UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

CARLA LIZANDRA DA SILVA

INVESTIGANDO O USO DE PRÁTICAS MODERNAS DE TESTES DE SOFTWARE NA INDÚSTRIA E NA ACADEMIA

QUIXADÁ

2025

CARLA LIZANDRA DA SILVA

INVESTIGANDO O USO DE PRÁTICAS MODERNAS DE TESTES DE SOFTWARE NA INDÚSTRIA E NA ACADEMIA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Orientadora: Profa. Dra. Carla Ilane Moreira Bezerra

QUIXADÁ

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S579i Silva, Carla Lizandra da.
Investigando o uso de práticas modernas de testes de software na indústria e na academia / Carla Lizandra da Silva. – 2025.
95 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Sistemas de Informação, Quixadá, 2025.
Orientação: Profa. Dra. Carla Ilane Moreira Bezerra.

1. Teste de software. 2. Ensino de testes. 3. Práticas modernas de testes. 4. Indústria de software. I. Título.

CDD 005

CARLA LIZANDRA DA SILVA

INVESTIGANDO O USO DE PRÁTICAS MODERNAS DE TESTES DE SOFTWARE NA INDÚSTRIA E NA ACADEMIA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas de Informação do Campus de Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas de Informação.

Aprovada em: 20 de Fevereiro de 2025

BANCA EXAMINADORA

Profa. Dra. Carla Ilane Moreira Bezerra (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Ivan do Carmo Machado
Universidade Federal da Bahia (UFBA)

Prof. Dr. Jeferson Kenedy Morais Vieira
Universidade Federal do Ceará (UFC)

À Deus, por sua infinita bondade e misericórdia, sem a quais eu não teria conseguido chegar até aqui. E à minha família, por todo apoio e amor que dedicaram a mim, por nunca duvidarem e sempre estarem comigo no momentos bons e nos ruins.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus, sem o qual eu não teria a capacidade nem a força para chegar até aqui. Toda honra e glória sejam dadas a Ele, que está no controle de tudo e me permitiu alcançar este momento.

À minha família, especialmente à minha mãe, Alderisa da Silva, e ao meu pai, Antonio Expedito da Silva, que sempre estiveram ao meu lado, torcendo e me incentivando, nunca me deixando desistir. Este trabalho é para vocês, que, de inúmeras formas, me proporcionaram as oportunidades necessárias para chegar até aqui e realizar esta conquista. Agradeço por cada oração para que eu pudesse alcançar esse objetivo.

Ao meu namorado, Weslley Itallo Vieira da Silva, que chegou em minha vida durante a graduação e, desde então, tem me apoiado em tudo o que precisei. Você foi essencial para que eu não desistisse e conseguisse concluir tanto a graduação quanto este trabalho. Obrigada por cada palavra de incentivo e pela paciência de sempre me explicar, mais uma vez, quando eu pedia.

À minha orientadora, Carla Ilane Moreira Bezerra, que desde o início se mostrou entusiasta e prestativa, tanto comigo quanto com este trabalho. Agradeço por todos os ensinamentos e dicas, por acreditar em mim e no meu potencial.

Aos meus amigos e colegas da graduação, que sempre estiveram ao meu lado e me ajudaram, cada um teve um papel importante na minha formação e na jornada até aqui. Em especial, gostaria de agradecer à minha amiga e colega Kasiliana Oliveira, que um dia me disse que eu tinha "jeito para QA". Mesmo sem perceber, você abriu meus olhos para uma área incrível da qual hoje não me vejo distante. Obrigada pelo incentivo.

Agradeço também aos meus colegas e supervisores do Núcleo de Práticas de Informática - NPI, no qual tive a oportunidade de aprimorar meus conhecimentos e habilidades. Em especial, agradeço ao meu supervisor, José Fernandes Almeida Junior, que acreditou e confiou em mim ao me conceder o meu primeiro cargo na área de Testes no NPI. Foi uma honra poder fazer parte e contribuir com meus conhecimentos.

Por fim, agradeço a todos que, de alguma forma, contribuíram para que eu chegasse até aqui. A cada palavra, a cada gesto de apoio, a cada oração e abraço, muito obrigado, e que Deus abençoe a todos.

“Assim, quer vocês comam, quer bebam, quer façam qualquer outra coisa, façam tudo para a glória de Deus. - 1 Coríntios 10.31”

(Bíblia Sagrada)

RESUMO

O teste de software desempenha um papel fundamental na garantia da qualidade dos sistemas, assegurando que o código funcione conforme especificado e prevenindo falhas que possam comprometer a experiência do usuário ou causar prejuízos às organizações. No entanto, apesar de sua relevância, a educação em teste de software ainda enfrenta desafios, especialmente na formação acadêmica. Este trabalho tem como objetivo investigar o uso de práticas modernas de testes de software tanto na indústria quanto na academia, identificando lacunas e desafios enfrentados pelos estudantes de graduação e profissionais da área. Para alcançar esse objetivo, foi realizada uma pesquisa em três etapas. Inicialmente, analisaram-se as ementas de disciplinas de teste de software em cursos de Engenharia de Software de instituições brasileiras, verificando a carga horária dedicada ao tema e os tópicos abordados. Em seguida, aplicou-se um *survey* com estudantes para identificar suas percepções sobre o ensino de testes, incluindo metodologias utilizadas, ferramentas apresentadas e nível de preparação para o mercado. Por fim, um segundo *survey* foi conduzido com profissionais da indústria para mapear as práticas mais adotadas no mercado, como a automação de testes, a aplicação de metodologias ágeis e o uso de *frameworks* específicos. Os resultados apontaram que, embora a automação de testes e as metodologias ágeis sejam amplamente utilizadas no mercado, o ensino acadêmico ainda prioriza abordagens mais tradicionais, com pouca ênfase na aplicação prática dessas técnicas. A comparação entre os dados coletados revelou que o que é ensinado na academia e as habilidades exigidas pelo mercado não estão bem alinhadas, dificultando a transição dos estudantes para a atuação profissional.

Palavras-chave: teste de software; ensino de testes; práticas modernas de testes; indústria de software.

ABSTRACT

Software testing plays a fundamental role in ensuring system quality, guaranteeing that the code functions as specified and preventing failures that could compromise user experience or cause losses to organizations. However, despite its relevance, software testing education still faces challenges, especially in academic training. This study aims to investigate the use of modern software testing practices in both industry and academia, identifying gaps and challenges faced by undergraduate students and professionals in the field. To achieve this goal, the research was conducted in three stages. Initially, the syllabi of software testing courses in Software Engineering programs at Brazilian institutions were analyzed, examining the hours dedicated to the subject and the topics covered. Next, a survey was conducted with students to identify their perceptions of software testing education, including the methodologies used, tools presented, and level of preparation for the job market. Finally, a second survey was carried out with industry professionals to map the most widely adopted market practices, such as test automation, the application of agile methodologies, and the use of specific frameworks. The results indicated that, although test automation and agile methodologies are widely used in the market, academic education still prioritizes more traditional approaches, with little emphasis on the practical application of these techniques. The comparison between the collected data revealed that what is taught in academia and the skills required by the industry are not well aligned, making the transition from students to professional practice more difficult.

Keywords: software testing; testing education; modern testing practices; software industry.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

1 INTRODUÇÃO

O teste de software consiste em um conjunto de procedimentos que asseguram que o código de computador esteja executando de acordo com as suas especificações e, ao mesmo tempo, evite qualquer comportamento indesejado (Myers *et al.*, 2011). Quando os defeitos ou *bugs* não são identificados e removidos, eles podem causar falhas, vulnerabilidades e danos a uma organização e seus usuários (Valle *et al.*, 2015). Dessa forma, o teste de software é o principal meio de verificação e validação usado durante o processo de desenvolvimento de software (Clarke *et al.*, 2017).

Os cursos de graduação em Tecnologia da Informação (TI) vêm abordando o ensino de teste de software através da disciplina de Engenharia de Software, nos tópicos de Verificação, Validação e Teste, porém ainda não existe uma integração com outras disciplinas (Valle *et al.*, 2015). Alguns dos tipos mais comuns de testes abordados nos cursos em computação são: teste de unidade, teste de integração, teste de sistema e teste de aceitação (Melo *et al.*, 2020).

O ensino de teste atualmente utiliza algumas abordagens de ensino, por exemplo: aprendizagem baseada em projetos (McManus; Costello, 2019), aprendizagem baseada em problemas (Cheiran *et al.*, 2017), revisão por pares (Smith *et al.*, 2012), entre outras. Das abordagens citadas, as que são mais frequentemente usadas são a aprendizagem baseada em projetos e a aprendizagem baseada em problemas, onde são utilizados alguns recursos na implantação, como slides, livros, tutoriais e questionários (Melo *et al.*, 2020).

Outras diversas metodologias e abordagens de ensino já foram propostas. A abordagem baseada em *gamificação* é um exemplo bastante conhecido, nela são utilizadas ferramentas como *CleanGame* (Santos *et al.*, 2019) e *GATE* (Chen, 2011) que, através de mecânicas, auxiliam na motivação, foco e engajamento. Um exemplo de mecânica utilizada por essas ferramentas é a *Score*, onde os alunos ganham pontos após realizarem tarefas específicas; outro exemplo seria a mecânica de *puzzles*, nesta mecânica são propostos desafios com regras simples que exigem ações específicas dos usuários (Fulcini *et al.*, 2023).

Outro tipo de abordagem que encontramos ao analisar a literatura é a da perspectiva integrada de conteúdo educacional de teste de software (Valle *et al.*, 2015), que baseia-se em integrar o ensino de teste de software juntamente com o ensino de programação, tendo em vista um melhor desenvolvimento das habilidades dos alunos. Essa é uma abordagem que tem o intuito de ajudar os alunos a melhorarem sua compreensão e análise em programação e ainda criar uma cultura de teste, onde os testes são inseridos desde o começo de seu aprendizado (Corte

et al., 2006).

A indústria de software tem se utilizado de práticas modernas da Engenharia de Software (Valente, 2020). Um exemplo de prática moderna no contexto de testes de software são os testes automatizados. A automação de testes de software possui um ciclo de vida próprio quando adotada em um programa de teste, que envolve estratégia e planejamento de metas, definição de requisitos de teste, análise, *design* e desenvolvimento (Dustin *et al.*, 1999). Um dos motivos para a crescente adoção dos testes automatizados na indústria se dá pela enorme vantagem sobre os testes manuais; enquanto os testes manuais são demorados e exigentes, tornando-os difíceis de executar, os testes automatizados são eficazes em tempo, custo e usabilidade (Gamido; Gamido, 2019).

Na literatura é possível identificar uma deficiência no aprendizado de testes dos alunos de computação, os mesmos não saem da universidade adequadamente preparados para testar, isso acaba muitas vezes tornando-se um desafio ou mesmo desvantagem ao ingressarem no seu primeiro emprego, já que muito provavelmente eles terão que realizar pelo menos algum tipo de teste (Carver; Kraft, 2011). São poucos os trabalhos da literatura que abordam o ensino de práticas modernas de software, e assim como qualquer outra área da computação, para que se possa obter um software de qualidade é necessário que os alunos sejam apresentados às novas tecnologias e ferramentas disponíveis no mercado (Arcuri, 2018; Fulcini *et al.*, 2023). Sabendo disso, é necessário que os cursos de computação forneçam um melhor preparo aos alunos, utilizando-se dessas abordagens, ferramentas e práticas modernas, pois a maioria das universidades ainda não dispõe de muitas disciplinas que incluam o assunto de teste de software profundamente (Lemos *et al.*, 2018).

Dessa forma, este trabalho tem como objetivo investigar o uso das práticas modernas de testes de software mais frequentes na indústria e aquelas abordadas na academia, a fim de identificar lacunas e desafios enfrentados por alunos de graduação em TI no aprendizado de testes de software. Para isso, propomos inicialmente a análise das ementas das disciplinas de teste de software ofertadas nos cursos de graduação em TI para identificar o que está sendo abordado nas disciplinas, em seguida a aplicação de um *survey* para identificar o nível de conhecimento e as percepções sobre o ensino de teste de software que os alunos conseguiram adquirir no decorrer da disciplina, com foco nas abordagens modernas apresentadas. Em seguida, um segundo *survey* será conduzido para mapear as práticas de testes mais adotadas pela indústria. Por fim, será realizada uma análise comparativa entre os dois *surveys*, visando identificar as possíveis

discrepâncias entre o que é ensinado na graduação e as competências exigidas pelo mercado.

1.1 Objetivos

Nesta Seção, é apresentado o objetivo geral e os objetivos específicos do presente trabalho.

1.1.1 Objetivo Geral

Este trabalho tem como objetivo investigar o uso de práticas modernas de testes de software, analisando sua aplicação tanto na indústria quanto na academia, para identificar lacunas, desafios e oportunidades de integração entre os dois contextos.

1.1.2 Objetivos Específicos

- Investigar as práticas de teste de software abordadas no ensino acadêmico.
- Analisar as práticas modernas de teste de software utilizadas na indústria, destacando ferramentas, técnicas e metodologias aplicadas no mercado.
- Comparar o ensino acadêmico com as práticas adotadas na indústria, identificando lacunas e oportunidades de alinhamento.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta Seção, serão apresentados os conceitos mais relevantes necessários para a compreensão deste trabalho. Na Seção 2.1, serão apresentadas algumas definições básicas sobre Teste de Software. Na Seção 2.2, serão apresentadas algumas práticas modernas de teste de software. A Seção 2.3 irá abordar as principais metodologias de ensino para teste de software presentes nos cursos de computação.

2.1 Testes de Software

Na literatura podemos encontrar diversas visões do que é teste de software e qual a sua finalidade. Segundo Sommerville (2011), a finalidade do teste de software é demonstrar que um programa de fato faz o que foi destinado a fazer, além de identificar possíveis falhas antes que ele seja colocado em uso. Myers *et al.* (2011) nos traz uma visão muito parecida ao afirmar que o teste de software vai consistir basicamente em um conjunto de procedimentos que asseguram que o código de computador esteja executando de acordo com as suas especificações e ainda que evite qualquer comportamento indesejado. Para Clarke *et al.* (2017), os testes desempenham um papel fundamental durante o processo de desenvolvimento de software, pois são o principal método de verificação e validação utilizado.

Com o objetivo de aprimorar ainda mais a qualidade do software, foram desenvolvidas técnicas de teste. Essas técnicas são empregadas para ampliar a cobertura dos testes, resultando em um aumento significativo na qualidade, confiabilidade e usabilidade. No Quadro 1 são descritas algumas dessas técnicas.

Quadro 1 – Exemplos de técnicas de teste de software

Técnicas de Teste Caixa-Preta	**Técnicas de Teste Caixa-Branca**
Partição de Equivalência	Teste de Fluxo de Controle
Análise de Valores Limite	Teste de Caminho
Tabela de Decisão	Teste de Fluxo de Dados

Fonte: Elaborado pela autora.

– **Teste Caixa-Preta.** Nesta técnica, o teste é realizado de forma a cobrir todas as funcionalidades da aplicação, porém sem entrar em detalhes de implementação (Verma *et al.*, 2017). Não é necessário que o testador conheça o código da aplicação, pois o foco se mantém

nas entradas disponíveis para a aplicação e nas suas saídas esperadas para cada valor de entrada (Jamil *et al.*, 2016). Nesta técnica, os testes são realizados do ponto de vista do cliente.

Como exemplo para a técnica de caixa-preta, conforme apresentado no Quadro 1 temos: partição de equivalência, a análise de valor limite e a tabela de decisão. A partição de equivalência é uma técnica que tem como objetivo reduzir a quantidade de casos de teste necessários. Ela funciona dividindo o conjunto de possíveis entradas do programa em classes de equivalência, com base nos valores de entrada. A partir dessas classes de equivalência, são gerados os casos de teste que representam diferentes cenários do domínio de entrada (Irawan *et al.*, 2018).

A análise de valor limite é uma técnica usada para identificar erros nos limites de entrada ou saída. Ela envolve selecionar casos de teste nos limites dos valores válidos. Essa técnica ajuda a encontrar problemas relacionados a limites, testar os limites ajuda a garantir o correto funcionamento do sistema (Myers *et al.*, 2011).

A tabela de decisão é usada para representar de forma organizada as combinações de entradas e suas respectivas saídas ou ações, ajudando a identificar e testar o comportamento do sistema com base em diferentes condições ou regras. A tabela estrutura as informações em colunas, representando as entradas, condições ou regras, e em linhas, representando as saídas ou ações. Essa organização facilita a compreensão das diversas combinações possíveis (Graham *et al.*, 2021).

– **Teste Caixa-Branca.** Já nesta técnica, o foco principal é a estrutura de uma aplicação, o teste é aplicado diretamente no código fonte do software (Verma *et al.*, 2017). Nessa técnica, os casos de teste garantem que todas as declarações feitas no programa foram executadas pelo menos uma vez durante o teste, garantindo assim o exercício de todas as condições lógicas estabelecidas (Pressman, 2005).

Como exemplo para a técnica de caixa-branca, conforme apresentado no Quadro 1, temos: O Teste de Fluxo de Controle que se concentra na execução de todos os caminhos de controle de um programa. Seu objetivo é garantir a correta execução e teste de declarações, decisões e estruturas de controle. Essa técnica envolve a seleção de caminhos de execução adequados para obter uma cobertura abrangente e identificar erros lógicos, decisões incorretas ou fluxos inadequados (Myers *et al.*, 2011).

O teste de caminho, segundo Sommerville (2007), é uma técnica que tem como objetivo

percorrer os caminhos de execução distintos em um componente ou programa. Quando todos os caminhos são percorridos, garante-se que todas as declarações no componente tenham sido executadas pelo menos uma vez.

A técnica de teste de fluxo de dados oferece ao testador acesso aos detalhes da estrutura do programa e destaca as variáveis utilizadas dentro dele. Essa abordagem de teste estrutural observa as variáveis definidas e sua utilização ao longo do programa. O testador registra as mudanças nos valores dessas variáveis por meio do conceito de grafo do programa, proporcionando uma compreensão mais abrangente da lógica e do fluxo dos dados no sistema (Sheoran *et al.*, 2020).

No Quadro 2 é feita uma breve comparação entre as técnicas de teste caixa-preta e caixa-branca:

Quadro 2 – Comparação das técnicas de testes caixa-preta e caixa-branca

Caixa-Preta	**Caixa-Branca**
Também é chamado de Técnica Baseada em Especificação.	Também é chamado de Técnica de Teste Estrutural
Conhecimento da estrutura interna e de programação não é necessário.	Conhecimento da estrutura interna e de programação é necessário
Principal foco na funcionalidade do sistema.	Principal foco na estrutura do código, *branches*, *loops*, condições etc
Conhecimento de implementação não é necessário.	É necessário ter conhecimento de implementação

Fonte: Adaptado de Verma *et al.* (2017).

2.2 Práticas Modernas de Teste de Software

As práticas de teste de software das quais iremos falar referem-se a abordagens e técnicas que vêm sendo utilizadas no processo de garantia de qualidade de software. Muitas pesquisas mostram o notável aumento na automação de testes na indústria, assim como também a aplicação dos testes seguindo as metodologias ágeis (Hynninen *et al.*, 2018). Além disso, esta seção abordará também os *Test Smells* e os *Flaky Tests*, práticas que vêm sendo cada vez mais estudadas devido ao seu impacto na confiabilidade e manutenção dos testes de software na indústria (Martins *et al.*, 2023), (Habchi *et al.*, 2022).

2.2.1 Testes Automatizados

O objetivo do teste é verificar se o software funciona corretamente, simulando o usuário final, esses testes podem ser manuais ou automatizados. O teste manual é demorado e

pode não detectar todos os *bugs* de forma eficaz. Já os testes automatizados permitem abranger mais áreas do sistema e executar testes em menos tempo, usando cenários de teste que podem ser refeitos sempre que necessário, com isso o teste automatizado torna-se mais vantajoso em termos de tempo, custo e usabilidade (Gamido; Gamido, 2019).

O Quadro 3 mostra as vantagens e desvantagens da utilização dos testes automatizados:

Quadro 3 – Vantagens e desvantagens da automação de testes

Vantagens	**Desvantagens**
Melhora a precisão e facilita a identificação rápida de *bugs* em comparação com o teste manual	Escolher a ferramenta certa requer um esforço considerável, tempo e um plano de evolução
Economiza tempo e esforço ao tornar o teste mais eficiente	Requer conhecimento da ferramenta de teste
Aumenta a cobertura de testes porque várias ferramentas de teste podem ser usadas simultaneamente, permitindo testar em paralelo diferentes cenários de teste	O custo de compra da ferramenta de teste e, no caso de métodos de reprodução, a manutenção dos testes é um pouco cara
O *script* de teste automatizado é repetível	É necessário ter proficiência para escrever os *scripts* de teste automatizados

Fonte: Adaptado de Umar e Zhanfang (2019).

Existem dois tipos de automação de testes, os testes baseados em código, os quais utilizam interfaces, bibliotecas, classes e módulos existentes para testar diferentes entradas e validar os resultados, e os testes baseados na interface gráfica do usuário (GUI) que geram eventos de interface como cliques e pressionamento de teclas. O *framework* de teste utiliza esses eventos para verificar o desempenho do programa (Sneha; Malle, 2017).

Em seu trabalho, Muneer (2014) encontrou 22 tipos de testes automatizados relatados na literatura, os quais são mostrados no Quadro 4.

Quadro 4 – Tipos de testes automatizados

1	***Code-Coverage Testing***	**12**	***Acceptance Testing***
2	***Module Testing***	**13**	***Model-based Testing***
3	***Functional Testing***	**14**	***Specification-based Testing***
4	***Security Testing***	**15**	***Concurrency Testing***
5	***GUI Testing***	**16**	***Fault-based Testing***
6	***Regression Testing***	**17**	***Interoperability Testing***
7	***Random Testing***	**18**	***Load Testing***
8	***Usability Testing***	**19**	***Performance Testing***
9	***Unit Testing***	**20**	***Stress Testing***
10	***Integration Testing***	**21**	***Memory Leakage Testing***
11	***System Testing***	**22**	***Environment Testing***

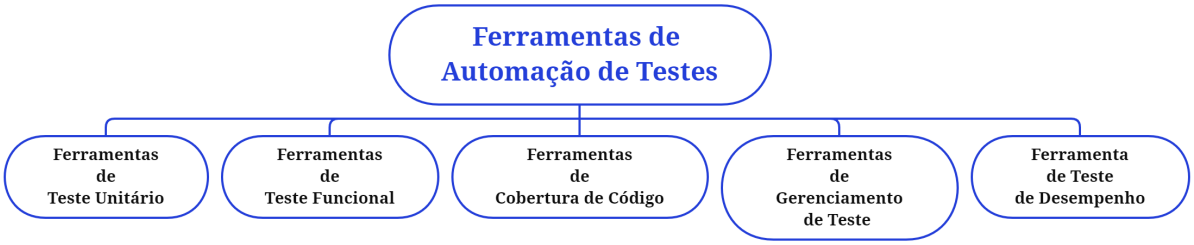
Fonte: Adaptado de Muneer (2014).

