascimento e Sousa, 2016):

- **Natureza**. A natureza da pesquisa possui caráter **básico** estratégico, procurando contribuir com avanço da ciência por meio do conhecimento obtido sem aplicação prática prevista. A produção teórica foi realizada a partir das respostas obtidas com um questionário *online* e relato de testadores da industria por meio de entrevistas;
- **Objetivos**. O trabalho realizado consiste em uma **pesquisa exploratória**, pois, com a coleta de dados, buscou-se caracterizar o contexto em que alguns profissionais da área de testes de GUI estão inseridos na indústria brasileira de desenvolvimento de software, bem como a identificação de fatores que impactam as suas atividades;
- **Procedimentos Técnicos**. Este trabalho pode ser classificado como **pesquisa de levantamento**, no qual, os testadores foram questionados sobre os contextos profissionais em que eles estão alocados e as variáveis que eles precisam lidar diariamente no desempenho de suas funções;
- **Forma e Abordagem do Problema**. A pesquisa posiciona-se por meio de uma abordagem **quali-quantitativa**, pois, na primeira fase do estudo, houve uma coleta dos dados utilizando questionário *online* e, em seguida, uma análise quantitativa das informações obtidas. Posteriormente, foram coletados dados adicionais por meio de entrevistas.

Na Figura 2.1, são apresentadas as atividades realizadas para atingir os objetivos informados no Capítulo 1. As seguintes etapas e atividades foram realizadas:

A) **Etapa 1 - Revisão de Literatura**. A revisão bibliográfica *adhoc* foi realizada visando à obtenção de um arcabouço de conhecimento sobre a área de qualidade e

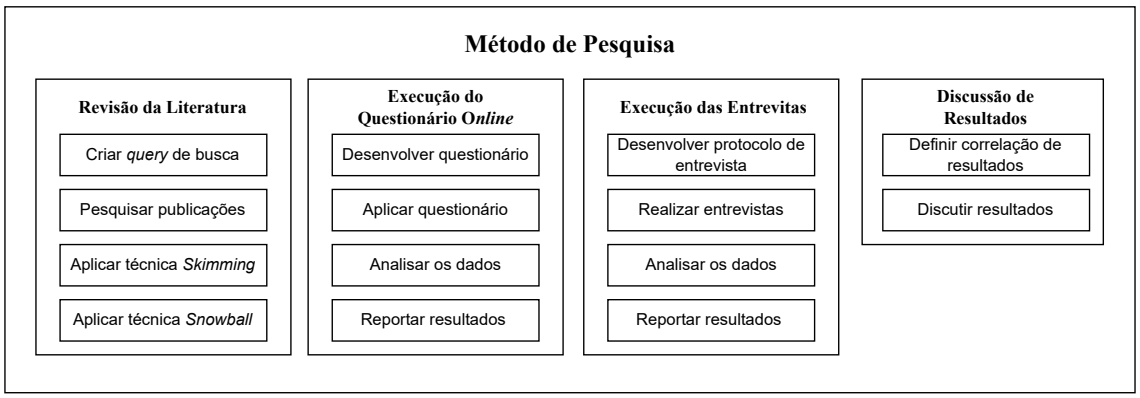

Figura 2.1: Método para realização da pesquisa

teste de software. Esse conhecimento permitiu aprofundamento nos temas pertinentes aos teste de GUI para executar as demais etapas do trabalho, respeitando as definições e os padrões de domínio. Nessa etapa, há quatro atividades:

- **Atividade 1 - Criar *query* de busca**. Para identificar trabalhos relacionados ao tema Teste de GUI, a seguinte *query* foi criada sem limitações de buscas referentes ao ano de publicação dos trabalhos para evitar a exclusão de trabalhos relevantes (publicação mais antiga utilizada é do ano de 1979 e a publicação mais atual é do ano de 2022):

  "query: { Title:("GUI testing" or "GUI test") OR Abstract:("GUI testing" or "GUI test") OR Fulltext:("GUI testing" or "GUI test") }

- **Atividade 2 - Pesquisar publicações**. Trabalhos relacionados ao tema da pesquisa foram encontrados utilizando a *query* criada na Atividade 1, por meio de plataformas de indexação, por exemplo, ACM Digital[1] e DBLP[2];

- **Atividade 3 - Aplicar técnica *skimming***. A técnica *Skimming* (of Notrh Carolina at Chapel Hill, 2014) foi utilizada nas publicações obtidas na Atividade 2 para identificar os trabalhos que apresentaram compreensão de testes de GUI, não somente mencioná-los de forma pontual. Após a identificação, foi obtida uma nova listagem de publicações relevantes para a dissertação;

- **Atividade 4 - Aplicar técnica *snowball***. A aplicação da técnica *Snow Ball* (University, 2014) visou identificar em cada uma das publicações obtidas na Atividade 3 as publicações referenciadas para serem utilizadas na revisão de literatura. Adicionalmente, foram identificadas na plataforma Google Scholar[3] as publicações com maior impacto na área de testes de software.

B) **Etapa 2 - Execução do questionário *online***. Foi utilizado um questionário para a coleta dos dados quantitativos. Objetivando alcançar quantidade razoável de

[1] `https://dl.acm.org/`
[2] `https://dblp.uni-trier.de/`
[3] `https://scholar.google.com.br/`

participações, optou-se pela realização de um questionário no formato *online*. Nessa etapa, há quatro atividades:

- **Atividade 1 - Desenvolver questionário**. O questionário foi elaborado com base nas orientações informadas por Kasunic (2005) e Kitchenham e Pfleeger (2002). Com o questionário, buscou-se entender a relação dos profissionais de testes com os testes de GUI e as perguntas foram organizadas nas seguintes categorias: i) *Caracterização dos respondentes*, ii) *Caracterização das companhias*, iii) *Experiência com testes de GUI manuais* e iv) *Experiência com testes de GUI automatizados*;
- **Atividade 2 - Aplicar questionário**. O questionário elaborado na Atividade 1 foi distribuído para profissionais da área de teste em diversos canais de comunicação.
- **Atividade 3 - Analisar os dados**. Os dados obtidos foram tabulados para a obtenção dos resultados posteriormente discutidos.
- **Atividade 4 - Reportar resultados**. Um relatório foi elaborado com os resultados quantitativos e qualitativos e, em seguida, esses resultados foram discutido.

C) **Etapa 3 - Execução das Entrevistas**. No intuito de coletar dados adicionais qualitativos e aprofundar alguns tópicos expostos no questionário, foram realizadas entrevistas com profissionais de testes. Nessa etapa, há quatro atividades:

- **Atividade 1 - Desenvolver protocolo de entrevista**. Nessa atividade, foi elaborado o protocolo da entrevista com base no *design* elencando por Adams (2015). Esse protocolo está disponível no Apêndice B;
- **Atividade 2 - Realizar entrevistas**. As entrevistas foram realizadas com profissionais de testes de forma semi-estruturada para que o entrevistado tivesse "espaço" para abordar outros assuntos;
- **Atividade 3 - Analisar os dados**. A codificação dos dados ocorreu a partir dos preceitos definidos por Corbin e Strauss (2008) para a criação de Teorias Fundamentadas de Dados. A codificação ocorreu em dois momentos, sendo uma codificação aberta e uma codificação axial;
- **Atividade 4 - Reportar resultados** Após a codificação dos dados, os resultados coletados foram evidenciados na Seção 5.6 do Capítulo 5;

D) **Etapa 4 - Discussão de Resultados** Elaborar uma análise a partir dos resultados obtidos no questionário *online* e nas entrevistas. Nessa etapa, há duas atividades:

- **Atividade 1 - Definir correlação de resultados**. Correlacionar os resultados do questionário *online* e das entrevistas para identificar similaridades e contrastes entre os diferentes contextos de profissionais;
- **Atividade 2 - Discutir resultados**. Discutir os resultados mais relevantes da pesquisa.

Capítulo 3

Revisão Bibliográfica

3.1 Qualidade de Software

A realização de testes de GUI consiste avaliar a qualidade de um software e verificar se há falhas na implementação desenvolvida. Uma forma de se avaliar a qualidade de um software é através da norma ISO/IEC 25010 (25000, 2011), que compreende as oito características de qualidade para um software a seguir:

- **Adequação Funcional**. Representa o grau em que um software fornece funções que atendem as necessidades esperadas e implícitas quando utilizadas em um certo contexto;
- **Desempenho**. Refere-se ao desempenho de um software em relação à quantidade de recursos utilizados nas condições esperadas;
- **Compatibilidade**. Grau em que um software pode trocar informações com outros componentes ou produtos enquanto compartilha o mesmo ambiente de hardware ou software;
- **Usabilidade**. Grau em que um software pode ser utilizado por usuários para atingir objetivos específicos com eficácia e eficiência, satisfazendo um contexto de uso;
- **Confiabilidade**. Capacidade do software de manter seu nível de desempenho sob condições estabelecidas por um período de tempo;
- **Segurança**. Corresponde a proteção de informações por parte de um software, disponibilizando os dados somente para quem possui autorização;
- **Manutenibilidade**. Representa o grau de eficácia e eficiência com que um software pode ser modificado para melhorá-lo, corrigi-lo ou adaptá-lo às mudanças de ambiente ou de requisitos;
- **Portabilidade**. Grau de eficiência e eficácia com que um software pode ser transferido de hardware, software ou ambiente operacional.

3.2 Fundamento de Teste

Nesta seção, são abordados conceitos e definições utilizados em qualquer tipo de testes, incluindo os testes de GUI. A seguir, é explicado como inconsistências são criadas em um software e quais abordagens de teste deve-se seguir para encontrá-las.

3.2.1 Erro, Defeito e Falha

A atividade de teste busca encontrar com antecedência inconsistências nos software que poderiam impactar na sua qualidade. A origem dessas inconsistências (popularmente chamado *bugs*) pode ser explicada com os termos `erro`, `defeito` e `falha`. Uma pessoa pode ter cometido um erro (engano) durante o ciclo de desenvolvimento do software, seja um analista de requisito durante a especificação, seja um desenvolvedor por não ter entendido o que deveria ser feito. Esse erro pode ser a origem de um defeito no código fonte que, posteriormente, com a sua execução, pode retornar uma falha (Klaus Olsen e Ulrich, 2019). Dessa forma, pode-se dizer que uma falha em um software ocorre por conta de um defeito no código fonte criado em decorrência de uma atividade errônea. Para Sharma (2021), os principais motivos para um erro acontecer em um software são:

- Prazos de entrega curtos;
- Falhas humanas;
- Profissionais inexperientes ou insuficientemente qualificados;
- Complexidade de código;
- Uso de tecnologias desconhecidas ou recém lançadas.

3.2.2 Verificação e Validação

Testes de GUI realizados em um software podem ter caráter de verificação ou validação (Coppola e Alégroth, 2022). Por mais que esses termos aparentam ser a mesma atividade, elas possuem diferentes objetivos. A verificação busca identificar se os requisitos especificados foram implementados no desenvolvimento do software. A validação busca garantir que um software atenda às necessidades de seus *steakholders* (Valente, 2020). A diferença existe, pois um software pode ser desenvolvido seguindo estritamente os requisitos identificados; no entanto, a especificação feita não atende as necessidades dos usuários ou o problema do cliente com o passar do tempo muda e uma nova solução é demandada.

3.2.3 Níveis de Testes

Um software pode ser submetido a diferentes níveis de testes. Klaus Olsen e Ulrich (2019) elenca os principais níveis de testes:

- **Teste de unidade**. Concentra-se em testar separadamente os componentes da aplicação para identificar problemas na lógica implementada. Um exemplo é teste realizado em classes e funções;
- **Teste de integração**. Avalia se as interações entre os componentes ou software ocorrem como esperado. Por exemplo, conexões com banco de dados, comunicação entre APIs (*Application Programming Interface*) e uso de elementos de infraestrutura em que o software esteja envolvido;
- **Teste de sistema**. Também chamado de teste de GUI (Valente, 2020), concentra-se em avaliar o comportamento ponta a ponta do software por meio de ações exibidas na interface gráfica;
- **Teste de aceitação**. Avaliação similar ao teste de GUI, mas executado para determinar se um software atende os critérios de aceitação a partir da perspectiva dos *steakholders*.

3.3 Testes de GUI

A interface gráfica de software é um dos principais objetos para a execução de testes (White et al., 2019). Por essa interface, pode-se simular como esses software serão utilizados na sua versão de produção, considerando que boa parte das interações dos usuários finais são feitas na interface gráfica (Qureshi e Nadeem, 2013). Com a popularidade dos dispositivos móveis com interfaces gráficas, como *smartphones*, *smartwatches* e computadores pessoais, *tablets*, testes de GUI tornaram-se um importante tipo de teste para garantir a qualidade dos software suportados por esses dispositivos.

Como em outros níveis de testes, nos testes de GUI o testador adiciona uma entrada no software e, em seguida, analisa as saídas dos testes (Memon et al., 2001). O objetivo é identificar se há conformidade entre o retorno recebido e o resultado esperado. Como em muitos casos o testador não conhece a estrutura interna do software, podendo avaliar o software somente pelos elementos da tela disponíveis, esse tipo de avaliação é chamada de teste de caixa-preta.

3.3.1 Teste de GUI Manual

As execuções dos testes de GUI podem ser feitas utilizando abordagem manual. Os testadores comumente realizam os testes interagindo, analisando e reportando os resultados para a equipe do software (Klaus Olsen e Ulrich, 2019). De acordo com Shruti Malve (2017), um dos ganhos de utilizar a abordagem manual está associado ao baixo custo de implementação, pois o testador precisa apenas do software a ser avaliado e um dispositivo (celular, *desktop* ou *tablet*) para acessá-lo.

No entanto, os testes de GUI manuais apresentam desvantagens. Uma delas está relacionada ao esforço do trabalho, pois consome tempo do testador, principalmente

se houver várias avaliações a serem executadas, afetando o desempenho da atividade de teste (Alégroth et al., 2013). Também, é uma abordagem sujeita a erros, pois conta com humanos para executá-la (Grechanik et al., 2009). Além disso, os testes manuais podem se tornar tediosos por causa de execuções repetitivas (Leitner et al., 2007). As duas maneiras para executar testes de GUI manuais são:

- **Execução manual exploratória**.

  Os testes são pensados e avaliados dinamicamente durante a sua execução, ou seja, eles são feitos com caráter mais informal e não pré-definido (Klaus Olsen e Ulrich, 2019). Como a execução não é totalmente planejada, não há escrita detalhada do que deve ser realizado nos testes e dos resultados esperados. Esse tipo de teste é recomendado quando há pouca ou nenhuma documentação do software a ser testado (Klaus Olsen e Ulrich, 2019). As vantagens são o incentivo ao pensamento crítico do testador e a oportunidade de aprender mais sobre a funcionalidade do software. No entanto, essa avaliação informal possui "fraquezas", pois a eficiência dos testes está diretamente relacionada à experiência e habilidade do testador;

- **Execução manual planejada**.

  A execução planejada dos testes de GUI requer do testador preparação prévia para a sua realização. Nessa forma de executar os testes, o testador identifica e documenta em cenários de testes o que deve ser testado. Em seguida, em casos de testes, são detalhados os passos que precisam ser feitos para atingir o propósito do teste (Rothermel et al., 2001). Um benefício é a possibilidade de reutilizar o teste em outros ciclos de avaliação, mesmo que de maneira manual. Para isso, é recomendada a criação de casos de testes em alto nível sem valores concretos para os dados de entrada e de saída.

3.3.2 Testes de GUI Automatizado

Por causa das deficiências nos testes manuais, há dedicação para o desenvolvimento de novas soluções para automatizar os testes de GUI (Canny et al., 2020). Os testadores realizam o desenvolvimento dos testes de GUI automatizados em diversas linguagens de programação (e.g., *Java*, *Javascript*, *Ruby* ou *Python*) a partir de casos de testes criados em linguagem natural. O teste de GUI automatizado permite a validação do software com rapidez, reduzindo o tempo necessário para aferir a qualidade do software (Karhu et al., 2009) e permitindo a reutilização de teste e execução em diferentes plataformas (Dallal, 2009). Além disso, pode-se aumentar a cobertura dos testes com o uso do processo automatizado (Sharma, 2014).

Apesar das vantagens informadas anteriormente, automatizar qualquer nível de teste, incluindo os executados na interface gráfica, não é atividade trivial, pois requer do testador diversos conhecimentos, como a experiência com ferramentas e distintas linguagens de programação e definição clara do que será automatizado (Tramontana

et al., 2019). Com o avanço das pesquisas e o desenvolvimento relacionado aos testes de GUI, houve mudanças nas soluções para automação de testes. A literatura organiza essas mudanças em três gerações (Alégroth et al., 2016):

- **Primeira Geração**. A interação com a funcionalidade do software era feita a partir das posições específicas do elemento na interface gráfica (eixos x e y), seja um botão, seja um campo de texto (Garousi et al., 2020a). Na década de 1990, essa solução foi um grande passo para complementar o teste manual e reduzir o esforço das equipes de teste. Entretanto, essa geração era sensível aos diferentes tamanhos de tela, fazendo com que o redimensionamento da janela do software prejudicasse os testes, impedindo que os *scripts* de teste identificassem a localização dos elementos que deveriam ser utilizados;

- **Segunda Geração**. A interação com a funcionalidade do software ocorria por meio de identificadores associados a cada elemento da tela, como o nome de um campo no código HTML (*HyperText Markup Language*) ou os seletores CSS (*Cascading Style Sheets*) (Alegroth et al., 2015). Embora essa interação resolvesse o problema citado na primeira geração, no momento do desenvolvimento dos testes automatizados, o testador ainda precisava configurar o seu código fonte para ser compatível com os diferentes dispositivos e navegadores em que o software fosse executado. No momento, essa geração de testes de GUI é a mais utilizada pela indústria (Garousi et al., 2020a). Há diversas ferramentas para atender essa proposta, tanto de código aberto, como *Selenium*[1], *Cypress*[2] e *Puppeteer*[3], quanto comerciais, como *Micro Focus Unified Functional Testing (UFT)* [4] e *TestComplete*[5];

- **Terceira Geração**. Também conhecida como testes de GUI visuais (*Visual GUI Testing* - VGT), está associada ao reconhecimento de imagens (Alegroth et al., 2014). A automação realizada pelo testador interage com os elementos do software após identificar seu formato gráfico, por exemplo, o *script* de teste saber que precisa clicar em um ícone quadrado de cor verde. Sua principal vantagem é no uso similar da aplicação como um usuário faria, junto com a validação visual dos elementos da interface. Uma das principais ferramentas que realizam essa atividade é o *Sikuli*[6]. Problemas como o não reconhecimento de elementos visuais e a baixa performance na execução do teste são questões que o testador deve considerar na escolha dessa abordagem.

[1] `https://www.selenium.dev/selenium-ide/`
[2] `https://www.cypress.io/`
[3] `https://pptr.dev/`
[4] `https://www.microfocus.com/`
[5] `https://smartbear.com/product/testcomplete`
[6] `http://sikulix.com/`

3.4 Trabalhos Relacionados

A literatura apresenta trabalhos que abordam a temática de testes de GUI. Dentre eles, pode-se listar estudos de natureza qualitativa e quantitativa desenvolvidos em análise de estudos de casos da indústria. A seguir, priorizando trabalhos referentes aos testes de GUI, são discutidas as principais diferenças entre esta pesquisa e os trabalhos relacionados ao tema.

Garousi et al. (2020a), motivados em sanar as necessidade da indústria em desenvolver testes de GUI automatizados, como foco em uma empresa de software da Turquia, avaliaram empiricamente as ferramentas *Sikuli* e *JAutomate*, buscando identificar qual seria a melhor solução para as demandas de projetos da companhia. A partir da análise estática das funções disponíveis em cada ferramenta, na observação do uso das soluções por parte dos profissionais e considerando a opinião dos automatizadores da área de teste, foi concluído que ambas soluções possuem limitações que poderiam impactar negativamente o desenvolvimento de software, por exemplo, gerando falsos negativos com a execução dos testes nesses software.

Alégroth et al. (2018) apresentou um estudo experimental, no qual o foco da pesquisa foi a avaliação da aplicabilidade, dos benefícios e das limitações dos testes de GUI visuais automatizados na indústria, principalmente no contexto de projetos de desenvolvimento de software ágeis que possuem integração contínua. Foi observado como a equipe de desenvolvimento web lida com o uso da ferramenta de testes de GUI visual *Eye Automate*[7]. Os resultados da análise qualitativa e quantitativa mostraram que *Eye Automate* provê benefícios como a capacidade de encontrar defeitos. No entanto, possui problemas como longo tempo de execução do *script* de teste, alto custos de manutenção e dificuldade para análise de resultados não determinísticos, como falsos positivos.

Alégroth et al. (2021) propuseram-se um guia de boas práticas para o desenvolvimento de testes de GUI automatizados de terceira geração para uma empresa da Suécia. Para isso, cinco casos de testes foram automatizados, feitos anteriormente de forma manual, com o uso das ferramentas *Sikuli* e *JAutomate*[8], visando identificar o impacto do guia de boas práticas no ciclo de testes. A partir da percepção dos profissionais de testes da indústria, os resultados qualitativos mostraram que as diretrizes recomendadas foram consideradas inadequadas, supérfluas ou desnecessárias por parte dos testadores. Com as análises realizadas, os autores concluíram que o guia produzido foi ineficaz durante a fase de testes do software, pois as práticas propostas no documento geraram esforços e custos extras na criação dos *scripts* de testes automatizados.

Coppola et al. (2020) entrevistou dez desenvolvedores de distintas empresas da Itália, buscando identificar os problemas mais recorrentes na execução de testes de GUI em plataformas móveis. Com os resultados obtidos, os autores puderam elencar vários

[7] `https://eyeautomate.com/`
[8] `https://jautomate.com/`

problemas associados à execução de testes de GUI, como a dificuldade de executar um mesmo teste em plataformas diferentes (e.g., *IOS* e *Android*), considerando as peculiaridades de cada sistema operacional. Além disso, foi observada dificuldades na manutenção dos *scripts* de testes automatizados.

Ye et al. (2021) também pesquisou sobre os testes de GUI executados em dispositivos móveis, focando nas dificuldades em testar aplicativos de entretenimento. Na condução da pesquisa, os autores realizaram entrevistas e aplicaram questionários com os integrantes da equipe de desenvolvimento de uma empresa. Os resultados obtidos mostraram que a localização dos elementos gráficos na interface são uma das maiores dificuldades nos testes realizados em plataformas móveis com *scripts* automatizados.

Apesar das pesquisas supracitadas estarem relacionadas aos testes de GUI na indústria, elas estão mais relacionadas com os testes automatizados, principalmente os de terceira geração, excluindo as demais gerações de testes de GUI em suas considerações. Por outro lado, o objetivo do trabalho realizado na dissertação foi analisar as práticas para a criação e execução de testes de GUI, considerando atividades manuais e automatizadas de qualquer geração feitas pelos profissionais da indústria.

3.5 Síntese de Capítulo

Neste capítulo, foram apresentados os principais conceitos referente a área de qualidade de software, como níveis de testes existentes, a diferença entre erro, defeito e falha e as distintas gerações de testes de GUI. O objetivo principal foi contextualizar o tema a ser abordado na dissertação. No próximo capítulo é detalhado as etapas da atividade do questionário *online*.

Capítulo 4

Questionário *Online*

4.1 Questões de Pesquisa

O questionário *online* foi produzido com o intuito de responder as seguintes questões de pesquisa apresentadas na seção 1.4:

- QP1.Como os profissionais executam os testes de GUI manuais?
- QP2.Como os profissionais executam os testes de GUI automatizados?
- QP3.Quais soluções os profissionais comumente utilizam para registrar os erros e as falhas encontrados no projeto de software?

4.2 *Design* do Questionário

Foram considerados apenas profissionais que realizam ou realizaram atividades de teste de software. Para isso, no início do questionário, foi inserida uma pergunta de controle para identificar se o respondente executou testes de GUI profissionalmente. Além dessa pergunta, o questionário tinha 32 questões, sendo 27 questões obrigatórias (84%) e 5 questões opcionais (16%). Essas questões são apresentadas no **Apêndice A**, que também mostra o termo de consentimento livre e esclarecido. Dentre essas questões, 25 questões (78%) foram fechadas (múltiplas alternativas) e 7 questões (22%) foram abertas (discursivas). As questões foram agrupadas em quatro seções:

- **Caracterização dos respondentes**. Há questões relacionadas aos dados demográficos dos respondentes e níveis educacionais;
- **Caracterização das companhias**. Há questões relacionadas aos locais de trabalho dos respondentes, qual o modelo de desenvolvimento utilizado na empresa, quem são os indivíduos responsáveis pelos testes e para quais plataformas os software eram desenvolvidos;

- **Experiência com testes de GUI manuais**. Há questões para identificar como os testes manuais são feitos pelos profissionais, além de entender como eles reportam os *bugs* encontrados durante a execução dos testes;
- **Experiência com testes de GUI automatizados**. Há questões sobre as ferramentas utilizadas no processo de testes de GUI automatizados, sua funcionalidade e limitações de uso, assim como as linguagens de programação empregadas no processo automatizado.

4.3 Distribuição do Formulário

Antes da distribuição do questionário, foi conduzido um estudo piloto para avaliar a qualidade das questões. Para isso, esse questionário foi enviado para alguns profissionais de testes. Dentre os dias 8 e 9 de março de 2021, quatro testadores responderam às perguntas e, em seguida, sugeriram mudanças na redação das questões. As sugestões foram avaliadas e atendidas visando deixar o texto mais entendível. Também, foi pedido a esses respondentes informarem quanto tempo levaram para responder as perguntas. Constatou-se que o questionário pode ser respondido em 10 minutos em média.

O aplicativo Google Formulários[1] foi utilizado para distribuir o questionário eletronicamente. Em seguida, ele foi enviado para três plataformas:

- *LinkedIn*: Mensagem direta para 675 profissionais e publicação em oito grupos de tecnologia;
- *Email*: Mensagem para 100 companhias de tecnologia;
- *Facebook*: Publicação em oito grupos sobre educação e tecnologia.

