UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO INTERDISCIPLINAR EM TECNOLOGIA DA INFORMAÇÃO

Maria Clara de Medeiros Meira

Um Panorama das Técnicas de Teste Caixa Preta para a Detecção de Falhas de Software

PAU DOS FERROS
2025

Maria Clara de Medeiros Meira

Um Panorama das Técnicas de Teste Caixa Preta para a Detecção de Falhas de Software

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito parcial para obtenção do título de Bacharela em Interdisciplinar em Tecnologia da Informação.

Orientador: Prof. Dr. Alysson Filgueira Milanez - UFERSA

PAU DOS FERROS

2025

M499p Meira, Maria Clara de Medeiros.
Um panorama das técnicas de teste caixa preta para a detecção de falhas de software / Maria Clara de Medeiros Meira. - 2025.
66 f. : il.

Orientador: Alysson Filgueira Milanez.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de , 2025.

1. Teste caixa preta. 2. Técnicas de testes. 3. Falhas. 4. Requisitos. I. Milanez, Alysson Filgueira, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

Maria Clara de Medeiros Meira

Um Panorama das Técnicas de Teste Caixa Preta para a Detecção de Falhas de Software

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito parcial para obtenção do título de Bacharela em Interdisciplinar em Tecnologia da Informação.

Defendida em: 14 / 07 / 2025

BANCA EXAMINADORA

Alysson Filgueira Milanez, Prof. Dr. (UFERSA)
Presidente

Laysa Mabel de Oliveira Fontes, Profa. Dra. (UFERSA)
Membro Examinador

David Candeia Medeiros Maia, Prof. Dr. (IFPB)
Membro Examinador

Para a minha avó Gracinha (*In memoriam*), que a cada livro que me presenteou, construiu um degrau para a minha formação, pessoal e acadêmica.

AGRADECIMENTOS

A Deus, toda honra e glória, pois sem a permissão e proteção Dele, nada seria possível.

À minha família, pelo amor e cuidado durante toda a minha vida, ao esforço que sempre fizeram e fazem todos os dias para que eu possa estudar; serei eternamente grata por acreditarem tanto em mim. Sonho em podê-los retribuir com uma vida boa e feliz. Vocês têm todo o meu amor e gratidão.

Aos meus amigos de São Vicente, os agradeço por se fazerem presentes apesar da distância, que nunca mudou nossa cumplicidade.

Aos amigos que construí na graduação, vocês são sinônimos de carinho e amparo.

À professora Mabel, um grandessíssimo carinho, respeito e admiração.

Ao meu orientador, Alysson, um profissional e ser humano ímpar. Obrigada não só por me mostrar o caminho, mas por me ensinar como caminhar.

Prefiro expressar minhas emoções pessoalmente, por isso, os breves agradecimentos. Portanto, a todos que contribuíram de alguma forma para a minha formação e realização deste trabalho, meu mais sincero, obrigada.

EPÍGRAFE

"Se você não tivesse capacidade, Deus não te daria a oportunidade. Seus medos você já conhece, experimente suas coragens."

(Santa Terezinha do Menino Jesus)

RESUMO

Assegurar a qualidade e confiabilidade de um software é fundamental para garantir seu funcionamento e desempenho. Porém, mediante um levantamento bibliográfico da literatura, é notória a carência de materiais no idioma português, e a escassez de materiais acerca da realização de testes de software, especialmente, os de caixa preta. Diante disso, este estudo visa analisar e investigar os testes de caixa preta e sua eficácia em detectar falhas de software, por meio de um estudo direcionado a um aprofundamento quanto à aplicação das suas diferentes técnicas e estratégias, e analisando qualitativamente o seu desempenho. Dessa forma, apresentando um panorama contendo as suas estratégias e metodologias para a realização dos testes funcionais, possibilitando a análise e investigação do seu funcionamento individual, para minimizar a ocorrência de erros. A aplicação das diferentes técnicas, metodologias e estratégias, bem como a análise do seu desempenho em diferentes cenários, permite que as abordagens sejam exploradas conforme a situação em que estão submersas, resultando em uma série de ações e investigações para garantir o funcionamento adequado do sistema. Assim, este panorama avaliativo construído mediante a sua categorização auxilia na verificação e validação dos requisitos e especificações dos softwares, para que, assim, os sistemas atinjam as métricas de qualidade que os farão funcionar em conformidade e de forma mais eficiente.