As ferramentas e *frameworks* de testes automatizados desempenham um papel fundamental no processo de garantia de qualidade de software. Essas soluções permitem que os

testes sejam executados de forma eficiente, repetível e confiável, economizando tempo e esforço significativos em comparação com os testes manuais.

Existem diversas opções de ferramentas disponíveis no mercado para auxiliar no desenvolvimento e execução de testes automatizados. Essas ferramentas de automação de testes de software podem ser divididas em diferentes categorias como mostrado na Figura 1 (Umar; Zhanfang, 2019).

Figura 1 – Categorias de ferramentas de automação de testes



Fonte: Adaptado de Umar e Zhanfang (2019).

Ferramentas de Teste Unitário. Os testes unitários são responsáveis por testar as partes mais fundamentais do código. Utilizando ferramentas específicas, eles garantem o correto funcionamento de uma unidade específica, além de verificar a estrutura do código. Alguns dos *frameworks* de teste unitário são: *JUnit*, *NUnit*, *JMockit*, *PHPUnit* e outros (Umar; Zhanfang, 2019).

Ferramentas de Teste Funcional. O teste funcional verifica se o software atende aos requisitos dos usuários. Os casos de teste são baseados nas especificações do componente. As ferramentas executam funções, comparando a saída obtida com o resultado esperado. Algumas das ferramentas de teste funcional são: *Cypress*, *Selenium*, *HP QuickTest Professional*, *TestComplete*, *Ranorex* e *Watir* (Umar; Zhanfang, 2019).

Ferramentas de Cobertura de Código. As ferramentas de cobertura de código são utilizadas para identificar as áreas do código de software abrangidas por testes automatizados. Elas quantificam a quantidade de linhas, declarações ou blocos de código testados por meio de conjuntos de testes automatizados, e revelam as partes do código que não são cobertas por testes automatizados, o que pode os tornar suscetíveis a defeitos. Algumas das ferramentas de cobertura de código são: Cobertura, *CodeCover*, EMMA, *PITest*, *Atlassian Clover*, e outras (Umar; Zhanfang, 2019).

Ferramentas de Gerenciamento de Teste. Ferramentas de gerenciamento de

testes automatizam as tarefas relacionadas aos testes, como criação de casos de teste, planos e estratégias de teste. Elas proporcionam uma forma organizada e de fácil acesso para armazenar e manter as atividades de teste. Alguns exemplos de ferramentas de gerenciamento de testes são: *Test Manager*, *Test Link*, *TETware*, *Test Environment Toolkit (TET)*, *QA Complete*, entre outras (Umar; Zhanfang, 2019).

Ferramentas de Teste de Desempenho. O teste de desempenho é conduzido para avaliar e determinar o comportamento do software em relação à capacidade de resposta e estabilidade sob diferentes condições e cargas de trabalho. É possível realizar o teste de desempenho utilizando diversas ferramentas como: *JMeter, Rational Performance Tester*, *HP LoadRunner*, *Silk Performer*, entre outras opções disponíveis (Umar; Zhanfang, 2019).

2.2.2 Test Smells

Os *test smells* podem ser definidos de maneira geral, como um agrupamento de problemas que são encontrados no código de teste, ou simplesmente um teste mau projetado (Hedayati *et al.*, 2015). Em sua pesquisa, Deursen *et al.* (2001) conseguiram identificar onze tipos de *test smells* em testes estáticos, que serão apresentados a seguir.

- ***Mystery Guest*.** Um teste ao usar recursos externos, como um arquivo que contém dados de teste, deixa de ser autocontido, o que pode dificultar o entendimento da funcionalidade testada. O uso de recursos externos pode criar dependências ocultas e aumentar as chances de falhas nos testes caso esses recursos sejam modificados ou removidos (Deursen *et al.*, 2001; Spadini *et al.*, 2020).
- ***Resource Optimism*.** Quando o código de teste faz suposições otimistas sobre a existência (ou ausência) e sobre o estado de recursos externos, isso pode causar um comportamento não determinístico nos resultados do teste (Deursen *et al.*, 2001; Palomba; Zaidman, 2017).
- ***Test Run War*.** Ocorrem quando os testes funcionam corretamente apenas quando você é o único a executá-los, mas falham quando outros programadores os executam. Provavelmente é causado por interferência de recursos, como quando alguns testes alocam recursos, como arquivos temporários, que também são usados por outros (Deursen *et al.*, 2001; Palomba; Zaidman, 2017).
- ***General Fixture*.** No *JUnit* é possível definir um método *setUp* que é executado antes de cada teste, criando um ambiente para a execução dos testes. No entanto, é importante evitar a criação de um ambiente de *setUp* muito genérico, onde diferentes testes acessam

apenas parte dele. Esses ambientes tornam a leitura e compreensão dos testes mais difíceis. Além disso, eles podem tornar os testes mais lentos, devido à execução de tarefas desnecessárias (Deursen *et al.*, 2001; Palomba *et al.*, 2018).

- ***Eager Test.*** A partir do momento que o método de teste abrange a verificação de vários métodos do objeto testado, isso acaba por prejudicar a legibilidade e compreensão, dificultando o uso como documentação. Isso aumenta a dependência dos testes entre si e os tornam mais difíceis de manter (Deursen *et al.*, 2001; Spadini *et al.*, 2020).
- ***Lazy Test.*** Essa situação ocorre quando diversos métodos de teste verificam a mesma função utilizando uma configuração de teste semelhante (por exemplo, verificam os valores de diferentes variáveis de instância). Esses testes geralmente só possuem sentido quando considerados em conjunto, tornando-se mais úteis ao serem agrupados utilizando o método *Inline* (Deursen *et al.*, 2001; Virgínio *et al.*, 2020).
- ***Assertion Roulette.*** É um tipo de problema que surge quando um método de teste contém vários *assertions* sem explicação. Quando algum dos *assertions* falha, não é possível identificar qual causou o problema (Deursen *et al.*, 2001; Santana *et al.*, 2020).
- ***Indirect Testing.*** Uma classe de teste é criada com o propósito de testar uma classe correspondente no código de produção. No entanto, é importante evitar que a classe de teste tenha métodos que realizam testes em outros objetos, especialmente quando existem referências a esses objetos na classe que está sendo testada.O fato de surgir esse problema também indica que pode haver problemas com o encapsulamento de dados no código de produção (Deursen *et al.*, 2001; Spadini *et al.*, 2018).
- ***For Testers Only.*** Quando uma classe de produção possui métodos que são utilizados exclusivamente por métodos de testes, esses métodos podem ser desnecessários e podem ser removidos, ou podem ser necessários apenas para configurar uma estrutura de teste.Dependendo da função desses métodos, pode ser indesejável tê-los no código de produção, onde outros possam utilizá-los. (Deursen *et al.*, 2001; Junior *et al.*, 2021).
- ***Sensitive Equality.*** É rápido e fácil escrever verificações de igualdade usando o método *toString*, normalmente é calculado o resultado real, depois é convertido para uma *string* e é comparado com uma *string* literal que representa o valor esperado. Esses testes podem depender de detalhes irrelevantes como vírgulas, aspas, e espaços. Qualquer alteração feita no método *toString* de um objeto resultará em falta nos testes. (Deursen *et al.*, 2001; Soares *et al.*, 2020).

– ***Test Code Duplication*.** Duplicações indesejáveis podem estar presentes no código de teste, especialmente nas partes responsáveis que configuram as estruturas de teste. (Deursen *et al.*, 2001; Soares *et al.*, 2022).

2.2.3 Flaky Tests

De maneira simplificada, *flaky tests* são testes inconsistentes, já que fornecem resultados não determinísticos (Qin *et al.*, 2021). Ao serem executados, esses testes apresentam resultados de falha e sucesso, apesar de não haver alterações no código fonte dos casos de teste (Parry *et al.*, 2021). O estudo de Fowler, Martin (2011) retrata esses resultados não determinísticos dos *flaky tests* como um grande e recorrente problema nos testes de regressão.

Os *flaky tests*, são um dos fatores significativos que prejudicam a produtividade dos desenvolvedores, devido à sua natureza inconsistente e instável. Esses testes são extremamente difíceis de reproduzir, levando os desenvolvedores a gastarem horas até descobrirem que as falhas ocasionais não estão relacionadas às alterações feitas, mas sim a um *flaky test* (Lam *et al.*, 2019).

Em sua pesquisa, Luo *et al.* (2014) identificou 10 causas de *flaky tests* e deu uma visão aprofundada das três causas mais frequentes. Foram analisados 201 *commits* para classificar as causas raiz dos *flaky tests*, e classificaram precisamente a causa raiz para 161 *commits*. As causas raiz dos *flaky tests* foram divididas em 10 categorias. No quadro abaixo é possível visualizar todas essas 10 categorias.

Quadro 5 – Categorias de causas dos *flaky tests*

1	***Async Wait***	6	***Time***
2	***Concurrency***	7	***IO***
3	***Test Order Dependency***	8	***Randomness***
4	***Resource Leak***	9	***Floating Point Operations***
5	***Network***	10	***Unordered Collections***

Fonte: Adaptado de Luo *et al.* (2014).

As três principais causas de *flaky tests* destacadas por Luo *et al.* (2014) são: *Async Wait*, *Concurrency*, *Test Order Dependency*.

– ***Async Wait*.** Dos 161 *commits* analisados, 74 deles, correspondendo a 45%, são classificados como *Async Wait*. Um *commit* é enquadrado nessa categoria quando o teste realiza uma chamada assíncrona e não espera corretamente pelo resultado antes de utilizá-lo. A seguir é descrito um exemplo de *flaky test* de *Async Wait* retirado da pesquisa de Luo *et al.*

(2014):

Código-fonte 1 – Exemplo de *Flaky Test Async Wait* - Adaptado de Luo *et al.* (2014)

```
1   @Test
2   public void testRsReportsWrongServerName() throws Exception {
3     MiniHBaseCluster cluster = TEST_UTIL.getHBaseCluster();
4     MiniHBaseClusterRegionServer firstServer =(
          MiniHBaseClusterRegionServer)cluster.getRegionServer(0);
5     HServerInfo hsi = firstServer.getServerInfo();
6     firstServer.setHServerInfo(...);
7
8     // Aguarda o sinal de retorno do servidor
9     Thread.sleep(2000);
10    assertTrue(firstServer.isOnline());
11    assertEquals(2,cluster.getLiveRegionServerThreads().size());
12    // O codigo funciona de maneira similar para o segundo servidor
13  }
```

O teste mostrado no Código-fonte 1 usa um *cluster* para iniciar um servidor chamado "*firstServer*" e aguarda sua resposta com o comando *Thread.sleep(2000)*. Se o servidor demorar a responder, o teste falhará. Esse problema pode ser corrigido substituindo o *Thread.sleep(2000)* por uma chamada que espera a resposta do servidor e adicionando um limite de tempo para o teste. Essas alterações eliminaram a inconsistência do teste (Luo *et al.*, 2014).

– ***Concurrency*.** Um total de 32 *commits*, representando 20% do total de 161 *commits*, pertencem à categoria *Concurrency*. Um *commit* é classificado nessa categoria quando o não determinismo do teste é causado pela interação inadequada de diferentes *threads*. Isso pode ocorrer devido a situações como *data races*, violações de atomicidade ou *deadlocks*. O não determinismo pode surgir tanto no *Component Under Test* (CUT) quanto no próprio código de teste. Dos 32 casos identificados, 10 (30%), são resultado de não determinismo no CUT, o que causa falhas intermitentes nos testes correspondentes. A falta de determinismo na execução do teste (ou código) pode ou não indicar um *bug*: o código pode ter vários comportamentos corretos possíveis. Caso o teste aceite apenas um subconjunto desses comportamentos como corretos, ocorrerá um resultado não determinístico e o teste será considerado inconsistente. (Luo *et al.*, 2014).

  O Código-fonte 2 realiza iterações em um mapa compartilhado por várias *threads*. O objetivo é verificar se a configuração (objeto "*conf*") foi alterada e, em caso afirma-

tivo, atualizar um novo objeto de configuração chamado "*newConf*" com as entradas correspondentes que estão ausentes.

Código-fonte 2 – Exemplo de *Flaky Test Concurrency* - Adaptado de Luo *et al.* (2014)

```
1    if (conf != newConf) {
2      for (Map.Entry<String, String> entry : conf) {
3        if ((entry.getKey().matches("hcat.*")) && (newConf.get(entry.
             getKey()) == null)) {
4          newConf.set(entry.getKey(), entry.getValue());
5        }
6      }
7      conf = newConf;
8    }
```

Se várias *threads* executarem esse código simultaneamente e modificarem o objeto "*conf*" ou "*newConf*" ao mesmo tempo, podem ocorrer resultados indeterminados e uma *ConcurrentModificationException* é lançada. Essa falta de determinismo no comportamento do teste pode levar a falhas intermitentes, onde o teste falha ou passa de forma inconsistente, dependendo da ordem de execução das *threads* e das modificações concorrentes feitas nos objetos de configuração. Para solucionar o problema, os desenvolvedores envolveram as linhas de código de 3 a 6 em um bloco sincronizado, garantindo que sejam executadas atomicamente (Luo *et al.*, 2014).

– ***Test Order Dependency*.** Um total de 19 dos 161(12%) dos *commits* pertencem à categoria *Test Order Dependency*. Essa categoria é atribuída quando o resultado de um teste depende da sequência em que os testes são executados. Em teoria, todos os testes em uma suíte de testes devem ser devidamente isolados e independentes um do outro, de modo que a ordem de execução não afete os resultados. No entanto, na prática, isso nem sempre é o caso (Luo *et al.*, 2014).

  Esse cenário ocorre quando os testes dependem de um estado compartilhado que não é configurado ou limpo corretamente. Esse estado compartilhado pode estar na memória principal (como campos estáticos em Java) ou em algum recurso externo (como arquivos ou bancos de dados). Esse problema também pode ocorrer quando um teste espera que o estado esteja conforme foi inicializado, mas, em algum momento, outro teste modifica esse estado ou ainda quando um teste depende que o estado seja configurado pela execução de outro teste, mas esse teste não é executado antes.Essas dependências entre os testes

resultam em um comportamento imprevisível quando a ordem de execução dos testes é alterada.

O trecho exibido em Código-fonte 3 pertence a uma classe de teste em que um teste chamado *testWrite* escreve em um arquivo via *fs*, preparando dados para serem lidos por outros testes. Os desenvolvedores assumiram que o teste *testWrite* sempre seria executado primeiro, porém o *JUnit* não garante uma ordem específica para a execução dos testes, se o *JUnit* executar algum teste de leitura antes do *testWrite*, o teste falhará. Para corrigir esse problema, os desenvolvedores removeram o teste original *testWrite* e adicionaram uma chamada para *testWrite* no método *@BeforeClass*, conforme mostrado na linha 10 (Luo *et al.*, 2014).

Código-fonte 3 – Adaptado de Luo *et al.* (2014)

```
1   @BeforeClass
2   public static void beforeClass() throws Exception {
3       bench = new TestDFSIO();
4       ...
5       cluster = new MiniDFSCluster.Builder(...).build()
6       FileSystem fs = cluster.getFileSystem();
7       bench.createControlFile(fs, ...);
8
9       // Verifica a funcionalidade de escrita nesta etapa, pois ela se
            faz necessaria para outros testes
10      testWrite();
11  }
```

2.3 Metodologias de Ensino Para Testes de Software

Existem diversas metodologias que podem ser aplicadas em diferentes contextos educacionais. Em seu trabalho Paschoal e Souza (2018) obteve como resultado de seu *survey* uma lista das abordagens utilizadas no ensino de teste. Na Tabela 1 é possível ver as abordagens citadas e quantos participantes mencionaram cada uma delas. Apesar dos vários tipos de abordagens citados pelos participantes, a abordagem tradicional continua sendo a mais usada.

Em seu trabalho, Valle *et al.* (2015) analisaram os currículos de referência propostos pela *Association for Computing Machinery* (ACM) e pela Sociedade Brasileira de Computação (SBC) para os cursos de graduação em Computação (Engenharia da Computação, Sistemas de Informação, Engenharia de Software, entre outros). Os currículos de referência têm o objetivo de

Tabela 1 – Abordagens utilizadas no ensino de teste

Abordagens de ensino	**Participantes**
Método de ensino tradicional	43
Aprendizagem baseada em projetos	20
Aprendizagem baseada em problemas	17
Revisão por pares	15
Ensino conjunto de teste com programação	13
Flipped classroom	3
Aprendizagem baseada em jogos	1

Fonte: Paschoal e Souza (2018).

fornecer suporte e diretrizes para a criação, implementação e avaliação de cursos de graduação em Computação.

Esses currículos recomendam a inclusão dos conteúdos de teste de software no curso de Engenharia de Software, sob o nome de Verificação, Validação e Teste de Software (VV&T). No entanto, essas recomendações não destacam explicitamente a importância de abordar de maneira integrada os temas relacionados ao teste de software com outras disciplinas, como Algoritmos, Programação Orientada a Objetos, Estruturas de Dados, entre outros, o que resulta na ausência de uma cultura consolidada de teste, impedindo os estudantes de adquirirem o hábito de testar seus programas de forma rigorosa desde o início do aprendizado da programação (Valle *et al.*, 2015).

Foram analisados os currículos das principais universidades brasileiras nos cursos de Ciência da Computação, Engenharia da Computação, Sistemas de Informação e Engenharia de Software, selecionadas a partir do Ranking Universitário Folha (RUF) de 2014. De maneira geral, o ensino de teste de software é abordado nas disciplinas de Engenharia de Software, especificamente no tópico de Verificação, Validação e Teste (VV&T), sem uma integração com outras disciplinas. No entanto, esses cursos dedicam menos de dez horas para o ensino de teste de software, o que não permite uma abordagem adequada, não há tempo suficiente para abordar adequadamente as práticas de teste de software (Valle *et al.*, 2015).

Valle *et al.* (2015) também analisaram os currículos no cenário internacional. Nos currículos das universidades analisadas, os temas relacionados ao teste de software são contemplados nas disciplinas de Engenharia de Software e/ou Programação, conforme preconizado pela ACM. No entanto, esses conteúdos são abordados de maneira isolada, sem uma conexão com outras disciplinas.

Observa-se que nos cursos de Ciência da Computação, tanto no Brasil quanto no

exterior, o teste de software é tratado apenas nas disciplinas de Engenharia de Software e/ou Fundamentos de Programação. A quantidade de aulas dedicadas ao teste de software é reduzida, dando aos alunos apenas uma visão superficial. Além disso, a educação em teste de software ocorre apenas no final dos cursos de graduação. Com isso, os alunos, de maneira geral, não recebem uma formação sólida para ingressarem no mercado de trabalho (Valle *et al.*, 2015).

3 TRABALHOS RELACIONADOS

Foram identificados na literatura alguns estudos que se relacionam com a proposta deste projeto de pesquisa. Nesta seção, são descritas as contribuições desses estudos e como eles se relacionam com o presente trabalho.

3.1 *A Survey on Software Testing Education in Brazil*

Em seu trabalho, Paschoal e Souza (2018) realizaram uma pesquisa em diversos cursos de ciência da computação oferecidos em diferentes universidades brasileiras para identificar os tópicos de teste de software que são ensinados e como são abordados. O principal objetivo era identificar os métodos utilizados para apresentar o conteúdo, os recursos de apoio empregados nas práticas de ensino, os desafios enfrentados e as estratégias de avaliação utilizadas para medir a aprendizagem dos estudantes.

O estudo foi conduzido por meio de uma pesquisa realizada em 2017 que envolveu a coleta de dados com professores de instituições de ensino superior brasileiras, como universidades, faculdades e escolas técnicas. Os dados foram coletados por meio de questionários e entrevistas com professores e coordenadores de cursos relacionados à área de testes de software.

Os resultados revelaram que os professores fazem esforços para ensinar os conceitos fundamentais e técnicas de teste de software, independentemente de ser em uma disciplina específica ou não. No entanto, quando o assunto é abordado em disciplinas de engenharia de software, nem todos os critérios de teste são abordados de forma abrangente. Essa lacuna também é observada em disciplinas específicas, onde nem sempre todos os critérios mais conhecidos são ensinados pelos professores.

A falta de uniformidade na carga horária das disciplinas pode ser uma das causas dessa falta de abrangência, já que diferentes disciplinas dedicam quantidades variadas de tempo ao ensino do assunto. Não surpreendentemente, os critérios de técnicas mais dispendiosas, como teste baseado em defeitos e teste baseado em modelos, são ensinados por mais da metade dos participantes quando o conteúdo é abordado em uma disciplina específica.

Através do *survey* realizado, foi possível identificar as dificuldades reais enfrentadas pelos professores ao ensinar o tema de teste de software. Uma das principais constatações foi que as ferramentas disponíveis para apoiar a condução das atividades não são de fácil utilização, além de algumas delas estarem desatualizadas.

Outro ponto relevante é a dificuldade que os professores encontram em motivar os alunos e fazer com que eles compreendam a importância do teste de software para a qualidade do software. Essa dificuldade pode estar relacionada à estratégia de ensino adotada, uma vez que a maioria dos professores ainda utiliza métodos tradicionais.

Por fim, tendo em vista os resultados do *survey*, os autores finalizam falando sobre novas perspectivas para a pesquisa em educação de teste de software, são elas: investir na experimentação de novas abordagens de ensino; estabelecer mecanismos de apoio que consigam contemplar atividades que simulam problemas reais; definir estratégias para o conteúdo de teste ser abordado em disciplinas específicas de teste em cursos de computação.

3.2 *Testing Education: A Survey on a Global Scale*

No trabalho de Melo *et al.* (2020) é realizado um estudo com o objetivo de obter uma visão abrangente do ensino de teste de software sob a perspectiva dos educadores. O estudo investiga como os educadores abordam o tema de teste de software, os materiais e detalhes que são abordados, os mecanismos de suporte utilizados, as práticas de ensino adotadas, bem como os desafios enfrentados e os instrumentos de avaliação utilizados.

Realizado entre abril e junho de 2020, o estudo consistiu em uma pesquisa conduzida com professores de teste de software, com o objetivo de investigar a abordagem adotada no ensino desse tema em sala de aula. Participantes de diferentes regiões globais foram incluídos no estudo, abrangendo quatro continentes, com exceção da África devido à falta de resposta. A maior representatividade de respondentes foi observada no continente europeu (49%), seguido pelo continente americano (32%), Ásia (15%) e Oceania (3%).

Os resultados da pesquisa revelaram a diversidade no ensino de teste de software em todo o mundo. Os tópicos e profundidade variam de acordo com a classificação das disciplinas. As principais técnicas ensinadas são o teste funcional e o teste baseado em falhas, com critérios mais específicos sendo abordados em disciplinas específicas. As abordagens de ensino mais comuns são baseadas em projetos e problemas, utilizando recursos educacionais como slides, capítulos de livros e vídeos.