4.4 Análise dos Dados

Os seguintes critérios foram adotados para o processamentos dos dados coletados:

1. Respostas em branco ou não associadas às perguntas não foram contabilizadas durante a codificação dos dados;
2. A soma do percentual de respostas poderiam passar de 100% quando a questão aceitava mais de uma alternativa;
3. A fim de evitar erros na contagem do percentual das perguntas fechadas, a soma foi feita por dois revisores, que, posteriormente, compararam manualmente os valores obtidos e discutiram sobre discrepâncias, buscando consenso;
4. A codificação das respostas abertas também foi realizada por dois revisores que compararam as interpretações obtidas dos dados.

[1] https://www.google.com/forms

4.5 Resultados

4.5.1 Perfil dos Respondentes

O questionário *online* foi compartilhado com profissionais que atuam com desenvolvimento de software e companhias de tecnologia. A partir dos convites enviados, foram recebidas 222 respostas de testadores localizados em boa parte dos estados brasileiros. Entre eles, é possível destacar São Paulo (32%), Rio Grande do Sul (18%) e Paraná (12%). A respeito do nível educacional dos testadores, foi observado que 5% possui Ensino Médio, 1% Ensino Técnico e 94% Ensino Superior.

Entre os respondentes com curso superior, 36% dos respondentes são formados em Análise e Desenvolvimento de Sistemas, 27% dos respondentes concluíram o curso de Sistemas da Informação e 13% dos respondentes terminaram o curso de Ciência da Computação. Além disso, 14% dos respondentes possuem outros cursos relacionados a tecnologia, como Segurança da Informação e Gerenciamento da Tecnologia. Os demais respondentes (10%) são graduados em cursos não associados à área de tecnologia, como Engenharia Química e Educação Física. Adicionalmente, 27% dos respondentes são certificados na área de testes (Tabela 4.1[2]) e 9% dos respondentes possuem mais de uma certificação.

Tabela 4.1: Certificação dos Testadores de Software

Certificação de Teste	# Testadores certificados
CTFL	58
CTFL-AT	17
CTAL-TAE	7
CTAL-TM	5
CTAL-TA	3
CBTS	3
CTFL-TM	1
CTFL-PT	1
CTFL-MBT	1
CTAL-TTA	1
CTAL-SEC	1
CTAL-AT	1
BSTQB	1

Acerca dos cargos dos testadores nas companhias em que atuam, foi identificado que 60,6% dos respondentes atuam como Analista de Testes, 10,9% dos respondentes atuam como Engenheiro de Qualidade, 8,6% dos respondentes atuam como Gerente, 7,2% dos respondentes atuam como Desenvolvedores de Software e 12,7% dos respondentes atuam em outros cargos (e.g., Analistas de Suporte e Analista

[2]Siglas das certificações podem ser consultadas em `https://bstqb.org.br/b9/tudo-sobre`

de Infraestrutura). Também, foi questionado se os respondentes utilizam alguma linguagem de programação além das atividades relacionadas aos testes. Como resultado, 72,5% dos respondentes indicaram que utilizam pelo menos uma linguagem de programação, sendo as respostas mais citadas *Java* (20%), *JavaScript* (18,3%), *Python* (12,6%) e *Ruby* (12,1%).

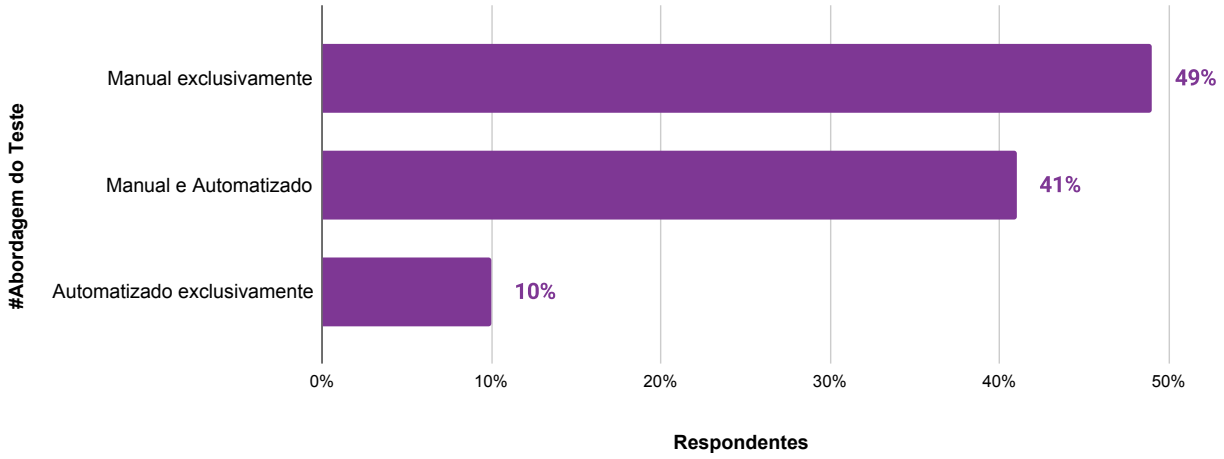

Figura 4.1: Abordagens para realização de testes de GUI

Na Figura 4.1, é apresentado como os testadores executam os testes de GUI. Quase metade dos respondentes realiza os testes de GUI exclusivamente de forma manual, enquanto a minoria faz exclusivamente de maneira automatizada. Além disso, foi solicitado aos respondentes informarem quantos anos de experiência possuem com testes de GUI, sendo que os respondentes que realizaram testes manuais e automatizados responderam duas vezes, uma por tipo de abordagem. Na Tabela 4.2, é indicado o tempo de experiência dos profissionais separados por abordagens adotadas para os testes de GUI, ou seja, testes manuais exclusivamente, testes automatizados exclusivamente e testes manuais e automatizados.

Tabela 4.2: Tempo de experiência dos respondentes

Tempo de experiência	Teste manual somente	Teste automatizado somente	Ambas abordagens	
			Manual	Automatizado
<1 ano	14%	19%	14%	14%
1 a 3 anos	46%	14%	28%	32%
3 a 5 anos	9%	19%	26%	24%
5 a 8 anos	22%	19%	18%	17%
>8 anos	9%	29%	14%	13%

Percebeu-se que parte do grupo de testes manuais tem entre 1 e 3 anos de experiência. No grupo automatizado, a experiência variou bastante entre os respondentes, porém há maior quantidade de respondentes compondo o grupo de testadores com mais de oito anos de experiência. Para os respondentes que executam testes manuais e

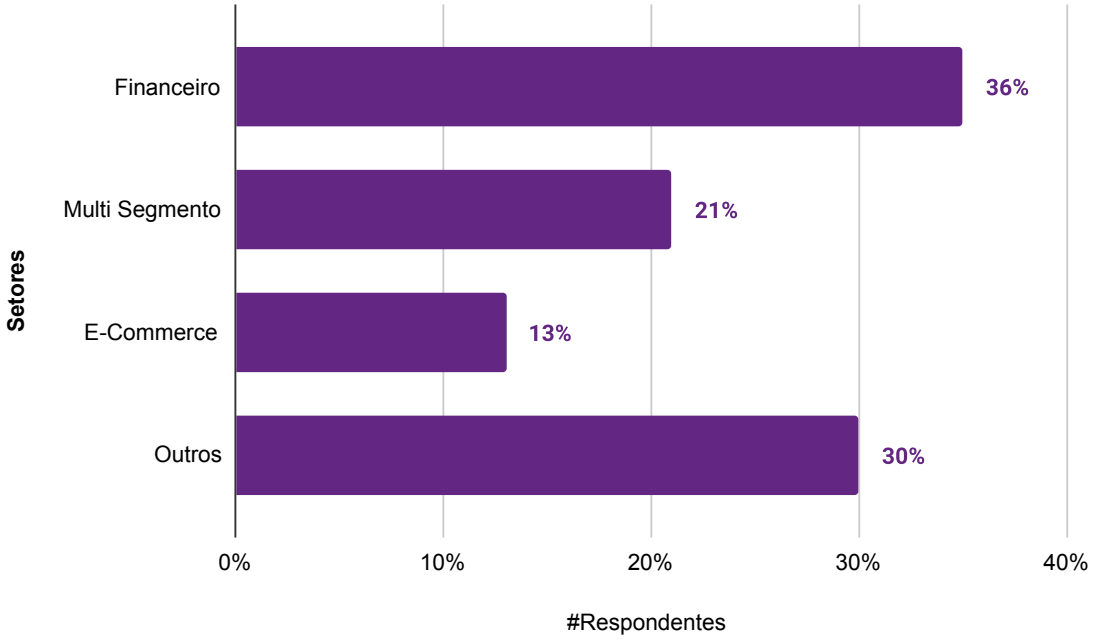

Figura 4.2: Segmento da empresa dos respondentes

automatizados, foi observado que a maioria possuem entre 1 a 5 anos de experiência em ambas abordagens de execução de testes.

4.5.2 Caracterização das Empresas

Nesta seção, são apresentados os resultados referentes às empresas em que os respondentes trabalham. Foi categorizado como "Multi Segmento" as empresas que atuam em mais de um setor. Como informado na Figura 4.2, os respondentes têm testando software de diferentes setores, como Financeiro, Multi Segmento e *E-Commerce*. Outros domínios totalizam 30%, como Recursos Humanos, Saúde, Logística, Comunicação, Alimentício e Games.

Sobre o modelo de desenvolvimento utilizado nas empresas, 23,64% (52 respondentes) citaram Scrum ou Kanban. Contudo, *Scrum* é um *framework* para gerenciamento de projetos e *Kanban* uma ferramenta de gerenciamento visual de trabalho. No entanto, sabendo que os profissionais de software associam *Scrum* e *Kanban* às metodologias ágeis, foram adicionadas essas respostas para a categoria "Àgil". Assim como informado na Figura 4.3, 43% dos respondentes trabalham com métodos Ágeis, 35% dos respondentes usam o modelo de desenvolvimento incremental, 15% dos respondentes usam o modelo em cascata e 7% dos respondentes usam outras soluções.

Também, foi questionado aos respondentes quem são os responsáveis pela realização dos testes de GUI. Dentre as respostas, 64% dos respondentes informaram que a

responsabilidade é exclusiva da equipe de qualidade de software e 16% dos respondentes relataram que tanto a equipe de qualidade quanto programador que desenvolveu a funcionalidade do software compartilham a responsabilidade. Os demais 20% dos respondentes citaram outras respostas, como o programador que não desenvolveu a funcionalidade que precisa ser testada (4,1%) e o time de suporte a usuário (3,4%). Sobre os software testados pelos respondentes, foi observado que 46,4% dos respondentes testam software Web, 33,8% dos respondentes avaliam software para dispositivos *mobile* e 19,8% dos respondentes software *desktop*.

4.5.3 QP1 - Como os profissionais executam os testes de GUI manuais?

Foi questionado aos respondentes sobre os testes manuais. Após a coleta de dados, foram divididos os respondentes em dois grupos: (i) aqueles que realizam ou realizaram apenas testes manuais e (ii) aqueles que realizam ou realizaram testes manuais e automatizados. No Grupo (i), a maioria dos respondentes (86%) relatou ser responsáveis pelo planejamento e pela execução dos testes de GUI, 11% dos respondentes relataram que executam apenas testes manuais e 3% dos respondentes informaram que apenas planejam os testes. No Grupo (ii), foram encontrados índices semelhantes, pois 93,6% dos respondentes relataram que realizam o planejamento e a execução dos testes, 3,2% dos respondentes apenas executam testes e 3,2% dos respondentes planejam apenas testes.

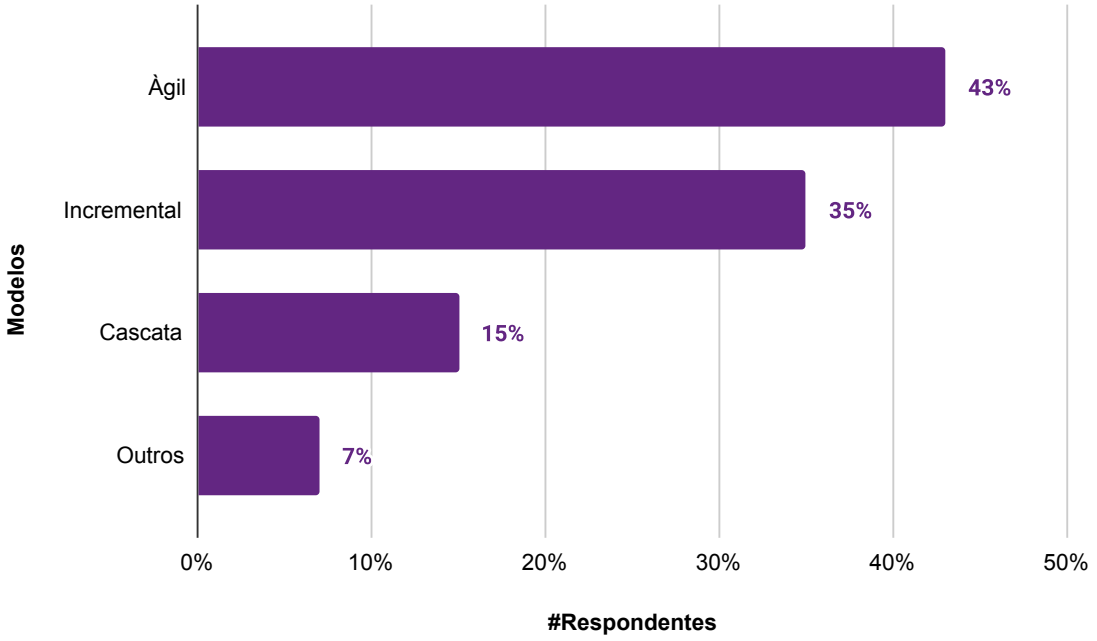

Figura 4.3: Modelos de desenvolvimento utilizados nos projetos

Outro pergunta aos profissionais estava relacionada às técnicas utilizadas para dar suporte aos testes manuais. No Grupo (i), 44% dos respondentes relataram que usam casos de teste para apoiar o processo de teste, 36% dos respondentes testam de forma exploratória e 19% dos respondentes costumam usar *checklist*. No Grupo (ii), foi identificado ligeiro aumento na quantidade de respondentes que usam casos de teste (52%), 22% dos respondentes realizam testes de forma exploratória e 16% dos respondentes usam *checklist*. Adicionalmente, 46% dos respondentes informaram que testam novas funções dos software e, parcialmente, as antigas funções, 31% dos respondentes testam todas as funções dos software e 23% dos respondentes responderam outros, por exemplo, "apenas funções antigas" e "o que impacta a funcionalidade desenvolvida".

4.5.4 QP2 - Como os profissionais executam os testes de GUI automatizados?

Foi questionado aos respondentes que realizam testes de GUI automatizados qual linguagem de programação é utilizada no processo de automação. Assim como foi realizado com os dados coletados com os testadores manuais, foram divididos os respondentes em dois grupos: (i) aqueles que apenas realizam testes automatizados e (ii) aqueles que realizam testes automatizados e manuais. Como pode ser observado na Figura 4.4, no Grupo (i) as linguagens mais citadas foram *Java*, *Python*, *Ruby*, *Javascript* e outras (12%) (*TypeScript*, *Kotlin* e C#), nessa ordem. Já no Grupo (ii) as mais citadas foram *Java*, *Javascript*, *Ruby* e *Python* . Eles também citaram outras (*TypeScript*, *Kotlin*, *C#*, *.net*, *Scala*, *Swift* e *Visual Basic*). Nessa questão, o respondente poderia informar mais de uma linguagem de programação.

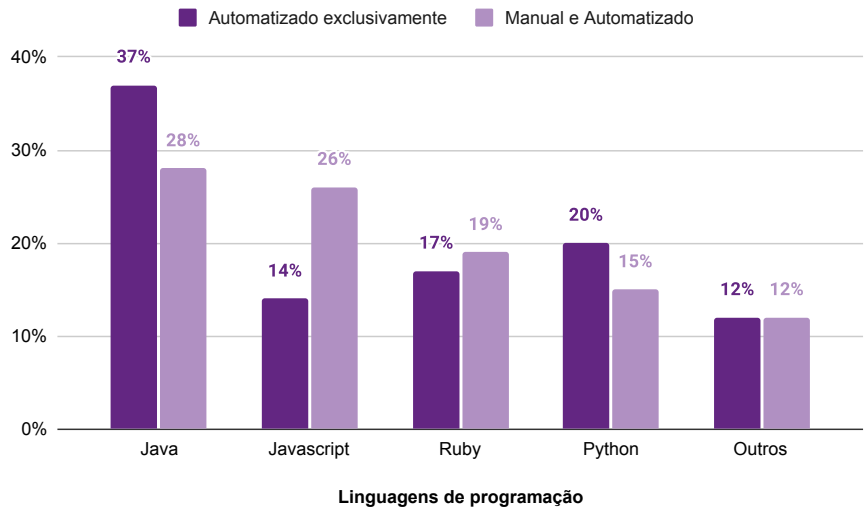

Figura 4.4: Linguagens de programação utilizada pelos respondentes para automação de testes de GUI

Também, foi questionado aos respondentes quem escolhe os *frameworks* e as ferramentas utilizadas no processo de testes de GUI automatizados. Para o Grupo (i), 22% dos respondentes responderam que a equipe de Qualidade foi a responsável pela escolha, 22% dos respondentes informaram que o testador individualmente, 17% dos respondentes disseram que a liderança da empresa determina as ferramentas, 13% dos respondentes responderam que o cliente do projeto decide e 26% dos respondentes informaram "Outros", como o proprietário do projeto e equipe de desenvolvimento. Para o Grupo (ii), 36% dos respondentes responderam que a equipe de Qualidade decide internamente, 20% dos respondentes informaram que o testador escolhe, 16% dos respondentes disseram que a liderança da empresa determina e 28% dos respondentes relataram "Outros".

Na Tabela 4.3, são mostrados os *frameworks* mais citados para o processo de automação de testes de GUI. As ferramentas mais citadas foram *Selenium*, *Appium*, *Cypress* e *TestComplete*. A categoria "Outros" compreende as ferramentas citadas por apenas um participante, por exemplo, *Autoit*, *Android Studio* e *Orange Testing*. Além disso, relataram que geralmente usam as soluções citadas, junto com *frameworks* adicionais, como *Cucumber* (47%), *Capybara* (23%) e *Behave* (8%). Também, 22% dos respondentes informaram outras ferramentas, por exemplo, *Watir*, *Percy* e *Rest Sharp*. Nessa questão, o respondente poderia escolher mais de uma opção.

Tabela 4.3: Ferramentas/*Frameworks* mais citados

Ferramentas/*Frameworks*	**#Respondentes**	**%**
Selenium	91	35,97%
Appium	53	20,95%
Cypress	24	9,49%
Outros	19	7,51%
TestComplete	13	5,14%
Sikuli	10	3,95%
SilkTest	8	3,16%
Robot Framework	7	2,77%
Microsoft Coded UI Tests	6	2,37%
Oracle Application Testing Suite	4	1,58%
UFT	4	1,58%
IBM Rational Functional Tester	3	1,19%
Ranorex	3	1,19%
Protractor	2	0,79%
TestCafe	2	0,79%
VS Code	2	0,79%
Webdriver IO	2	0,79%

Buscando entender os desafios em utilizar os *frameworks* de automação mencionadas, foi questionado aos respondentes se as soluções disponíveis atendem as necessidades do projeto. Os resultados para ambos os grupos (Grupo (i) e Grupo (ii)) são apresentados na Figura 4.5. De maneira geral, comparando as respostas, foi observado que (1) a maioria concorda que as ferramentas e os *frameworks* atendem às suas necessidades, (2) as respostas negativas foram mais significativas para o Grupo (ii) - 15% representam a soma dos respondentes que informaram *parcialmente* e *não atendem*.

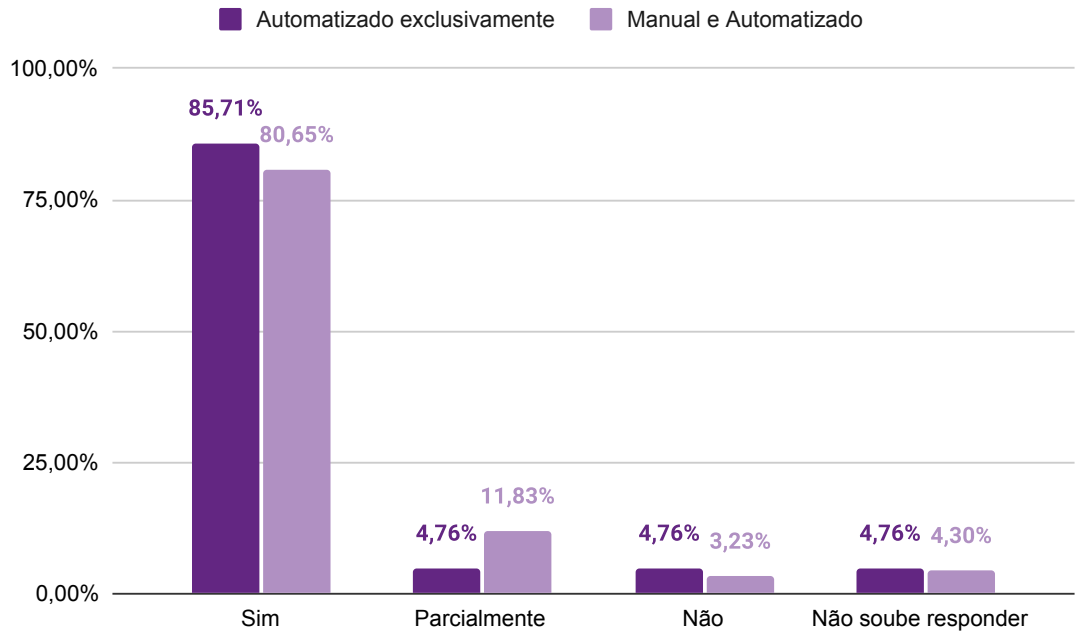

Figura 4.5: Percepção dos respondentes para com as ferramentas de automação de testes de GUI

O uso de *frameworks* é essencial em testes de GUI automatizados, embora os respondentes tenham apontadas algumas limitações referentes às soluções utilizadas no processo de automação. As respostas foram codificadas e associadas a uma ou mais subcaracterísticas de qualidade do produto (ISO/IEC 25010 [37]). Foram identificadas doze limitações diferentes (Tabela 4.4). Observou-se que mais de 40% dos respondentes não informaram limitações. No geral, eles relataram que (i) as ferramentas atendem às suas necessidades ou (ii) não conhecem limitações (falta de conhecimento). Por outro lado, as limitações de qualidade mais citadas referem-se à Completude Funcional (17,2%), Comportamento Temporal (7,8%) e Interoperabilidade (5,2%).

Em relação à proporção de testes de GUI automatizados, foram encontradas diferenças entre o Grupo (i) e o Grupo (ii). Os respondentes do Grupo (i) afirmaram que:

Tabela 4.4: Limitações das Ferramentas/*Frameworks*

Limitações	**#Respondentes**	**%**
Não informou Limitações	49	42,1%
Completude Funcional	20	17,2%
Outros	13	11,2%
Comportamento Temporal	9	7,8%
Interoperabilidade	6	5,1%
Adaptabilidade	5	4,3%
Adequação Funcional	3	2,6%
Modificabilidade	3	2,6%
Operabilidade	3	2,6%
Capacidade	3	0,9%
Correção Funcional	2	0,9%
Instalabilidade	2	0,9%
Aprendizagem	2	0,9%
Utilização de Recursos	2	0,9%

- Para 9% dos respondentes: 100% dos testes são automatizados;
- Para 29% dos respondentes: 70% a 90% dos testes são automatizados;
- Para 14% dos respondentes: 50% a 70% dos testes são automatizados;
- Para 24% dos respondentes: 30% a 50% dos testes são automatizados;
- Para 24% dos respondentes: 10% a 30% dos testes são automatizados.

Por outro lado, os respondentes do Grupo (ii) afirmaram que:

- Para 2% dos respondentes: 100% dos testes são automatizados;
- Para 17% dos respondentes: 70% a 90% dos testes são automatizados;
- Para 26% dos respondentes: 50% a 70% dos testes são automatizados;
- Para 24% dos respondentes: 30% a 50% dos testes são automatizados;
- Para 31% dos respondentes: 10% a 30% dos testes são automatizados;

Assim, para o Grupo (i), constata-se que quase 10% dos respondentes relataram que todos os testes são executados automaticamente e 38% dos respondentes relataram que de 70% a 100% de todos os testes são automatizados. Em comparação com o Grupo (ii), essa quantidade é menor (19%). Também, foi verificado que quase metade dos respondentes relatou que apenas 10% a 50% de todos os testes são automatizados.

Em relação à frequência com que os testes de GUI automatizados são realizados, 39% dos respondentes executam os testes após adicionar novas funções ao software, 25% dos respondentes executam testes após a correção de *bugs*, 22% dos respondentes

executam testes diariamente e 14% dos respondentes informaram outras frequências, por exemplo, três dias na semana, 15 dias na semana e no final dos *sprints*. Como os respondentes puderam escolher mais de uma opção, o somatório de limitações informadas ultrapassam a quantidade de respondentes do questionário *online*. Além dos testes de GUI, os respondentes afirmaram automatizar outros tipos de testes, como testes de API (37%), testes de integração (28%), teste de desempenho (21%) e testes de unidade (12%). Outros tipos receberam uma resposta cada, por exemplo, testes de regressão, testes de *chatbot* e teste de segurança, contabilizando 2% das respostas.

4.5.5 QP3 - Quais soluções os profissionais comumente utilizam para registrar os erros e as falhas encontrados no projeto de software?

Para os testes manuais, os respondentes informaram distintas maneiras de relatar as falhas encontradas durante os testes. As formas mais citadas e a quantidade de respondentes de cada uma são apresentadas na Figura 4.6. Por exemplo, cerca de 41% dos respondentes relataram *bugs* usando cartões no *Kanban*, 18% dos respondentes informaram especificamente a ferramenta *Jira*, 11% dos respondentes enviam mensagens pelo *Slack* e 6% dos respondentes utilizam *Mantis*.