Palavras-chave: Teste caixa preta; técnicas de testes; falhas; requisitos.

ABSTRACT

Ensuring the quality and reliability of software is essential to guarantee its operation and performance. However, by means of a bibliographic survey of literature, it is notorious the lack of materials in the Portuguese language, and the scarcity of materials on performing software testing, especially black box testing. Thus, this study aims to analyze and investigate black box testing and its effectiveness in detecting software failures, through a study aimed at deepening the application of its different techniques and strategies, and qualitatively analyzing its performance. Presenting an overview containing its strategies and methodologies for performing functional tests, enabling the analysis and investigation of its individual operation, to minimize the occurrence of errors. The application of different techniques, methodologies and strategies, as well as the analysis of their performance in different scenarios, ensures that they are explored according to the situation in which they are submerged, which results in a series of actions and investigations to ensure the proper functioning of the system. Therefore, this evaluative overview through its categorization helps in the verification and validation of its requirements and specifications, so that the systems achieve the quality metrics that will make them function in compliance and more efficiently.

Keywords: black box testing; testing techniques; failures; requirements.

LISTA DE FIGURAS

Figura 1 – Processo de Análise e Testes 42
Figura 2 – Validação de Palavras Palíndromas 47
Figura 3 – Fluxograma para Análise de Casos de Teste 48
Figura 4 – Grafo de Causa e Efeito para o Sistema de reserva de Voos 53
Figura 5 – Fluxograma para a Inserção e Listagem de um Contato Válido 54
Figura 6 – Análise de Usabilidade de Sistema de Formulários 57
Figura 7 – Sistema de Biblioteca Digital: Página de Login 59
Figura 8 – Sistema de Biblioteca Digital: Página Principal 59
Figura 9 – Plataforma de Cursos Digital: Página de Login 61
Figura 10 – Plataforma de Cursos Digital: Página Principal 61

LISTA DE TABELAS

Tabela 1 – Tabela de Validação de Senhas 45
Tabela 2 – Tabela de Decisão para o Sistema de Reservas de Voos 53

LISTA DE QUADROS

Quadro 1 – Heurísticas de Nielsen para Avaliar Métricas de Usabilidade 30
Quadro 2 – Comparação entre as técnicas mais utilizadas pelos autores nos trabalhos relacionados 39
Quadro 3 – Técnicas de Teste Caixa Preta 43
Quadro 4 – Classes de Equivalência para Validar Palíndromos 46
Quadro 5 – Teste Funcional Sistemático 49
Quadro 6 – Teste Funcional Sistemático Para Cálculos de Matrizes 50
Quadro 7 – Casos de Teste Combinatoriais para o Sistema de Agenda de Contatos . . . 53
Quadro 8 – Casos de Teste Combinatoriais para o Sistema de Biblioteca 55
Quadro 9 – Casos de Teste para um Utilitário de Calculadora 55
Quadro 10 – Avaliação das Heurísticas de Nielsen no sistema de Formulário 58
Quadro 11 – Avaliação das Heurísticas de Nielsen no Sistema BiblioTest 60
Quadro 12 – Avaliação das Heurísticas de Nielsen da Plataforma de Cursos Digital . . . 62

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Laço de Repetição para Validação de Senhas 44
Código-fonte 2 – Método para Verificação de Palíndromo 46
Código-fonte 3 – Validação de Email 48
Código-fonte 4 – Cálculo do Determinante de uma Matriz 3x3 49
Código-fonte 5 – Métodos para Confirmação de Reserva 51

SUMÁRIO

1 INTRODUÇÃO 15
1.1 Problematização 16
1.2 Objetivos 17
1.2.1 Objetivo Geral 17
1.2.2 Objetivos Específicos 17
1.3 Justificativa 18
1.4 Contribuição 19
1.5 Organização do Trabalho 19
2 FUNDAMENTAÇÃO TEÓRICA 21
2.1 Garantia de Qualidade 21
2.2 Granularidade dos Requisitos Funcionais 22
2.3 Teste de Software 22
2.4 Técnicas de Teste Caixa Preta 25
2.5 Testes de Usabilidade 28
2.5.1 Heurísticas de Nielsen 29
2.6 Eficácia dos Testes Funcionais 30
3 TRABALHOS RELACIONADOS 33
3.1 Processo de Teste de Software 33
3.2 Testes Funcionais 34
3.2.1 Teste Caixa Preta e Teste Caixa Branca 37
3.3 Análise da Usabilidade de Sistemas 37
3.4 Comparação Entre a Aplicação das Técnicas nos Trabalhos 38
4 METODOLOGIA 40
4.1 Levantamento Bibliográfico 40
4.2 Desenvolvimento dos Sistemas Analisados 41
4.3 Desenvolvimento dos Casos de Teste 41
4.4 Aplicação das Técnicas Caixa Preta 41
5 RESULTADOS 44
5.1 Particionamento por Equivalência 44
5.1.1 Validação de Senhas 44
5.1.2 Verificação de Palavras Palíndromas 46
5.2 Teste Funcional Sistemático 47
5.2.1 Validação de E-mail 47
5.2.2 Cálculo de Matriz 3x3 49