Além disso, a pesquisa mostrou que os métodos de avaliação são feitos principalmente por meio de trabalhos práticos e testes. As principais dificuldades relatadas incluem falta de tempo, alinhamento com a indústria, falta de ferramentas de suporte, falta de habilidades dos alunos e motivação.

Em conclusão, este estudo forneceu uma visão abrangente do ensino de testes de software com base nas respostas de educadores de várias universidades ao redor do mundo. As abordagens tradicionais de ensino ainda predominam, mas há uma presença crescente de métodos baseados em projetos. Foram identificados desafios comuns, como a falta de tempo para cobrir todo o conteúdo e a dificuldade em manter os alunos motivados.

3.3 *From Blackboard to the Office: A Look Into How Practitioners Perceive Software Testing Education*

No estudo realizado por Martins *et al.* (2021), o objetivo principal foi fornecer suporte à comunidade de teste de software e realizar uma análise sobre a percepção dos profissionais brasileiros em relação à educação em testes de software.

O estudo foi realizado por meio de uma pesquisa com especialistas, envolvendo 68 profissionais de teste de software do Brasil, com o objetivo de investigar a aprendizagem de teste de software e determinar se existe uma diferença clara entre (i) o que a indústria espera dos recém-formados e (ii) o que eles aprendem durante a graduação.

O estudo apresenta 3 principais descobertas, são elas: (i) a taxa de aprendizagem dos testadores de software novatos é semelhante à taxa de aprendizagem dos profissionais experientes; (ii) os testadores de software estudaram na academia menos da metade dos conteúdos que consideram importantes na indústria; e (iii) os testadores de software novatos geralmente fazem cursos extracurriculares apoiados pela indústria para complementar sua aprendizagem.

A pesquisa mostrou que, apenas 55,9% dos entrevistados estudaram teste de software na academia. As maiores dificuldades dos entrevistados estavam relacionadas à aplicação dos conceitos na prática e à busca por fontes de aprendizado que os apoiassem durante o processo de aprendizagem. Foi relatado também pelos participantes que a academia ensina poucos tópicos aplicáveis à carreira de teste de software. Dos 35 tópicos de teste de software que os respondentes gostariam de ter aprendido antes de se formarem, os resultados mostraram que menos da metade deles são abordados na academia (45,7%).

Em conclusão, os resultados revelam que a falta de conhecimento dos testadores de software iniciantes não parece estar relacionada à instituição de ensino, ao currículo de cursos específicos de computação ou à duração do curso. Independentemente desses fatores, a taxa de aprendizado é semelhante em todos os cenários. Surpreendentemente, os entrevistados relataram ter estudado apenas 45,7% dos conteúdos de teste de software abordados na pesquisa. Isso

sugere que os novos testadores de software frequentemente dependem de cursos extracurriculares apoiados pela indústria para complementar sua formação.

3.4 *On the Current State of Academic Software Testing Education in Sweden*

O estudo de Barrett *et al.* (2023) analisa o estado atual da educação em testes de software nas universidades suecas, destacando sua importância na formação de profissionais de tecnologia da informação. A pesquisa foi motivada pela necessidade de garantir a qualidade do software, dado que as atividades de verificação e validação consomem uma parte significativa dos recursos em projetos de desenvolvimento. Apesar da relevância do tema, o ensino de testes de software tem sido frequentemente negligenciado nos currículos de Ciência da Computação.

A pesquisa envolveu uma busca sistemática em 25 universidades suecas, das quais 14 oferecem cursos dedicados a testes de software. Os resultados mostraram que apenas 32% dos cursos são oferecidos em nível de graduação, enquanto 28% das universidades disponibilizam treinamento especializado. Em média, os cursos de testes de software representam cerca de 5% dos créditos totais exigidos para a graduação.

O estudo ainda aborda a escassez de testadores de software bem treinados, apontando que muitos graduados não possuem conhecimento adequado sobre testes. A análise revela que a maioria dos cursos se concentra em tópicos básicos, com pouca ênfase em áreas especializadas. Além disso, a pesquisa identificou que a educação em testes de software é predominantemente oferecida em formato presencial, limitando a flexibilidade para os alunos.

A metodologia utilizada incluiu a coleta de informações de sites oficiais de universidades e a análise de currículos. Os cursos foram categorizados em especializados e não especializados, com uma ênfase em tópicos como níveis de teste, tipos de teste, automação e gerenciamento de testes. A avaliação dos alunos varia entre trabalhos, projetos e exames, mas não existe uma abordagem uniforme.

A pesquisa conclui que, apesar de algumas universidades apresentarem um desempenho melhor que outras, a educação em testes de software na Suécia é limitada. Recomenda-se uma maior integração do ensino de testes de software nos currículos, começando desde os primeiros anos da formação em Ciência da Computação. Além disso, sugere-se que futuras pesquisas explorem mais a fundo as práticas de ensino e a colaboração entre universidades e a indústria para melhorar a formação em testes de software.

3.5 *Exploring Students' Opinion on Software Testing Courses*

O artigo Cammaerts *et al.* (2024) investiga as opiniões de estudantes sobre cursos de teste de software. Embora muitos estudos tenham avaliado abordagens pedagógicas e ferramentas educacionais, poucos consideraram a perspectiva dos alunos sobre os diversos aspectos desses cursos.

Para preencher essa lacuna, foi realizado um estudo exploratório com 103 estudantes de dez cursos diferentes. Utilizou-se um questionário que abordou desde o contexto inicial dos alunos até o design do curso e os resultados finais alcançados. A pesquisa revelou que, de modo geral, os estudantes possuem uma percepção positiva sobre os cursos, mas apontaram áreas de melhoria.

Entre os principais achados, destacam-se: a necessidade de diversificação nas abordagens e ferramentas usadas, já que o foco excessivo em testes unitários e na linguagem Java limita o aprendizado; a importância de experiências práticas mais realistas e complexas; e o desejo por um aprendizado mais interativo, sugerindo o uso de técnicas de *gamificação*.

Além disso, o estudo indicou que os estudantes valorizam atividades práticas e projetos em grupo, que permitem aplicar conceitos teóricos em cenários próximos ao mercado de trabalho. Também apreciam *feedback* constante, tanto de colegas quanto de professores, para aprimorar suas habilidades.

O artigo conclui que há espaço para aprimorar o ensino de teste de software, incluindo a diversificação de ferramentas e abordagens, a integração de experiências práticas mais próximas do mundo real e o uso de *gamificação* para aumentar o engajamento. Esses *insights* oferecem direções valiosas para o desenvolvimento de cursos mais eficazes e alinhados às necessidades dos estudantes.

3.6 Análise Comparativa

O Quadro 6 fornece uma comparação visual entre os trabalhos relacionados apresentados e o presente trabalho, utilizando alguns critérios elaborados pela autora.

Assim como nos trabalhos de Paschoal e Souza (2018) , Melo *et al.* (2020) , Martins *et al.* (2021) e Barrett *et al.* (2023), este trabalho também investiga as práticas de teste de software que são abordadas no ensino de teste nos cursos de graduação em TI, com a principal diferença sendo no público-alvo e que este trabalho também se dispõe a investigar as práticas modernas

utilizadas na indústria. O trabalho de Paschoal e Souza (2018) tem como público-alvo apenas os professores de instituições de ensino superior brasileiras, já o de Melo *et al.* (2020) tem como público-alvo também os professores de instituições de ensino superior, porém a nível global, e o de Martins *et al.* (2021) possui como público-alvo apenas os profissionais de teste de software que possuem diploma de Bacharel em um curso relacionado à ciência da computação, e por último o de Barrett *et al.* (2023) que investiga os currículos das universidades e Cammaerts *et al.* (2024) que investiga opiniões dos alunos sobre os aspectos individuais de um curso de teste de software. Neste trabalho, o público-alvo serão os estudantes que ainda estão cursando graduação em TI e profissionais que já atuam na área de teste de software que possuam diploma de graduação em TI. Por fim, este trabalho difere-se dos demais trabalhos citados ao ampliar o escopo direcionando-o para uma investigação tanto para estudantes de graduação em Tecnologia da Informação quanto para profissionais que já atuam na área e possuem diploma nessa área. Além de identificar práticas e deficiências no ensino de teste de software.

A seguir, o Quadro 6 sintetiza essas diferenças e semelhanças utilizando critérios comuns.

Quadro 6 – Análise comparativa entre os trabalhos relacionados e este trabalho.

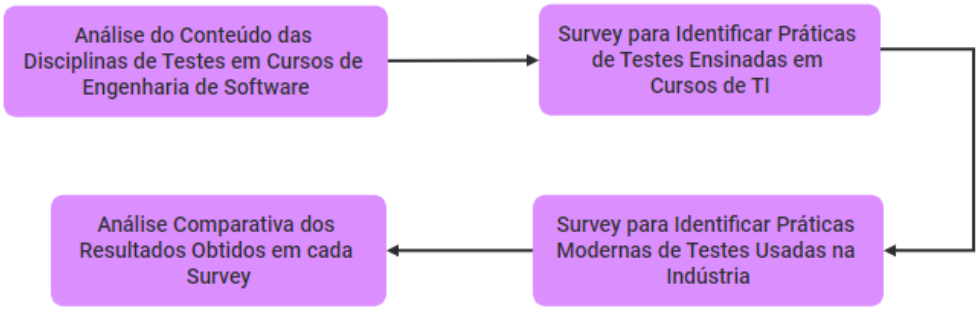
Critérios	**Paschoal e Souza (2018)**	**Melo *et al.* (2020)**	**Martins *et al.* (2021)**	**Barrett *et al.* (2023)**	**Cammaerts *et al.* (2024)**	**Trabalho Proposto**
Investiga Práticas de Teste de Software	**X**	**X**	**X**	**X**	**X**	**X**
Investiga as Principais Deficiências no Ensino de Teste	**X**	**X**	**X**		**X**	**X**
Investiga Práticas Modernas na Indústria						**X**
Auxilia o Ensino de Teste de Software			**X**		**X**	**X**

Fonte: Elaborado pelo autora.

4 METODOLOGIA

Nesta Seção, serão apresentadas as etapas adotadas para atingir os objetivos propostos neste estudo. A Figura 2 oferece uma visão geral das etapas, enquanto as próximas seções descrevem detalhadamente cada uma delas.

Figura 2 – Procedimentos metodológicos



Fonte: Elaborado pela autora.

4.1 Análise do Conteúdo das Disciplinas de Testes em Cursos de Engenharia de Software

A área de verificação e validação de software recebe uma atenção significativamente maior nos cursos de Engenharia de Software em comparação com outros cursos da área de computação. De acordo com o relatório da Association for Computing Machinery (2020), as diretrizes curriculares para Engenharia de Software reforçam a relevância de disciplinas voltadas para a garantia da qualidade e confiabilidade de sistemas de software, abrangendo práticas essenciais de verificação e validação. Essa ênfase distinta nos cursos de Engenharia de Software destaca a importância dessas práticas e justifica o foco desta análise nesse curso específico, dado seu compromisso mais profundo com o ensino de técnicas fundamentais para assegurar a qualidade de sistemas de software.

Dessa forma, utilizando como critério de escolha a nota de avaliação fornecida pelo Ministério da Educação (MEC), foram selecionados cursos de graduação em Engenharia de Software no Brasil com nota quatro ou cinco. A partir dessa seleção, foi realizada uma análise das ementas dos cursos para verificar se incluíam disciplinas voltadas para testes de software,

bem como a quantidade dessas disciplinas em cada curso.

Após essa identificação, conduziu-se uma nova análise, desta vez focada nos projetos pedagógicos dos cursos, com o objetivo de mapear os conteúdos e competências abordados em cada disciplina. Esse processo permitiu obter uma visão geral sobre o que, em teoria, é ensinado aos alunos durante a graduação. O capítulo 5 apresentará um detalhamento mais aprofundado dessa etapa, incluindo informações sobre os cursos analisados, as disciplinas identificadas e os conteúdos abordados em cada uma delas.

4.2 *Survey* para Identificar Práticas de Testes Ensinadas em Cursos de TI

Primeiramente, foram definidas as informações essenciais a serem obtidas por meio deste *survey*, bem como o público-alvo da pesquisa. Esse processo foi guiado por um estudo prévio da literatura, utilizando as pesquisas de Paschoal e Souza (2018), Melo *et al.* (2020) e Martins *et al.* (2021) como referência para identificar as principais questões a serem exploradas. Com esses dados estabelecidos, foi possível elaborar as perguntas de forma estruturada e alinhada aos objetivos da pesquisa. Antes da aplicação do *survey* ser realizada, um teste piloto foi conduzido com voluntários, para validar o questionário. Os resultados desse teste permitiram identificar ajustes necessários para melhorar a clareza das perguntas e garantir uma melhor compreensão por parte dos participantes. O *survey* foi disponibilizado online, em formato de questionário eletrônico, utilizando a ferramenta Google *Forms*.

A divulgação do questionário foi realizada através de canais eletrônicos de comunicação como *Linkedin* através de publicações e compartilhamento. A pesquisa também foi compartilhada por e-mail com os alunos da Universidade Federal do Ceará - Campus Quixadá, além de ter sido compartilhada por e-mail com coordenadores dos cursos de engenharia de software de outros campi, a fim de alcançar o maior número possível de participantes.

Após a aplicação do *survey*, as respostas dos participantes foram documentadas e organizadas de maneira estruturada, permitindo a rápida identificação e separação das respostas válidas daquelas consideradas inválidas por algum motivo. Em seguida, foi realizada uma análise detalhada dos dados, que foram também transformados em gráficos para facilitar a visualização e interpretação dos resultados. Essa abordagem possibilitou a identificação de padrões e tendências nos relatos dos participantes, destacando seus conhecimentos, principais dificuldades enfrentadas ao longo da disciplina e sugestões que na perspectivas dos estudantes aprimoraria o ensino de testes de software na graduação. O capítulo 6 apresentará de forma mais aprofundada as

informações sobre a aplicação deste *survey* e dos seus resultados obtidos.

4.3 *Survey* para Identificar Práticas Modernas de Testes Usadas na Indústria

Nesta etapa, um segundo *survey* foi desenvolvido desta vez voltado para profissionais da indústria de desenvolvimento de software que trabalham na área de testes, com o objetivo de identificar práticas modernas de teste mais utilizadas na indústria. Assim como no primeiro *survey*, inicialmente foram definidas as informações necessárias para a pesquisa utilizando os trabalhos de Paschoal e Souza (2018), Melo *et al.* (2020) e Martins *et al.* (2021) como referência para identificar as principais questões a serem exploradas.

Assim como no passo anterior (Seção 4.2), antes da aplicação deste segundo *survey*, um teste piloto foi conduzido com voluntários, para validar o questionário. Os resultados deste teste foram analisados e as adaptações foram feitas para garantir a eficácia e precisão do questionário final. A divulgação deste questionário também foi realizada através de canais eletrônicos de comunicação como *Linkedin* através de publicações e compartilhamento, e também em redes sociais através de grupos voltados para o conteúdo de teste de software/tecnologia. Novamente, a pesquisa também foi compartilhada por e-mail com os alunos da Universidade Federal do Ceará - Campus Quixadá, também foi compartilhada com participantes do primeiro *survey* que indicaram estar trabalhando na área de testes no momento da pesquisa os convidando a participarem desta pesquisa também, desta vez na perspectiva de profissionais no mercado.

Todas as respostas coletadas foram documentadas e organizadas de maneira estruturada, permitindo a rápida identificação e separação das respostas válidas daquelas consideradas inválidas por algum motivo. A análise dos dados foi feita, no capítulo 7 é apresentado mais detalhadamente esses resultados obtidos.

4.4 Análise Comparativa dos Resultados Obtidos em cada *Survey*

Nesta etapa, foi realizada uma análise comparativa entre os resultados obtidos nos dois *surveys*, com o objetivo de identificar possíveis divergências e semelhanças entre as percepções dos estudantes de Engenharia de Software e dos profissionais da indústria de software em relação às práticas de testes de software. A análise buscou destacar quais práticas de testes são mais enfatizadas na formação acadêmica e como essas práticas se alinham com aquelas aplicadas no ambiente profissional.

Primeiramente, foram organizados os dados de forma estruturada, possibilitando a comparação entre as respostas de ambos os grupos. Em seguida, foi feita uma análise qualitativa e quantitativa dos resultados. Foram analisados os dados numéricos, como a frequência de uso de determinadas ferramentas e técnicas de teste, bem como a opinião dos participantes sobre a importância dos diversos tipos de teste. Isso permitiu identificar, por exemplo, se há uma lacuna no ensino acadêmico em relação às ferramentas e metodologias mais utilizadas na indústria. Foram examinadas também as respostas abertas dos participantes, identificando temas comuns que se destacaram, como os desafios enfrentados na prática de testes, as competências mais valorizadas e as sugestões para aprimoramento tanto na formação acadêmica quanto nas práticas industriais.

Além disso, foi realizada uma análise crítica sobre as diferenças entre os conhecimentos adquiridos durante a graduação e as habilidades exigidas no mercado de trabalho. A comparação revelou se os estudantes estavam sendo preparados adequadamente para enfrentar os desafios da indústria, e se há uma necessidade de atualização nas práticas de ensino nos cursos de Engenharia de Software para alinhar melhor o conteúdo teórico à realidade do mercado. Com base nessa análise comparativa, foram elaboradas recomendações para melhorar o ensino de testes de software nos cursos de Engenharia de Software, levando em consideração as práticas mais modernas adotadas pela indústria. Essas recomendações têm como objetivo garantir que os futuros profissionais da área possuam não apenas os conhecimentos teóricos necessários, mas também a prática e as habilidades demandadas pelo mercado de trabalho.

5 INVESTIGANDO O CONTEÚDO DE TESTES DE SOFTWARE EM CURSOS DE GRADUAÇÃO DE ENGENHARIA DE SOFTWARE

Nesta seção, serão apresentadas informações mais detalhadas sobre a análise realizada das ementas das disciplinas de teste dos cursos selecionados, a forma como foi conduzida e os resultados obtidos.

5.1 Analise das Ementas das Disciplinas de Testes de Software dos Cursos de Graduação em Engenharia de Software

A Association for Computing Machinery (2020), em suas diretrizes curriculares, destaca a importância de que alunos de Ciência da Computação e Engenharia de Software desenvolvam habilidades em testes de software, verificação e validação, além de qualidade de software. Isso inclui a compreensão e aplicação de diferentes técnicas de teste, como testes de unidade, integração, sistema e aceitação, bem como o conhecimento de métodos formais e informais para verificar se o software atende aos requisitos e validar se ele resolve o problema do usuário. Diante dessa ênfase, a pesquisa concentrou-se na análise das ementas dos cursos de Engenharia de Software, garantindo assim que houvesse na ementa dos cursos pelo menos uma disciplina obrigatória voltada para testes e/ou verificação e validação.

Inicialmente, optou-se por selecionar, diretamente no site do Ministério da Educação (2023), os cursos de Engenharia de Software com avaliação igual a 4 ou 5. Posteriormente, o projeto pedagógico de cada curso selecionado foi analisado para identificar e confirmar a presença de disciplinas relacionadas a testes. A Tabela 2 apresenta as instituições que atenderam aos critérios mencionados.

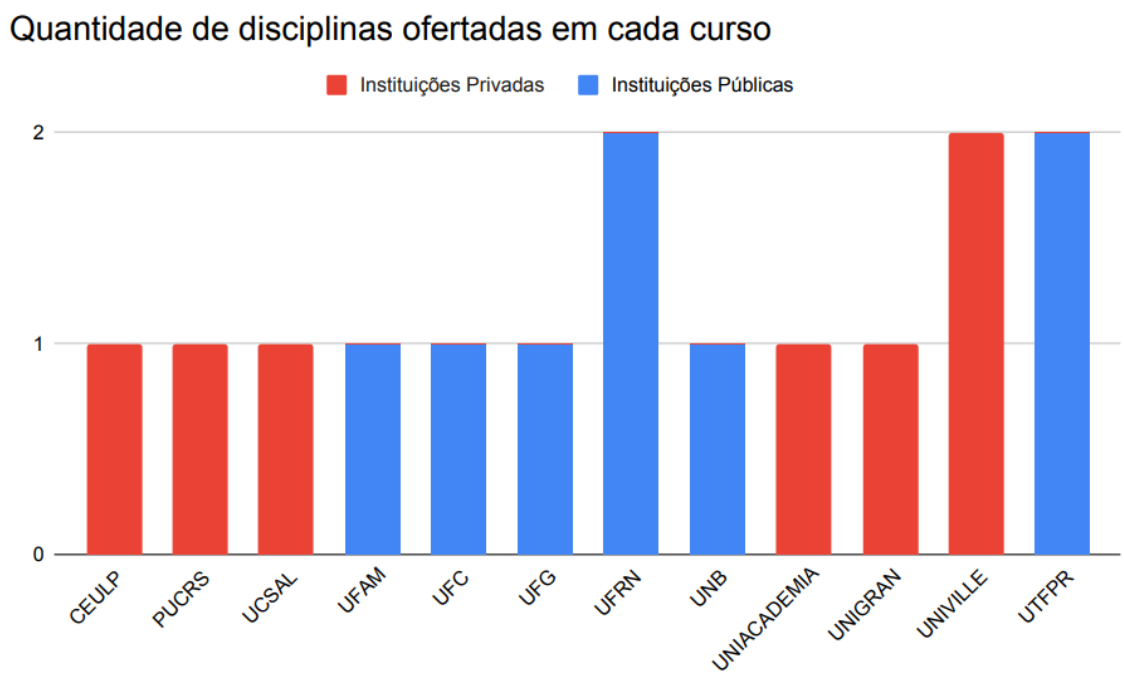
Tabela 2 – Instituições selecionadas para análise das ementas das disciplinas

Instituições	**Sigla**	**Nota no MEC**
Universidade de Brasília	UNB	Nota 05
Pontifícia Universidade Católica do Rio Grande do Sul	PUCRS	Nota 05
Centro Universitário Luterano de Palmas	CEULP	Nota 05
Universidade Federal de Goiás	UFG	Nota 05
Centro Universitário da Grande Dourados	UNIGRAN	Nota 05
Universidade Federal do Ceará	UFC	Nota 05
Universidade Tecnológica Federal do Paraná	UTFPR	Nota 04
Universidade da Região de Joinville	UNIVILLE	Nota 04
Centro Universitário Academia	UNIACADEMIA	Nota 04
Universidade Católica de Salvador	UCSAL	Nota 04
Universidade Federal do Rio Grande do Norte	UFRN	Nota 04
Universidade Federal do Amazonas	UFAM	Nota 04

Fonte: Elaborado pela autora

Algumas universidades, embora tenham obtido avaliação quatro ou cinco, não foram incluídas nesta lista devido a diversas razões. Em alguns casos, a busca pelo projeto pedagógico do curso ou pela matriz curricular necessária para verificar a presença de disciplinas relacionadas a testes não foi bem-sucedida. Em outras situações, a exclusão ocorreu devido a informações encontradas no site do MEC, indicando processos judiciais em andamento que poderiam comprometer a credibilidade do curso de Engenharia de Software e/ou da própria universidade. Ao todo, foram selecionadas 12 universidades que se encaixaram nos critérios, tendo 6 universidades com nota cinco e 6 universidades com nota quatro. A partir dessas informações, foi elaborado um gráfico mostrado na Figura 3 onde podemos ver a disposição das universidades entre públicas e privadas e a quantidade de disciplinas voltadas para teste.