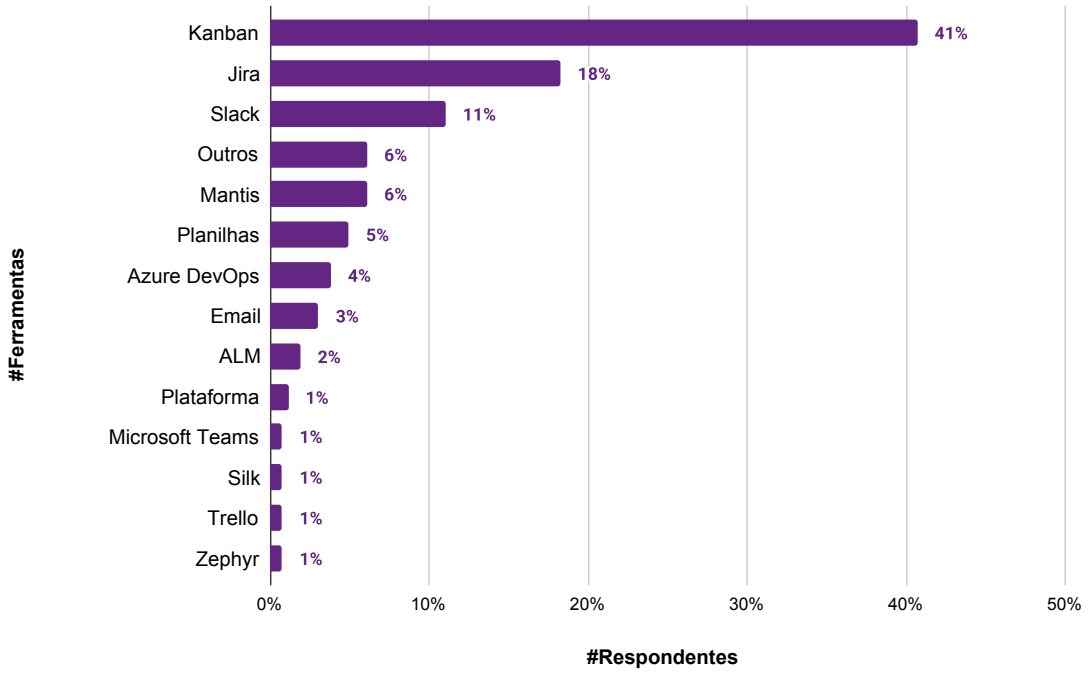

Figura 4.6: Documentação de *bugs* por testadores que executam teste de GUI manual

Para a execução de testes de GUI automatizados, foram citadas várias maneiras de relatar *bugs* (Figura 4.7). As formas mais citadas foram cartões em *Kanban* (40%),

Slack (15%), *Jira* (13%) e *e-mail* (9%). A categoria Outros (11%) inclui *Mantis*, *Gitlab*, *Trello* e conversas com a equipe de desenvolvimento por meio de ligações ou videochamadas.

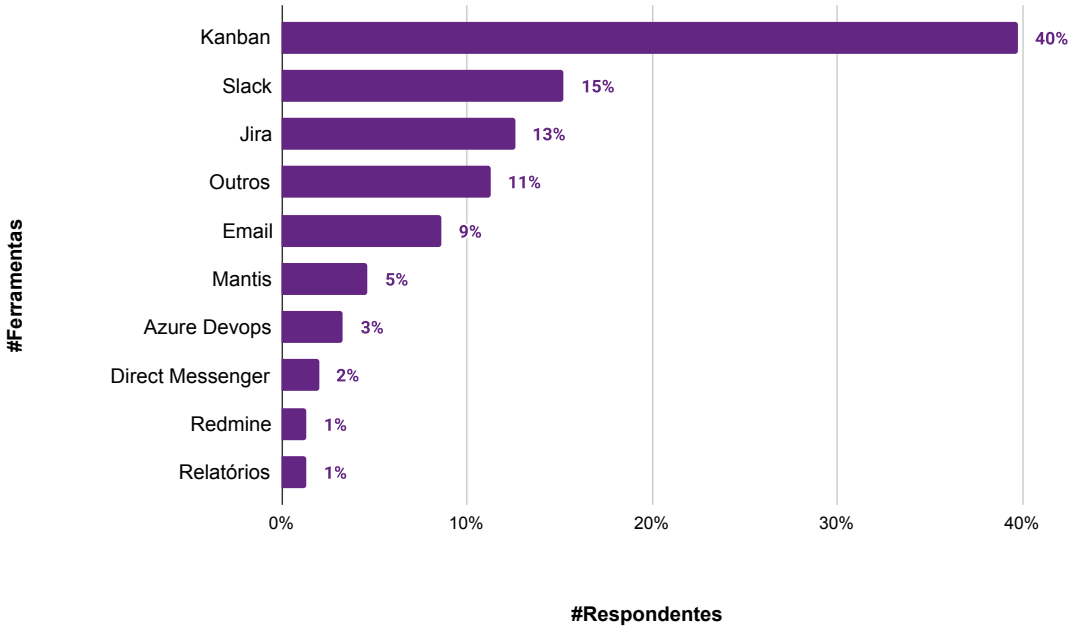

Figura 4.7: Documentação de *bugs* por testadores que executam teste de automatizado

4.6 Ameaças a Validade

Algumas ameaças a validade foram identificadas no decorrer da análise das repostas do questionário *online*:

- **Validade de Construção**. Para minimizar as dificuldades na extração dos dados do estudo, profissionais de software sem experiência com testes GUI, respostas brancas e respostas inválidas foram desconsiderados. O estudo considerou apenas empresas e testadores brasileiros. Ao enviar os convites, observou-se que a maioria das empresas selecionadas era das regiões Sul e Sudeste, o que pode ter uma cultura particular e impactar os resultados. No entanto, testadores de software com diferentes perfis, formação e experiência em testes de GUI responderam ao questionário, minimizando possível viés. Uma possível fragilidade foi identificada no formulário após a finalização da fase de questionário *online*, no qual observou-se que as perguntas referentes às plataformas em que os testadores realizam os testes e as tecnologias utilizadas para automação poderiam ter sido estruturadas para melhor entendimento dos temas.

- **Validade Interna**. Para reduzir o risco de interpretações errôneas das perguntas do questionário, foi realizado um estudo piloto com testadores com experiência profissional antes de enviá-lo aos possíveis participantes finais (respondentes). No estudo piloto, os testadores relataram que o tempo médio para responder a todas as perguntas foi 10 minutos, o qual não foi considerado um tempo excessivo de participação;
- **Validade Externa**. Embora o questionário tenha recebido diversas participações válidas (222 respondentes), elas podem não representar adequadamente toda a população de testadores de software da indústria brasileira. No entanto, acredita-se que os dados coletados podem auxiliar no entendimento do contexto atual de execução dos testes de GUI;
- **Confiabilidade**. Minimizando a ameaça relacionada ao uso de dados de pesquisa, dois pesquisadores realizaram a codificação das respostas abertas e o cálculo das porcentagens dos dados quantitativos para evitar interpretações errôneas das respostas coletadas. As eventuais divergências no processamento das respostas foram discutidas e um consenso foi obtido. Posteriormente, um terceiro pesquisador revisou as informações transcritas no texto para garantir a qualidade dos dados.

4.7 Síntese de Capítulo

Neste capítulo, foram apresentados a execução e os resultados obtidos a partir da resposta de 222 profissionais de testes no questionário *online*. Com esse questionário, pode-se identificar que parte dos testadores é formada em pelo menos um curso superior. No entanto, poucos possuem certificações adicionais na área de testes. Além disso, foi observado que cerca de 48% dos profissionais ainda executam testes de GUI de forma exclusivamente manual. No próximo capítulo, são abordados os detalhes referentes a atividade de entrevistas.

Capítulo 5

Entrevistas

5.1 Questões de Pesquisa

A realização das entrevistas visou complementar os resultados obtidos na etapa do questionário *online*. Para abordar tópicos não tratados anteriormente, foram utilizadas as seguintes questões norteadoras apresentadas na seção 1.4:

- QP4. Quais são as motivações para testar GUI manualmente ou de forma automatizada?
- QP5. Como os testes de GUI manuais são planejados?
- QP6. Quais são as principais limitações no processo de testes de GUI manuais?
- QP7. Como é o processo automatizado de testes de GUI?
- QP8. Quais são as principais limitações no processo de testes de GUI automatizados?

5.2 *Design* da Entrevista

No início do *design* da entrevista, foi elaborado um protocolo (apresentado no **Apêndice B**) que norteou as atividades pré e pós realização das conversas com os entrevistados. Com esse protocolo, foi definido o Termo de Consentimento Livre e Esclarecido (TCLE) com a intenção de informar os detalhes da pesquisa para os entrevistados, os seu direitos e a proteção de dados (**Apêndice B.2**).

As entrevistas ocorreram com o intuito de abordar temas não tratados de forma aprofundada no questionário *online*. Assim, a entrevista focou em uma identificação mais detalhada do perfil dos profissionais e suas motivações para tornarem-se testadores, além das limitações de projeto que impactam a realização das atividades profissionais da área de testes. Dessa forma, as perguntas da entrevista foram

organizadas em três seções de acordo com o perfil de cada entrevistado: (i) Perguntas de identificação do perfil do entrevistado, (ii) Perguntas para profissionais que executam testes de GUI manuais e (iii) Perguntas para profissionais que executam testes de GUI automatizados (**Apêndice B.3**).

5.3 Distribuição dos Convites

Antes de enviar convites aos possíveis participantes, foram realizadas entrevistas piloto com testadores para identificar e realizar ajustes necessários ao protocolo da entrevista. Três entrevistados de perfis distintos foram selecionados para responder às perguntas do roteiro de entrevista:

- Um entrevistado que realiza testes de GUI de forma manual e automatizada;
- Dois entrevistados que realizam testes de GUI exclusivamente manual.

Com a realização das entrevistas piloto, foi possível refinar as perguntas a partir das observações feitas pelos participantes. Além disso, foi identificado que o tempo médio para a realização do roteiro de questões foi de 40 minutos. Outro aspecto analisado durante as entrevistas piloto foi a eficiência das plataformas digitais utilizadas para as entrevistas, desde a coleta digital das assinaturas dos participantes à realização das videoconferências[1].

Em seguida, o convite de participação da entrevista foi enviado ao público alvo formado pelos testadores que executam testes de GUI manuais e automatizados. Esses convites foram priorizados para os testadores que responderam o questionário *online*. Em seguida, outros entrevistados de diferentes senioridades e históricos profissionais foram convidados a participar da entrevista. A estratégia de envio dos convites para as entrevistas foi análoga ao envio de convites para o questionário *online*, seguindo a seguinte ordem:

- Envio de *e-mail* para os 222 participantes que responderam o questionário *online*;
- Envio de *e-mail* para 100 empresas de tecnologia;
- Mensagem direta enviada manualmente para mais de 900 testadores no LinkedIn.
- Publicação de postagem em 22 grupos de tecnologias das redes sociais Facebook e LinkedIn;

5.4 Coleta dos Dados

Antes de efetivamente iniciar a entrevista, foi solicitado aos participantes assinarem o TCLE, no caso de concordarem com as informações contidas no documento.

[1] https://meet.google.com/

Como as entrevistas ocorreram por meio de videoconferências, os participantes assinaram o TCLE de maneira digital utilizando uma plataforma *online*[2]. As entrevistas ocorreram entre os dias 03 de fevereiro de 2022 e 07 de maio de 2022 por meio de videoconferência utilizando a plataforma GoogleMeet. Os critérios considerados para a coleta de dados foram:

- Saturação teórica: descobertas nas entrevistas começaram a diminuir;
- Falta de entrevistados: possíveis entrevistados não tinham interesse ou disponibilidade para participar da pesquisa no período de tempo estipulado para a coleta de dados.

No início da entrevista, foi informado aos entrevistados o objetivo da pesquisa, foi questionado se o entrevistado tinha alguma dúvida e foi solicitada a gravação da entrevista para uso posterior na codificação dos dados. As gravações das conversas ocorreram por meio do software livre OBS Studio[3] em dois computadores distintos, evitando a perda de dados que pudessem ocorrer por eventual falha em uma dos computadores do pesquisador (entrevistador). As gravações foram armazenadas em um HD externo e em plataformas na nuvem criptografadas, visando à criação de *backups* confiáveis. Em ambos armazenamentos, o acessos era restritos ao pesquisador, evitando qualquer tipo de "vazamento" de informações sensíveis dos entrevistados.

5.5 Análise dos Dados

Para iniciar a análise dos dados, foi realizada a transcrição das entrevistas gravadas. Na primeira versão da transcrição, foram escritas de forma integral as falas dos entrevistados no formato de texto corrido e, em seguida, a revisão dos textos. Foram organizadas as respostas dos entrevistados em uma planilha *online* com as respectivas perguntas para facilitar o entendimento dos dados. Depois, visando garantir maior qualidade dos dados, com a ajuda de um segundo pesquisador (aluno de mestrado), foi iniciada a codificação na plataforma *Delve*[4]. Para a realização da análise dos dados, foi utilizada uma abordagem de pesquisa com ênfase em dados qualitativos com o uso de elementos da Teoria Fundamentada em Dados (*Grounded Theory*) Corbin e Strauss (2008). As informações obtidas durante as entrevistas foram processadas em duas fases: (i) codificação aberta e (ii) codificação axial.

A codificação aberta visou extrair os dados em partes distintas e, depois, rotulá-los com códigos que os traduzissem. A abordagem utilizada na codificação aberta foi a indutiva, em que os códigos são criados após a coleta e o processamento das informações. A ordem de prioridade de métodos de codificação foi:

1. Codificação *In Vivo*. Foram utilizadas as próprias palavras do entrevistado para categorizar o dado analisado;

[2] https://www.portaldeassinaturas.com.br/
[3] `https://obsproject.com/`
[4] `https://delvetool.com/`

2. Codificação descritiva. Foi resumido o que foi dito pelo entrevistado em uma frase curta ou termo. Esse método foi utilizado somente quando não foi possível realizar a codificação *In Vivo*.

Com o fim da codificação aberta e a revisão dos códigos extraídos pelos dois pesquisadores, foi iniciada a codificação axial que visou agrupar e categorizar os códigos obtidos anteriormente, considerando os temas associados à fala dos entrevistados. A categorização dos códigos criados por ambos pesquisadores mais uma vez foi revisada e comparada com o objetivo de ter um resultado final consensual e com menos viés. A transcrição das entrevistas realizadas com os profissionais de testes estão disponibilizadas de forma *online*[5].

5.6 Resultados

As seções a seguir apresentam os resultados obtidos com a realização das entrevistas. Primeiramente, são apresentados aspectos referentes ao perfil dos entrevistados, como formação acadêmica, fontes de conhecimento utilizadas e as motivações que levaram o profissional a se tornar um testador. Em seguida, são respondidas as questões de pesquisa definidas no início desse capítulo (RQ4-RQ8), referentes a teste de GUI manual e teste de GUI automatizado. Na Figura 5.1, são apresentadas as categorias chave identificadas com a codificação dos dados das entrevistas.

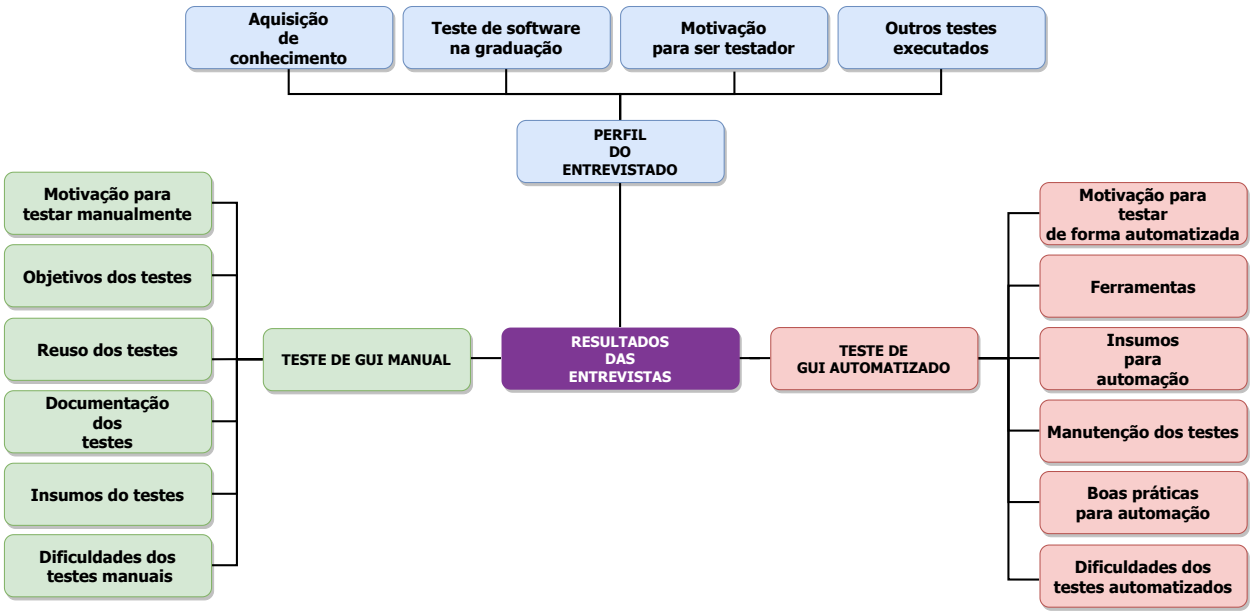

Figura 5.1: Aspectos gerais investigados com as entrevistas

[5]https://doi.org/10.5281/zenodo.7723574

5.6.1 Perfil dos Entrevistados

Foram entrevistados 20 testadores de software. Em relação ao tempo de experiência, há diferença significativa entre os perfis dos entrevistados. Enquanto os entrevistados que executam somente testes manuais estão há 4 anos e 5 meses atuando com GUI (em média), os entrevistados que realizam testes automatizados e manuais possuem 7 anos e 6 meses de experiência (em média). O único entrevistado que realiza exclusivamente testes automatizados atua na área há 3 anos.

Dentre os entrevistados, observou-se três tipos de testadores de software: (i) 1 entrevistado (0,05%) somente automatiza os testes; (ii) 6 entrevistados (0,3%) realizavam somente testes manuais; e (iii) 13 entrevistados (0,65%) realizam testes manuais e testes automatizados. Em relação à escolaridade, 5% (1 entrevistado) possui apenas o Ensino Médio, 10% (2 entrevistados) Superior incompleto, 14% (3 entrevistados) Superior em andamento, 57% (11 entrevistados) Superior Completo e 14% (3 entrevistados) Pós-Graduação. Dos entrevistados que possuem curso Superior e pós-graduação (Figura 5.2), foi identificado que 62% (8 entrevistados) são graduados em Sistemas da Informação, 15% (2 entrevistados) Análise e Desenvolvimento de Sistemas, 15% (2 entrevistados) Ciência da Computação, e 8% (1 entrevistado) Gestão de Processos.

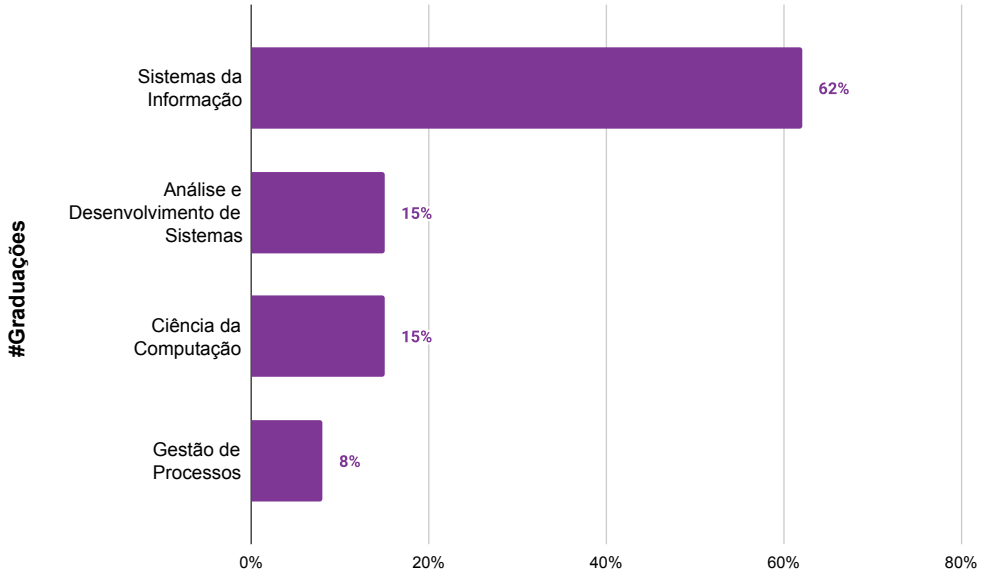

Figura 5.2: Graduação cursada pelos entrevistados

Contato dos entrevistados com teste de software durante a graduação

Para os entrevistados com curso superior, foi questionado como o conteúdo de teste de software foi ministrado na graduação. Como resposta, foi observado que os entrevistados vivenciaram quatro contextos durante o ensino superior (Figura 5.3): (i)

Havia disciplina focada em testes na graduação; (ii) **Testes de software não foi abordado durante a graduação**; (iii) **Tema abordado em disciplina correlacionada**; e (iv) **Tema desprezado pela instituição**.

No contexto **Havia disciplina focada em testes na graduação**, os entrevistados informaram que tiveram uma disciplina exclusiva para abordar o tema teste de software. No entanto, eles também afirmaram que essa disciplina era apresentada somente no final do curso e o conteúdo apresentado havia sido adquirido por eles com a prática profissional. Além disso, foi informado que essa disciplina repetia os conteúdos ministrados em outra disciplinas, como Engenharia de Software, sendo o assunto repassado com "superficialidade". O comentário do Entrevistado #8 exemplifica o contexto: *"[...] na faculdade, fui ter posteriormente uma matéria de Teste de Software, mas como eu já sabia como funcionava tudo na prática, ela acabou sendo um pouco irrelevante por eu já ter todo aquele conhecimento por conta do trabalho. Ela era super superficial, porque ministrava os mesmos conteúdos da matéria da Engenharia de Software, só que com outro nome, até os casos de testes eram mostrados muito mais parecidos com casos de uso do que outra coisa, então dessa maneira que eu vi não acho importante [...]".*

No contexto **Testes de software não foi abordado durante graduação**, alguns entrevistados afirmaram que o tema sobre qualidade e teste de software não foram repassados enquanto eles realizavam curso superior, pois a graduação tinha foco maior na área de programação, levando-os a se especializar em testes com outras fontes de conhecimento, como informado pelo Entrevistado #4 *"[...] durante minha graduação eu tive contato com lógica de programação, sql e até teste de usabilidade, mas nada diretamente para teste de software [...]".*

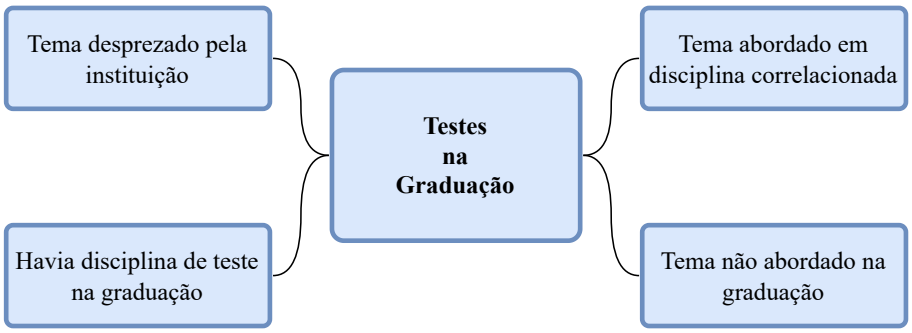

Figura 5.3: Contato dos entrevistados durante a graduação com o tema testes de software

Na contexto **Tema abordado em disciplina correlacionada**, os entrevistados informaram que tiveram contato com o tema teste de Software por meio da disciplina "Engenharia de Software". No entanto, não havia aprofundamento de conteúdo, pois era ministrado prioritariamente conceitos básicos, como as definições de teste caixa branca e caixa preta, sem o estudo de outros aspectos igualmente relevantes para testes de software. A resposta do Entrevistado #9 define esse contexto: *"[...] na*

faculdade eu vi só uma pincelada sobre testes, o professor na aula de Engenharia de Software levava uns papéis impressos falando sobre desenvolvimento e tinha uma coluna pequena na metade da folha que definia o que era teste de software, teste de caixa branca e preta, integração e essas coisas, então era uma coisa muito básica [...]".

No contexto **Tema desprezado pela instituição**, os entrevistados afirmaram notar falta de respeito pelo tema por parte de alguns professores, como dito pelo Entrevistado #11 *"[...] quando o professor comentava sobre os profissionais de testes se referia a eles como "as pessoas que ficam ali no canto testando [...]".* Além disso, foi informado pelo Entrevistado #9 que havia tratamento diferente em sala de aula para os estudantes que estavam trabalhando na área de teste de Software, conforme seu relato *"[...] esse professor que mencionei também era diretor da empresa em que eu estagiava, ele não dava importância nenhuma para área de teste, e quando descobriu que haviam testadores na sala de aula, ele virou as costas para a gente e também ia contra nós [...]".* Adicionalmente, o Entrevistado #15 afirmou que há desincentivo indireto por parte instituição de ensino para que os alunos não façam estágio na área de teste de software, pois não é possível concluir a disciplina de estágio do currículo acadêmico com experiência obtidas como testador de software, como mostrado em seu comentário: *"[...] vários colegas meus não conseguiram eliminar o estágio obrigatório do curso com as experiências que eles possuem na área de testes porque o curso não permite, eu só consegui porque atuo com automação, acho isso um absurdo [...]".*

Ainda abordando os aspectos educacionais, foi questionado aos entrevistados quais seriam os possíveis benefícios para graduados caso o tema teste de software fosse apresentado com mais ênfase nos cursos superiores de tecnologia. Os entrevistados informaram que uma disciplina com esse tema poderia contribuir na apresentação da área teste de software para novos profissionais, pois, em alguns casos, muitos testadores não sabem de sua existência, quais atividades são executadas ou a sua importância em projetos de software.