5.3 Grafo de Causa e Efeito e Tabela de Decisão 51
5.3.1 Reserva de Voos 51
5.4 Testes Combinatoriais e *Error Guessing* 52
5.4.1 Agenda de Contatos 53
5.5 Empréstimo de Livros 54
5.6 Teste Fuzzing 55
5.6.1 Utilitário de Calculadora 55
5.7 Aplicação dos Testes Caixa Preta em Relação à Usabilidade 56
5.7.1 Sistema de Formulário 57
5.7.2 Sistema de Biblioteca Digital 58
5.7.3 Plataforma de Cursos Digital 59
5.8 Discussões 61
6 CONCLUSÕES E TRABALHOS FUTUROS 64
REFERÊNCIAS 66

1 INTRODUÇÃO

Considera-se que os erros e falhas de software podem significar consequências gravíssimas durante o desenvolvimento ou entrega de um projeto, como os casos de alto perfil das falhas na implementação de sistemas de informação, a exemplo disto, a falha da *Hewlett-Packard* (HP) em 2004, que teve um impacto financeiro de 160 milhões de dólares (Dwivedi *et al.*, 2014). Objetificando a qualidade como um aspecto crítico para atestar o funcionamento do produto (Delamaro; Jino; Maldonado, 2013). Assim, pode-se definir que o teste de software é o processo de execução de determinado produto para verificar se o mesmo atingiu os objetivos das especificações e funcionou corretamente conforme o ambiente em que foi projetado (Neto, 2022).

A utilização de metodologias de teste ao longo do desenvolvimento dos sistemas torna-se indispensável para garantir sua adequação funcional, que deve ser operada criteriosamente e embasada tecnicamente (Delamaro; Jino; Maldonado, 2013), para estar conforme as métricas de qualidade de desenvolvimento de sistemas (Nielsen, 2007). Considerando o objetivo principal do teste de software como sendo revelar possíveis falhas ao longo do desenvolvimento do produto, para serem detectadas e corrigidas antes da entrega final (Delamaro; Jino; Maldonado, 2016). Existem diferentes formas de solucionar problemas, porém, revelá-los torna-se a primeira tarefa no processo de verificação e validação dos requisitos (Delamaro; Jino; Maldonado, 2016).

Diante disso, utilizar técnicas e metodologias eficazes de testes torna-se essencial. Dentre as abordagens disponíveis, os testes de caixa preta possuem a capacidade de detectar problemas sem a necessidade de conhecimento do código-fonte (Delamaro; Jino; Maldonado, 2013). Desse modo, os testes são feitos partindo do ponto de vista do usuário, ajudando a revelar inconsistências (AbuSalim; Ibrahim; Wahab, 2020), ou seja, após a apuração das funções dos sistemas, observa-se quais são fornecidas ou executadas, a depender dos requisitos do sistema, caso existam ou não. Nesse caso, evidencia-se que os testes funcionais são uma estratégia baseada nos requisitos e especificações e visam validar as funcionalidades do sistema, podendo ser aplicados a todos os níveis do processo de verificação e validação (Neto, 2022).

Os testes de caixa preta, também são conhecidos como testes funcionais, visto que são análises externas para avaliar o comportamento e execução das funcionalidades de um software por meio da criação de casos de teste e aplicação de técnicas para solucionar problemas (Delamaro; Jino; Maldonado, 2013). Os termos e argumentações da literatura que adotam os testes de caixa preta como essenciais no processo de desenvolvimento, destacam as vantagens

da utilização para poderem verificar que o sistema contempla os benefícios proporcionados durante e após as fases de teste, que serão menos exaustivos e demorados, e garantirão uma baixa granularidade (Khan; Khan, 2012), além de um desenvolvimento ágil dos casos de teste, favorecendo uma correção rápida e um retorno mais rápido do funcionamento.