Figura 3 – Gráfico Instituições



Fonte: Elaborado pela autora.

Ao analisar as ementas dos cursos de graduação em Engenharia de Software selecionados, foi possível identificar se de fato os cursos possuíam pelo menos uma disciplina voltada para testes e, através do projeto pedagógico de cada curso, identificamos os conteúdos ofertados por cada disciplina. A Tabela 3 mostra uma lista das disciplinas voltadas para teste de software que são ministradas em cada um dos cursos das instituições selecionadas para a pesquisa. Uma versão detalhada dos conteúdos abordados em cada disciplina pode ser consultada no Apêndice C.

Tabela 3 – Disciplinas ofertadas em cada curso das instituições

Instituições	**Disciplinas**	**CH**	**Site**
CEULP	Teste de Software	60h	https://bit.ly/4jsiLZI
PUCRS	Verificação e Validação de Software	60h	https://bit.ly/4jmvKMH
UCSAL	Testes e Qualidade de Software	60h	https://bit.ly/4h49TrF
UFAM	Verificação, Validação e Teste de Software	90h	https://bit.ly/3Q5o83X
UFC	Verificação e Validação	64h	https://es.quixada.ufc.br/
UFG	Teste de Software	64h	https://bit.ly/3E45CpL
UFRN	Teste de Software I	60h	https://bit.ly/4h2wyVl
UFRN	Teste de Software II	60h	https://bit.ly/4h2wyVl
UNB	Testes de Software	30h	http://software.unb.br/
UNIACADEMIA	Teste de Software	36h	http://bit.ly/42iPquQ
UNIGRAN	Verificação e Validação	80h	https://bit.ly/4gaSi05
UNIVILLE	Teste de Software I	72h	https://bit.ly/3E5DyCt
UNIVILLE	Teste de Software II	72h	https://bit.ly/3E5DyCt
UTFPR	Teste de Software	60h	https://bit.ly/4gonut3

Fonte: Elaborado pela autora.

Ao analisar as ementas dos cursos de Engenharia de Software, observou-se que os conteúdos relacionados a testes de software apresentam, em sua maioria, uma base teórica sólida que enfatiza conceitos fundamentais, como os diferentes tipos de testes (unidade, integração, sistema, regressão) e as técnicas para geração de casos de teste. Contudo, a partir da análise comparativa dos projetos pedagógicos, nota-se uma expressiva variação na profundidade e na abordagem prática desses conteúdos.

No total, foram selecionados 12 cursos de Engenharia de Software, totalizando 14 disciplinas relacionadas a testes. Desses, apenas 2 cursos possuem mais de uma disciplina voltada para teste de software. Das 14 disciplinas, 12 puderam ser analisadas mais detalhadamente por meio do projeto pedagógico do curso, enquanto duas não tiveram informações detalhadas disponíveis. Entre os conteúdos mais recorrentes estão técnicas, geração de casos de teste, testes de regressão e ferramentas. Na Tabela 4 é possível visualizar os 10 conteúdos mais abordados nas disciplinas, classificados de acordo com a frequência de aparição.

Um aspecto relevante a ser destacado é que, embora diversas disciplinas enfatizem a automação de testes, nenhuma delas cita de forma explícita o uso de outras práticas modernas, como *test smells* e *flaky tests*. Essa ausência evidencia uma limitação na formação dos alunos, pois restringe o panorama das técnicas atuais em testes de software à única vertente da automação.

Por um lado, muitas disciplinas concentram-se na exposição dos fundamentos, proporcionando uma visão ampla dos métodos e técnicas tradicionais. Essa oferece aos alunos uma compreensão dos conceitos básicos, mas frequentemente deixa lacunas quanto à aplicação

Tabela 4 – Conteúdos mais abordados nas disciplinas

Conteúdos	**Quantidade de menções**
Técnicas de teste	08
Geração de casos de teste	08
Teste de regressão	05
Ferramentas de testes	05
Testes de sistema	05
Teste de unidade	04
Teste de integração	04
Desenvolvimento orientado a testes	04
Automação dos testes	04
Testes de aceitação	04

Fonte: Elaborado pela autora.

prática dos conhecimentos. Por outro lado, em alguns cursos onde o conteúdo é dividido em mais de uma disciplina ou em módulos mais específicos, há uma tendência a integrar teoria e prática de forma mais efetiva. Nessas iniciativas, os conteúdos teóricos são complementados por atividades práticas como o desenvolvimento orientado a testes (TDD), a aplicação de *frameworks* de automação e a elaboração de planos de teste, o que contribui para uma preparação mais alinhada às demandas reais do mercado.

Outro aspecto que evidencia a variação na profundidade dos conteúdos é a carga horária dedicada às disciplinas. Observa-se que, enquanto alguns cursos apresentam uma carga horária de 60 horas – como os oferecidos pelo CEULP, PUCRS, UCSAL e UTFPR –, outros oferecem uma abordagem mais extensa, com 80 horas como no caso da UNIGRAN, 90 na UFAM e um total de 120 horas divididas entre as duas disciplinas da UFRN e um total de 144 horas divididas entre as duas disciplinas da UNIVILLE. Contudo, há cursos com cargas horárias reduzidas, como o da UNB (30 horas) e UNIACADEMIA (36 horas), o que pode refletir um tratamento mais superficial do tema. Essa discrepância ressalta a necessidade de uma padronização que permita um aprofundamento adequado, possibilitando que os alunos não apenas compreendam os conceitos teóricos, mas também os apliquem de maneira robusta na prática.

Por fim, outro ponto de destaque diz respeito à inclusão de tópicos de práticas modernas, que aparecem de forma pontual e, em geral, com menor aprofundamento. Essa discrepância evidencia uma possível defasagem entre a formação acadêmica e as exigências do setor de tecnologia, que cada vez mais valoriza habilidades práticas e o domínio de ferramentas capazes de suportar a dinâmica dos processos de desenvolvimento modernos.

Portanto, a análise dos conteúdos curricularmente oferecidos revela que, embora os cursos selecionados garantam a presença de uma disciplina voltada para testes, há uma diversidade significativa na forma e na profundidade com que os temas são tratados. Essa disparidade aponta para a necessidade de uma maior integração entre teoria e prática, bem como para a atualização dos conteúdos, incorporando de forma mais explícita as demais práticas modernas além da automação, a fim de formar profissionais mais preparados para os desafios contemporâneos do mercado de software.

6 *SURVEY* INVESTIGANDO O ENSINO DE PRÁTICAS MODERNAS NA DISCIPLINA DE TESTE DE SOFTWARE

Nesta seção, será apresentado o primeiro *survey* , aplicado a participantes que já concluíram ou estão cursando uma graduação na área da computação (Apêndice A), juntamente com os resultados obtidos. A coleta de informações foi feita através de um questionário online na plataforma do *Google Forms*, a análise dos dados coletados nos traz uma visão do nível de conhecimento adquirido pelos alunos nas disciplinas de teste de software.

6.1 Desenvolvimento e Estruturação do Questionário

O *survey* (Apêndice A) tem como principal objetivo compreender melhor a experiência dos participantes em relação ao ensino de testes de software durante a graduação, bem como sua familiaridade com práticas modernas e ferramentas associadas a essa área. Dividido em seções específicas, o questionário aborda desde informações básicas sobre a graduação até questões mais detalhadas sobre o conhecimento em testes de software.

Inicialmente foram coletadas informações sobre a graduação dos participantes, incluindo o status atual do curso e a presença de disciplinas voltadas para testes de software. Esses dados foram essenciais para contextualizar as respostas e entender melhor a formação acadêmica dos participantes, além de possibilitar a diferenciação do nível de conhecimento em testes de software entre os participantes que estão no início e aqueles que já estão em estágios mais avançados do curso de graduação.

Na sequência, a seção sobre disciplinas de teste de software visa aprofundar o conhecimento sobre a experiência dos participantes nesse campo específico, incluindo uma autoavaliação do nível de conhecimento adquirido e sobre a abordagem teórica e prática das disciplinas. Ainda nesta seção, os participantes também tiveram a oportunidade de compartilhar os desafios encontrados durante o aprendizado e indicar se sentiram falta de algum conteúdo específico durante a disciplina.

A seção sobre conhecimento em testes de software busca identificar o grau de familiaridade dos participantes com diferentes conceitos, práticas e técnicas de teste, incluindo níveis de teste, técnicas de teste estrutural e funcional, práticas modernas como testes automatizados, *test smells* e *flaky tests*, além do conhecimento sobre ferramentas e *frameworks* de teste.

Por fim, os participantes tiveram a oportunidade de trazer sugestões de melhorias no ensino de teste de software, visando fornecer *insights* valiosas para possíveis aprimoramentos no

currículo acadêmico, levando em consideração a percepção dos próprios participantes sobre o assunto. No Quadro 7 é apresentado o *survey* de forma resumida.

Quadro 7 – Questões do *Survey*.

ID	Questão
	Identificação
Q01	Nome
Q02	Estado
Q03	Informe o curso de graduação.
Q04	Informe a instituição de ensino do seu curso de graduação
Q05	Você está trabalhando na área da sua graduação atualmente?
Q06	Caso esteja trabalhando atualmente, qual cargo você ocupa?
	Sobre a graduação
Q07	Status da graduação
Q08	O seu curso possui alguma disciplina voltada para testes de software?
Q09	Caso ainda esteja cursando a graduação, você já cursou a disciplina voltada para testes de software?
	Sobre a disciplina de testes
Q10	Como você avalia o seu nível de conhecimento adquirido na disciplina de teste de software?
Q11	Quanto ao conteúdo abordado na disciplina de teste de software durante sua graduação, você acha que foi mais voltado para a teoria ou para a prática?
Q12	Você encontrou desafios ao aprender o conteúdo da disciplina de teste de software?
Q13	Por favor, descreva as dificuldades que você encontrou durante a disciplina de teste de software.
Q14	Houve alguma falta em relação a conteúdos, técnicas, abordagens e/ou ferramentas que você gostaria de ter aprendido na disciplina de teste de software?
Q15	Qual conteúdo, técnica, abordagem ou ferramenta específica você gostaria de ter aprendido na disciplina de teste de software?
	Conhecimentos em teste de software
Q16	Quais níveis de teste de software você conhece e/ou teve contato?
Q17	Quais técnicas de teste de software você conhece e/ou teve contato?
Q18	Quais testes estruturais (ou teste de caixa-branca) você conhece e/ou teve contato?
Q19	Quais testes funcionais (ou teste de caixa-preta) você conhece e/ou teve contato?
Q20	Quais práticas modernas de teste você conhece?
Q21	Sobre testes automatizados, quais dos seguintes testes você conhece e/ou teve contato?
Q22	Sobre Test *Smells*, quais dos seguintes testes você conhece e/ou teve contato?
Q23	Sobre *Flaky Tests*, quais dos seguintes testes você conhece e/ou teve contato?
Q24	Você conhece alguma ferramenta e/ou *framework* para testes de software?
Q25	Você conhece alguma ferramenta de gerenciamento de teste?
Q26	Tendo em vista o que você viu na graduação e as práticas e técnicas mostradas nessa pesquisa, na sua opinião, o ensino de teste de software necessita de melhorias?
	Sugestões de melhorias
Q27	O que você sugere que seja feito para trazer melhorias no ensino de teste de software?

Fonte: Elaborado pela autora.

6.2 Teste Piloto

Ao finalizar o desenvolvimento e estruturação do questionário, um teste piloto foi realizado visando identificar possíveis falhas e, com isso, garantir que o questionário atenda aos objetivos propostos antes da divulgação para os participantes. O teste piloto foi aplicado a três estudantes da Universidade Federal do Ceará que se disponibilizaram para participar.

A aplicação do teste piloto permitiu identificar, por meio das considerações dos participantes, a presença de alguns erros ortográficos. Além disso, foi sugerida a modificação de algumas perguntas de múltipla escolha para o formato de caixas de seleção. Por fim, recomendou-se incluir uma breve e simples definição para as questões relacionadas a conteúdos específicos. A versão final do questionário está disponível no Apêndice A deste trabalho.

6.3 Divulgação

O *survey* foi amplamente divulgado, com o objetivo de atingir um público-alvo composto por alunos e ex-alunos dos cursos de graduação em computação. Para ampliar o alcance, foi realizado um pedido formal à Universidade Federal do Ceará para que a pesquisa fosse divulgada entre os estudantes dos diversos campi que oferecem cursos na área de computação.

Além disso, a pesquisa foi divulgada amplamente por meio de redes sociais como *LinkedIn* e *WhatsApp*, utilizando postagens e compartilhamentos em grupos para alcançar o público-alvo. Essa estratégia possibilitou um maior engajamento e visibilidade, contribuindo para uma maior diversidade de participantes no estudo.

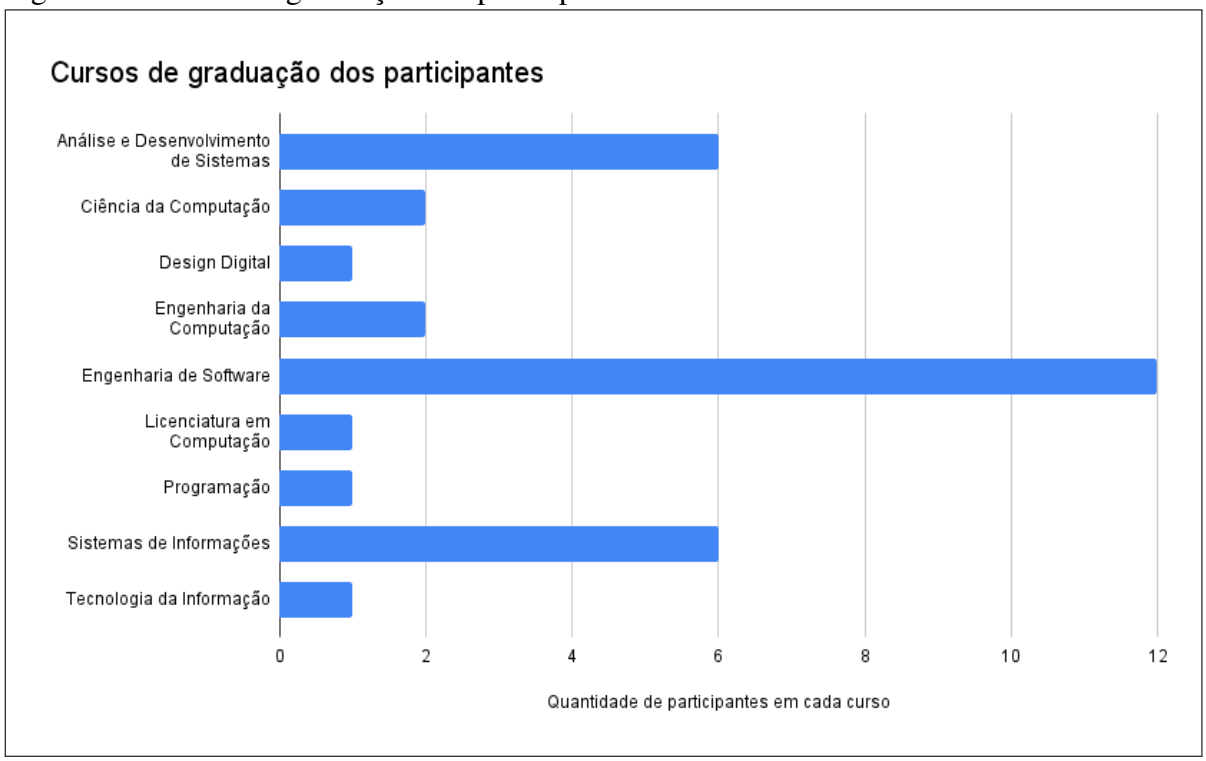
6.4 Perfil dos Participantes

O questionário recebeu um total de 32 respostas válidas, abrangendo participantes de cinco estados brasileiros. **Q02.**A maioria das respostas veio do Ceará (CE), representando 84,4% (27) dos participantes. Em seguida, São Paulo (SP) contribuiu com 6,3% (2) das respostas, enquanto Mato Grosso do Sul (MS), Piauí (PI) e o Distrito Federal (DF) tiveram cada um 3,1% (1) dos participantes.

A Figura 4 (**Q03**) mostra que 37,5% (12) dos participantes declarou cursar Engenharia de Software. Sistemas de Informação, Análise e Desenvolvimento de Sistemas e Sistemas de Informação apareceram em seguida, cada um com 18,8% (6). Outros cursos mencionados foram Ciência da Computação e Engenharia da Computação, ambos com 6,3% (2). Além disso, os cursos Design Digital, Licenciatura em Computação, Tecnologia da Informação e Programação foram declarados por 3,1% (1) dos participantes cada.

Q04. Dos participantes, 43,8% (14) são da Universidade Federal do Ceará (UFC), seguida pelo campus Quixadá da mesma instituição, com 18,8% (6). A UniCatólica de Quixadá, contribuiu com 6,3% (2) dos participantes. As demais instituições de ensino tiveram a mesma

Figura 4 – Cursos de graduação dos participantes



Fonte: Elaborado pela autora.

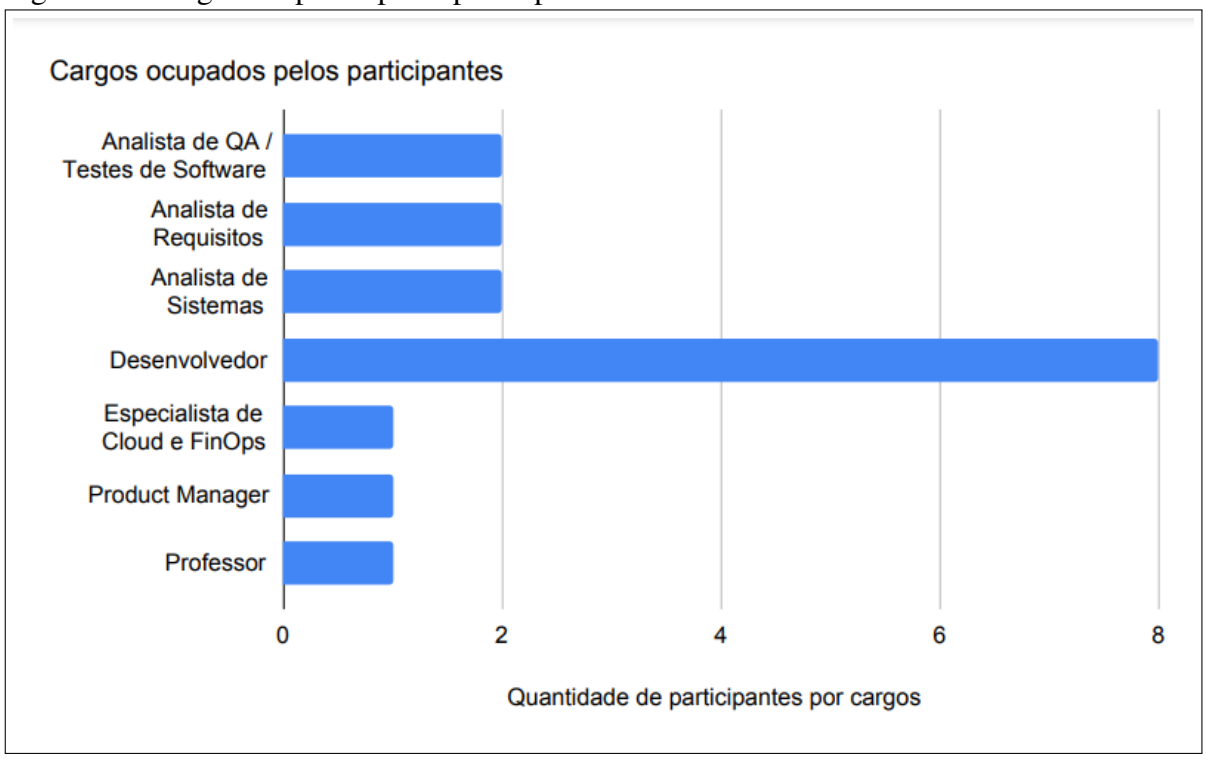
proporção de participação, com 3,1% (1) cada: Universidade Católica de Brasília (UCB), Universidade Estadual do Ceará (UECE), Universidade Virtual do Estado de São Paulo (Univesp), Instituto Federal do Piauí (IFPI), Estácio, Universidade Cidade de São Paulo (UNICID), Universidade Federal do Ceará - Campus de Itapajé, FIAP, Unicesumar e Unifanor. Essa diversidade de instituições públicas e privadas participantes enriquece a pesquisa, oferecendo uma amostra mais representativa e abrangente, refletindo diferentes contextos acadêmicos.

56,3% (18) dos participantes não estavam atuando na área relacionada à sua graduação no momento da coleta dos dados (**Q05**). A Figura 5 mostra os cargos ocupados pelos 43,8% (14) restantes (**Q06**). Quanto ao status da graduação, 78,1% (25) declarou estar com a graduação em andamento. Outros 3,1% (1) informaram estar formados há menos de 1 ano, enquanto 12,5% (4) estão formados há 1 a 3 anos. Adicionalmente, 3,1% (1) dos participantes possuem formação concluída há mais de 6 anos, e outros 3,1% (1) há mais de 10 anos (**Q07**).