Outros entrevistados também enfatizaram a necessidade de preparar profissionais para realizar entregas com mais qualidade em seus projetos de desenvolvimento de software no mercado de trabalho. Como foi identificado nas respostas, alguns desenvolvedores aprendem a testar somente quando é exigido pelas empresas em que eles atuam. Além disso, para alguns entrevistados, faz-se necessário difundir a cultura da qualidade em seus projetos.

Aquisição de conhecimento dos entrevistados para atuar com testes de GUI

Foi observado que os testadores para desempenhar as suas atividades de testes, utilizam diversas fontes de conhecimento para conhecer mais sobre os testes de GUI. Essas fontes foram organizadas em quatro categorias (Figura 5.4): (i) **Conteúdo Online**; (ii) **Trabalho**; (iii) **Ensino Formal**; e (iv) **Outros**.

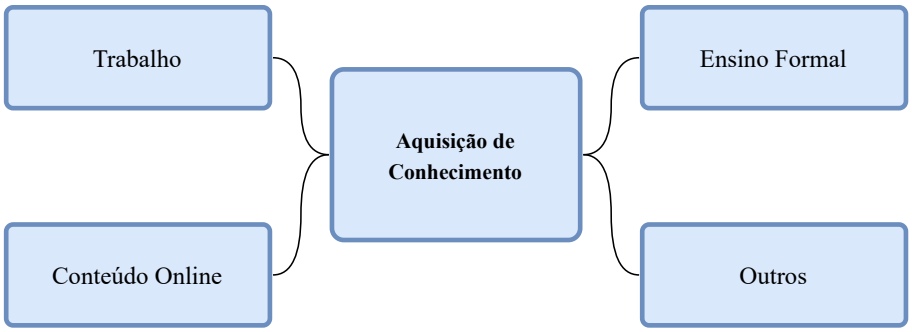

Figura 5.4: Fontes de conhecimento para os entrevistados atuarem como testadores

Dentre as categorias mais citadas estão **Conteúdo Online** e **Trabalho**, ambas com 35% das respostas (Tabela 5.1). Para o tipo de aprendizado obtido de forma *online*, a origem de conhecimento "cursos" foi a mais mencionada (55% dos entrevistados) (principalmente associando à plataforma *Udemy*), 14% dos entrevistados mencionaram *blogs*, 9% dos entrevistados mencionaram *sites* de buscas, 9% dos entrevistados mencionaram artigos, 9% dos entrevistados mencionaram vídeos do *Youtube* e 5% dos entrevistados mencionaram documentação técnica.

Na categoria **Trabalho**, 41% dos entrevistados afirmaram aprender sobre testes a partir de conhecimento tácito (com outro profissional passando a sua experiência sobre a área), 41% dos entrevistados citaram que aprenderam sobre testes executando as demandas passadas no dia a dia de trabalho e 18% dos entrevistados aprenderam com a participação em treinamentos na empresa que atuavam.

A categoria menos mencionada está relacionada ao **Ensino Formal**, por exemplo, por meio de um Curso Superior. Dentre os entrevistados, 67% afirmaram aprender sobre testes enquanto faziam uma Graduação, 17% informaram que aprenderam em uma Pós-Graduação e 17% disseram que aprenderam em Curso Técnico. As fontes de conhecimento que não expressam os demais tipos de aquisição foram categorizados como **Outros**. Das respostas afirmadas para essa categoria, 42% dos entrevistados aprenderam de forma autodidata, 42% dos entrevistados aprenderam por certificações, 8% dos entrevistados aprenderam em livros e 8% dos entrevistados aprenderam em apostilas.

Motivações para o entrevistado se tornar um testador de software

No aspecto profissional do perfil do entrevistado, foram investigados os motivos que impactaram os profissionais na escolha da função de testador de software. Nove motivos foram identificados (Figura 5.5): (i) **Recomendação de amigo**; (ii) **Interesse próprio na área**; (iii) **Insatisfação com emprego atual**; (iv) **Demanda da empresa**; (v) **Motivos financeiros**; (vi) **Dificuldades com programação**; (vii) **Convidado por uma empresa**; (viii) **Recrutamento errôneo**; e (ix) **Não conseguiu entrar em outra área**.

Tabela 5.1: Fontes de conhecimento dos entrevistados

Tipo de Aquisição	Origem	%
Trabalho	Conhecimento Tácito Atividades do dia a dia Treinamento interno na empresa	35%
Conteúdo Online	Site de busca Artigos Cursos Blogs Vídeos Youtube Documentação Técnica	35%
Outros	Autodidata Certificações Livros Apostilas	19%
Ensino Formal	Curso Técnico Faculdade Pós Graduação	10%

A categoria **Recomendação de amigo** foi a mais citada pelos entrevistados, na qual eles afirmaram que iniciaram na área teste de Software por incentivo ou convite de algum conhecido que trabalhava na indústria de desenvolvimento de software como testador. Como pode ser observado nas entrevistas, essa recomendação para a área teste de software ocorreu tanto no começo do ensino superior e na carreira dos profissionais, quanto para os entrevistados que atuavam na área de tecnologia em algum outro contexto.

A categoria **Interesse próprio** motivou os entrevistados a testarem software. A partir do conteúdo sobre testes visto na graduação (mesmo superficialmente como informado pelos entrevistados) e do estudo autodidata realizado, os entrevistados entusiasmaram-se, levando-os a procurarem vagas para atuarem como testadores.

Na categoria **Insatisfação com emprego atual**, foi observado que os entrevistados estavam insatisfeitos com as atividades que realizavam em suas equipes, seja por falta de conhecimento para desempenhá-las ou por não enxergarem valor nas atividades. Além disso, havia insatisfação como um todo sobre o andamento do projeto em que eles estavam alocados, pois, em alguns casos, havia momentos ociosos não utilizados para o desenvolvimento profissional dos entrevistados, não havendo o incentivo para a ampliação de sua competência.

Por causa da falta de testadores nas equipes de desenvolvimento de software, foi identificado que, por conta da **Demanda da empresa** em que atuavam, os entrevistados foram transferidos entre setores e alocados para executarem atividades refe-

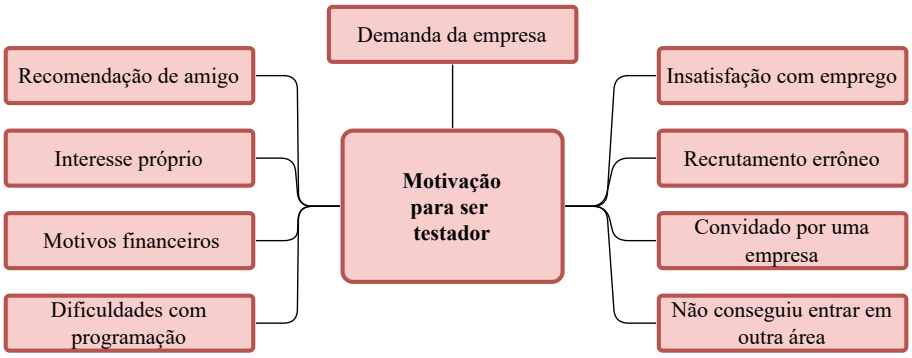

Figura 5.5: Motivos que levaram o entrevistado a iniciar na área de testes de software

rentes à área teste de software. Os entrevistados afirmaram que, no momento dessa transferência interna de setor, eles não possuíam as habilidades necessárias para desempenhar as atividades, mas a empresa ainda assim fez a sua alocação para avaliar a qualidade do software.

A busca de melhores salários compõem a categoria **Motivos financeiros**. Para os entrevistados insatisfeitos com os seus salários, a promessa de remuneração melhor na área de teste de software fizeram com que eles buscassem uma posição para atuar como testador de software. Essa motivação foi citada por entrevistados com carreira no setor de tecnologia e formados em outros setores (*e.g.*, Artes Cênicas), fazendo-os migrar para a área de teste de Software.

Na categoria **Dificuldade com programação**, entrevistados informaram que, pelo fato de não terem conhecimento para desempenhar a função de desenvolvedor de software, eles decidiram se especializar para assumir o papel de testador de software em projetos de tecnologia. Além disso, foi mencionado pelos entrevistados que eles não gostam de codificar, incentivando a sua participação na área de testes de software.

A categoria **Convidado por uma empresa** reflete as respostas referentes aos entrevistados que inicialmente não pensavam em atuar como testador, mas foram convidados por empresas que possuíam vagas para essa posição e que eles não haviam se candidatado, como informado pelo Entrevistado #20: *"[...] Para ser sincero eu não tive uma motivação específica, foi algo que aconteceu. Eu estava para entrar em uma empresa de TI pois eu tinha muita experiência com suporte e manutenção, fiz entrevista com o gestor, mas a vaga que eu ia entrar acabou fechando. Três meses depois o gestor me ligou questionando se eu tinha interesse em uma vaga para trabalhar com o teste de software em uma e acabei aceitando a oferta [...]".*

A categoria **Recrutamento errôneo** faz referência aos entrevistados que participaram de processos seletivos acreditando que a vaga era para um setor da empresa; no entanto, a seleção tinha como objetivo preencher posições para a área teste de software. Como informado pelo Entrevistado #14 somente na contratação realizada pela equipe de recursos humanos da empresa, foi notificado que a vaga era para ser

testador: *"[...] no momento da seleção foi informado que a vaga não era para banco de dados como eu esperava, mas para teste de software [...]"*. Em outro relato, o entrevistado informou que estava fazendo um curso para se especializar na área de tecnologia; no entanto, o entrevistado estava realizando um programa de estágio e não estava ciente disso.

A busca por uma vaga de estágio e a falta de oportunidade contemplam a categoria **Não conseguiu entrar em outra área**. Os entrevistados, que não obtiveram uma vaga de estágio em outra área de tecnologia, aceitaram atuar como testador no primeiro convite que receberam, conforme informado pelo Entrevistado #5: *"[...] entrei na área de testes através de um estágio. Todos os estágios que eu estava procurando exigiam experiências além do que eu tinha, e este da área de testes de software foi o único que eu fui aprovada [...]"*.

Outros testes executados pelos entrevistados

Além de testes de GUI, os entrevistados costumam executar outros tipos de teste (Tabela 5.2), sendo o mais citado o teste de API (15 entrevistados - 49%). Os demais tipos são teste de carga citado por 6 entrevistados (20%), teste de usabilidade citado por 4 entrevistados (13%) e os testes de compatibilidade, performance, unidade, acessibilidade, fumaça e de componente citado por 1 entrevistado (3%) cada. O teste de API pode ter tido mais citações por causa da popularidade do uso de interfaces para a troca de informações por parte das empresas, fazendo com que os testadores precisem avaliar se as requisições enviadas ou recebidas estão com os dados esperados.

Tabela 5.2: Outros tipos de testes realizados pelos entrevistados

Teste de	**#Respondentes**	**%**
API	15	49%
Carga	6	20%
Usabilidade	4	13%
Compatibilidade	1	3%
Performance	1	3%
Unidade	1	3%
Acessibilidade	1	3%
Fumaça	1	3%
Componente	1	3%

Caracterização das Empresas

Foi investigado o posicionamento das empresas em que os entrevistados atuam em relação à execução dos testes de GUI. Observou-se que parte das empresas priorizam a execução manual de teste de GUI, de acordo com os entrevistados que executam testes manuais. Segundo eles, não há alocação de horas no projeto para a automação, explicitando que a gerência prefere testes manuais. Além do mais, quando os

produtos em desenvolvimento são novos, exige avaliação manual em um primeiro momento.

Além disso, um testador mencionou que, em relação aos resultados, a organização em que trabalha procura por atividades que tragam retorno a curto prazo, forçando a equipe de teste a somente realizar testes manuais. Também, foi informado por outro entrevistado que, pela falta de testadores na sua empresa, a prioridade é "apagar incêndios". Assim, o foco está em testar correções de falhas urgentes, não havendo espaço para evolução dos processos de qualidade de software.

Por fim, um entrevistado afirmou que os testes manuais são realizados sem formalidade em seu projeto, pois a execução deles é feita pelo gerente do projeto e não por um profissional qualificado da área de testes.

5.6.2 QP4 - Quais são as motivações para testar GUI manualmente ou de forma automatizada?

No estudo, foi identificada que a maioria doe profissionais ainda realizam testes de GUI de maneira manual. Assim, investigou-se, durante as entrevistas, as motivações que incentivam os testadores a escolher a abordagem manual ou automatizada no momento da execução dos testes de GUI. Primeiramente, são apresentados os motivos para a execução exclusiva dos testes de GUI manual e, em seguida, são abordadas as razões que levaram os testadores a automatizarem testes de GUI.

Motivos para testar somente de maneira manual

Mesmo com a popularidade de ferramentas e técnicas para automatizar testes de GUI, muitos profissionais ainda não automatizam testes. Foram identificadas categorias referentes às razões pelas quais os entrevistados não realizam testes automatizados (Figura 5.6): (i) **Preferência profissional**; (ii) **Problemas educacionais** e (iii) **Contexto de projeto**.

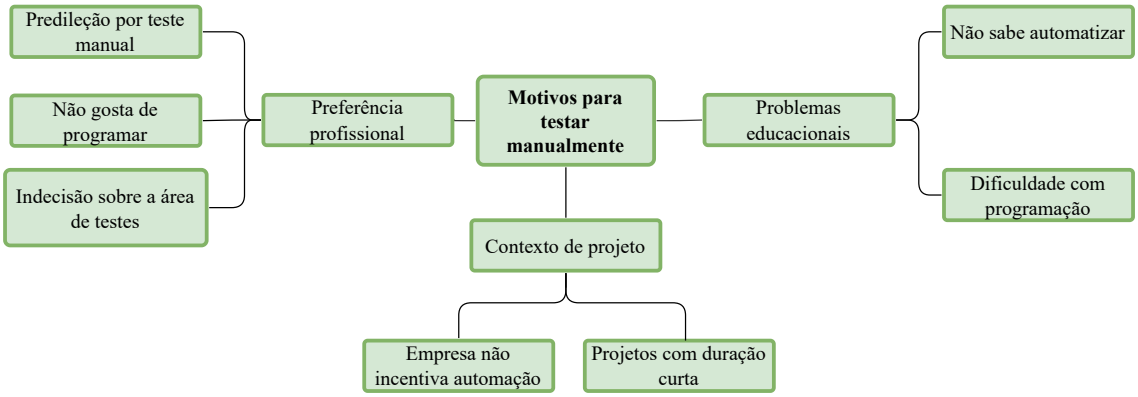

Figura 5.6: Motivos que levam os entrevistados a executar somente testes de GUI manuais

1. **Preferência profissional**

   A seguir, estão os motivos pessoais e profissionais que fazem os entrevistados a não automatizarem os testes. Dentre as tipos encontrados estão: **Predileção por teste manual**, **Não gosta de programar** e **Indecisão sobre a área de testes**.

   - `Predileção por teste manual:` alguns entrevistados preferem realizar tarefas associadas aos testes manuais do que automatizados, ou seja, sentem-se mais à vontade em executar atividades mais atreladas a interpretação de regras de negócio e especificação de requisitos do que a codificação de cenários de testes automatizados;
   - `Não gosta de programar:` como informado por um entrevistado, o ato de programar, seja ele um teste automatizado ou a codificação de um software, não é uma atividade que dê prazer ou satisfação para ele. Por conta dessa preferência, o entrevistado somente executa testes manuais;
   - `Indecisão sobre a área de testes:` entrevistado recém iniciado na área de teste informou que ainda está conhecendo o universo em volta da qualidade de software. Ele estava indeciso se irá continuar realizando somente testes manuais ou se irá se profissionalizar para a automação de testes.

2. **Problemas educacionais**

   Na categoria **Problemas educacionais**, são abordadas as limitações e a falta de conhecimento que impedem a produção de testes de GUI automatizados por parte dos entrevistados. A seguir, são detalhados os problemas: **Não sabe automatizar** e **Dificuldade com programação**.

   - `Não sabe automatizar:` há entrevistados que ainda estão se familiarizando com os testes manuais e não possuem conhecimento sobre automação de testes, impactando na realização de *scripts* automatizados de testes de GUI;
   - `Dificuldade com programação:` alguns entrevistados listaram ter problemas em entender e utilizar a lógica de programação, bem como aplicá-la com alguma linguagem de programação na criação de testes automatizados. Termos como "resistência" e "pavor" foram associados pelos entrevistados com a ação de programar.

3. **Contexto de projeto**

   Na categoria **Contexto de projeto**, é detalhado quais são as variáveis organizacionais que impactam na produção de testes de GUI automatizados, dentre elas estão: **Empresa não incentiva a automação** e **Projetos com duração curta**.

   - `Empresa não incentiva a automação:` muitas empresas alocam os seus testadores somente em atividades relacionadas à criação e execução de testes manuais, não os permitindo que automatizem os testes de

GUI, mesmo que eles tenham as competências necessárias para o serviço. Além disso, como os testadores estão sempre investindo os seus esforços na realização de testes manuais, eles ficam sobrecarregados pelas atividades repetitivas e cansativas, tendo pouco tempo para automatizar qualquer teste;

- `Projetos com duração curta:` projetos que possuem curto prazo para serem entregues e a equipe responsável pela criação do software que não irá dar manutenção no código fonte posteriormente tendem a não receberem automação de teste de GUI, segundo os entrevistados. Nesse caso, o ciclo de desenvolvimento de software não comporta o processo de automatização dos testes, o qual precisa de tempo maior para ser eficiente e trazer benefícios.

Motivos para testar de maneira automatizada

Foram identificados três categorias relacionadas aos motivos que levam os entrevistados a desenvolver competências para executar as tarefas de automação de testes de GUI em seus projetos (Figura 5.7): (i) **Motivos educacionais**; (ii) **Motivos de projeto** e (iii) **Motivos profissionais**.

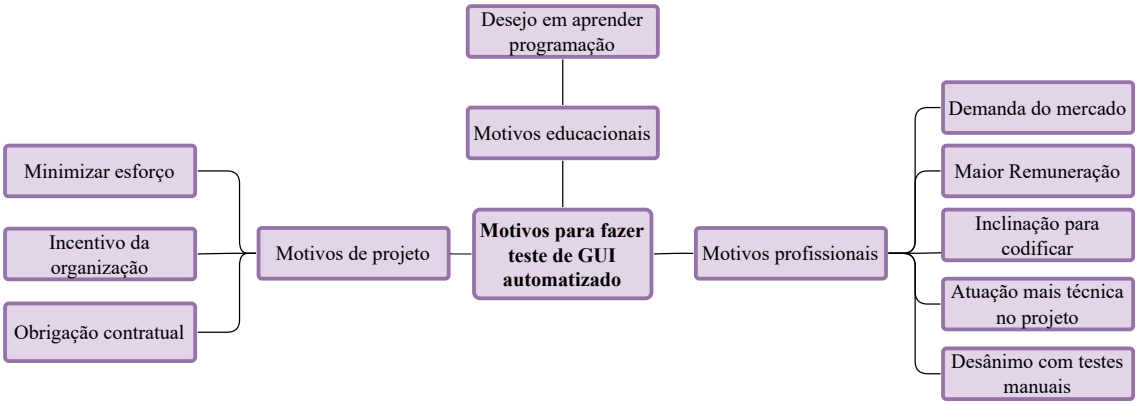

Figura 5.7: Motivos para ser um testador que automatiza teste de GUI

1. **Motivos Educacionais**

   A seguir é apresentado as razões relacionados a problemas educacionais que levaram o testador de software a criar testes de GUI automatizados em seus projetos.

   - `Desejo em aprender programação:` aprender a codificar em uma linguagem de programação motivou um entrevistado a conhecer a área de desenvolvimento de software, levando-o a utilizar a experiência obtida com criação de software para a atividade de automação de testes de GUI.

2. **Motivos de Projeto**

   O contexto de projeto de desenvolvimento de software em que o testador se encontra, também impacta na forma em que o profissional irá executar os testes de GUI. A seguir são apresentadas as motivações de projeto informadas pelos entrevistados:

   - `Minimizar esforço:` diminuir o tempo para executar testes e aumentar a quantidade de validações realizadas no software foram os principais motivos informados pelos entrevistados para que eles dessem início na área de automação de testes. Com a automação, além de conseguir as vantagens mencionadas, o time de teste pode realizar outras atividades menos repetitivas do que os testes de GUI manuais;
   - `Incentivo da organização:` alguns entrevistados foram incentivados pelas empresas em que trabalham a aprenderem sobre automação para aplicarem o conhecimento nos projetos da empresa, visando os ganhos que a automação de teste propicia no fluxo de desenvolvimento de software;
   - `Obrigação contratual:` foi informado que alguns contratos com clientes somente seriam realizados mediante a criação de testes automatizados para avaliar a qualidade do software a ser produzido. Como a automação era algo indispensável para o dono do projeto, a empresa exigia que os testadores criassem testes automatizados de GUI.

3. **Motivos Profissionais**

   Muitos testadores priorizaram a execução de testes de GUI por conta de motivos referentes a sua carreira profissional. Abaixo são apresentados as motivações informadas pelo profissionais da área de testes.

   - `Demanda do mercado:` para alguns entrevistados, o próprio mercado de trabalho gerou a necessidade dos testadores terem as competências necessárias para a automação de testes. Segundo eles, cada vez mais as vagas de emprego divulgadas elencam, dentre os requisitos obrigatórios, conhecimento em programação e automação de testes, além dos gestores das empresas enfatizarem a necessidade do candidato provar que conhecem os *framework* para automação de testes de GUI;
   - `Maior remuneração:` os entrevistados perceberam que os salários de testadores que sabem automatizar testes são maiores do que os de testadores que executam testes de GUI manuais. Mesmo em empresas que não há espaço para automatizar, há valorização salarial maior para profissionais que, pelo menos, tenham as habilidades necessárias para a automação. Por isso, alguns entrevistados informaram que o salário foi um dos motivos para focar em testes automatizados;
   - `Inclinação para codificar:` entrevistados afirmaram que gostam de codificar e a área de automação de testes foi o espaço que eles encontraram

para contemplar essa vontade;

- `Atuação mais técnica no projeto:` desconectar-se das atividades mais relacionadas ao gerenciamento de projeto e teste manual foi um dos motivos identificados na entrevista. Para um entrevistado, o desejo de ser uma referência técnica no projeto de desenvolvimento e conhecer as ferramentas e todo o processo de automação seriam os motivos para automação de testes de GUI;
- `Desânimo com testes manuais:` A falta de interesse em executar testes manuais também foi informada pelos entrevistados, pois realizar somente validações na interface de forma manual pode ser cansativo. Além disso, pode gerar sentimento de estagnação de carreira por conta das atividades repetitivas e "não técnicas".

5.6.3 QP5 - Como os testes de GUI manuais são planejados?

Para entender com mais profundidade como os testes de GUI manuais são planejados para a sua execução posterior nos projetos de desenvolvimento de software, as respostas dos entrevistados foram separadas em categorias referente à atividade de planejamento. Nas seções a seguir, são abordados as categorias (Figura 5.8): (i) **Objetivos do teste**; (ii) **Insumos para os testes**; (iii) **Formato da documentação dos testes de GUI** e o (iv) **Reuso da documentação produzida para os teste de GUI**.

Figura 5.8: Categorias de codificação identificadas para o planejamento dos testes de GUI manuais

Objetivos do teste

Ao planejar e executar o teste de GUI, é necessário identificar o objetivo dos testes, ou seja, qual é o foco da avaliação de qualidade realizada. Foram identificados cinco tipos de objetivos (Figura 5.9): (i) **Funcionalidades essenciais**; (ii) **Validação protótipo**; (iii) **Elementos disponíveis na interface**; (iv) **Responsividade do software** e (v) **Usabilidade do software**.

Dentre os tipos de objetivos, destaca-se o objetivo **Funcionalidades essenciais**. Os entrevistados priorizam a validação das regras de negócio com cenários de testes básicos e quantidade mais reduzida de cenários alternativos, como mencionado pelo Entrevistado #15: *"[...] se eu pegasse uma feature de login para testar, eu criava*

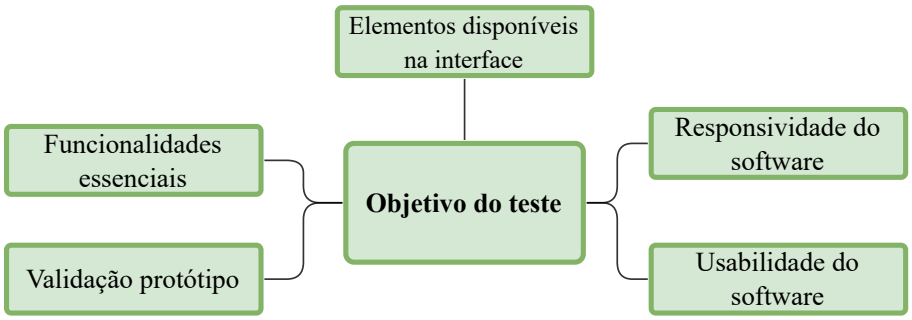

Figura 5.9: Objetivos dos testes de GUI manuais

um teste informando uma senha correta, outro para uma senha incorreta ou só um com o usuário errado".

O objetivo **Elementos disponíveis na interface** representa a avaliação da disponibilidade de uso dos campos exibidos em tela, bem como a grafia dos respectivos textos associados, como o texto de um *placeholder* ou o título de um *input*. Além disso, outras características não diretamente relacionadas ao teste de GUI são observadas pelos entrevistados. Um exemplo é a responsividade do software, por causa da diversidade de dispositivos existentes no mercado, pois é necessário verificar o software em telas com tamanhos distintos, como computadores ou *smartphones*.