6.5 Experiência acadêmica dos participantes na disciplina de teste de software

Considerando que, neste ponto do questionário continuaram apenas os participantes que já concluíram a graduação ou que estão cursando atualmente e já realizaram a disciplina relacionada a testes, as próximas questões foram elaboradas com o objetivo de compreender a

Figura 5 – Cargos ocupados pelos participantes



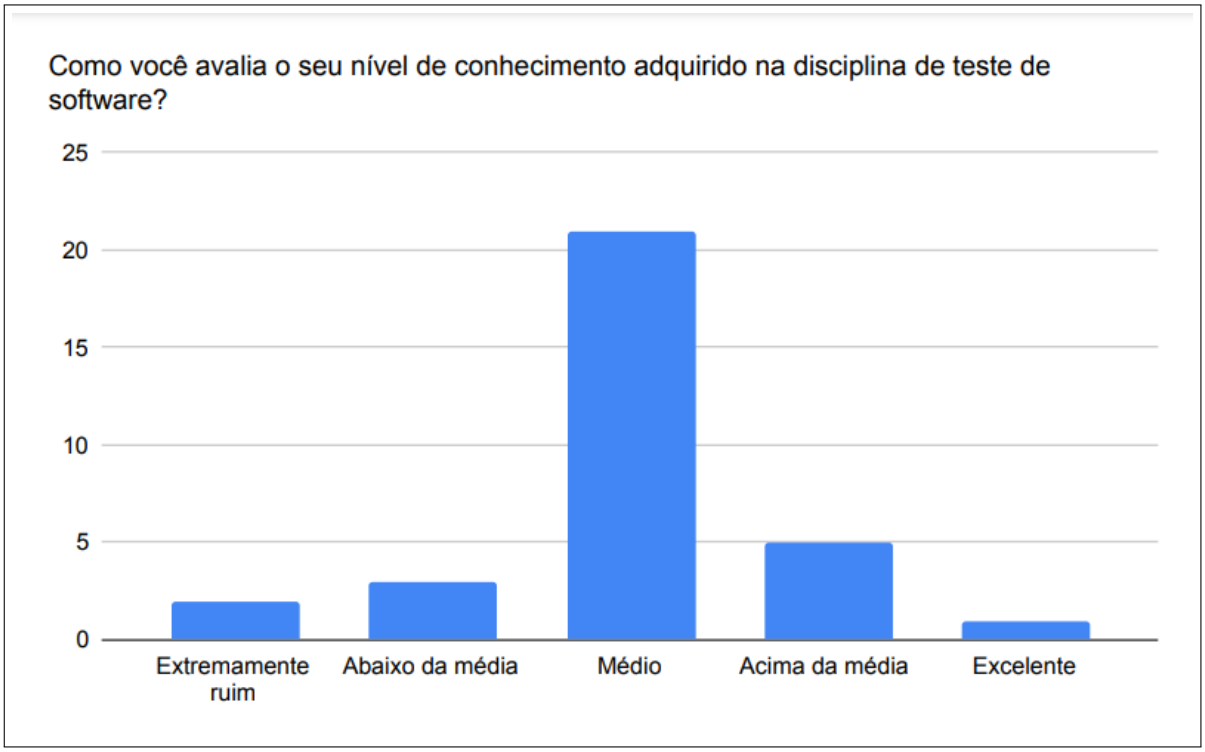
Fonte: Elaborado pela autora.

percepção dos participantes sobre a experiência e os aprendizados adquiridos na disciplina.

A análise dos dados revela que a percepção dos participantes em relação à disciplina de Teste de Software se concentra predominantemente em um nível de conhecimento mediano (**Q10**). Essa avaliação sugere que, embora os conteúdos tenham sido assimilados, ainda há margem para aprofundamento e aprimoramento dos conceitos abordados. Observa-se, inclusive, que um grupo menor de alunos se sente mais confiante, classificando seu nível de conhecimento como acima da média ou até excelente, o que pode indicar que determinados métodos de ensino ou abordagens específicas estão beneficiando esses estudantes. Por outro lado, a presença de avaliações negativas—com alguns participantes considerando seu desempenho abaixo da média ou até extremamente insatisfatório—aponta para a necessidade de uma revisão cuidadosa dos processos pedagógicos empregados. Esse contraste entre as avaliações positivas e negativas destaca a importância de identificar quais elementos da disciplina podem ser ajustados para atender melhor às diferentes expectativas e perfis dos alunos. Ver a Figura 6.

A **Q11** mostra que a maioria dos participantes considerou a disciplina de Teste de Software como mais voltada para a teoria. 50% (16) dos participantes indicaram que o conteúdo foi predominantemente teórico, e apenas 12,5% (4) consideraram a disciplina mais prática. Por outro lado, 37,5% (12) avaliaram que a abordagem foi equilibrada, contemplando de maneira

Figura 6 – Nível de conhecimento adquirido na disciplina de teste de software



Fonte: Elaborado pela autora.

eficaz tanto os aspectos teóricos quanto práticos. Esses dados indicam uma ênfase no aspecto teórico do ensino da disciplina, embora uma parte significativa dos respondentes reconheça esforços para integrar teoria e prática.

Ao serem questionados sobre os desafios encontrados ao aprender o conteúdo da disciplina (**Q12**), 53,1% (17) dos participantes afirmaram não ter enfrentado dificuldades, enquanto 46,9% (15) relataram ter encontrado algum tipo de desafio. Embora a maioria dos alunos tenha considerado o conteúdo acessível, uma parcela significativa enfrentou algum tipo de obstáculo durante o aprendizado na disciplina.

Analisando as respostas abertas fornecidas pelos participantes que tiveram alguma dificuldade (**Q13**), os principais desafios relatados foram:

– *Dificuldades com o equilíbrio entre teoria e prática*

  Vários participantes mencionaram que a ênfase teórica da disciplina dificultou a compreensão inicial do conteúdo. Comentários como *"Muita teoria, pouca prática"* e *"No início a disciplina parecia mais difícil de compreender quando explicada somente de forma teórica"* evidenciam a necessidade de uma abordagem mais prática. Outros destacaram que os exemplos utilizados na disciplina estavam distantes de cenários reais, com observações como *"Os cenários apresentados costumam ser adaptados para funcionar dentro da*

metodologia de ensino, mas geralmente estão longe de um cenário real do mercado."

– *Barreiras técnicas e recursos insuficientes*

Alguns participantes relataram dificuldades específicas com ferramentas e configurações, como *"Adaptação a uma nova ferramenta de testes"* e *"As configurações necessárias para que o teste fosse executado (ambiente de teste)"*. Além disso, a falta de acesso a recursos modernos e gratuitos foi apontada como um obstáculo: *"Falta de práticas e recursos mais acessíveis para realizar os testes, a maioria era paga."*

Outros desafios mencionados foram relacionados a fatores externos, como a pandemia ou greves, que afetaram a qualidade da aprendizagem: *"Fiz durante a pandemia, não foi muito proveitosa a disciplina para mim."* e ***"Greve, por isso não vi tudo, ficava no fundo da sala."*** Questões pessoais também surgiram, como *"Mais dificuldades pessoais, nenhuma relação com a disciplina."*

Metade dos participantes confirmou que sentiu falta de algum conteúdo durante a disciplina de testes (**Q14**), seja uma técnica, abordagem específica e/ou uma ferramenta. Enquanto a outra metade dos participantes afirmou estar satisfeita com o que foi ofertado no decorrer da disciplina.

Muitos participantes destacaram o desejo de aprender ferramentas e técnicas mais atualizadas, como TDD, BDD, testes de carga (**Q15**). A carência de práticas voltadas para a aplicação em cenários reais também foi um ponto recorrente, evidenciando a necessidade de alinhar o ensino acadêmico às demandas do mercado. Além disso, houve menções à necessidade de uma abordagem mais prática na montagem de ambientes de teste e na criação de documentação e relatórios. Alguns participantes relataram uma experiência limitada com ferramentas específicas que não foram suficientemente aprofundadas para consolidar o aprendizado.

6.6 Análise do Conhecimento de Teste de Software dos Participantes

A análise dos dados revela que a grande maioria dos participantes já teve contato com os diferentes níveis de teste de software (**Q16**), demonstrando uma base significativa de conhecimento na área. Apenas um pequeno percentual indicou não ter tido nenhuma experiência com esses testes. Destaca-se que o Teste de Aceitação é o mais conhecido, com a maioria dos participantes mencionando-o; em seguida, os testes de Integração e Unidade também apresentaram índices expressivos, evidenciando que a abordagem de testes estruturada, desde componentes individuais até a integração de módulos, é valorizada. Por outro lado, a menor

menção ao Teste de Usabilidade pode apontar para uma lacuna ou menor ênfase nessa área específica.

A análise das respostas para as questões **Q17**, **Q18** e **Q19** revela uma disparidade no conhecimento e na experiência dos participantes em relação às diferentes técnicas de teste de software. Enquanto a maioria declarou ter tido contato com testes funcionais (caixa-preta) e estruturais (caixa-branca), um percentual relevante afirmou não ter familiaridade com nenhuma das técnicas ou não se recordar das diferenças entre elas. Esse dado sugere uma possível dificuldade na fixação dos conceitos ou uma abordagem didática que poderia ser aprimorada para garantir maior retenção do aprendizado.

Em relação aos testes estruturais (**Q18**), uma parcela expressiva dos participantes indicou não ter tido contato com esse tipo de teste. Entre aqueles que relataram experiência, o teste de caminho foi o mais mencionado, seguido pelos testes de fluxo de controle e fluxo de dados. Essa distribuição sugere que, apesar de alguns conceitos estruturais serem abordados, ainda há uma lacuna significativa no contato prático com essas técnicas, o que pode impactar a aplicação eficiente desses testes no desenvolvimento de software.

Já no que se refere aos testes funcionais (**Q19**), os resultados apontam para uma situação semelhante. Embora técnicas como análise de valor limite e partição de equivalência tenham sido citadas por uma parte dos respondentes, um número considerável de participantes declarou não ter tido contato com testes funcionais. Isso reforça a necessidade de um maior aprofundamento nesse tema, visto que os testes de caixa-preta são amplamente utilizados na validação de requisitos e na garantia da qualidade do software.

Foi solicitado aos participantes que informassem as práticas de teste modernas que conheciam (**Q20**), e 15,6% (5) indicaram não conhecer nenhuma prática moderna de teste. Entre os que relataram conhecimento, testes automatizados foram mencionados por 84,4% (27), evidenciando sua ampla adoção e reconhecimento. Além disso, 34,4% (11) mencionaram o uso de *Test Smells*, enquanto 12,5% (4) citaram práticas relacionadas a *Flaky Tests*. Podemos perceber que a maioria dos participantes está familiarizada com a automação de testes; porém, há ainda um conhecimento menos difundido sobre práticas específicas como identificação de *test smells* e *flaky tests*.

A análise das respostas para a **Q21** indica que a maioria dos participantes já teve algum contato com testes automatizados, embora uma parcela ainda não tenha experiência com esse tipo de abordagem (15,6%). Entre os que relataram conhecimento, os testes mais

citados foram os de Integração, Usabilidade, Unidade, Funcional e Aceitação, destacando-se como as práticas mais difundidas e essenciais no desenvolvimento de software. Além disso, os testes de Interface Gráfica do Usuário e de Regressão também apareceram em uma quantidade considerável de respostas, evidenciando a preocupação com a automação para garantir a estabilidade e a experiência do usuário. Apesar disso, testes mais específicos, como os de Segurança, Interoperabilidade e Concorrência, foram mencionados por uma parcela muito menor de participantes. Essa distribuição sugere que, embora os conceitos básicos de testes automatizados sejam bem disseminados, há espaço para ampliar o ensino e a prática de abordagens mais especializadas.

A análise da **Q22** revela que a maioria dos participantes (62,5%) não teve contato com nenhum tipo de *Test Smells*, o que pode indicar uma lacuna no ensino ou na prática desses conceitos, que são fundamentais para a qualidade dos testes automatizados. Entre aqueles que relataram conhecimento sobre *Test Smells*, o mais citado foi *Test Code Duplication*, o que sugere uma conscientização sobre a duplicação de código nos testes, um problema comum que pode comprometer a manutenção e a eficiência da suíte de testes. Outros *Test Smells* mencionados incluem *Lazy Test*, *Mystery Guest* e *Eager Test*. No entanto, *Test Smells* mais específicos, como *Indirect Testing*, foram pouco citados, sugerindo que conceitos mais avançados podem não ser amplamente abordados ou reconhecidos pelos participantes.

A análise da **Q23** revela que a grande maioria dos participantes (84,4%) não teve contato com *Flaky Tests*, indicando que o conceito ainda não é amplamente discutido ou reconhecido entre os respondentes. Isso é preocupante, pois *Flaky Tests* podem comprometer a confiabilidade dos testes automatizados, dificultando a detecção de falhas reais e reduzindo a confiança na suíte de testes. Entre os que relataram conhecimento sobre o tema, *Async Wait* foi o *Flaky Test* mais citado. Outros tipos mencionados incluem *IO*, *Concurrency* e *Test Order Dependency*. No entanto, *Flaky Tests* mais específicos, como *Network* e *Randomness*. Os dados indicam que há uma oportunidade de aprofundar o ensino sobre *Flaky Tests*, especialmente porque esses testes podem impactar negativamente a qualidade do software. A conscientização e a aplicação de boas práticas para evitar testes instáveis são essenciais para garantir a confiabilidade dos testes automatizados e a eficiência do processo de desenvolvimento.

Quando questionados sobre o conhecimento de ferramentas e *frameworks* para testes de software (**Q24**), a maioria dos participantes afirmou ter experiência com essas tecnologias. Entre as mais mencionadas, destacaram-se o *Selenium*, *JUnit* e *Cypress*, seguidos de outras

ferramentas e *frameworks* como *TestNG*, *Gatling*, *JMeter*, *RestAssured*, *Mocha*, *Jest*, *Ranorex* e até *Delphi*. Vale ressaltar que alguns participantes responderam de forma genérica, apenas confirmando o conhecimento, sem detalhar as ferramentas específicas. Por outro lado, uma parcela considerável indicou não ter familiaridade com ferramentas ou *frameworks* para testes de software. Quando o assunto foi o conhecimento sobre ferramentas de gerenciamento de testes (**Q25**), uma quantidade menor de participantes afirmou conhecer alguma dessas ferramentas. As mais citadas foram *Test Link* e *Test Manager*, com menções também a outras ferramentas como Jira, *Confluence* e *Jenkins*. No entanto, a grande maioria dos participantes indicou não ter conhecimento sobre essas ferramentas de gerenciamento, evidenciando uma possível lacuna nesse aspecto da prática de testes.

Após responderem sobre seus conhecimentos em testes de software, os participantes foram questionados se, com base no que aprenderam na graduação e nas práticas e técnicas apresentadas nesta pesquisa, consideram que o ensino de testes de software precisa de melhorias (**Q26**) e 87,5% (28) acreditam que o ensino de teste de software na graduação necessita de melhorias. Apenas 12,5% (4) responderam que não veem necessidade de mudanças no ensino dessa área.

Q27. Por último, foi proposto aos participantes que contribuíssem com opiniões sobre como o ensino de testes de software poderia ser melhorado. De modo geral, muitos participantes apontaram como principal mudança a inclusão de mais práticas durante a disciplina. Os participantes relataram compreender a importância do conteúdo teórico, mas que teriam tido um aproveitamento melhor da disciplina se houvesse mais atividades práticas e inclusão de ferramentas mais modernas que estejam sendo utilizadas no mercado, como podemos observar no comentário abaixo:

"A teoria sempre é deveras importante, mas durante minha disciplina, senti mais falta da prática em si. Gostaria de conhecer muito mais ferramentas. Pois para mim, só foi usado Pytest, unittest, selenium e testlink (o qual considerei deplorável)."

Outro participante comentou:

"A teoria deve ser sim abordada mas também em paralelo com a prática. Abordagem de casos de uso desde a elaboração teórica até a abordagem prática."

Outro ponto bastante comentado pelos participantes foi a necessidade de incluir disciplinas específicas sobre testes de software na grade obrigatória, garantindo que o tema seja tratado com mais profundidade:

"Acredito que uma disciplina voltada apenas para essa temática, pois meu contato foi através da disciplina de Engenharia de Software."

Outro participante disse ainda:

"Teoria é sempre importante, é claro. E também é importante ter em mente que o tempo disponível para uma disciplina é extremamente limitado. Uma coisa talvez interessante seria ter mais de uma disciplina voltada para isso. Claro, isso também não é algo simples, mas às vezes o conteúdo é grande demais para uma disciplina só."

Outras sugestão interessantes foram citadas como a integração do ensino de testes a projetos realizados em outras disciplinas, como desenvolvimento web ou mobile para que fosse possível aplicar os conceitos aprendidos em situações mais próximas da realidade do mercado.

Os participantes demonstraram ter um conhecimento geral sobre testes de software, com familiaridade com os tipos de testes mais básicos e amplamente utilizados. Porém, o conhecimento sobre algumas práticas mais atuais como *Test Smells* e *Flaky Tests*, e o uso de algumas ferramentas ainda são menos difundidas.

7 *SURVEY* INVESTIGANDO PRÁTICAS MODERNAS DE TESTE DE SOFTWARE NA INDÚSTRIA

Nesta seção, será apresentado o segundo questionário aplicado a profissionais atuantes na indústria de testes de software. O objetivo é explorar detalhadamente as práticas modernas utilizadas, metodologias adotadas, ferramentas empregadas e os desafios enfrentados no dia a dia desses profissionais.

7.1 Desenvolvimento e Estruturação do Questionário

Assim como no primeiro *survey*, este questionário também foi criado tendo por base as pesquisas realizadas por Paschoal e Souza (2018), Melo *et al.* (2020) e Martins *et al.* (2021), foi possível desenvolver um questionário detalhado com o objetivo de investigar as práticas modernas de testes de software na indústria. Este questionário foi dividido em seções específicas para obter informações abrangentes sobre as práticas e ferramentas utilizadas pelos profissionais de testes de software na indústria atual.

O questionário começa com um Termo de Consentimento (ver Apêndice B), onde são estabelecidos os parâmetros e expectativas para a participação na pesquisa. Este termo garante aos participantes que suas respostas serão tratadas com confidencialidade e que sua identidade será mantida em sigilo. Ao aceitar o termo, os participantes indicam sua compreensão e concordância com os termos estabelecidos, demonstrando seu consentimento informado e voluntário para contribuir com o estudo.

A primeira seção do questionário é dedicada à Identificação. Nesta seção, solicitamos algumas informações básicas sobre o participante, como nome (opcional), estado de residência, curso de graduação e tempo de experiência na área de testes de software. Essas informações são fundamentais para contextualizar as respostas e compreender o perfil dos profissionais que participam da pesquisa.

A segunda seção, que aborda sobre a experiência na área de Testes dos participantes, buscamos entender as práticas e metodologias utilizadas pelos profissionais em suas organizações. Perguntamos sobre as metodologias seguidas (como *Agile*, *DevOps*, *Scrum*, etc.), as linguagens de programação mais utilizadas, as ferramentas de gestão de testes empregadas e os diferentes tipos de testes executados. Esta seção permite mapear as práticas correntes e as ferramentas preferidas pelos profissionais da área.

A seção seguinte, fala sobre Técnicas de Teste foca nas abordagens específicas

adotadas pelos profissionais para conduzir os testes. Questionamos sobre a utilização de técnicas de teste estrutural (caixa-branca) e funcional (caixa-preta), bem como os tipos de testes estruturais e funcionais que os participantes já executaram. Esta seção é crucial para entender a profundidade e a variedade das técnicas aplicadas no dia a dia dos profissionais.

A próxima seção é sobre Testes Automatizados, que busca identificar quais tipos de testes automatizados são realizados pelos participantes. As opções incluem uma ampla gama de testes, como testes de cobertura de código, testes de usabilidade, testes de carga, entre outros. Esta seção ajuda a capturar a extensão da automação de testes nas práticas da indústria.

Seguindo, a seção sobre *Test Smells* investiga a familiaridade dos participantes com conceitos de problemas em testes de software, como *Mystery Guest*, *Resource Optimism*, e outros. Da mesma forma, a seção sobre *Flaky Tests* explora a experiência dos participantes com testes automatizados instáveis, que podem ocasionalmente falhar ou passar sem razão aparente. Estas seções fornecem percepções sobre os desafios e as nuances enfrentados pelos profissionais na prática de testes de software.

Para a seção sobre Ferramentas e *Frameworks* de Teste, perguntamos aos participantes sobre as ferramentas e frameworks que utilizam para executar testes, como *Cypress*, *JUnit*, *Selenium*, entre outros. Esta seção oferece uma visão das preferências tecnológicas e das ferramentas de suporte utilizadas na indústria.

Para concluir, a seção de Sugestões e Opiniões abre espaço para que os participantes compartilhem suas principais dificuldades encontradas no dia a dia como profissionais de testes e o que consideram essencial para qualquer pessoa que deseje se tornar um bom profissional na área de testes de software. Este *feedback* é valioso para entender os desafios enfrentados e identificar áreas potenciais para melhorias na formação e nas práticas da indústria.

Em resumo, o questionário foi elaborado para abranger uma variedade de tópicos relevantes ao teste de software na indústria, desde informações básicas sobre os profissionais até detalhes específicos sobre práticas e ferramentas utilizadas. No Quadro 8 é apresentado o *survey* de forma resumida.

7.2 Teste Piloto e Divulgação do *Survey*

Um teste piloto foi conduzido com o objetivo de identificar possíveis falhas no questionário que pudessem comprometer sua clareza e entendimento pelos participantes. Dois voluntários participaram do teste, e nenhum erro foi relatado. O questionário pode ser consultado

Quadro 8 – Questões do *Survey*.

ID	Questão
Identificação	
Q01	Nome
Q02	Estado
Q03	Informe o curso de graduação.
Q04	Há quanto tempo você atua na área de Testes?
Experiência na área de testes	
Q05	Na organização onde você trabalha é seguida alguma metodologia?
Q06	Quais linguagens de programação são mais frequentemente usadas por você?
Q07	Você utiliza alguma ferramenta de gestão de testes para acompanhar o progresso e relatar resultados?
Q08	Abaixo estão alguns testes que podem ser executados em diferentes níveis durante a fase de desenvolvimento. Marque aqueles que você já executou.
Q09	Técnicas de teste representam abordagens ou métodos específicos para conduzir os testes. Marque abaixo as técnicas que você já utilizou.
Q10	Ainda falando sobre técnicas de testes. Quais testes estruturais (ou teste de caixa-branca) você já executou?
Q11	Ainda falando sobre técnicas de testes. Quais testes funcionais (ou teste de caixa-preta) você já executou?
Q12	Sobre testes automatizados, quais dos seguintes testes você já executou?
Q13	Sobre Test *Smells*, quais dos seguintes testes você já executou?
Q14	Sobre *Flaky Tests*, quais dos seguintes testes você já executou?
Q15	Você utiliza alguma ferramenta e/ou *framework* para executar testes?
Sugestões e opiniões	
Q16	Quais principais dificuldades você encontra no seu dia a dia como um profissional de testes?
Q17	O que você diria que é uma obrigação para qualquer pessoa aprender para se tornar um bom profissional de testes de software? Qual o essencial?

Fonte: Elaborado pela autora

no Apêndice B deste trabalho.

Após a realização e aprovação do teste piloto, o *survey* foi divulgado com o objetivo de atingir profissionais de tecnologia que atuam como testadores. A divulgação ocorreu por meio de redes sociais como *LinkedIn* e *WhatsApp*, utilizando postagens e compartilhamentos em grupos específicos para alcançar o público-alvo.

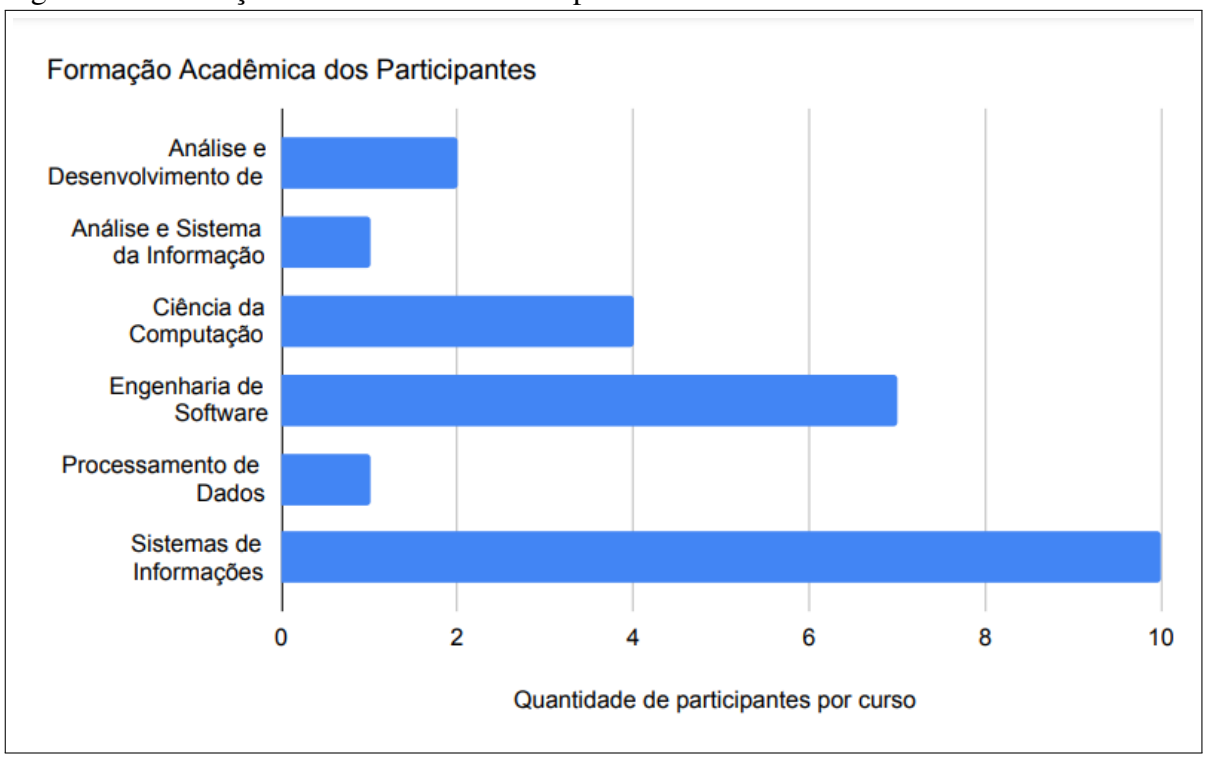
7.3 Perfil dos Participantes

O questionário recebeu um total de 25 respostas válidas, abrangendo participantes de cinco estados brasileiros (**Q02**). A maioria das respostas veio do Ceará (CE), representando 68% (17) dos participantes. Em seguida, Pernambuco (PE) contribuiu com 12% (3) das respostas, enquanto Minas Gerais (MG) participou com 8% (2) dos respondentes. São Paulo (SP) teve 8% (2) dos participantes, e o Distrito Federal (DF) respondeu por 4% (1) das respostas.

A Figura 7 mostra que os cursos mais representados (**Q03**) foram Sistemas de Informação, com 40% (10) dos participantes, seguido por Engenharia de Software, com 28% (7). Em menor número, participaram da pesquisa os cursos de Ciência da Computação, com 16% (4), Análise e Desenvolvimento de Sistemas, com 8% (2), e Análise e Sistemas de Informação e

Processamento de Dados, cada um com 4% (1).

Figura 7 – Formação Acadêmica dos Participantes



Fonte: Elaborado pela autora.

Quanto ao tempo de atuação na área de Testes (**Q04**), a maior parte dos participantes tem experiência entre 1 e 3 anos, um total de 44% (11) dos participantes. 28% (7) dos participantes possuem mais de 5 anos de experiência, e 24% (6) dos participantes afirmaram ter entre 3 e 5 anos de experiência. Também houve uma resposta de um participante com menos de 1 ano de experiência, representando 4% (1) (Figura 8).

7.4 Experiência Profissional dos Participantes na Área de Teste de Software

7.4.1 Metodologias, Ferramentas e Tecnologias Utilizadas

Analisando os dados apresentados, é possível perceber que os profissionais da área de testes de software adotam uma abordagem diversificada e adaptável em seus processos. Em relação às metodologias (**Q05**), observa-se que as organizações preferem combinar diferentes práticas, destacando-se o uso de métodos ágeis como *Scrum*, *Agile* e *Kanban*. Essa mescla permite uma maior flexibilidade e a personalização dos processos, integrando inclusive práticas como *DevOps* e, em menor escala, metodologias tradicionais como *Waterfall*, o que evidencia a busca por soluções que se ajustem às necessidades específicas de cada projeto.

Figura 8 – Tempo de experiência na área de testes

Fonte: Elaborado pela autora.

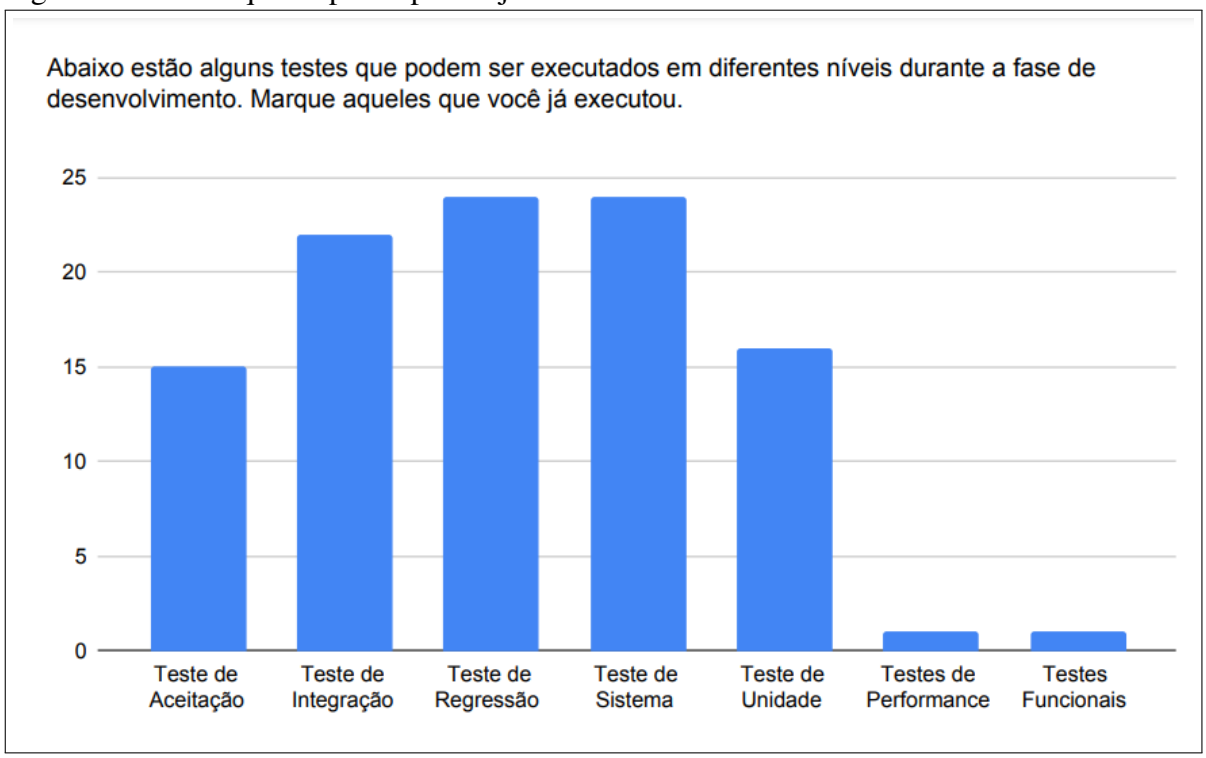
Sobre linguagens de programação (**Q06**), os respondentes demonstram uma tendência a utilizar tecnologias consolidadas, com destaque para *JavaScript* e *Java*, que são amplamente empregadas tanto no desenvolvimento de aplicações web quanto em sistemas corporativos robustos. Ao mesmo tempo, a escolha por trabalhar com múltiplas linguagens por parte de alguns profissionais indica a necessidade de versatilidade e a capacidade de se adaptar a diferentes contextos tecnológicos, enquanto outros optam por se especializar em uma única ferramenta, reforçando a importância do alinhamento com os objetivos e o ecossistema da organização. Quanto às ferramentas de gestão de testes (**Q07**), o cenário aponta para uma predominância no uso de soluções integradas, onde o Jira se destaca como a opção preferencial para o acompanhamento de projetos e a integração de fluxos de trabalho. Ferramentas específicas, como *TestLink* e *TestRail*, também são empregadas, seja de forma isolada ou combinadas, permitindo que as equipes ajustem suas estratégias de gestão conforme as particularidades dos processos de testes.

7.4.2 Experiência dos Participantes com Tipos e Técnicas de Testes

Os dados analisados demonstram que os profissionais possuem uma ampla experiência na execução de diversos tipos de testes (**Q08**), conforme mostra a Figura 9, evidenciando uma prática que vai além da aplicação de um único método. Os participantes já trabalharam com

testes de regressão e de sistema, que se destacam em sua prática, além de integrarem testes de integração, unidade e aceitação. Essa diversidade reflete uma abordagem abrangente, em que nenhum profissional se restringe a uma única forma de validação.

Figura 9 – Testes que os participantes já executaram



Fonte: Elaborado pela autora.

No que diz respeito às técnicas de teste (**Q09**), observa-se uma forte preferência pelos métodos funcionais (caixa-preta), embora haja também um conhecimento relevante sobre as técnicas estruturais (caixa-branca). A familiaridade com ambas as abordagens, presente na maioria dos respondentes, indica uma tendência a combinar diferentes estratégias para aprimorar a eficácia dos testes.

Uma análise mais detalhada dos testes estruturais (**Q10**) revela que os métodos baseados no fluxo de controle e no fluxo de dados são os mais comuns, seguidos pelo teste de caminho. Contudo, as variações na aplicação dessas técnicas sugerem diferentes níveis de especialização entre os profissionais, além de possíveis discrepâncias na percepção sobre a classificação dessas abordagens. Em relação aos testes funcionais (**Q11**), a técnica de análise de valores limite se sobressai, sendo complementada por métodos como a partição de equivalência e o uso de tabelas de decisão. A diversidade nos relatos sobre a execução dessas técnicas pode indicar interpretações distintas quanto à integração dos métodos dentro dos testes funcionais.

Sobre testes automatizados (**Q12**), os testes funcionais continuam sendo os mais

aplicados, seguidos por práticas voltadas para a regressão, interface gráfica, integração e sistema. Embora abordagens como testes baseados em modelos e testes *end-to-end* apareçam com menor frequência, a presença de testes de carga, desempenho e estresse evidencia a diversidade das práticas de automação adotadas. Em relação à quantidade de tipos de testes que cada um já havia executado, apenas 2 dos participantes informaram ter realizado apenas um dos testes apresentados, enquanto os demais indicaram ter executado uma variedade de testes, sugerindo que a maioria dos participantes utiliza, ou já utilizou, uma variedade dos testes automatizados apresentados em sua prática cotidiana.

Quanto aos *test smells* (**Q13**), a maioria dos participantes não trabalha com essa abordagem. Entre os que a utilizam, o *test code duplication* se destaca, enquanto outras práticas, como *lazy test* e *assertion roulette*, aparecem em menor escala, apontando para diferentes níveis de experimentação com essas técnicas. Em relação aos *flaky tests* (**Q14**) também ocorreram variações, já que uma parte significativa dos respondentes não os utiliza. Dentre os que os empregam, técnicas relacionadas a *async wait* e *network* são as mais recorrentes, acompanhadas por outras estratégias que, embora menos frequentes, demonstram o esforço em lidar com inconsistências e instabilidades nos testes.

Por fim, a questão das ferramentas e *frameworks* (**Q15**) mostra que o *Cypress* é amplamente preferido, com *Selenium* e *JUnit* também figurando como opções relevantes, conforme a Figura 10. A utilização de outras ferramentas menos populares reflete a busca por soluções diversificadas que atendam às demandas específicas de cada contexto.

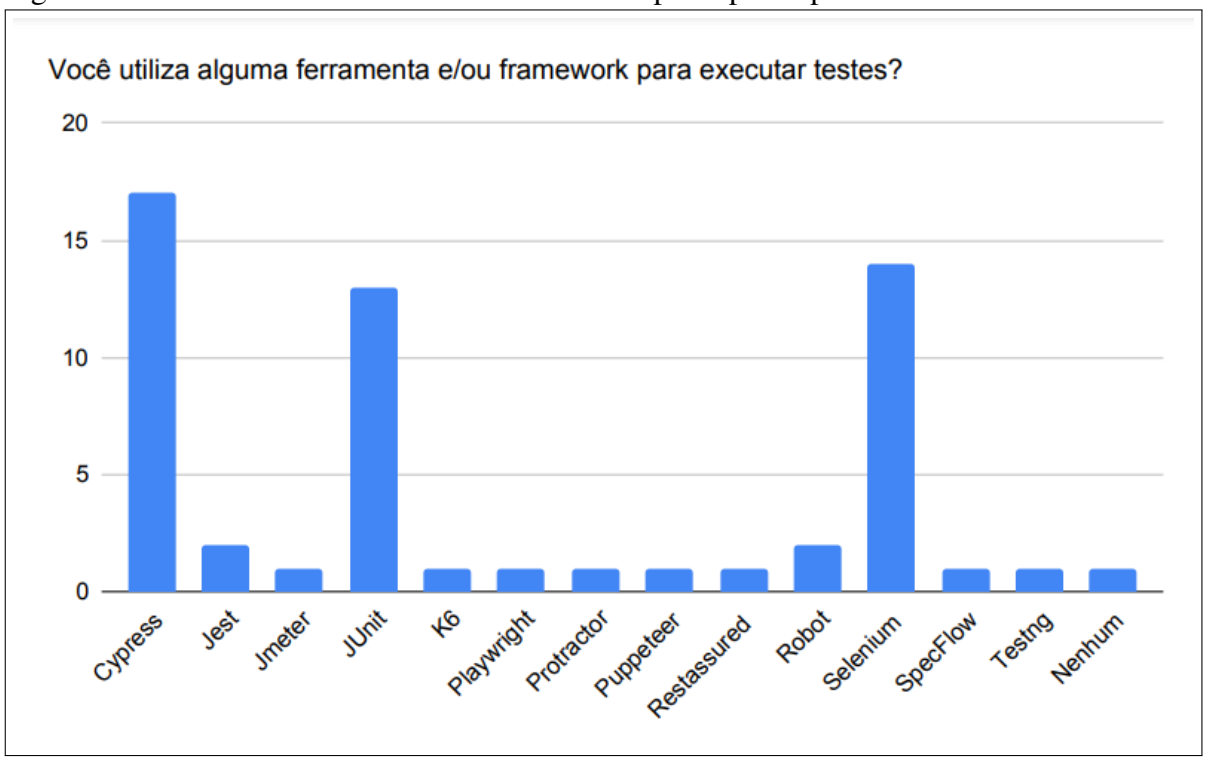
Em resumo, a análise evidencia que os profissionais da área de testes de software adotam uma abordagem diversificada, combinando diferentes tipos de testes, técnicas e ferramentas. Essa prática integrada demonstra um comprometimento com a eficiência e a qualidade, adaptando-se continuamente às demandas de um ambiente de desenvolvimento dinâmico e em constante evolução.

7.5 Sugestões e Opiniões

7.5.1 Dificuldades Enfrentadas por Profissionais de Testes (Q16)

Os participantes da pesquisa destacaram diversas dificuldades encontradas no dia a dia como profissionais de testes, evidenciando desafios relacionados a processos, comunicação, cultura organizacional e aspectos técnicos. Um dos pontos mais recorrentes é a falta de documen-

Figura 10 – Ferramentas e *Frameworks* utilizados pelos participantes



Fonte: Elaborado pela autora.

tação, problemas de comunicação com a equipe, resistência à cultura de qualidade e a pressão de prazos curtos. Essas dificuldades foram classificadas em 4 categorias principais: 1. Problemas com a documentação, 2. Problemas de comunicação, 3. Problemas culturais e 4. Outros.

1. Problemas com a documentação. Documentação incompleta, ausente ou desatualizada é uma dificuldade comum. Muitos profissionais mencionam lacunas na documentação de requisitos, processos e casos de uso. A presença de requisitos incompletos, mudanças não documentadas e a volatilidade das histórias de usuários dificultam o planejamento e a execução dos testes. Os participantes relatam carência de documentação do Sistema, de casos de uso e de regras de negócio, além da falta de empenho em manter as documentações atualizadas. Como podemos observar nos comentários abaixo:

"Falta de organização das empresas, requisitos incompletos, mudanças não documentadas, falta de documentação."

Outro participante comentou ainda:

"Falta de documentação do sistema, dos casos de usos, e das regras de negócio; Falta de interesse em manter as documentações atualizadas; Dificuldade de encontrar tempo pra adicionar/melhorar automação de testes e processos."

2. Problemas de comunicação. Alguns participantes relataram dificuldades na

comunicação dentro da equipe, mencionando que, muitas vezes, é mais desafiador manter uma comunicação eficiente do que lidar com o próprio processo de trabalho. Essa dificuldade seria tanto com membros da equipe quanto com superiores que não exercem a liderança da melhor maneira e não fornecem um direcionamento adequado e especializado. Além disso, enfrentam obstáculos ao discutir aspectos do código e do sistema, o que pode impactar a colaboração e a resolução de problemas. Abaixo seguem algumas respostas relatadas pelos participantes:

"Problemas mais em comunicação com o time, do que no processo em si."

"Comunicação, às vezes é difícil discutir um código quando não se tem um denominação estabelecida das partes do código e do sistema."

3. Problemas culturais. Culturalmente, muitos profissionais enfrentam resistência em empresas onde não há uma cultura consolidada de qualidade de software. Há uma dificuldade em convencer equipes sobre a importância dos testes e a necessidade de sua inclusão em todas as etapas do desenvolvimento é um obstáculo comum. Além disso, há resistência de algumas equipes em aceitar a ajuda do QA em todas as etapas do desenvolvimento demonstra uma falta de integração e compreensão do papel do testador. Abaixo seguem algumas respostas relatadas pelos participantes:

"Conscientizar o time de que os testes são necessários e que devem fazer parte do processo, inclusive dos devs."

"Trabalhar em empresas onde não há cultura de qualidade de software."

"Sem dúvidas é manter e seguir totalmente o processo de QA que decidimos colocar em prática. Quando se trabalha com entregas constantes aos clientes, fica um pouco difícil focar 100% no processo de qualidade. Fazemos o possível para não pular etapas do nosso processo, mas nem sempre é possível. Precisamos sempre nos adaptar às adversidades que surgem mas também tentamos nunca perder o foco no processo. Acaba que precisamos fazer modificações constantes para que o processo funcione da forma mais eficiente e ágil possível, para que seja possível realizar as entregas sem perder a qualidade."

4. Outros. Outras dificuldades, embora menos recorrentes, foram citadas pelos participantes, como a falta de reconhecimento do trabalho dos profissionais de testes que pode ser desmotivadora e afetar a qualidade do trabalho. Bem como a dificuldade em implementar a integração contínua e pipelines com testes automatizados. Alguns comentários dos participantes abaixo:

"Falta de reconhecimento."

"Ter integração contínua, pipelines com os testes automatizados."

7.5.2 Habilidades Essenciais Para um Bom Profissional de Testes de Software (Q17)

Os participantes identificaram diversas habilidades e conhecimentos essenciais para qualquer pessoa que deseje se tornar um bom profissional de testes de software. Um ponto amplamente destacado foi a necessidade de embasamento teórico sólido, que permita ao QA tomar decisões fundamentadas e se posicionar de maneira profissional, evitando "achismos" sobre o que testar e como testar.

"Princípio de testes (além de criar uma base forte de testes), princípio de automação de testes (tanto web, api e mobile)."

"Os conceitos por trás de cada tipo de teste."

Também houveram muitos comentários sobre princípios fundamentais como os descritos no *Syllabus* do ISTQB, sendo citados como um guia indispensável para aprender os fundamentos de testes, ciclo de vida do desenvolvimento de software, técnicas de teste, gerenciamento de testes, entre outros.

"Estude o Syllabus, não adianta nada automatizar, sem saber os princípios."

"Praticamente tudo o que está no Syllabus do ISTQB. Ele é um guia de estudos para quem quer tirar a certificação CTFL, mas também contém basicamente tudo o que um QA/Tester iniciante precisa saber para trabalhar na área."

A capacidade de adaptação às mudanças nos processos das empresas e a constante evolução no aprendizado são vistas como essenciais. A curiosidade, a proatividade para estudar negócios e técnicas de desenvolvimento de software, e a arte de questionar também foram mencionadas como qualidades fundamentais para um bom profissional de testes. Além disso, habilidades interpessoais, como boa comunicação, paciência, e a habilidade de lidar com pessoas, foram destacadas como igualmente importantes.

"A arte de questionar torna o trabalho de teste essencial para um bom profissional."

"Aprender a se comunicar, e ser adaptável."

"Curiosidade, estudar técnicas, pró ativo para estudar negócios e técnicas de desenvolvimento de software."

8 ANÁLISE COMPARATIVA DOS RESULTADOS DE CADA *SURVEY*

Este capítulo apresenta uma análise comparativa dos resultados obtidos a partir dos dois *surveys* executados: o primeiro, voltado para o ensino de testes de software na graduação, e o segundo, direcionado às práticas modernas adotadas na indústria. A análise integra os dados dos Capítulos 6 e 7 com as observações previamente elaboradas, evidenciando as principais discrepâncias e pontos de convergência entre os ambientes acadêmico e corporativo.

De forma geral, o *survey* aplicado aos acadêmicos revelou que 87,5% dos participantes reconhecem a necessidade de melhorias no ensino de testes de software. Os resultados apontam que o conteúdo ministrado durante a graduação é predominantemente teórico, o que tem sido identificado como uma barreira para a assimilação prática dos conceitos. A ausência de disciplinas dedicadas exclusivamente a testes de software e a escassez de atividades práticas e uso de ferramentas modernas contribuem para que os estudantes não desenvolvam, de maneira satisfatória, competências essenciais para o mercado. Além disso, apesar dos alunos estarem familiarizados com os fundamentos e técnicas básicas, uma grande parte nunca teve contato com práticas mais avançadas, como *Test Smells* e *Flaky Tests*, e com ferramentas de gerenciamento de testes – aspectos considerados cruciais para a formação completa de um profissional da área.