Com relação ao objetivo **Usabilidade do software**, observou-se que muitas vezes o testador precisa de competências além das exigidas para a realização de testes de GUI, como ser responsável por verificar a usabilidade do software testado. O comentário do Entrevistado #4 discorre sobre isso: *"[...] aqui onde trabalho tem um UX Design, mas nas minhas experiências passadas não, então eu tinha que também pensar na usabilidade do software e até na acessibilidade durante o planejamento dos cenários [...]"*.

No objetivo **Validação de protótipo**, os entrevistados explicaram que precisam avaliar se os protótipos desenvolvidos como primeira versão do software atendem a documentação que mantém os requisitos especificados anteriormente. Sobre o objetivo **Responsividade do software**, os entrevistados avaliam no teste se o software, durante o seu uso em distintos dispositivos, respeita as dimensões de tela. A avaliação da responsividade do software ocorre por conta da diversidade de aparelhos com diferentes tamanhos que poderiam acessar a aplicação, principalmente se for um software Web.

Insumos utilizado nos testes

Os entrevistados utilizam vários tipos de documentação como material de apoio para a criação dos testes de GUI (Figura 5.10), por exemplo: (i) **História de usuário**; (ii) **Regra de negócio**; (iii) **Reporte do desenvolvedor** (iv) **Requisitos**; (v) **Critério de aceite**; e (vi) **Participação refinamento**.

Figura 5.10: Insumos utilizados para a criação dos testes manuais

Parte dos entrevistados (#7 a #20) afirmou que utilizam o insumo **Histórias de Usuários** como material de apoio para o planejamento dos testes de GUI. A maioria dos entrevistados trabalha com metodologias Ágeis, utilizando frequentemente as história de usuário para a especificação de requisitos.

Outro insumo relacionado aos métodos ágeis é a **Participação de refinamento**, nas quais os testadores obtém detalhes referentes às novas funções da *sprint* na reunião de refinamento de requisitos, junto ao dono do produto (*product owner*). O Entrevistado #3 informa que: *"[...] no refinamento técnico com os desenvolvedores, é explorado o ponto de vista técnico da história e como ela será feita, e nessa seção eu já defino quais serão os cenários de testes que queremos explorar e o como será testada a história [...]"*.

O insumo **Critérios de aceite** é complementar ao insumo **Histórias de usuários** no processo de criação dos testes de GUI (Altexsoft, 2021). Com esses critérios, pode-se ter definição correta para a tarefa, conforme informado pelo Entrevistado #3: *"[...] o refinamento de negócio é focado com o Product Owner e o Analista de Negócio para entender do ponto de vista de negócio o que a gente vai fazer na sprint; nessa parte focamos na criação de critérios de aceitação bem definidos, para conseguirmos provar posteriormente a conclusão da história de usuário [...]"*.

Na visão dos entrevistados, metodologias ágeis podem ter algumas desvantagens, por exemplo, a falta de detalhes que impactam no entendimento do software e na realização dos testes. Isso foi ressaltado pelo Entrevistado #6: *"[...] o que eu sinto falta hoje são documentos mais explicativos para apoio dos testes na metodologia Ágil, hoje não há informação sobre as features ou só o básico do básico que pode ser utilizado para a escrita dos testes, então acho que tem esse déficit [...]"*.

Insumos mais "tradicionais" também foram encontrados, como **Requisitos** e **Regras de negócio**, formatados para atender ao modelo de desenvolvimento Cascata. Além disso, o Entrevistado #18 informou que, para criar e executar os testes, ele solicita ao desenvolvedor que alterou o software o que foi feito na nova função, conforme

seu relato: *"[...] pedindo que o desenvolvedor me informasse o que ele queria que eu testasse, porque a aplicação aqui é robusta e eu preciso saber o que ele implementou para eu testar. Pelo reporte do desenvolvedor sobre a alteração dele, eu fazia os testes e entregava um relatório de execução [...]".*

Formato da documentação dos testes de GUI

Em relação ao formato em que os teste de GUI são escritos, na Figura 5.11, são apresentados os formatos mais citados pelos entrevistados para documentar testes de GUI: (i) **Gherkin**; (ii) **Mapa mental**; (iii) **Tópicos dos testes**; e (iv) **Step by Step**. O formato em que os testes de GUI são escritos é geralmente impactada pela metodologia utilizada no processo de desenvolvimento de software. Entrevistados que trabalham com alguma metodologia Ágil, costumam utilizar **Gherkin** como o formato de padronização para a escrita dos testes de GUI. **Gherkin** é uma linguagem criada para a descrição de comportamentos do software, que abstrai detalhes de codificação e implementação, focando no uso das funções do software a partir da perspectiva de um usuário (Chandorkar et al., 2022).

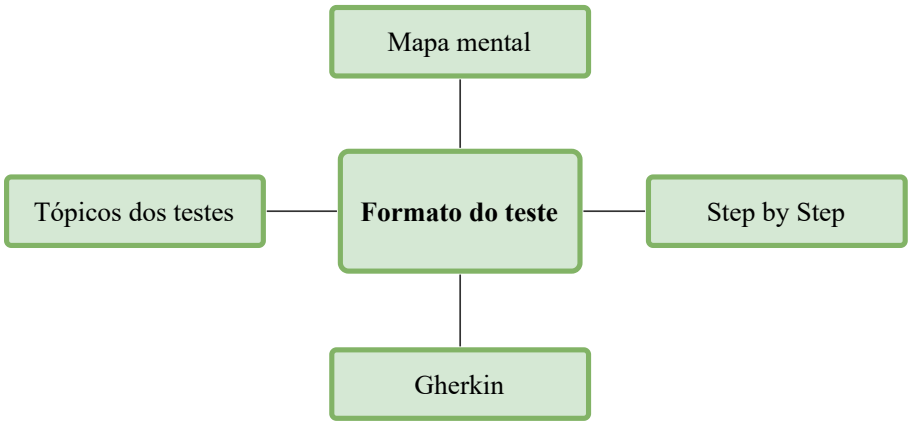

Figura 5.11: Formatos definidos para a criação de testes de GUI manuais

O formato **Mapa Mental** é o mais popular para escrever entre os entrevistados. Esse formato é utilizado para facilitar o processo de validação, conforme informado pelo Entrevistado #10 *"[...] Também já utilizei mapas mentais para me ajudar a pensar nos cenários, identificando os pontos de teste que eu precisava executar para não ficar algo tão extenso [...]"*. Buscando agilidade na escrita dos testes, o Entrevistado #9 informou que tenta reduzir o esforço nessa parte do processo: *"[...] geralmente não descrevo todos os casos de testes, porque com a metodologia Ágil você não precisa escrever todos os detalhes, faço só os tópicos para eu lembrar do que eu preciso testar [...]"*. Na **Figura C.1**, é apresentado um exemplo da utilização do formato **Mapa Mental** para a definição de cenários de testes de GUI.

Por outro lado, os entrevistados que utilizam o modelo Cascata definem os cenários de testes com todos os passos em detalhes para a execução. Por exemplo, o Entrevistado #15 informou que: *"[...] prefiro o método tradicional de escrever passos,*

até porque se o teste falhasse, o desenvolvedor saberia em que ponto o teste tinha quebrado." Além disso, para apoiar a escrita dos testes de GUI, os entrevistados utilizam técnicas de suporte para a definição dos cenários, como: (i) Regra de Pareto; (ii) Análise de valores limites; (iii) Partição de equivalência; (iv) Tabelas de decisões; e (v) Matriz de rastreabilidade.

Reuso da documentação produzida para os teste de GUI

Quanto ao reuso dos testes de GUI, na Figura 5.12, são apresentadas as situações em que os entrevistados utilizam os testes criados anteriormente: (i) **Para testes de regressão**; (ii) **Sem reuso**; (iii) **Uso esporádico**; (iv) **Controle pessoal**; (v) **Automação de teste**; e (vi) **Como documentação**.

Figura 5.12: Momentos em que os testes manuais são reaproveitados no projeto

Os testes de GUI manuais, após criados, podem ser utilizados novamente no projeto de desenvolvimento de software. Assim, os testadores podem utilizar mais de uma vez os cenários de testes documentados. Dos 20 entrevistados, 8 entrevistados (40%) reutilizam os testes para avaliação de regressão, 4 entrevistados (20%) não reutilizam os testes criados, 2 entrevistados (10%) reutilizam parcialmente os teste existentes, 2 entrevistados (10%) consultam os testes em um controle pessoal de forma esporádica, 2 entrevistados (10%) reaproveitam os testes no processo de automação e 2 entrevistados (10%) utilizam os testes como uma documentação do software em avaliação, disponibilizando os testes como um manual para os usuários.

5.6.4 QP6 - Quais são as principais limitações no processo de testes de GUI manuais?

Considerando que a realização de testes de GUI manuais é feita em duas etapas, sendo a primeira o levantamento dos cenários de testes e, posteriormente, a de execução, as limitações identificadas para os testes de GUI manuais foram categorizadas de forma a contemplar as duas etapas informadas.

Dificuldades e soluções na criação de teste manual de GUI

As maiores dificuldades que os entrevistados encontram em seus projetos na escrita dos cenários testes e as possíveis soluções para esses problemas foram organizadas em duas categorias (Figura 5.13): (i) **Limitações de Processo**; e (ii) **Limitações Profissionais**.

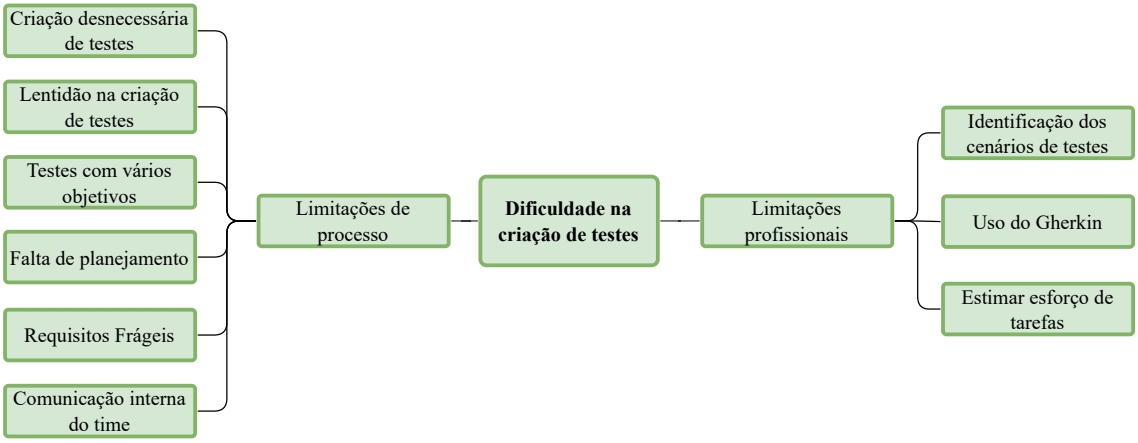

Figura 5.13: Problemas relacionados a criação de testes de GUI

1. **Limitações Profissionais**
   A categoria **Limitações profissionais** referencia a falta de habilidade dos entrevistados para a criação dos testes de GUI. Três tipos de limitações profissionais foram encontrados: **Identificação dos cenários**, **Uso do *Gherkin*** e **Estimar esforço de tarefas**. São elas:

   - `Identificação dos cenários`: Encontrar os cenários de software que devem ser testados é um desafio, principalmente pela complexidade dos software sob testes que geram longa curva de aprendizado e pela falta de documentação nos projetos. A consequência é testes com ações pouco detalhadas e com assertividades mal definidas. Para evitar os problemas mencionados, uma maneira comumente utilizada para identificar os requisitos é por meio do uso prático do software para validar qual é a saída retornada pela interação do testador, pois é comum ter dificuldade para escrever cenários de testes até ter bom entendimento do produto sendo utilizado. O envolvimento do cliente/usuário no momento da criação dos requisitos também foi apontado como possível solução para facilitar o processo de criação de testes. O cliente próximo ao projeto pode, por exemplo, revisar as informações e verificar se há algo incorreto. Além disso, os testadores podem ser incluídos nas reuniões de refinamento e ajudar a sanar quaisquer dúvidas sobre a nova função a ser desenvolvida no software;

- *Uso do Gherkin*[6]: Criar um bloco de teste no formato Gherkin que não seja extremamente específico pode ser desafiador para os testadores de software. O cenário de teste precisa ser generalista o suficiente para ser reaproveitamento em outras avaliações de testes; ainda assim, o teste precisa conter detalhes razoáveis para o testador lê-lo e executar as ações esperadas para o uso do software;

- `Estimar esforço de tarefas:` Identificar o esforço a ser gasto para a concepção e o planejamento dos testes é outra dificuldade relacionada ao processo de teste, principalmente quando não se tem ideia das funções do software a ser avaliadas.

2. **Limitações de Processo**
   A categoria **Limitações de processo** é referente aos problemas que ocorrem no projeto em que os entrevistados estão alocados como: **Criação desnecessária de testes**, **Lentidão na criação de teste**, **Testes com vários objetivos**, **Falta de planejamento**, **Requisitos frágeis** e **Comunicação interna do time**. São elas:

   - `Criação desnecessária de testes:` por vezes, os testadores criam testes no formato *Gherkin*, mas as pessoas do projeto não consultam os insumos produzidos. Para os entrevistados, isso é considerado desperdício de esforço e consome o tempo que poderia ser utilizado em outras atividades da qualidade de software. A utilização de testes automatizados de GUI como documentação viva do projeto é uma das recomendações dos entrevistados para resolver o problema em questão. Ao invés de ter um requisito, um cenário de teste manual e o mesmo teste codificado para a automação, é possível minimizar o esforço do time investido na criação e na manutenção de vários artefatos com o uso centralizado de *scripts* automatizados para o entendimento do software;

   - `Testes com vários objetivos:` testar distintos aspectos de um software em uma única avaliação pode impactar na qualidade do teste. Planejar em um mesmo teste a avaliação de regras de negócios e várias interfaces pode impactar no foco da atividade e consequentemente na qualidade do resultado final do teste;

   - `Lentidão na criação de testes:` documentar diversos cenários de testes é um desafio em projetos com muitas funções para serem testadas. Por isso, alguns entrevistados sugerem o uso de soluções baseadas em inteligência artificial para a identificação e escrita de cenários de testes, visando diminuir o esforço dessas atividades. Além disso, outra sugestão é alterar o formato da escrita do teste, com a substituição dos cenários de testes detalhados por mapas mentais. Geralmente, eles são mais fácil de entender, por serem mais visuais, além do tempo de produção deste material ser menor em comparação aos testes escritos de forma

[6] `https://cucumber.io/docs/gherkin/reference/`

mais tradicional. Armazenar cenários de testes para serem reutilizados posteriormente também pode minimizar o trabalho da escrita dos testes, principalmente em projetos que exigem grande volume de criações e execuções de testes, como em testes de regressão;

- `Falta de planejamento:` as atividades da equipe de qualidade de software podem não ser consideradas nos prazos finais para a entrega do produto. Às vezes, pode ser requisitado que a concepção, a escrita e a execução dos cenários dos testes devam ocorrer em um curto intervalo de tempo (e.g., um único dia de trabalho), gerando esforço por parte dos testadores e impactando na qualidade final do produto;
- `Requisitos frágeis:` um dos grandes problemas relacionados ao processo é os requisitos frágeis do projeto. A documentação desses requisitos geralmente não possui detalhes suficientes para realizar cobertura de teste satisfatória. Por exemplo, o Entrevistado #2 informou que há falta de conhecimento até mesmo pelas profissionais que deveriam especificar os requisitos *"[...] geralmente a equipe de requisitos não conhece as regras de negócios e mesmo informando quais necessidades que precisam ser sanadas com a funcionalidade, ainda deixam lacunas na documentação que dificultam o planejamento dos testes e até o desenvolvimento da feature. Além disso, os clientes também não sabem informar o que esperam do software."* Documentações incompletas ou mal escritas podem impactar de diversas formas no processo, por exemplo retrabalho na escrita/execução dos testes, pois o software não atende ao desejo do cliente, e *bugs* podem passar despercebido pelo testador, pois ele não possui informação suficiente para realizar os testes. Uma solução apontada para resolver esse problema é utilizar *mockups*. Isso pode facilitar o entendimento de como o software final minimamente deveria ser ou quais funções deveriam estar disponíveis na interface para o usuário, principalmente se o software ainda estiver em fase embrionária. Também, há a possibilidade de identificar o máximo de informações no momento da especificação dos requisitos, como o que precisa ser feito e as necessidades do usuário, para que os testes ocorram em contexto mais consolidado, deixando menor margem de erro para desenvolvedores e testadores. As histórias de usuário poderiam ser distribuídas para a equipe com a sequência de passos suficiente para entender os fluxos do software. Adicionalmente, há o agravante de que, em algumas equipes, não há um profissional especializado em levantar e especificar requisitos, gerando documentações rasas nos projetos, segundo os entrevistados. Posições como o *Scrum Master* lidam com os requisitos, ao invés de ter posição focada na escrita de histórias de usuário;
- `Comunicação ineficiente do time:` um problema informado pelos entrevistados que impacta os requisitos indiretamente é a falta de comunicação entre os membros do time. São realizadas alterações nas especificações

não compartilhadas. As documentações estão geralmente rasas em detalhes e ainda passam por alterações constantes não compartilhadas com o time de qualidade ou até mesmo com o desenvolvedor alocado para a tarefa de codificação. O Entrevistado #7 informa a dificuldade em seu contexto: *"[...] o que ocorre bastante é que a feature já está sendo desenvolvida, o seu protótipo está pronto no Figma e a história de usuário já está pronta, mas após uma ou duas semanas de desenvolvimento ocorrem alterações na interface do software, com isso é necessário mudar os cenários de testes que já estão prontos, a documentação e tudo mais [...]"*. Os requisitos do software sendo desenvolvido e, posteriormente, utilizados no processo de planejamento dos testes devem ser comunicados pelos interessados da equipe para ter alinhamento de expectativas de como o software deverá funcionar na sua versão final. Assim, caso haja alterações nas especificações, a equipe deve ser notificada para evitar problemas relacionados ao versionamento incorreto de informações.

Dificuldades e soluções na execução de teste manual de GUI

Em relação à execução do teste manual de GUI, foram encontrados três tipos de limitações (A Figura 5.14): (i) **Limitações de Processo**; (ii) **Limitações Técnicas** e (iii) **Limitações Organizacionais**.

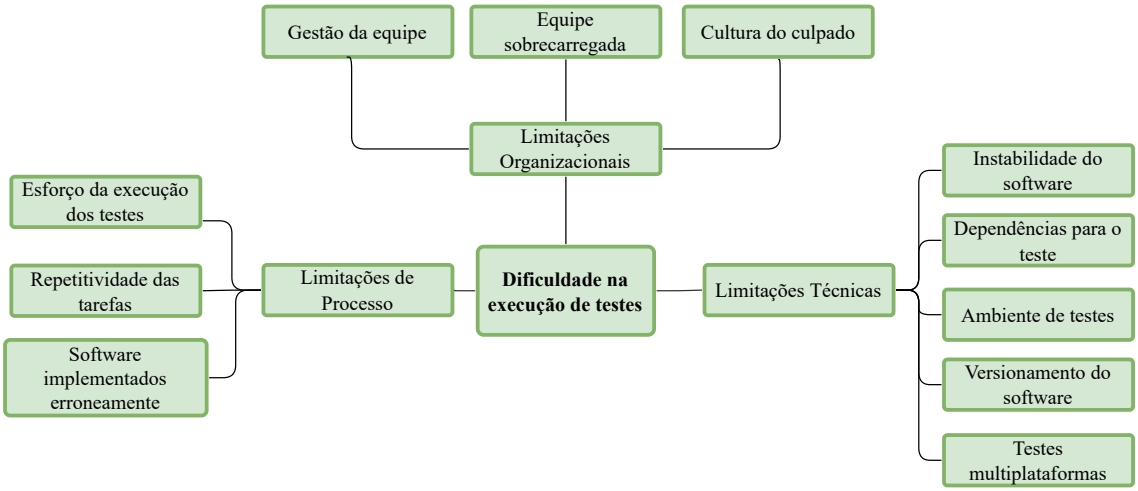

Figura 5.14: Problemas relacionados a execução de testes de GUI

1. **Limitações de Processo**
   Assim como nas limitações para as criações de testes, a execução de testes de GUI também possui dificuldades geradas pela organização do processo utilizado no projeto. A seguir, são apresentadas as limitações de processo: **Software implementados erroneamente**, **Esforço da execução dos testes** e **Repetitividade das tarefas**.

- `Software implementado erroneamente:` com base na própria intuição ou "achismo", o desenvolvedor codifica o software desrespeitando os requisitos especificados, os quais estão de acordo com as expectativas do cliente. Como o software foi produzido de forma errônea, perde-se tempo de projeto, pois o testador pode "desperdiçar" uma execução de teste em um software que não está na sua versão final, devolvendo a tarefa para o desenvolvedor para refazer todo o trabalho codificado;
- `Esforço da execução dos testes:` a necessidade de avaliar de forma manual vários aspectos durante a execução do teste, como regras de negócio e elementos visuais, representa um problema para os projetos de desenvolvimento de software. Segundo os entrevistados, pela falta de objetividade no teste, perde-se tempo analisando detalhes que, na opinião deles, não são relevantes para o software. Além disso, pontos mais críticos são deixados de lado por conta desse excesso de responsabilidade do teste. Pelo fato de muitos desenvolvedores não respeitarem os requisitos no momento da codificação, o testador pode não confiar no trabalho do seu colega de equipe. Assim, para evitar que alguma falha passasse para a versão final do software, o testador pode achar necessário testar todo o software de novo, mesmo as funções não diretamente relacionadas àquela que o desenvolvedor havia alterado;
- `Repetitividade das tarefas:` testar a mesma função mais de uma vez, seja ou não em um teste de regressão, para tentar identificar novos erros, pode impactar na qualidade da execução dos testes. Executar os mesmos testes várias vezes não garante que novas falhas sejam identificadas. Além disso, repetir as mesmas atividades pode deixar o testador com o "olhar viciado" e perder o senso crítico sobre o que está testando.

2. **Limitações Organizacionais**
   A categoria **Limitações organizacionais** corresponde às dificuldades geradas pela cultura da organização em que os entrevistados trabalham, junto com as escolhas ou a falta de ação da gerência dos projetos. As limitações identificadas foram: **Gestão de equipe**, **Equipe sobrecarregada** e **Cultura do culpado**.

   - `Gestão de equipe:` segundo os entrevistados, equipes formadas por profissionais sem experiência podem impactar na qualidade da execução dos testes. Posições importantes, como gerentes de projetos e *scrum master*, quando preenchidas por pessoas sem experiência, podem levar a atrasos, mudanças de metodologia de desenvolvimento durante a execução do projeto e, consequentemente, a necessidade da realização de horas extras;
   - `Equipe sobrecarregada:` por conta de deficiências no planejamento das atividades e consequentes pressões sobre os profissionais, muitos afirmaram estar desmotivados e com baixa produtividade. Nesse contexto, eles costumam receber o software com bastante atraso e com pouco tempo para a execução dos testes. Alguns afirmaram que precisam trabalhar de

madrugada para conseguir avaliar os cenários de testes antes do prazo final;

- `Cultura do culpado:` Quando um problema ocorre no projeto, alguns entrevistados afirmaram que, ao invés de encontrar uma solução adequada de forma colaborativa entre os interessados, ocorre uma tentativa de culpabilizar apenas uma pessoa pela crise existente. Para os entrevistados, isso acaba deixando o ambiente de trabalho tóxico e desmotivador para todos.

3. **Limitações Técnicas**
   A categoria **Limitações técnicas** corresponde aos problemas gerados a partir da estrutura em que os testes de GUI são executados. As seguintes limitações foram encontradas: **Instabilidade do software**, **Dependência para o teste**, **Ambiente de testes**, **Versionamento do software** e **Testes multiplataforma**.

   - `Instabilidade do software:` a performance da versão do software a ser avaliada pode gerar lentidão na execução dos testes de GUI. Esse problema pode confundir o testador se a baixa resposta do software é um *bug*, que deve ser registrado para os desenvolvedores avaliarem ou um comportamento mapeado pelo time, atrapalhando a finalização dos testes. Para os entrevistados, o desenvolvedor deve garantir que o ambiente de teste e o software estejam funcionando como esperado antes de solicitar ao time de qualidade a avaliação da sua tarefa;

   - `Dependências para o teste:` a falta de massa de dados e as integrações externas realizadas com software de terceiros para os testes é a maior queixa dos entrevistados sobre a execução manual de GUI. Basicamente, as pré-condições necessárias para a execução dos testes são desconsideradas no momento do planejamento do projeto; e o impacto vai desde a falta de usuários credenciados para o *login* no software, o uso imparcial de funções pela falta de permissões ao tempo excessivo para configurar o software, que muita das vezes é maior do que o próprio tempo do teste. Uma forma de diminuir esse problema é dar acesso ao testador para criar as dependências necessárias para o teste. Caso ele não tenha competência técnica, devem ser repassadas as devidas capacitações para que ele consiga produzir os dados para o testes;

   - `Ambiente de testes:` em muitos projetos, não há ambientes de testes para que os testadores possam executar as suas avaliações. Além disso, quando o ambiente de teste é criado, a estrutura de hardware pode responder de forma determinística, com quedas constantes e lentidão para acesso. Como observado em algumas respostas, o testador não possui barreiras para o entendimento do software ou da tarefa de testes, mas quanto ao ambiente de homologação que fica fora do ar. É esperado pelos entrevistados que o ambiente de homologação tenha estrutura mais próxima possível do ambiente de produção, com estabilidade suficiente

para os testes sejam executados. Outra solução é o uso de *Docker*[7], para que a manutenção dos ambientes seja feita de maneira mais dinâmica e com mais controle. Além disso, foi pontuado que esses problemas poderiam ser resolvidos com a participação de um profissional *DevOps* no time, o qual ficaria responsável pela infraestrutura e acessos do projeto;

- `Versionamento do software:` o testador recebe o software para avaliar sem as últimas alterações presentes no ambiente de produção. Assim, o profissional testa a aplicação em uma versão antiga e defasada em relação ao que os usuários finais utilizam. É de responsabilidade do desenvolvedor saber utilizar o software de controle de versão de código, como também gerenciar a versão da *branch* que ele está desenvolvendo;
- `Testes multiplataforma:` a necessidade de testar um software em diversas telas existe porque há imensa quantidade de dispositivos no mercado que possuem tamanhos de telas diferentes. Na maioria das vezes, há duas soluções para essa situação (i) comprar diferentes dispositivos, como *smartphones* de diferentes marcas, o que envolve grande investimento ou (ii) utilizar uma ferramenta para teste *cross browser* (utilizar uma plataforma que simula diferentes dispositivos). No entanto, essa simulação possui algumas ressalvas, por exemplo problemas de conexão e latência, pois, geralmente, esses serviços são hospedados em outros países.