Em contrapartida, o *survey* realizado com profissionais da indústria demonstrou um cenário de maior maturidade prática. Os profissionais apresentam conhecimento consolidado em diversas técnicas de teste, desde os níveis básicos (como testes de unidade, integração e sistema) até metodologias avançadas, evidenciando uma aplicação efetiva das práticas de testes. Observa-se que 100% dos profissionais utilizam testes automatizados e ferramentas de gestão, o que reforça a importância da prática e da atualização contínua na rotina de trabalho. Além disso, conceitos modernos, como *Test Smells* e *Flaky Tests*, têm sido mais explorados na indústria – com taxas de adoção de 40% e 60%, respectivamente – demonstrando que o ambiente corporativo exige um domínio mais profundo e prático dos métodos de teste.

A comparação entre os dois *surveys* ressalta uma discrepância importante: enquanto os acadêmicos estão inseridos num ambiente predominantemente teórico, com 81,3% de conhecimento das técnicas básicas, os profissionais da indústria indicam taxas superiores (92% de conhecimento das técnicas e 100% de adoção de testes automatizados). Esse contraste é ainda mais evidente quando se analisam os testes estruturais e funcionais, onde os profissionais apresentam percentuais mais elevados (72% e 76%, respectivamente) em comparação com os acadêmicos (59,4% e 65,6%, respectivamente). Outro ponto crítico é o uso de ferramentas de

gerenciamento de testes, que, na indústria, é adotado por todos os participantes, enquanto na academia apenas 31,2% tiveram contato com tais ferramentas.

Tais diferenças evidenciam a necessidade urgente de uma reavaliação dos currículos acadêmicos em cursos de computação. É necessário que as instituições de ensino passem a incorporar atividades práticas, laboratórios, projetos integrados e, sobretudo, disciplinas específicas dedicadas aos testes de software. Essas medidas possibilitariam uma aproximação mais realista do ambiente de trabalho, preparando os estudantes para os desafios do mercado, onde a aplicação prática e a familiaridade com ferramentas modernas são essenciais.

Além disso, a análise destaca a importância da conscientização sobre o papel fundamental dos testes de software na qualidade dos sistemas. Tanto na academia quanto na indústria, a valorização da área de QA e a busca contínua por atualização – por meio de cursos, certificações e treinamentos – são elementos cruciais para a formação de profissionais competentes e para a melhoria dos processos de desenvolvimento de software. O (Quadro 9) destaca as principais diferenças nas questões mais relevantes.

Quadro 9 – Análise comparativa dos resultados dos *surveys*

Questões	**Academia**	**Indústria**
Níveis de Teste	96,9%	100%
Técnicas de Teste	81,3%	92%
Testes Estruturais	59,4%	72%
Testes Funcionais	65,6%	76%
Testes Automatizados	84,4%	100%
Test Smells	37,5%	40%
Flaky Tests	15,6%	60%
Ferramentas e/ou *Frameworks*	81,3%	96%
Ferramentas de Gerenciamento	31,2%	100%

Fonte: Elaborado pelo autora.

Em resumo, a comparação entre o ensino acadêmico e as práticas da indústria revela uma clara divergência entre a teoria oferecida nas universidades e as exigências práticas do mercado. Este cenário reforça a necessidade de uma integração mais estreita entre a academia e a indústria, permitindo a construção de um ambiente educacional dinâmico, capaz de alinhar os conteúdos teóricos com as demandas reais do setor. Tal alinhamento não só contribuirá para a formação de profissionais mais preparados, mas também para a elevação da qualidade dos sistemas desenvolvidos e a redução de custos decorrentes de falhas em produção.

9 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho investigou o uso de práticas modernas de testes de software tanto na academia quanto na indústria, objetivando identificar lacunas, desafios e oportunidades de integração entre os dois contextos. A pesquisa foi conduzida em três etapas complementares: a análise das ementas das disciplinas de teste de software nos cursos de Engenharia de Software, a aplicação de um *survey* com estudantes de graduação e, finalmente, um *survey* com profissionais atuantes na área. A partir desses estudos, foi possível estabelecer uma comparação aprofundada entre o ensino acadêmico e as práticas empregadas no mercado.

Os principais achados apontam para um descompasso relevante entre o que é ensinado na graduação e as demandas reais do mercado de trabalho. Enquanto o ensino de teste de software nas instituições de ensino evidencia uma ênfase predominante em abordagens teóricas – com pouca integração de práticas modernas e ferramentas atualizadas – os profissionais da indústria demonstram um domínio sólido de técnicas avançadas, como automação de testes, uso de *frameworks*, *Test Smells* e *Flaky Tests*. Além disso, enquanto a maioria dos acadêmicos não tem contato com ferramentas de gerenciamento de testes, 100% dos profissionais relataram utilizá-las em sua rotina.

Essas discrepâncias refletem não apenas uma lacuna no currículo das universidades, mas também evidenciam a necessidade de uma aproximação maior entre o ambiente acadêmico e o mercado. A carência de disciplinas específicas sobre testes de software e a insuficiência de atividades práticas impedem que os estudantes desenvolvam, de forma eficaz, as competências exigidas pelas organizações. Em contrapartida, os profissionais buscam continuamente atualizar seus conhecimentos por meio de cursos extracurriculares, certificações e treinamentos, o que reforça a importância da prática na consolidação dos conceitos de teste.

Diante desses resultados, recomenda-se que as instituições de ensino reavaliem seus currículos, incorporando atividades práticas, laboratórios, projetos interdisciplinares e, especialmente, disciplinas dedicadas exclusivamente aos testes de software. A promoção de parcerias com a indústria pode ser uma estratégia valiosa para aproximar os conteúdos acadêmicos das demandas reais do mercado, contribuindo para a formação de profissionais mais preparados e alinhados com as práticas modernas.

Diante desses achados, este estudo espera contribuir para o debate e melhoria contínua do ensino de testes de software e na preparação dos futuros profissionais para os desafios do mercado. Espera-se que os resultados apresentados possam estimular mudanças na forma como

os testes são ensinados, incentivando uma abordagem mais equilibrada entre teoria e prática, além de um maior alinhamento com as necessidades do setor de tecnologia.

9.1 Ameaças à Validade

Limitação da Amostra ao Curso de Engenharia de Software A restrição da amostra ao curso de Engenharia de Software representa uma ameaça à validade dos resultados, pois pode comprometer a generalização dos achados para outros cursos de Tecnologia da Informação. As abordagens curriculares e metodológicas podem variar significativamente entre os diferentes cursos, limitando a abrangência dos dados e a representatividade dos desafios enfrentados no ensino de testes de software.

Para mitigar essa limitação, recomenda-se a ampliação da amostra em estudos futuros, incluindo outros cursos da área da computação. Adicionalmente, propõe-se realizar uma análise comparativa curricular e a integração de múltiplas fontes de dados (professores, alunos e profissionais da indústria), a fim de fortalecer a validade externa dos resultados e oferecer uma visão mais abrangente do cenário educacional.

Exclusão da Perspectiva dos Estudantes dos Cursos Selecionados A não inclusão das percepções dos respectivos estudantes dos cursos selecionados para análise das ementas constitui outra ameaça, uma vez que esses alunos seriam fundamentais para compreender a eficácia e os desafios das práticas de ensino de testes de software em cada um desses cursos analisados. Essa exclusão pode levar a uma análise não tão eficaz, deixando de captar *insights*.

Como estratégia de mitigação, recomenda-se a inclusão de uma amostra representativa também dos alunos de cada curso analisado em pesquisas futuras, por meio de *surveys*. A triangulação dos dados, combinando as opiniões de alunos, professores e profissionais da indústria, permitirá uma avaliação mais robusta e abrangente, contribuindo para o aprimoramento contínuo dos processos de ensino e para o alinhamento com as demandas do mercado.

Influência do Tempo Decorrido Desde a Graduação dos Participantes do *Survey* da Academia Uma outra ameaça à validade dos resultados decorre da presença de participantes no *survey* que já haviam concluído a graduação há um tempo significativo. Isso é relevante porque os conhecimentos relatados na pesquisa podem ter sido adquiridos após a conclusão da graduação, e não durante a disciplina voltada para teste de software. Como resultado, os dados obtidos podem refletir o nível de conhecimento atual dos participantes, que pode ser superior ao nível de conhecimento que possuíam no momento em que cursaram a disciplina, levando a

uma distorção nos resultados. A análise pode indicar que o nível de conhecimento dos alunos era menor do que o demonstrado na pesquisa, já que as percepções podem estar mais alinhadas com as habilidades adquiridas após a formação acadêmica.

Para mitigar essa ameaça, sugere-se a inclusão de critérios de seleção mais rigorosos para os participantes do *survey*, garantindo que aqueles que participaram da pesquisa estejam, de fato, em uma posição que reflita o conhecimento adquirido durante a graduação. Uma possível abordagem seria segmentar os dados com base no tempo desde a graduação, permitindo uma análise mais precisa das informações.

9.2 Trabalhos Futuros

Esta pesquisa apresentou uma análise detalhada das práticas de teste de software, tanto no âmbito acadêmico quanto na indústria, evidenciando as semelhanças, diferenças e desafios enfrentados em cada contexto. Contudo, há várias áreas que podem ser aprofundadas para expandir o conhecimento sobre o estado atual da educação e prática em testes de software, além de explorar maneiras de melhorar esses processos no futuro.

Replicação do estudo em maior escala: Uma das principais direções para trabalhos futuros envolve a replicação deste estudo em uma escala mais ampla. A ampliação da amostra de participantes é crucial para garantir que os resultados reflitam uma realidade mais diversificada e representativa, considerando a variação de contextos acadêmicos e industriais. A inclusão de profissionais de diferentes setores da indústria de software, além de acadêmicos de universidades com diferentes perfis e abordagens, pode fornecer uma visão mais completa sobre os avanços e desafios enfrentados na implementação de práticas de testes de software. A coleta de dados em diferentes regiões geográficas e com grupos etários distintos também pode ajudar a identificar variações no ensino e aplicação dos testes, contribuindo para uma melhor compreensão do panorama global da área.

Metodologias de ensino: Um aspecto fundamental a ser explorado em pesquisas futuras refere-se às metodologias de ensino utilizadas no treinamento de testes de software. O estudo atual indicou que muitos estudantes enfrentam dificuldades, principalmente devido à ênfase excessiva em abordagens teóricas e à carência de ferramentas práticas para aplicação dos conceitos aprendidos. Portanto, uma linha de pesquisa importante seria investigar metodologias de ensino que integrem de maneira mais eficaz teoria e prática.

REFERÊNCIAS

ARCURI, A. An experience report on applying software testing academic results in industry: we need usable automated test generation. **Empirical Software Engineering**, Springer, v. 23, p. 1959–1981, 2018.

Association for Computing Machinery. **ACM Computing Curricula 2020**. 2020. Disponível em: https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf. Acesso em: 18 abr. 2023.

BARRETT, A. A.; ENOIU, E. P.; AFZAL, W. On the current state of academic software testing education in sweden. In: IEEE. **2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. [*S. l.*], 2023. p. 397–404.

CAMMAERTS, F.; TRAMONTANA, P.; PAIVA, A. C.; FLORES, N.; RICÓS, F. P.; SNOECK, M. Exploring students' opinion on software testing courses. In: **Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering**. [*S. l.*: *s. n.*], 2024. p. 570–579.

CARVER, J. C.; KRAFT, N. A. Evaluating the testing ability of senior-level computer science students. In: IEEE. **2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T)**. [*S. l.*], 2011. p. 169–178.

CHEIRAN, J. F. P.; RODRIGUES, E. de M.; CARVALHO, E. L. de S.; SILVA, J. P. S. da. Problem-based learning to align theory and practice in software testing teaching. In: **Proceedings of the XXXI Brazilian Symposium on Software Engineering**. [*S. l.*: *s. n.*], 2017. p. 328–337.

CHEN, N. Gate: game-based testing environment. In: **Proceedings of the 33rd international Conference on Software Engineering**. [*S. l.*: *s. n.*], 2011. p. 1078–1081.

CLARKE, P. J.; DAVIS, D. L.; CHANG-LAU, R.; KING, T. M. Impact of using tools in an undergraduate software testing course supported by wrestt. **ACM Transactions on Computing Education (TOCE)**, ACM New York, NY, USA, v. 17, n. 4, p. 1–28, 2017.

CORTE, C. K. D.; BARBOSA, E. F.; MALDONADO, J. C. **Ensino integrado de fundamentos de programação e teste de software**. Tese (Doutorado), 2006.

DEURSEN, A. V.; MOONEN, L.; BERGH, A. V. D.; KOK, G. Refactoring test code. In: CITESEER. **Proceedings of the 2nd international conference on extreme programming and flexible processes in software engineering (XP2001)**. [*S. l.*], 2001. p. 92–95.

DUSTIN, E.; RASHKA, J.; PAUL, J. **Automated software testing**: Introduction, management, and performance: Introduction, management, and performance. [*S. l.*]: Addison-Wesley Professional, 1999.

Fowler, Martin. **Eradicating Non-Determinism in Tests.(2011)**. 2011. Disponível em: https://martinfowler.com/articles/nonDeterminism.html. Acesso em: 9 jun. 2023.

FULCINI, T.; COPPOLA, R.; ARDITO, L.; TORCHIANO, M. A review on tools, mechanics, benefits, and challenges of gamified software testing. **ACM Computing Surveys**, ACM New York, NY, 2023.

GAMIDO, H. V.; GAMIDO, M. V. Comparative review of the features of automated software testing tools. **International Journal of Electrical and Computer Engineering**, IAES Institute of Advanced Engineering and Science, v. 9, n. 5, p. 4473, 2019.

GRAHAM, D.; BLACK, R.; VEENENDAAL, E. V. **Foundations of software testing ISTQB Certification**. [*S. l.*]: Cengage Learning, 2021.

HABCHI, S.; HABEN, G.; PAPADAKIS, M.; CORDY, M.; TRAON, Y. L. A qualitative study on the sources, impacts, and mitigation strategies of flaky tests. In: IEEE. **2022 IEEE Conference on Software Testing, Verification and Validation (ICST)**. [*S. l.*], 2022. p. 244–255.

HEDAYATI, A.; EBRAHIMZADEH, M.; SORI, A. A. Investigating into automated test patterns in erratic tests by considering complex objects. **International Journal of Information Technology and Computer Science (IJITCS)**, v. 7, n. 3, p. 54, 2015.

HYNNINEN, T.; KASURINEN, J.; KNUTAS, A.; TAIPALE, O. Software testing: Survey of the industry practices. In: IEEE. **2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. [*S. l.*], 2018. p. 1449–1454.

IRAWAN, Y.; MUZID, S.; SUSANTI, N.; SETIAWAN, R. System testing using black box testing equivalence partitioning (case study at garbage bank management information system on karya sentosa). In: **The 1st International Conference on Computer Science and Engineering Technology Universitas Muria Kudus**. [*S. l.*: *s. n.*], 2018.

JAMIL, M. A.; ARIF, M.; ABUBAKAR, N. S. A.; AHMAD, A. Software testing techniques: A literature review. In: IEEE. **2016 6th international conference on information and communication technology for the Muslim world (ICT4M)**. [*S. l.*], 2016. p. 177–182.

JUNIOR, N. S.; MARTINS, L.; ROCHA, L.; COSTA, H.; MACHADO, I. How are test smells treated in the wild? a tale of two empirical studies. **Journal of Software Engineering Research and Development**, v. 9, p. 9–1, 2021.

LAM, W.; GODEFROID, P.; NATH, S.; SANTHIAR, A.; THUMMALAPENTA, S. Root causing flaky tests in a large-scale industrial setting. In: **Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis**. [*S. l.*: *s. n.*], 2019. p. 101–111.

LEMOS, O. A. L.; SILVEIRA, F. F.; FERRARI, F. C.; GARCIA, A. The impact of software testing education on code reliability: An empirical assessment. **Journal of Systems and Software**, Elsevier, v. 137, p. 497–511, 2018.

LUO, Q.; HARIRI, F.; ELOUSSI, L.; MARINOV, D. An empirical analysis of flaky tests. In: **Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering**. [*S. l.*: *s. n.*], 2014. p. 643–653.

MARTINS, L.; BRITO, V.; FEITOSA, D.; ROCHA, L.; COSTA, H.; MACHADO, I. From blackboard to the office: A look into how practitioners perceive software testing education. In: **Evaluation and Assessment in Software Engineering**. [*S. l.*: *s. n.*], 2021. p. 211–220.

MARTINS, L.; CAMPOS, D.; SANTANA, R.; JUNIOR, J. M.; COSTA, H.; MACHADO, I. Hearing the voice of experts: Unveiling stack exchange communities' knowledge of test smells. In: IEEE. **2023 IEEE/ACM 16th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)**. [*S. l.*], 2023. p. 80–91.

MCMANUS, J. W.; COSTELLO, P. J. Project based learning in computer science: a student and research advisor's perspective. **Journal of Computing Sciences in Colleges**, Consortium for Computing Sciences in Colleges, v. 34, n. 3, p. 38–46, 2019.

MELO, S. M.; MOREIRA, V. X.; PASCHOAL, L. N.; SOUZA, S. R. Testing education: a survey on a global scale. In: **Proceedings of the XXXIV Brazilian Symposium on Software Engineering**. [*S. l.*: *s. n.*], 2020. p. 554–563.

Ministério da Educação. **Ministério da Educação - e-MEC**. 2023. Disponível em: https://emec.mec.gov.br/emec/nova. Acesso em: 12 out. 2023.

MUNEER, I. Systematic review on automated testing (types, effort and roi). 2014.

MYERS, G. J.; SANDLER, C.; BADGETT, T. **The art of software testing**. [*S. l.*]: John Wiley & Sons, 2011.

PALOMBA, F.; ZAIDMAN, A. Notice of retraction: Does refactoring of test smells induce fixing flaky tests? In: IEEE. **2017 IEEE international conference on software maintenance and evolution (ICSME)**. [*S. l.*], 2017. p. 1–12.

PALOMBA, F.; ZAIDMAN, A.; LUCIA, A. D. Automatic test smell detection using information retrieval techniques. In: IEEE. **2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. [*S. l.*], 2018. p. 311–322.

PARRY, O.; KAPFHAMMER, G. M.; HILTON, M.; MCMINN, P. A survey of flaky tests. **ACM Transactions on Software Engineering and Methodology (TOSEM)**, ACM New York, NY, v. 31, n. 1, p. 1–74, 2021.

PASCHOAL, L. N.; SOUZA, S. d. R. S. de. A survey on software testing education in brazil. In: **Proceedings of the 17th brazilian symposium on software quality**. [*S. l.*: *s. n.*], 2018. p. 334–343.

PRESSMAN, R. S. **Software engineering**: a practitioner's approach. [*S. l.*]: Palgrave macmillan, 2005.

QIN, Y.; WANG, S.; LIU, K.; MAO, X.; BISSYANDÉ, T. F. On the impact of flaky tests in automated program repair. In: IEEE. **2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)**. [*S. l.*], 2021. p. 295–306.

SANTANA, R.; MARTINS, L.; ROCHA, L.; VIRGÍNIO, T.; CRUZ, A.; COSTA, H.; MACHADO, I. Raide: a tool for assertion roulette and duplicate assert identification and refactoring. In: **Proceedings of the XXXIV Brazilian Symposium on Software Engineering**. [*S. l.*: *s. n.*], 2020. p. 374–379.

SANTOS, H. M. dos; DURELLI, V. H.; SOUZA, M.; FIGUEIREDO, E.; SILVA, L. T. da; DURELLI, R. S. Cleangame: Gamifying the identification of code smells. In: **Proceedings of the XXXIII Brazilian Symposium on Software Engineering**. [*S. l.*: *s. n.*], 2019. p. 437–446.

SHEORAN, S.; MITTAL, N.; GELBUKH, A. Artificial bee colony algorithm in data flow testing for optimal test suite generation. **International Journal of System Assurance Engineering and Management**, Springer, v. 11, p. 340–349, 2020.

SMITH, J.; TESSLER, J.; KRAMER, E.; LIN, C. Using peer review to teach software testing. In: **Proceedings of the ninth annual international conference on International computing education research**. [*S. l.*: *s. n.*], 2012. p. 93–98.

SNEHA, K.; MALLE, G. M. Research on software testing techniques and software automation testing tools. In: IEEE. **2017 international conference on energy, communication, data analytics and soft computing (ICECDS)**. [*S. l.*], 2017. p. 77–81.

SOARES, E.; RIBEIRO, M.; AMARAL, G.; GHEYI, R.; FERNANDES, L.; GARCIA, A.; FONSECA, B.; SANTOS, A. Refactoring test smells: A perspective from open-source developers. In: **Proceedings of the 5th Brazilian Symposium on Systematic and Automated Software Testing**. [*S. l.*: *s. n.*], 2020. p. 50–59.

SOARES, E.; RIBEIRO, M.; GHEYI, R.; AMARAL, G.; SANTOS, A. Refactoring test smells with junit 5: Why should developers keep up-to-date? **IEEE Transactions on Software Engineering**, IEEE, v. 49, n. 3, p. 1152–1170, 2022.

SOMMERVILLE, I. **Sommerville**: Software engineering. [*S. l.*]: Pearson Studium, 2007.

SOMMERVILLE, I. **Engenharia de software**. Pearson Prentice Hall, 2011. ISBN 9788579361081. Disponível em: https://books.google.com.br/books?id=H4u5ygAACAAJ.

SPADINI, D.; PALOMBA, F.; ZAIDMAN, A.; BRUNTINK, M.; BACCHELLI, A. On the relation of test smells to software code quality. In: IEEE. **2018 IEEE international conference on software maintenance and evolution (ICSME)**. [*S. l.*], 2018. p. 1–12.

SPADINI, D.; SCHVARCBACHER, M.; OPRESCU, A.-M.; BRUNTINK, M.; BACCHELLI, A. Investigating severity thresholds for test smells. In: **Proceedings of the 17th International Conference on Mining Software Repositories**. [*S. l.*: *s. n.*], 2020. p. 311–321.

UMAR, M. A.; ZHANFANG, C. A study of automated software testing: Automation tools and frameworks. **International Journal of Computer Science Engineering (IJCSE)**, v. 6, p. 217–225, 2019.

VALENTE, M. T. Engenharia de software moderna. **Princípios e Práticas para Desenvolvimento de Software com Produtividade**, v. 1, p. 24, 2020.

VALLE, P. H. D.; BARBOSA, E. F.; MALDONADO, J. C. Cs curricula of the most relevant universities in brazil and abroad: Perspective of software testing education. In: IEEE. **2015 International Symposium on Computers in Education (SIIE)**. [*S. l.*], 2015. p. 62–68.

VERMA, A.; KHATANA, A.; CHAUDHARY, S. A comparative study of black box testing and white box testing. **International Journal of Computer Sciences and Engineering**, v. 5, n. 12, p. 301–304, 2017.

VIRGÍNIO, T.; MARTINS, L. A.; SOARES, L. R.; SANTANA, R.; COSTA, H.; MACHADO, I. An empirical study of automatically-generated tests from the perspective of test smells. In: **Proceedings of the XXXIV Brazilian Symposium on Software Engineering**. [*S. l.*: *s. n.*], 2020. p. 92–96.

APÊNDICE A – *SURVEY* - INVESTIGANDO O ENSINO DE PRÁTICAS MODERNAS NA DISCIPLINA DE TESTES DE SOFTWARE

Q1. Termo de consentimento

Caro(a) Participante,

Esse questionário tem como objetivo identificar o conteúdo e práticas modernas na disciplina de testes de software. Este questionário faz parte do Trabalho de Conclusão de Curso (TCC) da Universidade Federal do Ceará - Campus Quixadá do curso de Sistemas de Informação. Suas respostas são valiosas e ajudarão a fornecer *insights* cruciais para a pesquisa. Por favor, responda de maneira honesta e reflexiva, baseando-se em sua experiência acadêmica. Suas informações serão tratadas com confidencialidade e serão utilizadas estritamente para fins acadêmicos. Ao responder a este questionário, você está consentindo com a participação na pesquisa. Suas respostas serão anônimas e não serão associadas a sua identidade pessoal.

Termo de Consentimento: Eu li e compreendi as informações fornecidas acima. Ao continuar e responder a este formulário, concordo em participar voluntariamente desta pesquisa. Entendo que minhas respostas serão usadas apenas para fins de pesquisa e que minha identidade será mantida em sigilo.

Em caso de dúvidas entre em contato pelo o e-mail: lizandra27@alu.ufc.br

1. Identificação

Nesta seção, solicitaremos algumas informações para entender um pouco mais sobre você, estudante que está cursando graduação na área da Computação ou já formado. Suas respostas são fundamentais para a nossa análise e nos ajudarão a contextualizar suas percepções e experiências no decorrer da graduação com a disciplina de teste de software.

Q2. Nome (opcional)

_____

Q3. Estado

( ) 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

Questão 4. Informe seu curso de graduação

( ) Ciência da Computação

( ) Design Digital

( ) Engenharia da Computação

( ) Engenharia de Software

( ) Redes de Computadores

( ) Sistemas de Informação

( ) Outros...

Questão 5. Informe a instituição de ensino do seu curso de graduação

---

Questão 6. Você está trabalhando na área da sua graduação atualmente? (Estágios também são válidos)

( ) Sim

( ) Não

Questão 7. Caso esteja trabalhando atualmente, qual cargo você ocupa?

( ) Analista de Qa/Testes de Software

( ) Analista de Sistemas

( ) Arquiteto de Software

( ) Desenvolvedor

( ) Engenheiro de Software

( ) UI/UX

( ) Outros...

---

2. Sobre a sua graduação

Nesta seção, serão coletadas algumas informações mais específicas sobre a sua graduação

Questão 8. Status da graduação

( ) Início do curso

( ) Em andamento

( ) Formado a menos de 1 ano

( ) Formado de 1 a 3 anos

( ) Formado a mais de 6 anos

( ) Formado a mais de 10 anos

Questão 9. O seu curso possui alguma disciplina voltada para testes de software?

Essa disciplina pode ser obrigatória ou optativa

(Ex.: Teste de Software, Verificação e Validação)

- Uma disciplina é caracterizada como obrigatória quando o aluno é obrigado a cursá-la para concluir a graduação.

- Uma disciplina é caracterizada como optativa quando o aluno pode escolher se deseja cursá-la.

( ) Sim, disciplina obrigatória
( ) Sim, disciplina optativa
( ) Não, meu curso não possui nenhuma disciplina voltada para testes

Questão 10. Caso ainda esteja cursando a graduação, você já cursou a disciplina voltada para testes de software?
(Ex.: Teste de Software, Verificação e Validação)

( ) Sim
( ) Não
( ) Sim, já sou formado(a)

3. Sobre a disciplina de voltada para teste de software da sua graduação

Nesta seção, buscamos detalhes específicos sobre a disciplina de teste de software durante sua graduação, bem como sua experiência ao cursá-la.

Questão 11. Como você avalia o seu nível de conhecimento adquirido na disciplina de teste de software?
( ) Excelente
( ) Acima da média
( ) Médio
( ) Abaixo da média
( ) Extremamente ruim

Questão 12. Quanto ao conteúdo abordado na disciplina de teste de software durante sua graduação, você acha que foi mais voltado para a teoria ou para a prática?
( ) Mais teórico
( ) Mais prático
( ) A disciplina abordou de maneira eficaz tanto os aspectos teóricos quanto práticos.

Questão 13. Você encontrou desafios ao aprender o conteúdo da disciplina de teste de software?

( ) Sim

( ) Não

Questão 14. Por favor, descreva as dificuldades que você encontrou durante a disciplina de teste de software.

Questão 15. Houve alguma falta em relação a conteúdos, técnicas, abordagens e/ou ferramentas que você gostaria de ter aprendido na disciplina de teste de software?

( ) Sim

( ) Não

Questão 16. Qual conteúdo, técnica, abordagem ou ferramenta específica você gostaria de ter aprendido na disciplina de teste de software?

---

4. Conhecimento em Testes de Software

Esta seção visa avaliar o conhecimento adquirido pelos participantes durante sua formação em Engenharia de Software, especificamente em relação a práticas, técnicas e conceitos relacionados a teste de software.

Questão 17. Quais níveis de teste de software você conhece e/ou teve contato?

Os níveis de teste indicam em qual fase do desenvolvimento de software os testes serão aplicados.

( ) Teste de aceitação

( ) Teste de integração

( ) Teste de regressão

( ) Teste de sistema

( ) Teste de unidade

( ) Nenhum

( ) Outros...

Questão 18. Quais técnicas de teste de software você conhece e/ou teve contato?

As técnicas de teste representam abordagens ou métodos específicos para conduzir os testes.

( ) Técnica de teste estrutural (ou teste de caixa-branca)
( ) Técnica de teste funcional (ou teste de caixa-preta)
( ) Nenhum
( ) Outros...

Questão 19. Quais testes estruturais (ou teste de caixa-branca) você conhece e/ou teve contato?

Nesta técnica, o foco principal é a estrutura da aplicação, o teste é aplicado diretamente no código fonte.

( ) Teste de fluxo de controle
( ) Teste de caminho
( ) Teste de fluxo de dados
( ) Nenhum
( ) Outros...

Questão 20. Quais testes funcionais (ou teste de caixa-preta) você conhece e/ou teve contato?

Nesta técnica, o teste é realizado para cobrir todas as funcionalidades da aplicação, porém sem entrar em detalhes de implementação. Testes no ponto de vista do usuário final.

( ) Partição de equivalência
( ) Análise de valores limite
( ) Tabela de decisão
( ) Nenhum
( ) Outros...

Questão 21. Quais práticas modernas de teste você conhece?

( ) Testes Automatizados
( ) *Test Smells*

( ) *Flaky Test*

( ) Nenhum

( ) Outros...

Questão 22. Sobre testes automatizados, quais dos seguintes testes você conhece e/ou teve contato?

Testes automatizados referem-se à prática de executar testes de software de forma automatizada, sem a necessidade de intervenção manual. São testes conduzidos por ferramentas e *scripts* que executam os casos de testes.

( ) Teste de Cobertura de Código

( ) Teste de Módulo

( ) Teste Funcional

( ) Teste de Segurança

( ) Teste de Interface Gráfica do Usuário (GUI)

( ) Teste de Regressão

( ) *Random Testing*

( ) Teste de Usabilidade

( ) Teste de Unidade

( ) Teste de Integração

( ) Teste de Sistema

( ) Teste de Aceitação

( ) Teste Baseado em Modelos

( ) Teste Baseado em Especificações

( ) Teste de Concorrência

( ) Teste Baseado em Falhas

( ) Teste de Interoperabilidade

( ) Teste de Carga

( ) Teste de Desempenho

( ) Teste de Estresse

( ) Teste de Vazamento de Memória

( ) Teste de Ambiente

( ) Nenhum

( ) Outros...

Questão 23. Sobre *Test Smells*, quais dos seguintes testes você conhece e/ou teve contato?

Test smell são indícios de problemas nos testes de software, apontando possíveis fragilidades ou deficiências.

( ) *Mystery Guest*

( ) *Resource Optimism*

( ) *Test Run War*

( ) *General Fixture*

( ) *Eager Test*

( ) *Lazy Test*

( ) *Assertion Roulette*

( ) *Indirect Testing*

( ) *For Testers Only*

( ) *Senstitive Equality*

( ) *Test Code Duplication*

( ) Nenhum

( ) Outros...

Questão 24. Sobre *Flaky Tests*, quais dos seguintes testes você conhece e/ou teve contato?

Flaky tests são testes automatizados que ocasionalmente falham ou passam sem razão aparente, indicando instabilidade e inconsistência.

( ) *Async Wait*

( ) *Concurrency*

( ) *Test Order Dependency*

( ) *Resource Leak*

( ) *Network*

( ) *Time*

( ) *IO*

( ) *Randomnes*

( ) *Floating Point Operations*

( ) *Unordered Collections*

( ) Nenhum

( ) Outros...

Questão 25. Você conhece alguma ferramenta e/ou *framework* para testes de software?

Por exemplo: *JUnit*, *Cypress*, *Selenium*.

Questão 26. Você conhece alguma ferramenta de gerenciamento de teste?

Por exemplo: *Test Manager*, *Test Link*.

Questão 27. Tendo em vista o que você viu na graduação e as práticas e técnicas mostradas nessa pesquisa, na sua opinião, o ensino de teste de software necessita de melhorias?

( ) Sim

( ) Não

5. Sugestões de melhorias

Nesta seção, coletaremos sugestões que os participantes julguem necessárias para o ensino de teste de software na graduação.

Questão 28. O que você sugere que seja feito para trazer melhorias no ensino de teste de software?

APÊNDICE B – *SURVEY* - INVESTIGANDO PRÁTICAS MODERNAS DE TESTES DE SOFTWARE NA INDÚSTRIA

Questão 1. Termo de consentimento

Caro(a) Participante,

Esse questionário tem como objetivo identificar as práticas utilizadas pelos profissionais de testes de software na indústria. Este questionário faz parte do Trabalho de Conclusão de Curso (TCC) da Universidade Federal do Ceará - Campus Quixadá do curso de Sistemas de Informação. Suas respostas são valiosas e ajudarão a fornecer *insights* cruciais para a pesquisa. Por favor, responda de maneira honesta e reflexiva, baseando-se em sua experiência na indústria. Suas informações serão tratadas com confidencialidade e serão utilizadas estritamente para fins acadêmicos. Ao responder a este questionário, você está consentindo com a participação na pesquisa. Suas respostas serão anônimas e não serão associadas a sua identidade pessoal.

Termo de Consentimento: Eu li e compreendi as informações fornecidas acima. Ao continuar e responder a este formulário, concordo em participar voluntariamente desta pesquisa. Entendo que minhas respostas serão usadas apenas para fins de pesquisa e que minha identidade será mantida em sigilo.

Em caso de dúvidas entre em contato pelo o e-mail: lizandra27@alu.ufc.br

1. Identificação

Nesta seção, solicitaremos algumas informações para entender um pouco mais sobre você. Suas respostas são fundamentais para a nossa análise e nos ajudará a contextualizar suas percepções e experiências atuando na área de teste de software.

Questão 2. Nome (opcional)

_____

Questão 3. Estado

( ) 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

Questão 4. Informe seu curso de graduação

( ) Ciência da Computação

( ) Design Digital

( ) Engenharia da Computação

( ) Engenharia de Software

( ) Redes de Computadores

( ) Sistemas de Informação

( ) Outros...

Questão 5. Há quanto tempo você atua na área de Testes?

( ) Menos de 1 ano

( ) 1 - 3 anos

( ) 3 - 5 anos

( ) Mais de 5 anos

2. Sobre a sua experiência na área de Testes

Nesta seção, gostaríamos de conhecer um pouco da sua experiência como profissional de teste de software e saber quais práticas você têm utilizado atualmente.

Questão 6. Na organização onde você trabalha é seguida alguma metodologia?

( ) *Agile*

( ) *DevOps*

( ) *Extreme Programming*

( ) *Kanbam*

( ) *Scrum*

( ) *Waterfall* ou Cascata

Questão 7. Quais linguagens de programação são mais frequentemente usadas por você?

( ) Java

( ) *JavaScript*

( ) *Python*

( ) Outros...

Questão 8. Você utiliza alguma ferramenta de gestão de testes para acompanhar o progresso e relatar resultados?

( ) Jira

( ) *TestRail*

( ) *Test Manager*

( ) *Test Link*

( ) Nenhuma

( ) Outros...

Questão 9. Abaixo estão alguns testes que podem ser executados em diferentes níveis durante a fase de desenvolvimento. Marque aqueles que você já executou.

( ) Teste de Aceitação

( ) Teste de Integração

( ) Teste de Regressão

( ) Teste de Sistema

( ) Teste de Unidade

( ) Nenhuma

( ) Outros...

Questão 10. Técnicas de teste representam abordagens ou métodos específicos para conduzir os testes. Marque abaixo as técnicas que você já utilizou.

( ) Técnica de teste estrutural (ou teste de caixa-branca)

( ) Técnica de teste funcional (ou teste de caixa-preta)

( ) Nenhuma

( ) Outros...

Questão 11. Ainda falando sobre técnicas de testes. Quais testes estruturais (ou teste de caixa-branca) você já executou?

( ) Teste de fluxo de controle

( ) Teste de caminho

( ) Teste de fluxo de dados

( ) Nenhuma

( ) Outros...

Questão 12. Ainda falando sobre técnicas de testes. Quais testes funcionais (ou teste de caixa-preta) você já executou?

( ) Partição de equivalência

( ) Análise de valores limite

( ) Tabela de decisão

( ) Nenhuma

( ) Outros...

Questão 13. Sobre testes automatizados, quais dos seguintes testes você já executou?

( ) Teste de Cobertura de Código

( ) Teste de Módulo

( ) Teste Funcional

( ) Teste de Segurança

( ) Teste de Interface Gráfica do Usuário (GUI)

( ) Teste de Regressão

( ) *Random Testing*

( ) Teste de Usabilidade

( ) Teste de Unidade

( ) Teste de Integração

( ) Teste de Sistema

( ) Teste de Aceitação

( ) Teste Baseado em Modelos

( ) Teste Baseado em Especificações

( ) Teste de Concorrência

( ) Teste Baseado em Falhas

( ) Teste de Interoperabilidade

( ) Teste de Carga

( ) Teste de Desempenho

( ) Teste de Estresse

( ) Teste de Vazamento de Memória

( ) Teste de Ambiente

( ) Nenhuma

( ) Outros...

Questão 14. Sobre *Test Smells*, quais dos seguintes testes você já executou?

Test smell são indícios de problemas nos testes de software, apontando possíveis fragilidades ou deficiências.

Saiba mais sobre *Test Smells*:

https://testsmells.org/pages/testsmells.html

( ) *Mystery Guest*

( ) *Resource Optimism*

( ) *Test Run War*

( ) *General Fixture*

( ) *Eager Test*

( ) *Lazy Test*

( ) *Assertion Roulette*

( ) *Indirect Testing*

( ) *For Testers Only*

( ) *Senstitive Equality*

( ) *Test Code Duplication*

( ) Nenhuma

( ) Outros...

Questão 15. Sobre *Flaky Tests*, quais dos seguintes testes você já executou?

Flaky tests são testes automatizados que ocasionalmente falham ou passam sem razão aparente, indicando instabilidade e inconsistência.

Saiba mais sobre *Flaky tests*:

https://docs.gitlab.com/ee/development/testing_guide/flaky_tests.html

( ) *Async Wait*

( ) *Concurrency*

( ) *Test Order Dependency*

( ) *Resource Leak*

( ) *Network*

( ) *Time*

( ) *IO*

( ) *Randomnes*

( ) *Floating Point Operations*

( ) *Unordered Collections*

( ) Nenhuma

( ) Outros...

Questão 16. Você utiliza alguma ferramenta e/ou *framework* para executar testes?

( ) *Cypress*

( ) *JUnit*

( ) *Selenium*

( ) Nenhuma

( ) Outros...

---

3. Sugestões e opiniões

Nesta seção, gostaríamos de coletar algumas sugestões e opiniões de cada participante sobre algumas questões específicas.

Questão 17. Quais principais dificuldades você encontra no seu dia a dia como um profissional de testes?

Questão 18. O que você diria que é uma obrigação para qualquer pessoa aprender para se tornar um bom profissional de testes de software? Qual o essencial?

APÊNDICE C – ASSUNTOS ABORDADOS NAS DISCIPLINAS

– **CEULP - Teste de Software**

Inspeção de software; Princípios e técnicas de testes de software: teste de unidade, teste de integração, teste de regressão; Desenvolvimento orientado a testes; Automação dos testes; Geração de casos de teste; Teste de *intefaces* humanas; Teste de aplicações para a web; Testes alfas, beta e de aceitação; Ferramentas de testes; Planos de testes; Gerenciamento do processo de testes; Registro e acompanhamento de problemas.

– **PUCRS - Verificação e Validação de Software**

Apresentação dos conceitos fundamentais de qualidade de software no contexto de processo de software. Discussão do conceito de Dependabilidade para compreender a qualidade de um produto de software, com ênfase na confiabilidade. Discussão dos conceitos de verificação e validação como técnicas de remoção de falhas. Apresentação dos níveis de verificação e validação. Estudo teórico-prático de técnicas e ferramentas para verificação e validação de software em diferentes plataformas.

– **UCSAL - Testes e Qualidade de Software**

O site oficial do curso não disponibiliza o Projeto Pedagógico ou qualquer outro documento que detalhe os conteúdos abordados nas disciplinas. Dessa forma, não foi possível obter informações específicas sobre os tópicos cobertos na disciplina de Teste de Software.

– **UFAM - Verificação, Validação e Teste de Software**

Fundamentos da qualidade de software; Qualidade do processo; Maturidade do Processo de Software; Qualidade do produto; Inspeção de software; Princípios e técnicas de testes de software: teste de unidade; teste de integração; teste de regressão; Planejamento de Verificação e Validação; Desenvolvimento orientado a testes; Automação dos testes; Geração de casos de teste; Ferramentas de testes; Planos de testes; Gerenciamento do processo de testes; Registro e acompanhamento de problemas.

– **UFC - Verificação e Validação**

Objetivos e restrições de Verificação e Validação; Planejamento de Verificação e Validação; Documentação de estratégias de Verificação e Validação, testes e outros artefatos; Medidas

e Métricas; Análise estática de código; Atividades de Verificação e Validação ao longo do ciclo de vida de um produto; Revisão de software; Testes de unidade; Análise de cobertura; Técnicas de teste funcional (caixa preta); Testes de integração; Desenvolvimento de casos de teste baseados em casos de uso e estórias de usuários; Testes de sistema; Testes de aceitação; Testes de atributos de qualidade; Testes de regressão; Ferramentas de teste (combinação com ferramentas de integração contínua); Análise de relatórios de falha; Técnicas para isolamento e falhas (depuração); Análise de defeitos; Acompanhamento de problemas (*tracking*); IEEE Std 1012.

– **UFG - Teste de Software**

Processo de construção: definições básicas, atividades e documentação. Processo de Teste de Software: definições básicas, técnicas de teste, teste baseado em intuição e experiência do engenheiro de software, atividades do processo, documentação e ferramentas.

– **UFRN - Teste de Software I; Teste de Software II**

Teste de Software I:

Introdução e motivação ao Teste de Software; Conceitos básicos de teste: defeito, falha, casos de teste, critérios de teste; Teste de Unidade e Integração; Técnicas de Teste: funcional (caixa-preta), estrutural (caixa-branca); Técnica Funcional: partições em classes de equivalência, análise do valor limite, combinações; Técnica estrutural: critérios de teste baseado em fluxo de controle, critérios de teste baseado em fluxo de dados; Ferramentas e *frameworks* para teste de unidade e integração; Técnicas para geração de testes de unidade e integração; Refatoração e testes; Testes de regressão.

Teste de Software II:

Processos de Teste: papéis, atividades e artefatos gerados; Testes de Sistema e Aceitação; Automação de Testes de Sistema e Aceitação; Testes de Carga, Desempenho, Segurança; Automação de Testes de Sistema e Aceitação; Metodologias ágeis e Testes de Software; Revisão de Software: inspeção, revisão em time, *walkthrough*; Programação baseada em assertivas; Análise Estática; Testes nos Modelos de Qualidade.

– **UNB - Testes de Software**

Conceitos básicos; Princípios, técnicas e ferramentas de testes de software; Desenvolvi-

mento orientado a testes (TDD); Utilização de dublês (mocks) para testes; Testes orientados a requisitos não funcionais; Uso de ferramentas para apoiar testes de software.

– **UNIACADEMIA - Teste de Software**

O site oficial do curso não disponibiliza o Projeto Pedagógico ou qualquer outro documento que detalhe os conteúdos abordados nas disciplinas. Dessa forma, não foi possível obter informações específicas sobre os tópicos cobertos na disciplina de Teste de Software.

– **UNIGRAN - Verificação e Validação**

Introdução a verificação e validação de software; Teste no ciclo de vida; Técnicas de teste de software; Estratégias de teste de software; Planejamento de teste; Depuração, manutenção e teste de regressão; Teste e validação do aspecto comportamental de sistemas; Ferramentas de teste de software; Conceitos de qualidade de produto; Dimensões de qualidade de *garvin*; Fatores de qualidade de *mccall*; Normas iso/iec 9126 e iso 2500; Garantia de qualidade.

– **UNIVILLE - Teste de Software I; Teste de Software II**

Teste de Software I:

Princípios e Técnicas de teste; Conceitos básicos de teste: defeito, falha, casos de teste, critérios de teste; Técnicas de teste: funcional (caixa preta), estrutural (caixa branca; Estratégias de teste: teste unitário, teste de integração, teste de regressão, desenvolvimento orientados a testes.

Teste de Software II:

Automação dos testes; Geração de casos de testes; Teste de aplicações para a web; Metodologias ágeis e teste de software; Ferramentas de testes; Plano de testes; Gerenciamento do processo de testes; Registro e acompanhamento de problemas.

– **UTFPR - Teste de Software**

Fundamentos de teste de software: Verificação/Validação/Teste, terminologia de defeitos/erros/falhas, casos de teste, etapas do teste e limitações. Testes automatizados: Estratégias de teste e Testes de unidade (*mocks*, *stubs*, assertivas), integração e sistema. Técnicas de geração de casos de teste: critérios de teste caixa-preta (particionamento em classes de

equivalência, valor limite, tabela de decisão) e caixa-branca (fluxo de controle, fluxo de dados, cobertura de comandos, decisões, caminho básico, caminhos independentes). Teste *end-to-end*: seletores de interface, ferramentas de automatização, Integração contínua e teste baseado em modelo.