5.6.5 QP7 - Como é o processo automatizado de testes de GUI?

Para ter entendimento sobre o planejamento dos testes de GUI automatizados, foram avaliados quatro categorias referente ao processo de automação (Figura 5.15): (i) **Ferramentas para automação de testes**; (ii) **Insumos para os testes automatizados**; (iii) **Manutenção dos testes automatizados**; e (iv) **Boas práticas para automação de testes**. Vale ressaltar que, na fase das entrevistas, 70% dos entrevistados relataram realizar testes de GUI automatizados.

Figura 5.15: Categorias de codificação identificadas para o planejamento dos testes de GUI automatizados

[7] `https://www.docker.com/`

Ferramentas para automação de GUI

Em complemento ao questionário *online*, no processo de entrevistas, foi investigado sobre as ferramentas utilizadas no processo de automação de testes. Foi identificado que os entrevistados utilizam soluções principalmente da segunda geração de ferramentas de testes automatizados, com exceção da ferramenta *Eye Automate* que faz parte da terceira geração (Tabela 5.3). *Frameworks* de suporte como o *Cucumber* e *Behave* também foram informados pelos participantes; no entanto, elas auxiliam no uso de *BDD* e histórias de usuários e não são ferramentas automatizadas propriamente ditas. Além disso, os entrevistados informaram que utilizam as ferramentas mencionadas com as seguintes linguagens de programação: 33% (7 entrevistados) Java, 19% (4 entrevistados) Python, 19% (4 entrevistados) Javascript, 19% (4 entrevistados) C#, 5% (1 entrevistado) Cobol e 5% (1 entrevistado) Ruby.

Tabela 5.3: Ferramentas para automação citadas pelos entrevistados

Ferramentas	**Tipo de Automação**	**#Entrevistados**
Selenium	Web	11
Robot Framework	Web	10
Cypress	Web	9
Selenium IDE	Web	3
Capybara	Web	2
Protactor	Web	2
Appium	Mobile	2
Playwright	Web	1
Katalon	Web/Mobile	1
UFT	Web	1
Eye Automate	Web	1
Espresso Café	Mobile	1
Test Cafe	Web	1
Test Complete	Web	1

Insumos para testes automatizados de GUI

Na Figura 5.16, é apresentado o conjunto de insumos de projeto utilizados para suportar o processo de automação de teste de GUI. Alguns entrevistados informaram que realizam a automação sem o uso de qualquer tipo de documentação formal criada previamente. Dentre os que utilizam, o uso de cenários manuais de teste de GUI foi o mais citado por eles. Além dos cenários, histórias de usuário e outras regras de negócio foram informadas como documentos de apoio para a automação de GUI. Também, foi citado o emprego do planejamento do *Product Owner*.

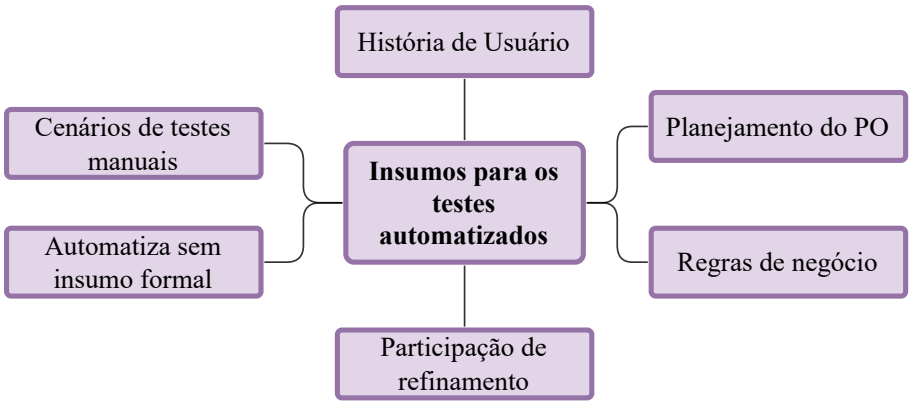

Figura 5.16: Insumos utilizados no processo de automação de testes de GUI

Manutenção dos testes automatizados de GUI

Considerando que um dos maiores esforços após a criação dos testes automatizados é a manutenção do código fonte criado, na Figura 5.17 é apresentado em que momento ocorre o planejamento para a revisão dos testes automatizados de GUI nos projetos de desenvolvimento de software em que os testadores estão alocados. O maior motivador para a manutenção do código de teste é os falsos negativos retornados após a execução do teste, mencionado por oito dos quatorze entrevistados. Assim, após a análise dos resultados da execução, é identificada uma falha no código de teste e não no software sendo avaliado, mostrando um caráter mais corretivo do que reativo para a manutenção.

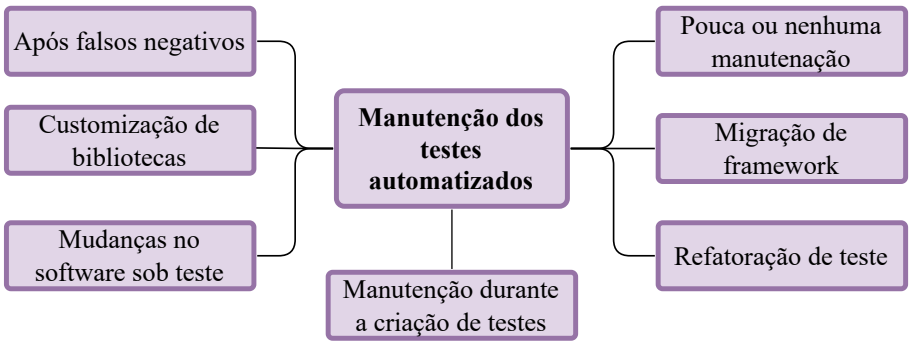

Figura 5.17: Motivos para a realização de manutenção nos testes automatizados

Também, foram identificadas alterações relacionadas aos *frameworks* de testes: três dos entrevistados informaram realizar customizações nos *frameworks* e bibliotecas de testes, o que gera alterações no código de testes. Um desses entrevistados relatou que, ao migrar de *framework* de teste, por exemplo de *Selenium* para *Playwright*, sua equipe também teve que mudar os testes automatizados.

Mudanças no software por parte do desenvolvedor também geram manutenções, como informado por três entrevistados que realizam correções no código de teste

automatizado. Outros três entrevistados mencionaram manutenções mais reativas, quando, ao criar um novo teste automatizado, também realiza a revisão dos códigos existentes. Um entrevistado informou que as manutenções ocorrem com viés de refatoração, ou seja, recriando todos os cenários de testes automatizados seguindo boas práticas. Por fim, três entrevistados afirmaram que, em seus projetos, há pouca ou nenhuma manutenção de código, o foco da equipe é na criação de novos testes.

Boas práticas para automação de teste de GUI

Foi investigado se os entrevistados possuem em seus projetos algum documento de boas práticas, manuais ou qualquer tipo de material para prover suporte ao testador na tarefa de automatização de testes. As categorias de documentação identificados para esse tópico foram (Figura 5.18): (i) ***Wiki* sobre automação**; (ii) **Regra de escrita de *Gherkin***; (iii) **Documentação escrita após resolução de problema** e (iv) **Sem orientação para automação**. São elas:

- `Wiki sobre automação:` cinco entrevistados informaram que utilizam documentações produzidas internamente orientando como utilizar os *frameworks* de testes, regras para criação de variáveis para os testes automatizados e como gerenciar o repositório que comporta os códigos de testes automatizados. Segundo eles, o objetivo dessa documentação é nortear, principalmente, os testadores recém admitidos nas equipes. Além disso, um entrevistado informou que há semanalmente apresentações sobre automação de testes ministradas por profissionais mais experientes a fim de compartilhar o conhecimento com os demais colegas de trabalho;
- `Regra de escrita de Gherkin:` segundo um entrevistado, a única boa prática repassada para a equipe de teste era em relação a quantidade de *links* de um cenário de teste em *Gherkin*, o qual não deveria ultrapassar mais de cinco comandos por testes. Contudo, não havia outros materiais ou regras de suporte para o testador;

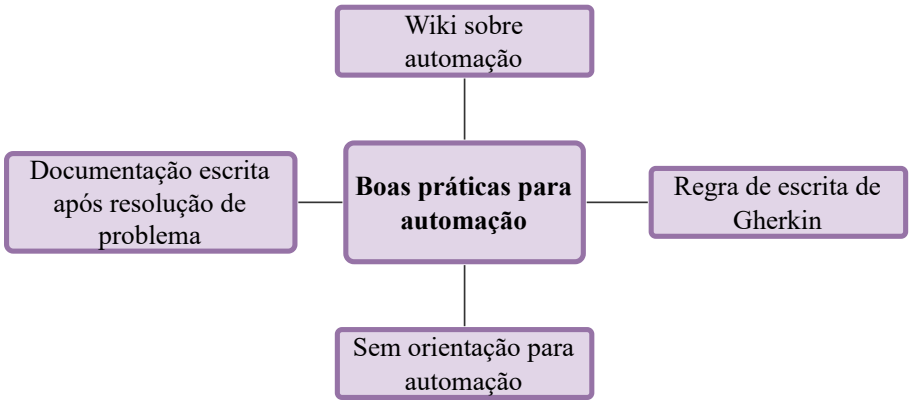

Figura 5.18: Boas práticas para automação de testes

- `Documentação escrita após resolução de problema:` como na equipe não havia uma documentação formal para uso dos testadores, os integrantes da área de teste decidiram que, após a resolução de um problema relacionado a automação, deveria ser registrada a respectiva resolução em uma plataforma de acesso irrestrito a todos. Dessa forma, aos poucos seria criada uma base de conhecimento na empresa;
- `Sem orientação para automação:` seis entrevistados informaram que não há boas práticas documentadas em seus projetos. Caso fosse necessário alguma orientação adicional, o testador deveria entrar em contato com outro colega de equipe de forma verbal ou esperar que no processo de *code review* do código de teste, o revisor indicasse o que deveria ser alterado na automação criada.

5.6.6 QP8 - Quais são as principais limitações no processo de testes de GUI automatizados?

Assim como abordado para os testes de GUI manuais, nesta seção, são apresentadas as limitações que dificultam os entrevistados a criarem testes de GUI automatizados. As três categorias de limitações encontradas foram (Figura 5.19): (i) **Limitações do software sob teste**; (ii) **Limitações educacionais** e (iii) **Limitações organizacionais**. São elas:

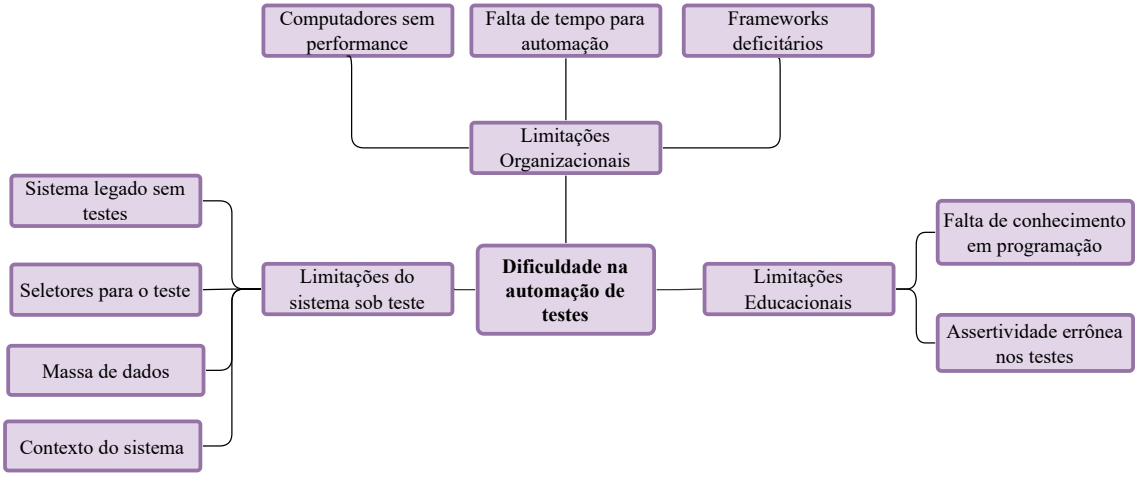

Figura 5.19: Dificuldades que impactam o processo de automação de testes de GUI

1. **Limitações Organizacionais**

   A categoria **Limitações organizacionais**, assim como para os testes de GUI manuais, é referente aos problemas gerados a partir de ações tomadas ou a falta de decisão das empresas em que os testadores atuam.

   - `Computadores sem performance:` a produtividade da atividade de automatizar testes de GUI pode ser impactada negativamente pelos computadores disponibilizados aos funcionários. Segundo os entrevistados,

máquinas com pouco recurso computacional, como memória e processamento, não suportam a utilização dos *frameworks* de testes automatizados, fazendo com que a automação leve mais tempo para ser concluída. Para os entrevistados, seria interessante a disponibilização de computadores mais novos que atendam as necessidades profissionais dos testadores;

- `Falta de tempo para automação:` para os entrevistados automatizando testes de GUI nas empresas, o tempo disponibilizado para a criação de *scripts* automatizados é insuficiente. Isso dificulta a evolução constante de validações automatizadas no projeto. Esse problema ocorre pela interrupção das atividades de automação para a alocação do profissional em tarefas referentes aos testes manuais. Como solução, eles indicam que deve ter melhor planejamento da empresa para ter tempo suficiente para automação de testes de GUI;
- `Frameworks deficitários:` algumas empresas com um setor de testes mais conservador, utilizam ferramentas antigas que não possuem as funções necessárias para um testador automatizar testes de GUI. Assim, o testador pode levar tempo extra com customizações de bibliotecas ou uso de outras soluções terceiras. Uma solução é o uso de ferramentas mais modernas que facilite a automação de testes, em detrimento aos *frameworks* legados que não atendem as demandas da área.

2. **Limitações Educacionais**

   A categoria **Limitações educacionais** faz menção a falta de conhecimento técnico dos entrevistados para realizar as atividades de testes de GUI automatizados.

   - `Falta de conhecimento com programação:` entrevistados informaram que, apesar de seus companheiros de equipe conhecerem as ferramentas de automação de testes, alguns não conhecem o suficiente as linguagens ou lógica de programação necessárias para desempenhar as tarefas referentes à automação. Isso foi relatado principalmente no que diz respeito a customizar comandos ou configurar integrações contínuas. O profissional de teste automatizado precisa de pelo menos um conhecimento básico em programação;
   - `Assertividade errônea para o teste:` pela falta de experiência de alguns testadores, a automação de testes produzida por eles podem conter problemas relacionados ao objetivo do teste e aos critérios de assertividade escolhidos. Esses critérios são utilizados para saber se uma avaliação do software foi aprovada ou falhou. Uma forma de minimizar esse problema, segundo os entrevistados, é instruir os profissionais menos experientes pode meio de transmissão de conhecimento na equipe.

3. **Limitações do software sob teste**

   Dificuldades envolvendo o software testado e o ambiente em que ele será executado compõem as categorias de limitações do software sob teste.

   - `Software legado sem testes:` iniciar a automação em um software que não há testes é uma atividade desafiadora para os entrevistados. Nesse contexto, é necessário dispor de tempo para preparar o ambiente de teste e as suas respectivas dependências, a fim de executar os *scripts* de testes automatizados.

   - `Seletores para o teste:` Em muitos software, não há seletores para que os *frameworks* de testes possam interagir com os elementos exibidos na interface. Isso pode impedir a execução de testes automatizados no projeto. Esse problema ocorre quando o desenvolvedor não adiciona seletores relevantes nos elementos da tela, como pela biblioteca de desenvolvimento *Front-End React* [8], que exporta páginas HTML sem estruturação propícia para os testes automatizados. Uma solução é o desenvolvedor adicionar informações no HTML do software, como *testing-id*, para que o testador possa automatizar com mais facilidade as avaliações do software. Além disso, o testador deve conseguir criar *xpaths* consistentes, caso não haja um *testing-id* disponível;

   - `Massa de dados:` assim como informado pelos entrevistados que executam testes manuais, a falta de massa de dados impacta no processo de automação de testes. Segundo eles, simular dados de terceiros na criação de testes pode ser desafiador;

   - `Contexto do software:` entender o domínio de um software é algo que pode levar tempo para um testador novo; principalmente, se o projeto não possui requisitos e regras de negócios documentados. Não é uma tarefa fácil automatizar fluxos de um software que ninguém sabe como ele deve funcionar. Uma recomendação dada pelos entrevistados é a empresa ter documentação do software disponível para os testadores. Além disso, se possível, o testador deve realizar verificação manual do software, para ele ter em mente o que deve ser automatizado posteriormente.

5.7 Ameaças a Validade

Assim como na etapa do questionário *online*, para as entrevistas há ameças a validade, conforme informado a seguir.

- **Validade de Construção**. As perguntas levantadas para as entrevistas foram escolhidas a fim de possibilitar a coleta de dados de distintos aspectos dos testes de GUI. No entanto, foi notado que algumas respostas dos entrevistados

[8] `https://pt-br.reactjs.org/`

fugiram do tema questionado; possivelmente, isso ocorreu por falta de compreensão da pergunta pelo entrevistado e, por esse motivo, as respostas sem conexão com a pergunta foram desconsideradas;

- **Validade Interna**. Pela dificuldade em encontrar testadores dispostos a participar das entrevistas, não houve divisão igualitária dos perfis dos entrevistados, ou seja, não há a mesma quantidade de participantes que executam somente testes de GUI manuais, somente testes de GUI automatizados e que realizam ambos testes. Para minimizar essa dificuldade, foi utilizado o critério de saturação teórica para concluir a coleta de dados;
- **Validade Externa**. Assim como na etapa do questionário *online*, a quantidade de participações não representam toda a população de testadores de software, mas é pequeno recorte de profissionais da industria;
- **Confiabilidade**. Para garantir a qualidade dos resultados, dois pesquisadores realizaram a codificação das transcrições obtidas a partir das entrevistas e, em seguida, os códigos extraídos foram comparados a fim de corrigir divergências dos resultados preliminares. Posteriormente, outros dois pesquisadores realizaram a revisão da categorias e as suas respectivas codificações para minimizar vieses ou interpretações errôneas dos dados provenientes das conversas com os testadores.

5.8 Síntese de capítulo

Neste capítulo, foram descritas as atividades relacionadas às entrevistas, como a estratégia para convidar participantes e a realização dos testes piloto a coletados dados. Também, foi abordada a análise dos dados e os resultados obtidos. No próximo capítulo, são discutidos os resultados obtidos na etapa do questionário *online* e nas entrevistas.

Capítulo 6

Discussão dos Resultados

6.1 Perfil dos testadores

Nesta seção, são discutidos os resultados referentes aos pontos que tangem o perfil dos participantes da pesquisa, tanto na etapa do questionário *online* quanto nas entrevistas, como tempo de experiência, a formação acadêmica e outros testes realizados nos projetos além dos testes de GUI.

Tempo de experiência

Para os respondentes do questionário *online*, foi identificada quantidade significativa de testadores que possuem entre 1 a 5 anos de experiência com testes de GUI, mostrando que, provavelmente, esses profissionais procuraram a área de teste em um contexto de recém-formados na graduação ou em transição de carreira. Analisando os resultados das entrevistas, foi identificado que a maioria dos profissionais que executam exclusivamente testes manuais possuem entre 2 a 4 anos de experiência, enquanto que os testadores que executam testes de GUI manuais e automatizados, possuem entre 5 e 10 anos de experiência. Assim, é possível que o tempo de experiência profissional dos testadores impactem na forma em que eles executam os testes de GUI, sendo que os testes de GUI automatizados são desenvolvidos por profissionais que atuam a mais tempo na área de teste e que possuem maior senioridade, pois, provavelmente obtiveram conhecimentos durante a jornada profissional suficientes para testar o software além da abordagem manual.

Formação acadêmica

A maioria dos testadores que trabalham com testes de GUI possui ensino superior. Observando os resultados do questionário *online*, há poucos casos em que os respondentes concluíram apenas o ensino médio ou cursos técnicos (6%). De forma similar, foi observado nas entrevistas que apenas um dentre vinte entrevistados não possuía ensino superior. Em relação às graduações mais cursadas pelos participantes, para

o questionário *online*, Análise e Desenvolvimento de Sistemas foi o curso mais citado; enquanto, para os entrevistados, Sistemas de Informação foi a graduação mais informada.

Foi investigado se a formação acadêmica dos respondentes do questionário *online* teria impacto na forma em como os testes de GUI são executados. O curso superior de Análise e Desenvolvimento de Sistemas foi o que mais graduou testadores de diferentes perfis, tanto os que realizam exclusivamente testes manuais, tanto exclusivamente testes automatizados ou os que usam ambas abordagens. Porém, não foi possível aferir se a formação do profissional interfere na forma em que o testador executa testes de GUI, ou se testadores formados em cursos tecnólogos ou não tendem a testar GUI com uma abordagem específica. Esse ponto pode ser melhor investigado em trabalhos futuros. Pode-se observar que os testadores que automatizam testes estão no nível pleno e sênior (mais de três e cinco anos de experiência profissional respectivamente) e formaram-se no curso de Sistemas da Informação (36%), seguido de Análise de Sistema (32%), Ciência da Computação (19%), Engenharia da Computação (3%) e Outros cursos (10%), como Banco de dados e Redes de Computadores.

Além disso, dentre os profissionais que participaram da pesquisa, tanto no fase de questionário *on-line* quanto nas entrevistas, poucos respondentes têm certificações em teste, embora exista um conselho ISTQB local no país (BSTQB - *Brazilian Software Testing Qualifications Board*). Esse conselho traduziu para português brasileiro os materiais de estudo e as provas, além de permitir o pagamento da taxa de inscrição em real.

Fonte de conhecimento

A partir dos relatos dos profissionais entrevistados, os testadores geralmente desenvolvem as suas competências para atuarem na área de teste utilizando conteúdos da literatura cinza, como *blogs* e cursos criados por outros profissionais de testes. Além disso, a orientação de outros testadores no ambiente de trabalho e a execução de tarefas de testes nos projetos de software são as principais formas de aprendizado sobre a área. Os resultados mencionados podem ser justificados principalmente pela maneira em que os cursos superiores de tecnologia abordam o tema de testes. Como pontuado pelos entrevistados, em um curso de Análise e Desenvolvimento de Sistema, o discente precisa participar obrigatoriamente de diversas disciplinas sobre programação; no entanto, não há disciplinas exclusivas para tratar o tema teste de software, obrigando os interessados a procurar material complementar sobre teste fora da instituição formal de ensino.

Outros testes realizados

Além dos testes de GUI, os testadores executam outros tipos de testes. Nesse estudo, as avaliações em API foram mencionadas pelos testadores, a partir de ferramentas

como *Postman*[1] e *Insomnia*[2]. Essas ferramentas não exigem que o usuário tenha *expertise* em programação, tanto que 50% dos entrevistados que realizam somente testes manuais afirmaram testar API's, mesmo eles não tendo experiência com automação de testes ou linguagens de programação.

Também, foi identificado que, por meio da realização de requisições diretamente nas API's, alguns testadores criam massas de dados para serem usadas posteriormente na execução de teste de GUI manual. As demais avaliações, como teste de carga, performance e unidade, foram menos citadas, possivelmente pela necessidade de criar *scripts* com uma linguagem de programação para testar determinado aspecto do software.

6.2 Modelos de processos nas companhias

O modelo de processo Àgil foi o mais utilizado nos projetos de desenvolvimento de software dos respondentes do questionário *online* e dos entrevistados. O modelo Ágil foi criado em 2001 visando responder de forma mais eficiente as demandas de criação de software e, por conta disso, várias organizações começaram implementar métodos ágeis em seus projetos.

Como foi apresentado no capítulo de resultados, muitos testadores que realizam testes de GUI estão alocados em projetos em que possuem falhas organizacionais e de processo, falhas que geram a seguinte questão: as empresas em que os profissionais trabalham realmente são ágeis ou estão em um contexto de "falso ágil"? O "falso ágil"ocorre quando há a implementação de métodos "agilistas", mas sem mudança cultural da empresa para que ocorra a geração de resultados pelo uso do Ágil.

O primeiro princípio do Manifesto Ágil, que vai de encontro com as limitações informadas pelos testadores, é ***Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto***. Considerando que ocorrem mudanças nos requisitos do software e elas não são compartilhadas com os integrantes do time, é possível aferir que a equipe não se comunica de forma eficaz e que os profissionais de diferentes setores não possuem sinergia na realização das suas tarefas, gerando falta de troca de informações pertinentes ao projeto para todos os envolvidos.

Outro princípio não respeitado é ***Construir projetos em torno de indivíduos motivados, dando a eles o ambiente e o suporte necessário e confiando neles para fazer o trabalho***, primeiramente, pela "cultura do culpado"informado pelos entrevistados, que se sentem atacados pela empresa ao serem culpabilizados de forma agressiva pelo projeto e, consequentemente, desmotivados com ambiente de trabalho tóxico. Além disso, há falta de suporte por parte da empresa quando não são disponibilizados computadores que não provêm recursos necessários para

[1] `https://www.postman.com/`
[2] `https://insomnia.rest/`

testadores desempenhem suas atividades, como a criação de testes de GUI automatizados.

O Manifesto Ágil também informa que ***As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis***; contudo, essa também não é uma realidade para todos os testadores, pois os desenvolvedores não seguem as boas práticas no momento da criação do software, adicionando identificadores no código para utilizá-los no processo automatizado de testes de GUI, dificultando a automação dos cenários de testes do software. Dessa forma, é possível considerar que os testadores trabalham em empresas que se auto intitulam ágeis, mas não seguem as boas práticas dos métodos em suas rotinas de trabalho e não passaram pela mudança cultural necessária para ter os benefícios de uma atuação "agilista".

Mesmo com as fragilidades apontadas, a maioria dos testadores informou que atua em times Àgeis, principalmente aqueles que automatizam testes de GUI Como pode ser observado na Figura 6.1, os projetos Ágeis se destacam principalmente na implementação de testes de GUI de abordagem mista, ou seja, são executadas as avaliações de forma manual e automatizada, além de alguns respondentes que atuam somente de maneira automatizada. O modelo de desenvolvimento Incremental também possui quantidade significativa de testadores automatizando testes de GUI; diferentemente dos projetos que utilizam o modelo de desenvolvimento Cascata que investem esforços em execuções manuais. Vale destacar que nos modelos de desenvolvimento, há muitos profissionais que executam testes exclusivamente de maneira manual, mostrando que essa abordagem é utilizada independentemente do modelo de desenvolvimento utilizada nas empresas.

6.3 Motivações para escolher a abordagem de execução de testes de GUI

Como apresentado nos resultados da pesquisa (Seção 5.6), foi possível identificar, nas entrevistas, as motivações que levam os testadores a escolher entre a abordagem manual ou automatizada no momento de executar os testes de GUI. Primeiramente, são apresentados os motivos para a realização de testes de GUI manuais e, em seguida, as motivações para os testadores automatizarem os testes de GUI em seus projetos.

Motivos para testar GUI de forma manual

Mesmo com a crescente adoção de técnicas e ferramentas disponíveis no mercado para automação, muitos testadores ainda realizam verificações manuais na interface do software. Na etapa do questionário *online* e das entrevistas, somente pequena parcela de testadores realizam testes de GUI exclusivamente de forma automatizada, 9% e 5% respectivamente.

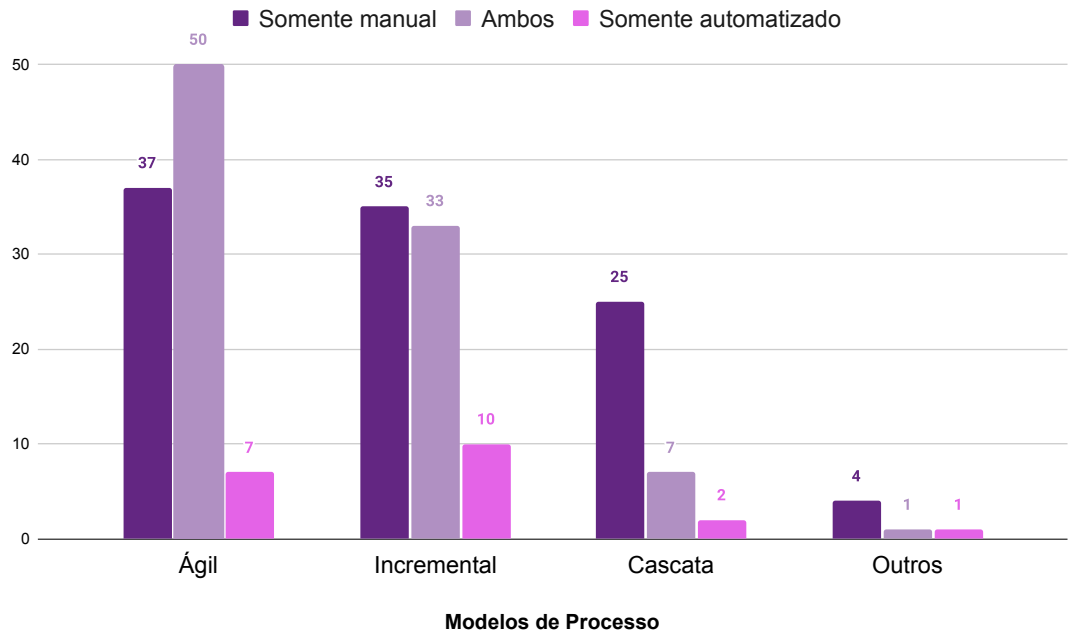

Figura 6.1: Abordagens de execução de testes de GUI utilizadas por modelo de processo de desenvolvimento

Como apresentado na Seção 5.6.2, há diversos motivos que levam os testadores a executarem os testes de GUI exclusivamente de forma manual. Primeiramente, em muitos casos, os projetos em que os testadores atuam não permitem que ocorra a criação de testes de GUI automatizados. Dentre os principais motivos, encontram-se: (i) projetos de curta duração que não comportam a atividade de automação dos testes de GUI; e (ii) falta de interesse por parte da organização de capacitar e alocar os funcionários para automatizar os testes. A gerência enxerga mais valor nos testes de GUI manuais e desconsidera os possíveis ganhos da automação de testes.

Além das questões de projeto, há problemas referentes ao aspecto educacional dos testadores. Alguns profissionais possuem dificuldades em executar atividades relacionadas à lógica de programação e, por isso, não criam efetivamente *scripts* automatizados em seus projetos. Provavelmente, Durante a formação técnica do testador, ele não teve êxito de maneira satisfatória em matérias de programação, mesmo com a obrigatoriedade de cursar disciplinas que envolvem codificação, por exemplo, análise de algoritmo, programação orientada a objetos e afins. Há duas razões para isso ter acontecido: (i) falta de empenho por parte do aluno durante a graduação; ou (ii) problemas no ensino (ou ambos). Por conta dessas dificuldades, alguns testadores mencionaram ter "repulsa" por programação e, consequentemente, não se especializam na automação de testes.

Motivos para testar GUI de forma automatizada

Como mencionado, os testadores que realizam testes de GUI automatizados possuem mais tempo de experiência na área de teste do que os testadores que executam exclusivamente testes de GUI manuais. Esse tempo maior de atuação na indústria permite que os testadores tenham olhar mais maduro sobre o mercado. Por trás dos motivos para a automação, podem estar o desejo por melhores salários, exigências das empresas sobre conhecimento de automação no momento da contratação e os ganhos para o projeto por ter um processo automatizado de testes, como menor tempo de execução e aumento da produtividade. O interesse em programar também foi destacado pelos entrevistados como uma motivação para implementar testes automatizados. Essa motivação está diretamente relacionada à autoestima dos testadores, visto que, com a automação, os testadores enxergam valor técnico maior em suas atividades.

6.4 Processo de teste de GUI manual

Analisando os objetivos definidos para os testes de GUI manuais, foi observado que os testadores em seus projetos estão avaliando distintos aspectos do software na mesma execução de testes, como a verificação dos elementos da tela, responsividade e usabilidade, além das funções padrão do software. Por conta desse contexto, é possível afirmar que os testadores posicionam-se em suas equipes com perfil polivalente no que diz respeito às tarefas de execução de testes.

Após a definição do objetivo do testes de GUI, os testadores utilizam diferentes bases de testes para criar os cenários a serem executados posteriormente. Foi observado que o modelo de desenvolvimento utilizado no projeto impacta no tipo de insumo utilizado pelo testador, como em projetos que utilizam o modelo de desenvolvimento Cascata que fornecem documentação tradicional como requisitos e regras de negócios, enquanto no modelo Ágil é fornecido aos profissionais histórias de usuários e critérios de aceite para a definição dos testes de GUI manuais. Além disso, cerimônias para definir *sprints* são utilizadas pelos testadores para conhecer um pouco mais do que será implementado no software e, por sua vez, obter mais detalhes do que precisará ser testado por ele no futuro.

Como informado anteriormente, o modelo de desenvolvimento também impacta na forma em que o cenário de teste é criado dentro dos projetos. Nos projetos que utilizam o modelo Ágil, vários testadores estruturam os seus testes em mapas mentais, que tendem a ser escritos mais rapidamente do que com escrita cursiva dependendo da complexidade do que será testado. A flexibilidade também é outro benefício dos mapas mentais, pois o testador consegue editar e organizar os cenários de forma mais dinâmica com alteração dos nós do mapa mental.

6.5 Limitações nos testes de GUI manuais

Muitos dos problemas identificados na criação e na execução dos testes de GUI estão associados a falhas de processo. Mesmo com a produção de diversos conteúdos por parte da literatura formal e cinza sobre as melhores práticas de Engenharia de Software em relação ao processo de teste, ainda há falhas na especificação de requisitos que levam à criação de requisitos frágeis, software implementados considerar a documentação previamente produzida e a falta de planejamento das tarefas. Assim, a etapa de teste de software é negligenciada e não realizada durante o ciclo de desenvolvimento do software.

Além disso, há dificuldades associadas à cultura da empresa em que os testadores trabalham. Com equipes sobrecarregadas, os poucos testadores dos projetos são alocados nos testes de GUI manuais e, por conta do imediatismo em avaliações manuais, não há espaço no cronograma do projeto para que os testadores consigam investir esforços nos testes de GUI automatizados. A escolha da abordagem manual em detrimento à automação é por conta da falta de funcionários. Isso, a longo prazo, pode dificultar a reprodutividade dos testes, seja pelo volume de testes necessários ou pelo cansaço físico e mental crescente do testador que irá executar a bateria de testes. Outro problema criado a partir da falta de mão de obra nos projetos é a alocação de profissionais com pouca experiência em funções que necessitam qualificações específicas. Dessa forma, é provável que não haja avaliação efetiva dos cenários relevantes ou críticos para o cliente nem a criação de testes de GUI automatizados para o software.

A maturidade da equipe também pode impactar o processo de teste. Por exemplo, colegas de trabalho inexperientes que não realizam o reporte apropriado de suas atividades levam os testadores a abandonar as suas tarefas de testes para descobrir o andamento real do projeto. Além disso, por vezes, o software é passado para a equipe de teste de GUI em uma versão do software instável, sem que os desenvolvedores tenham criado e executado testes de unidade, por exemplo.

Finalmente, também foram encontrados problemas relacionados às atividades comuns do testador. Dois exemplos dessas dificuldades são (i) ambientes de testes menos estáveis do que a estrutura de produção e (ii) dependência de integrações não disponibilizadas no momento do teste. Os responsáveis pelo projeto, como *Scrum Masters* ou Gerentes de Projetos, deveriam tentar minimizar o impacto negativo gerado por essas limitações, identificando no início do projeto as dependências e precondições necessárias para a realização dos testes de GUI no processo do desenvolvimento do software.

6.6 Processo de teste de GUI automatizado

A seguir, são tratados aspectos referentes aos testes de GUI automatizados, como as linguagens de programação utilizadas para automatizar os testes e as ferramentas que suportam a execução dos *scripts* de testes.

Linguagens de programação utilizadas

A linguagem de programação Java foi a mais popular entre os testadores que responderam ao questionário *online*. Também, foi identificado o uso de linguagens de tipagem dinâmica, como *Javascript*, *Python* e *Ruby*. Esse resultado foi similar ao observado nas entrevistas, nas quais os entrevistados citaram com destaque *Java*, *Python* e *Javascript* ocupando a primeira, segunda e terceira posição, respectivamente.

Além de *Java* ser uma linguagem popular, ela é compatível com o *Selenium*, uma das soluções de automação mais implementadas pelos testadores. Além disso, *Java* pode ser executada em qualquer sistema operacional e possui uma comunidade bastante ativa. *Python*, *Javascript* e *Ruby* também aparecem em destaque. Essas linguagens possuem uma curva de aprendizado menor comparadas com outras linguagens, como *C*. Elas também são compatíveis com várias ferramentas para automação de testes, como *Selenium* e *Robot Framework*, e têm crescente adesão pela indústria.

Ferramentas para automação

Dos testadores que relataram que a equipe de QA (*Quality Assurance*) ou os próprios testadores são os responsáveis pela escolha das ferramentas de automação de teste, metade usa métodos Ágeis. A autonomia para decidir como fazer o trabalho pode estar diretamente associada aos princípios do modelo de desenvolvimento aplicado. O manifesto Ágil[3] afirma que a equipe deve refletir sobre como tornar-se mais eficaz e ajustar suas soluções para cumprir suas demandas. O projeto deve proporcionar o ambiente e o suporte necessários para que os testadores sintam-se confiantes e decidam como realizar o trabalho.

Tanto no questionário *online* quanto nas entrevistas, foi observado que a segunda geração de testes de GUI é a mais utilizada para automação, principalmente com o uso do *Selenium* e do *Robot Framework*. Algumas vantagens dessa ferramenta são licença de código aberto, compatibilidade com vários sistemas operacionais (*Windows*, *Linux* e *Mac*) e uma comunidade ativa que ajuda no suporte técnico e na melhoria da solução. Em seguida, tem-se o *Robot Framework*, uma ferramenta que permite ao testador customizar códigos facilmente utilizando a linguagem *Python*. O *Cypress* também foi bastante citado. Ele permite automatizar testes de GUI, componentes e API em um único projeto de código fonte. Isso facilita o gerenciamento e a manutenção de diferentes tipos de testes que são organizados em uma única estrutura.

[3] `https://agilemanifesto.org/`

Além dessas ferramentas, os testadores relataram um conjunto de *frameworks* adicionais, como *Cucumber*, *Capybara* e *Behave*. Esses *frameworks* utilizam o *Gherkin* para a implementação de *BDD*[4] no processo de qualidade. O *BDD* permite o desenvolvimento de testes automatizados por meio de cenários de fácil leitura e compreensão por qualquer membro do projeto, como analista de requisitos, desenvolvedor ou testador. Portanto, projetos que utilizam *BDD* buscam uma abordagem colaborativa para definir e compartilhar os testes. Outra característica do *BDD* é a atenção especial às necessidades do negócio do cliente, de forma que os requisitos de mercado são informações necessárias para o desenvolvimento de testes.

6.7 Limitações nos testes de GUI automatizados

Foram identificados problemas organizacionais que impactam o processo de testes de GUI automatizados, assim como ocorre para os testes de GUI manuais. Por exemplo, computadores antigos são geralmente disponibilizados para os testadores realizarem as suas atividades. Contudo, pelo baixo desempenho, esses computadores não atendem aos requisitos mínimos necessários para a execução das ferramentas, levando o testador a gastar mais tempo na produção de *scripts* automatizados.

Os testadores também criticaram as ferramentas utilizadas para a automação. Segundo eles, as organizações não estão dispostas a dialogar sobre mudanças nos *frameworks* para criação de testes de GUI. Assim, perde-se funções disponíveis em outras soluções que facilitariam a atividade de teste. Também, não há recomendação por parte dos líderes de projeto para que os desenvolvedores adicionem *testing-id* nas páginas HTML no momento da criação de novas funções. Tal situação, dificulta a interação entre a ferramenta de automação de testes e o software sob teste.

A falta de documentação de boas práticas também é um problema recorrente nos projetos. Dentre os entrevistados que automatizam testes de GUI, 46% informaram que não há guias para auxiliar profissionais novos e veteranos a desempenharem as suas atividades da empresa. Além disso, qualquer tentativa de criar documentação é uma iniciativa por parte dos próprios funcionários e não da gerência.

6.8 Descobertas e recomendações finais

Com base nas citações dos participantes da pesquisa, foi possível identificar os contextos em que os testadores atuam profissionalmente e as respectivas dificuldades que eles precisam lidar no dia a dia de trabalho. Algumas sugestões para melhorar a execução dos testes de GUI são:

- **Formação profissional do testador:** como foi informado anteriormente, parte dos testadores são graduados em cursos de tecnologia e possuem tempo de experiência significativo atuando na área de testes. No entanto, muitos possuem

[4] `https://cucumber.io/docs/bdd/`

dificuldades na realização de suas tarefas de projeto, desde a criação de testes manuais, por não conseguirem identificar os cenários de testes, a codificação de *scripts* automatizados, pela falta de conhecimento em lógica e linguagem de programação. Para minimizar esse cenário, é recomendado buscar novas competências para atuar na área do testes por parte do testador, identificar essas fragilidades na formação dos profissionais e promover treinamentos e cursos para o aperfeiçoamento do colaborador;

- **Testabilidade do software:** por conta das limitações citadas pelos testadores para os software sob testes que eles avaliam, é possível que a testabilidade dos software seja baixa, considerando a instabilidade das aplicações, ambientes de testes sem recursos, falta de massa de dados e a inexistências de seletores para que as ferramentas de automação de testes possam interagir com o software. Para que haja qualidade no processo de testes, as pré-condições para a realização das avaliações precisam ser consideradas previamente no momento do planejamento do projeto pelo time;
- **Processo empregado no desenvolvimento de software:** independentemente do modelo de desenvolvimento utilizado nas empresas, distintas limitações foram identificadas na forma em que os projetos de desenvolvimento de software são gerenciados. Falhas como falta de planejamento das tarefas, comunicação ineficiente do time, ênfase na realização de atividades com pouco valor, equipes sobrecarregadas e ambientes conflituosos foram alguns exemplos levantados pelos testadores durante a pesquisa. Como no Cascata, no Incremental e nos métodos Ágeis são debatidos soluções para esses problemas, é recomendado que os responsáveis pelas equipes revisem em que momento a execução do projeto distanciou-se das melhores práticas adotadas pelos métodos que suportam as atividades de desenvolvimento de software e, em seguida, seja avaliado quais medidas podem ser implementadas a curto e médio prazo visando melhorar a saúde do projeto;
- **Investimento na automação de testes de GUI:** A atividade de automação de testes exige diversas elementos para ser realizada, como habilidade em desenvolvimento por parte do testador, uso de ferramentas e custos adicionais de implementação. Porém, os benefícios de testes de GUI automatizados podem ser maiores. Esses vão desde a redução de erros gerados por humanos até a possibilidade da execução de mais testes a cada nova versão do software. Mesmo com as vantagens mencionadas, muitas empresas aparentemente não possuem interesse em utilizar esse tipo de abordagem nos seus projetos, visto que testadores afirmaram maior incentivo por parte das suas organizações na execução de testes manuais, falta de tempo nos projetos para a automação dos testes de GUI e disponibilização de computadores antigos para a codificação de *scripts* de testes. Manter somente a execução de testes de GUI manuais pode se tornar débito técnico para o projeto. Assim, recomenda-se que as empresas avaliem a possibilidade de, pelo menos, automatizar os cenários de testes de maior valor para o software para minimizar o esforço do time.

6.9 Síntese de capítulo

Nesse capítulo, forma discutidos os principais resultados obtidos nas atividades do questionário *online* e nas entrevistas, como o perfil dos testadores e aspectos referentes aos testes de GUI manuais e testes de GUI automatizados. No próximo capítulo, são apresentadas as considerações finais da pesquisa.

Capítulo 7

Considerações Finais

7.1 Conclusões

Os resultados desta dissertação contribuem para entendimento comum das práticas que a indústria possui para a realização de testes de GUI. A partir das 222 participações no questionário *online* e das 20 entrevistas com profissionais da indústria brasileira de software, foi possível identificar e catalogar as experiências dos testadores de GUI.

Foi observado no questionário que muitos testadores de software executam testes de GUI principalmente de forma manual, utilizando casos de testes no formato *Gherkin* ou verificações exploratórias. Nas entrevistas, foi explorado um pouco mais e identificado que, dentre os principais motivos que levam os testadores a realizarem testes de GUI manuais, há dificuldades com as atividades que envolvem programação e preferência pessoal por avaliações manuais. Notou-se também dificuldades referentes à organização em que os testadores trabalham e, principalmente, em relação aos processos deficitários dos projetos em que eles estão alocados, como falta de profissionais para desempenharem a alta carga de atividades do projeto, problemas com os computadores disponibilizados para os testadores e comunicação ineficiente entre os membros das equipes.

Em relação aos testes de GUI automatizados, foi identificado que os testadores que realizam a tarefa de automação de testes possuem mais tempo de experiência do que os testadores que realizam testes de GUI manuais exclusivamente. As principais justificativas para os testadores automatizarem testes de GUI são: desejo por melhores salários, desenvolvimento da carreira, a redução do tempo para testar um software e o interesse em programação. As soluções mais citadas para a automação de testes de GUI foram utilizando a linguagem *Java* e a ferramenta *Selenium*. Ademais, diversos respondentes afirmaram que utilizam o *BDD* como técnica adicional para a criação de testes de GUI automatizados.

Além disso, cerca de metade dos testadores que automatizam testes e participaram

do questionário *online* relataram que as soluções disponíveis não têm limitações. Aqueles que relataram limitações, queixaram-se do longo tempo necessário para a execução da suíte de testes e da impossibilidade de testar o software em mais de uma aba do navegador de Internet.

7.2 Contribuições

Os resultados trazidos nesta dissertação apresentam contribuições para a área de testes de software, principalmente para testes de GUI. A pesquisa foi produzida a partir da perspectiva dos testadores da indústria brasileira de software. Eles compartilharam informações a respeito do contexto de trabalho e do histórico de carreira profissional. As principais contribuições da pesquisa são:

1. Apresentação do perfil dos testadores de software da indústria por meio de um recorte, no qual foi informado o nível de escolaridade, as fontes de conhecimento utilizadas para atuar como testadores e os motivos para atuarem na área de teste de software;
2. Apresentação das formas em que os testadores realizam os testes de GUI manuais e automatizados;
3. Apresentação de aspectos referentes ao processo de testes de GUI manual, como objetivo do testes e insumos de suporte utilizados na criação dos cenários de testes;
4. Apresentação de aspectos referentes ao processo de testes de GUI manual e automatizado, como as linguagens de programação e ferramentas mais utilizadas pelos testadores que criam testes de GUI automatizados;
5. Apresentação das principais limitações que impactam nas atividades dos testadores de software, tanto na abordagem manual quanto automatizada;
6. Apresentação das motivações para os testadores escolherem entre a abordagem manual ou automatizada para os testes de GUI.

A indústria pode utilizar os resultados informados na pesquisa visando melhorar o seu processo de teste de GUI, principalmente ao evitar as limitações informadas pelos testadores tanto na abordagem manual quanto na automatizada. Além disso, a acadêmia poderá desenvolver soluções para minimizar as dificuldades informadas pelos testadores e facilitar as atividades diárias dos testadores nos projetos de desenvolvimento de software.

7.3 Trabalhos Futuros

Na realização de trabalhos futuros, será considerado que muitos testadores ainda realizam testes de GUI exclusivamente de forma manual pelo fato de terem dificul-

dade ou até mesmo não possuírem conhecimento sobre programação. Esse contexto será o início para a produção de novos estudos que pretendem:

- Prover meios de incentivar as instituições de ensino formal a abordar testes de software nas disciplinas dos cursos de graduação;
- Investigar os motivos que levam os testadores de software a não saber programar, visto que eles cursaram disciplinas relacionadas a lógica de programação durante a graduação;
- Desenvolver um conjunto de *guidelines* que vise orientar os testadores que executam somente testes de GUI manuais a também automatizar testes.

Referências

25000, S. I. (2011). Iso/iec 25010. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?start=6. Last checked on Sep 08, 2021.

Adams, W. C. (2015). *Conducting Semi-Structured Interviews*, chapter 19, páginas 492–505. John Wiley Sons, Ltd.

Alégroth, E., Feldt, R., e Olsson, H. H. (2013). Transitioning manual system test suites to automated testing: An industrial case study. In *Sixth IEEE International Conference on Software Testing, Verification and Validation*, ICST'13, páginas 56–65. IEEE Computer Society.

Alegroth, E., Feldt, R., e Ryrholm, L. (2014). Visual gui testing in practice: challenges, problemsand limitations. *Empirical Software Engineering*, 20.

Alegroth, E., Gao, Z., Oliveira, R., e Memon, A. (2015). Conceptualization and evaluation of component-based testing unified with visual gui testing: An empirical study. In *IEEE 8th International Conference on Software Testing, Verification and Validation*, ICST'15.

Alferidah, S. K. e Ahmed, S. (2020). Automated software testing tools. In *2020 International Conference on Computing and Information Technology (ICCIT-1441)*, páginas 1–4.

Alotaibi, A. A. e Qureshi, R. (2017). Novel framework for automation testing of mobile applications using appium. *International Journal of Modern Education and Computer Science*, 9:34–40.

AltexSoft (2020). Quality-assurance-quality-control-and-testing-whitepaper. https://www.altexsoft.com/media/2016/10/Quality-Assurance-Quality-Control-and-Testing-Whitepaper.pdf. Last checked on Aug 29, 2021.

Altexsoft (2021). Acceptance criteria for user stories: Purposes, formats, examples, and best practices.

Alégroth, E., Feldt, R., e Kolström, P. (2016). Maintenance of automated test suites in industry: An empirical study on visual gui testing.

Alégroth, E., Karlsson, A., e Radway, A. (2018). 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)*, páginas 172–181.

Alégroth, E., Petersén, E., e Tinnerholm, J. (2021). A failed attempt at creating guidelines for visual gui testing: An industrial case study. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, páginas 340–350.

Amalfitano, D., Fasolino, A. R., Tramontana, P., De Carmine, S., e Memon, A. M. (2012). Using gui ripping for automated testing of android applications. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ASE 2012, página 258–261, New York, NY, USA.

Arnatovich, Y. L. e Wang, L. (2018). A systematic literature review of automated techniques for functional gui testing of mobile applications.

Bach, T. (2022). Testing in very large software projects. Dissertação de Mestrado, The Faculty of Mathematics and Computer Science, https://archiv.ub.uni-heidelberg.de/volltextserver/31757/7/dissertation-tbach-color-druck.pdf.

Banerjee, I., Nguyen, B., Garousi, V., e Memon, A. (2013). Graphical user interface (gui) testing: Systematic mapping and repository. *Information and Software Technology*, 55(10):1679–1694.

Busjaeger, B. e Xie, T. (2016). Learning for test prioritization: An industrial case study. FSE 2016, página 975–980, New York, NY, USA. Association for Computing Machinery.

Canny, A., Palanque, P., e Navarre, D. (2020). Model-Based Testing of GUI Applications Featuring Dynamic Instanciation of Widgets. In *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2020)*, páginas 95–104, Porto (virtual), Portugal. IEEE.

Chandorkar, A., Patkar, N., Sorbo, A. D., e Nierstrasz, O. (2022). An exploratory study on the usage of gherkin features in open-source projects. In *IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022, Honolulu, HI, USA, March 15-18, 2022*, páginas 1159–1166. IEEE.

Cheng, Y.-P., Li, C.-W., e Yi-Cheng, C. (2019). Apply computer vision in gui automation for industrial applications. *Mathematical Biosciences and Engineering*, 16:7526.

CISQ (2022). The cost of poor software quality in the u.s. `https://www.synopsys.com/software-integrity/resources/analyst-reports/cost-poor-quality-software.html?intcmp=sig-blog-cisq22`. Last checked on Jan 02, 2022.

Coppola, R. e Alégroth, E. (2022). A taxonomy of metrics for gui-based testing research: A systematic literature review. *Information and Software Technology*, 152:107062.

Coppola, R., Ardito, L., Morisio, M., e Torchiano, M. (2020). Mobile testing: New challenges and perceived difficulties from developers of the italian industry. *IT Professional*, 22(5):32–39.

Corbin, J. e Strauss, A. (2008). Basics of qualitative research (3rd ed.): Techniques and procedures for developing grounded theory.

Coutinho, J., Andrade, W., e Machado, P. (2022). A survey of requirements engineering and software testing practices in agile teams. SAST '22, página 9–18, New York, NY, USA. Association for Computing Machinery.

Dallal, J. A. (2009). Automation of object-oriented framework application testing. In *2009 5th IEEE GCC Conference Exhibition*, páginas 1–5. IEEE Computer Society.

Do Nascimento, F. e Sousa, F. (2016). *Metodologia Da Pesquisa Cientifica: TEORIA E PRATICA*. Thesaurus, first edição.

Etheredge, J. (2018). Software complexity is killing us. `https://www.simplethread.com/software-complexity-killing-us/`. Last checked on Aug 29, 2021.

Garousi, V., Afzal, W., Çağlar, A., İhsan Berk Işık, Baydan, B., Çaylak, S., Boyraz, A. Z., Yolaçan, B., e Herkiloğlu, K. (2020a). Visual gui testing in practice: An extended industrial case study.

Garousi, V., Rainer, A., Lauvås, P., e Arcuri, A. (2020b). Software-testing education: A systematic literature mapping. *Journal of Systems and Software*, 165:110570.

Glenford, M. (1979). *The Art of Software Testing*. New York: Word Association, second edição.

Grechanik, M., Xie, Q., e Fu, C. (2009). Creating gui testing tools using accessibility technologies. In *International Conference on Software Testing, Verification, and Validation Workshops*, páginas 243–250. IEEE Computer Society.

Gunasekaran, S. e Bargavi, V. (2015). Survey on automation testing tools for mobile applications. *International Journal of Advanced Engineering Research and Science*, 2(11):2349–6495.

Huang, C.-Y., Chang, J.-R., e Chang, Y.-H. (2010). Design and analysis of gui test-case prioritization using weight-based methods. *Journal of Systems and Software*, 83(4):646–659.

ISTQB (2018). Istqb® worldwide software testing practices survey 2017-18. https://www.istqb.org/references/surveys/istqb%C2%AE-worldwide-software-testing-practices-survey-2017-18.html. Last checked on Aug 29, 2021.

ISTQB (2020). Facts & figures istqb. https://www.istqb.org/about-us/facts-figures.html. Last checked on Aug 29, 2021.

Jones, C. (2015). Wastage: The impact of poor quality on software economics. *Software Quality Professional*, 18(1).

Junior, N., Costa, H., Karita, L., Machado, I., e Soares, L. (2021). *Experiences and Practices in GUI Functional Testing: A Software Practitioners' View*, página 195–204. Association for Computing Machinery, New York, NY, USA.

Karhu, K., Repo, T., Taipale, O., e Smolander, K. (2009). Empirical observations on software testing automation. In *International Conference on Software Testing Verification and Validation*.

Kasunic, M. (2005). Designing an effective survey. Relatório técnico, Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst.

Kitchenham, B. A. e Pfleeger, S. L. (2002). Principles of survey research part 2: Designing a survey. *SIGSOFT Softw. Eng. Notes*, 27(1):18–20.

Klaus Olsen, M. P. e Ulrich, S. (2019). *Certified Tester Foundation Level Syllabus*. International Software Testing Qualifications Board, https://bstqb.org.br/b9/doc/syllabus_ctfl_2018br.pdf, third edição.

Leitner, A., Ciupa, I., Meyer, B., e Howard, M. (2007). Reconciling manual and automated testing: The autotest experience. In *40th Annual Hawaii International Conference on System Sciences*, HICSS '07.

Memon, A. M., Pollack, M. E., e Soffa, M. L. (2001). Hierarchical gui test case generation using automated planning. 27(2):144–155.

Mohammad, D. R., Al-Momani, S., Tashtoush, Y. M., e Alsmirat, M. (2019). A comparative analysis of quality assurance automated testing tools for windows mobile applications. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*.

Molyneaux, I. (2015). *The Art of Application Performance Testing: From Strategy to Tools*. O'Reilly Media, second edição.

Moreira, R., Paiva, A., Nabuco, M., e Memon, A. (2017). Pattern-based gui testing: Bridging the gap between design and quality assurance. *Software Testing, Verification and Reliability*, 27.

of Notrh Carolina at Chapel Hill, U. (2014). Skimming.

Qureshi, I. A. e Nadeem, A. (2013). Gui testing techniques: A survey. *International Journal of Future Computer and Communication*, páginas 142–146.

Ramler, R. e Wolfmaier, K. (2006). Economic perspectives in test automation: Balancing automated and manual testing with opportunity cost. In *Proceedings of the 2006 International Workshop on Automation of Software Test*, AST '06, página 85–91, New York, NY, USA. Association for Computing Machinery.

Rothermel, G., Untch, R., Chu, C., e Harrold, M. (2001). Prioritizing test cases for regression testing. *IEEE Transactions on Software Engineering*, 27(10):929–948.

Santos, I., M. Melo, S., S. L. Souza, P., e R. S. Souza, S. (2022). A survey on the practices of software testing: a look into brazilian companies. *Journal of Software Engineering Research and Development*, 10:11:1 – 11:15.

Schaefer, H. (2005). Risk based software testing: Strategies for prioritizing tests against deadlines. `http://www.methodsandtools.com/archive/archive.php?id=31`. Last checked on Sep 08, 2021.

Sharma, L. (2021). Error, defect, and failure. `https://www.toolsqa.com/software-testing/istqb/error-defect-failure/`. Last checked on Aug 29, 2021.

Sharma, R. M. (2014). Quantitative analysis of automation and manual testing. Relatório Técnico Volume 4, International Journal of Engineering and Innovative Technology (IJEIT).

Shruti Malve, P. S. (2017). Investigation of manual and automation testing using assorted approaches. *International Journal of Scientific Research in Computer Science and Engineering*, 5:81–87.

Sneha, K. e Malle, G. M. (2017). Research on software testing techniques and software automation testing tools. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, páginas 77–81.

Tanaka, H. (2019). X-brot: Prototyping of compatibility testing tool for web application based on document analysis technology. páginas 18–21.

Tramontana, P., Amalfitano, D., Amatucci, N., e Fasolino, A. R. (2019). Automated functional testing of mobile applications: a systematic mapping study. *Softw. Qual. J.*, 27(1):149–201.

University, R. (2014). Literature search: Snowball and citation search.

Valente, M. T. (2020). Engenharia de software moderna. `https://engsoftmoderna.info/`. Last checked on Aug 29, 2021.

White, T. D., Fraser, G., e Brown, G. J. (2019). Improving random gui testing with image-based widget detection. ISSTA 2019, página 307–317, New York, NY, USA. Association for Computing Machinery.

Ye, J., Chen, K., Xie, X., Ma, L., Huang, R., Chen, Y., Xue, Y., e Zhao, J. (2021). An empirical study of gui widget detection for industrial mobile games. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2021, página 1427–1437, New York, NY, USA. Association for Computing Machinery.

Apêndice A

Questionário Online

A1 APRESENTAÇÃO DA PESQUISA

Prezado participante,

Atualmente, há diferentes maneiras de realizar testes de software, uma das mais conhecidas é testes funcionais de GUI (Graphical User Interface), onde o testador, utilizando a interface gráfica, interage com os elementos das telas do software para avaliar a qualidade funcional.

Diante disso, a pesquisa a seguir busca entender quais são os diferentes perfis de testadores de software, como também, identificar como os testes funcionais de GUI são executados atualmente pelos profissionais da área.

A sua participação é importante para entendermos mais a respeito do cenário de testes funcionais e para podermos pensar em melhorias e soluções a serem propostas futuramente.

O tempo médio de preenchimento do formulário é de 10 minutos.

A pesquisa não divulgará nenhum dado do entrevistado!

Atenciosamente, Nailton Soares de Almeida Junior Mestrando em Ciência da Computação pela Universidade Estadual de Feira de Santana.

A2 TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

1. Você está sendo convidado(a) a participar voluntariamente da pesquisa sobre o perfil dos testadores e suas práticas de execução de testes funcionais de GUI.
2. Sua participação não é obrigatória.
3. A qualquer momento você pode desistir de participar e retirar seu consentimento.
4. Sua recusa não trará nenhum prejuízo em sua relação com o pesquisador ou com a instituição.

5. Sua participação consistirá em responder um questionário com questões abertas e objetivas.
6. Para minimizar qualquer desconforto e manter sua privacidade, o questionário apresentará caráter anônimo.
7. Os benefícios relacionados à sua participação estão apenas em contribuir com a pesquisa científica. Será permitido acesso aos resultados desta pesquisa por meio de publicações científicas realizadas a partir desse estudo.
8. Ao continuar respondendo este questionário, você concorda com as informações aqui descritas, porém a qualquer momento você poderá interromper a pesquisa sem ônus algum.
9. Este questionário utiliza o pacote de aplicativo Google Docs, portanto a coleta e o uso de informações do Google estão sujeitos à Política de privacidade do Google (https://www.google.co.uk/policies/privacy/)
10. Abaixo, seguem os dados de contato dos responsáveis por esta pesquisa, com os quais você pode tirar suas dúvidas sobre sua participação:

    -Nailton Soares de Almeida Junior - nailtonalmeidajr@gmail.com
    -Larissa Rocha - lrsoares@uefs.br (Professora da Universidade Estadual de Feira de Santana - orientadora)
    -Heitor Augustus Xavier Costa - heitor@ufla.br (Professor da Universidade Federal de Lavras)
    -Ivan Machado - ivanmachado@dcc.ufba.br (Professor da Universidade Federal da Bahia)

Você concorda com o termo de consentimento?* (Múltipla escolha)

- Sim
- Não

Você faz ou já realizou testes funcionais de GUI?* (Múltipla escolha)

- Sim
- Não

A3 PERFIL DO TESTADOR DE SOFTWARE

1 Em qual estado do país você mora?* (Múltipla escolha)

- Acre (AC)
- Alagoas (AL)
- Amapá (AP)
- Amazonas (AM)

- Bahia (BA)
- Ceará (CE)
- Distrito Federal (DF)
- Espírito Santo (ES)
- Goiás (GO)
- Maranhão (MA)
- Mato Grosso (MT)
- Mato Grosso do Sul (MS)
- Minas Gerais (MG)
- Pará (PA)
- Paraíba (PB)
- Paraná (PR)
- Pernambuco (PE)
- Piauí (PI)
- Rio de Janeiro (RJ)
- Rio Grande do Norte (RN)
- Rio Grande do Sul (RS)
- Rondônia (RO)
- Roraima (RR)
- Santa Catarina (SC)
- São Paulo (SP)
- Sergipe (SE)
- Tocantins (TO)

2 Qual é a sua formação?* (Múltipla escolha)

- Ensino Médio
- Ensino Técnico
- Graduação
- Especialização
- Mestrado

- Doutorado

3 Caso tenha nível superior, qual graduação já cursou? (Caixa de seleção)

- Ciência da Computação
- Engenharia da Computação
- Análise e Desenvolvimento de Sistemas
- Sistemas de Informação
- Rede de Computadores
- Outros

4 Você possui alguma certificação na área de testes de software? Caso tenha, informe as certificações separando-as por ponto e vírgula (;) (Questão aberta)

5 Além de testar software, você desenvolve? Em qual linguagem de programação?* (Caixa de seleção)

- C
- C++
- C#
- Java
- JavaScript
- Perl
- PHP
- Python
- Ruby
- Visual Basic
- Nenhuma das alternativas
- Outros

A4 CONTEXTO DO LOCAL DE TRABALHO

1 Qual cargo ou função você desempenha atualmente na empresa em que trabalha?* (Questão aberta)

2 Qual a área que o software da sua empresa atende?* (Caixa de seleção)

- Comunicação
- Educacional

- Financeira
- Games e Entretenimento
- Hospitalar
- Médica
- Outros

3 Qual modelo de desenvolvimento de software é utilizado atualmente na empresa em que trabalha?* (Múltipla escolha)

- Espiral
- Incremental
- XP
- Outros

4 Na sua atual experiência profissional, quem é o responsável pelos testes funcionais de GUI?* (Caixa de seleção)

- Equipe de testes (QA) exclusivamente
- Desenvolvedor que criou a funcionalidade
- Desenvolvedor que não criou a funcionalidade
- Equipe de suporte ao usuário
- Outros

5 Na sua atual experiência profissional, o que é verificado na execução dos testes funcionais de GUI?* (Caixa de seleção)

- As novas funcionalidades e algumas funções antigas do sistema
- As novas funcionalidades e todas as funções antigas do sistema
- Somente as funções antigas do sistema
- Outros

6 Em qual plataforma os testes funcionais de GUI são executados?* (Caixa de seleção)

- Desktop
- Mobile
- Web
- Outros

7 Atualmente, de que forma você realiza os testes funcionais de interface (GUI)?* (Múltipla escolha)

- Manualmente
- Automatizada
- Ambos

A5 EXPERIÊNCIA COM TESTES MANUAIS

1 Quantos anos de experiência você possui em testes funcionais de GUI?* (Múltipla escolha)

- Menos de 1 ano
- Entre 1 e 3 anos
- Entre 3 anos e 5
- Entre 5 e 8 anos
- Mais de 8 anos

2 Qual ação você realiza nos testes funcionais de GUI?* (Múltipla escolha)

- Somente planeja os testes
- Planeja e executa os testes
- Somente executa os testes

3 De que forma você executa manualmente os testes GUI?* (Caixa de seleção)

- De forma exploratória
- Com casos de teste
- Com checklists

4 Como os bugs identificados na execução dos testes funcionais de GUI são reportados?* (Caixa de seleção)

- Chamado na plataforma Mantis
- Criação de card no Kanban
- Com envio de e-mail
- Controle com planilhas
- Mensagem no Slack
- Outros

A6 EXPERIÊNCIA COM TESTES AUTOMATIZADOS

1 Quantos anos de experiência você possui em testes funcionais GUI?* (Múltipla escolha)

- Menos de 1 ano
- Entre 1 e 3 anos
- Entre 3 anos e 5
- Entre 5 e 8 anos
- Mais de 8 anos

2 Em qual linguagem de programação, você automatiza os testes funcionais de GUI?* (Caixa de seleção)

- Java
- JavaScript
- Perl
- PHP
- Python
- Ruby
- Visual Basic
- Outros

3 Na sua atual experiência profissional, quais ferramentas são utilizadas na automação de testes funcionais de GUI?* (Caixa de seleção)

- Appium
- EyeAutomated
- IBM Rational Functional Tester
- Microsoft Coded UI Tests
- Oracle Application Testing Suite
- Ranorex
- Selenium
- SilkTest
- Sikuli
- TestComplete
- UFT
- Outros

4 Além das ferramentas informadas anteriormente, quais outros frameworks são utilizados na automação dos testes funcionais de GUI? (Caixa de seleção)

- Cucumber
- Behave
- Capybara
- Watir
- Outros

5 Quem propôs o uso das ferramentas utilizadas na automação dos testes funcionais de GUI na sua empresa?* (Questão aberta)

6 Com que frequência os testes funcionais de GUI automatizados são executados na sua empresa? (Caixa de seleção)

- Diariamente
- 3 dias por semana
- 5 dias por semana
- Após inclusão de nova funcionalidade
- Após correção de bugs
- Outros

7 De todos os testes funcionais de GUI executados, qual a proporção de testes você acredita que é feita de forma automatizada?* (Múltipla escolha)

- Entre 10% a 30% de todos os testes
- Entre 30% a 50% de todos os testes
- Entre 50% a 70% de todos os testes
- Entre 70% a 90% de todos os testes
- Entre 100% de todos os testes

8 Quais são as maiores limitações das ferramentas que você utiliza para a automação dos testes funcionais de GUI? Comente sobre:* (Questão aberta)

9 Você considera que as ferramentas utilizadas para a automação dos testes funcionais de GUI atendem a sua necessidade? Comente sobre:* (Questão aberta)

10 Como os bugs identificados na execução automatizada dos testes funcionais de GUI são reportados?* (Caixa de seleção)

- Mensagens de e-mail
- Chamado na plataforma Mantis
- Criação de card no Kanban

- Mensagem no Slack
- Outros

11 Além dos testes funcionais de GUI, quais outros você realiza?* (Caixa de seleção)

- Testes de unidade
- Testes de API
- Testes de carga / performance / stress
- Testes de integração
- Nenhum
- Outros

12 Além dos testes funcionais de GUI, quais outros testes de software são realizados na sua empresa?* (Caixa de seleção)

- Testes de unidade
- Testes de API
- Testes de carga / performance / stress
- Testes de integração
- Nenhum
- Outros

A6 AVALIAÇÃO DO QUESTIONÁRIO ONLINE

1 Você tem algum comentário sobre o tema da pesquisa ou sobre as questões do formulário? (Questão aberta)

2 Caso tenha interesse em receber os resultados desta pesquisa ou se disponibilize para responder questionários futuros, favor informe o seu e-mail: (Questão aberta)

Apêndice B

Protocolo de Entrevistas

B.1 Roteiro da entrevista

1. **Objetivo geral das entrevistas:**
    - Entender os processos relacionados ao planejamento e execução de testes de GUI, tanto manuais, quanto automatizados, por profissionais no contexto industrial.
2. **Objetivos específicos das entrevistas:**
    - Identificar como o profissional iniciou a sua carreira na área de testes de software;
    - Identificar quais foram as fontes de estudo para a atuação do profissional na área;
    - Entender a percepção do profissional a respeito do teste de GUI manual;
    - Entender a percepção do profissional a respeito do teste de GUI automatizado.
3. **Hipóteses:**
    - Os profissionais que executam exclusivamente testes de GUI manuais não possuem os conhecimentos técnicos de programação e ferramentas suficientes para a automação dos cenários de testes;
    - As organizações em que os profissionais atuam não oferecem um ambiente propício para a automação de testes de GUI.
4. **Público alvo:**
    - Profissionais que executam testes de GUI na indústria.
5. **Norteadores para a realização da entrevista:**

- A entrevista será feita de forma semi-estruturada, permitindo o entrevistado responder às perguntas da forma que desejar;
- O entrevistador não deverá interromper o entrevistado;
- O entrevistador não deverá orientar a resposta do entrevistado;
- Deverá ser reduzido vieses e evitar julgamento de valor sobre as respostas do entrevistado.

6. **Critérios para o convite de profissionais para as entrevistas:**
   - Convidar profissionais que responderam o questionário online da etapa anterior da pesquisa;
   - Convidar profissionais da área de testes através de redes sociais;
   - Encaminhar convites para empresas de tecnologia, principalmente as do ramo de testes de software.
7. **Atividades pré-entrevista:**
   - Após a confirmação de participação na pesquisa por parte do profissional, a entrevista será agendada através de e-mail;
   - O termo de consentimento será encaminhado via e-mail para o profissional para que seja assinado de maneira eletrônica.
   - Testar a plataforma de videoconferências que será feita a entrevista posteriormente;
8. **Realização da entrevista:**
   - Informar os objetivos da pesquisa ao entrevistado;
   - Questionar o entrevistado se ele possui alguma dúvida;
   - Informar que a entrevista será gravada, como também as regras de confidencialidade;
   - Realizar as perguntas do roteiro.
9. **Após a entrevista:**
   - Solicitar críticas ou sugestões a respeito das perguntas feitas para o entrevistado;
   - Informar que o participante receberá o produto da pesquisa após o encerramento do estudo através de e-mail;
   - Informar que a gravação está sendo encerrada.

B.2 Termo de consentimento livre e esclarecido

O Sr.(a) está sendo convidado(a) para participar da pesquisa "UM ESTUDO SOBRE AS PRÁTICAS DE TESTES FUNCIONAIS DE GUI NA INDÚSTRIA BRASILEIRA DE SOFTWARE". Nesta pesquisa pretendemos identificar as motivações em executar os testes de GUI de maneira manual ou automatizada. O intuito que nos leva a estudar este tópico, é a relevância em identificar qual o contexto atual da execução dos testes de interface na indústria, a fim de entender como o processo é feito e como poderia ser melhorado. Para esta pesquisa, serão coletados dados a respeito das práticas de testadores para os testes de GUI através de entrevistas semi-estruturadas. As entrevistas realizadas serão gravadas para a posterior extração dos dados, mas de forma alguma as gravações serão publicadas on-line ou compartilhadas com terceiros. Os benefícios relacionados à sua participação estão apenas em contribuir com a pesquisa científica. Será permitido acesso aos resultados desta pesquisa por meio de publicações científicas realizadas a partir desse estudo. Para participar deste estudo o(a) Sr.(a) não terá nenhum custo, nem receberá qualquer vantagem financeira. O Sr.(a) terá o esclarecimento sobre o estudo em qualquer aspecto que desejar e estará livre para participar ou recusar-se a participar. Poderá retirar seu consentimento ou interromper a participação a qualquer momento. A sua participação é voluntária e a recusa em participar não acarretará qualquer penalidade ou modificação na forma em que é atendido pelo(a) Universidade Estadual de Feira de Santana e pelo pesquisador, que tratará a sua identidade com padrões profissionais de sigilo. Caso o (a) Sr.(a) tenha alguma dúvida ou necessite de qualquer esclarecimento ou ainda deseje retirar-se da pesquisa, por favor, entre em contato com os pesquisadores abaixo a qualquer tempo.

Pesquisador Responsável – Nailton Almeida, nailtonalmeidajr@gmail.com, (075) 99260-2607 Também em caso de dúvida, o(a) Senhor(a) poderá entrar em contato com o Comitê de Ética em Pesquisa com Seres Humanos da Universidade Estadual de Feira de Santana (UEFS) . O Comitê de Ética em Pesquisa (CEP) busca defender os interesses dos participantes de pesquisa. O CEP é responsável pela avaliação e acompanhamento dos aspectos éticos de todas as pesquisas envolvendo seres humanos. O Comitê de Ética em Pesquisa da Universidade Estadual de Feira de Santana (UEFS) está localizado na Avenida Transnordestina, S/N, Bairro: Novo Horizonte, Feira de Santana - Bahia - Módulo 1, MA 17. Horário de funcionamento: De Segunda-feira a Sexta-feira das 13h30min às 17h30min. Telefone: (75) 3161-8124. E-mail: cep@uefs.br.

B.3 Formulário de questões da entrevista

Seção 1. Perguntas de identificação do perfil do entrevistado

1. Há quantos anos você trabalha na área de teste de software?
2. Como você começou a trabalhar com testes de GUI?

3. Qual o seu nível de escolaridade?
4. Onde você obteve conhecimento para atuar com testes de software?
5. Você acha importante esse conteúdo ser abordado na universidade?
6. Atualmente, você testa de forma manual ou automatizada?

Seção 2. Perguntas para profissionais que executam testes de GUI manuais

1. Como é o seu processo de criação de testes de GUI manuais?
2. O seu processo de criação de testes de GUI é documentado? A documentação é revisada posteriormente à sua criação?
3. Quais as principais dificuldades para a criação de testes de GUI manuais? Como seria possível resolver esses problemas?
4. Quais as principais dificuldades para a execução dos testes? Como seria possível facilitar a execução dos testes?
5. Você faz algum outro tipo de teste manual que não seja teste de GUI?
6. Você tem familiaridade com as tecnologias utilizadas para a automação de testes de GUI?
7. O que impede você de automatizar testes de GUI atualmente? Você tem vontade de realizar essa atividade?
8. O seu ambiente de trabalho incentiva mais o desenvolvimento de testes manuais ou automatizados?
9. Qual o seu sentimento em relação a execução dos testes de GUI manuais no seu projeto?

Seção 3. Perguntas para profissionais que executam testes de GUI automatizados

1. Como é o seu processo de criação de testes de GUI automatizados?
2. Como posteriormente é feita a manutenção dos testes criados?
3. O seu processo de criação de testes de GUI automatizados é documentado?
4. Quais as principais dificuldades para a criação dos testes automatizados?
5. Como seria possível facilitar o processo de criação de testes de GUI automatizados?
6. Quais foram as motivações para você automatizar testes de GUI?
7. Quais tecnologias você se familiarizou para automatizar testes de GUI? O que orientou essas escolhas?
8. Como o seu ambiente de trabalho incentiva o desenvolvimento de testes automatizados?
9. Você prefere executar os testes de forma manual ou automatizada? Por que?

Apêndice C

Imagens de Apoio

C.1 Mapas mentais utilizados na escrita de teste de GUI

A Figura C.1 exemplifica como cenários de testes de GUI podem ser escritos através de mapas mentais. Como pode ser observado, em um cenário hipotético de um software que permite o acesso mediante uso de credenciais de segurança, o testador pode definir em um mapa mental que testaria a **Funcionalidade de Login** através de duas condições de testes, sendo elas **login válido** e **login inválido**. Em seguida os cenários de testes foram definidos nas respectivas condições informadas, como realizar **login no software com o usuário incorreto e uma senha correta**.

Figura C.1: Exemplo de mapa mental utilizado na criação dos testes de GUI