Analisando cada um das técnicas individualmente, será possível construir uma perspectiva mais abrangente dos testes funcionais, desse modo, estabelecendo um panorama no que se refere ao seu desempenho e eficiência. A partir da implementação em diferentes cenários, este trabalho busca promover a compreensão quanto a quais contextos as técnicas se tornam mais apropriadas, permitindo uma escolha mais assertiva e eficaz conforme as particularidades de cada sistema, otimizando o processo de teste, para a correção de erros.

Assim, estudar e aplicar as metodologias de teste caixa preta torna-se indispensável, o que permite aprimorar a capacidade de identificação e correção de erros (Delamaro; Jino; Maldonado, 2013). Além disso, colocando em prova as técnicas, avaliando-as individualmente e elaborando um panorama de investigação conforme o seu funcionamento, minimiza-se a ocorrência de possíveis falhas. Por meio do presente estudo, destaca-se a importância da realização dos testes funcionais, somado à aplicação de diferentes técnicas e metodologias, bem como o uso em diferentes circunstâncias, de modo a garantir o funcionamento adequado do sistema e auxiliar na verificação e validação dos seus requisitos e especificações (Neto, 2022).

1.1 Problematização

Analisando a complexidade dos atuais materiais de estudos acerca de teste de software, e a escrita sendo majoritariamente no idioma inglês (Myers; Sandler; Badgett, 2011), o presente trabalho visa apresentar um novo panorama condensado e objetivo acerca das metodologias e estratégias dos testes de caixa preta no idioma português. Com isso, proporcionando recursos na língua portuguesa para incentivar a aprendizagem e aplicações metodológicas e profissionais.

Nesse sentido, a exploração dos conceitos teóricos e a aplicação prática das abordagens em sistemas pré-definidos e construídos especificamente para esta análise, adequando as estratégias e técnicas de teste caixa preta para detectar falhas e, consequentemente, atestar a qualidade de sistemas, cria um panorama com potencial para diminuir a incidência de erros em uma das etapas mais importantes da elaboração de um produto.

A partir disso, neste estudo, apresenta-se um panorama das abordagens metodológicas dos testes de caixa preta. Ademais, a sua utilização em diferentes cenários, visa facilitar o

entendimento conforme a individualidade de cada técnica. Valendo salientar que a atividade de teste não prova a ausência de defeitos, somente revela a existência dos mesmos (Delamaro; Jino; Maldonado, 2013). Dessa forma, fornecendo uma perspectiva abrangente da implementação dessas técnicas em diferentes cenários, colocando em prova sua eficácia e como respondem às respectivas situações em que foram colocadas.

1.2 Objetivos

Esta pesquisa visa investigar e construir um amplo panorama teórico e prático acerca dos testes de caixa preta, bem como suas diferentes metodologias, técnicas e estratégias. Além disso, busca-se analisar o desempenho de cada abordagem individualmente para compreender em que cenários e situações podem ser inseridas no processo de verificação e validação, para auxiliarem na detecção de falhas. Resultando em um objetivo maior, com a construção de um material que possa ser usado como apoio para estudantes e profissionais da área de teste de software, no idioma português, que aborda desde os conceitos teóricos até aplicações práticas conforme o seu desempenho.

1.2.1 Objetivo Geral

Buscando revelar as falhas presentes em diferentes sistemas, esta pesquisa visa investigar a aplicação dos testes caixa preta em diferentes sistemas.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral, serão abordados os seguintes aspectos:

- Realizar um levantamento bibliográfico acerca de teste de software e testes de caixa preta;
- Estudar e apresentar individualmente cada uma das técnicas e abordagens de teste caixa preta, para construir casos de teste e estratégias para cada abordagem;
- Aplicar as técnicas e estratégias de testes de caixa preta em diferentes cenários de sistemas finalidades individuais, observando seu comportamento e eficácia para a detecção de falhas;
- Analisar qualitativamente as técnicas de teste caixa preta e como se mostram úteis para a detecção de falhas.

1.3 Justificativa

A análise da eficácia das novas metodologias e aplicações dos testes de caixa preta, bem como a sua adição a um panorama estratégico, serão exibidas mediante a realização das aplicações das técnicas em diferentes sistemas. Dessa forma, objetificando a sua adequação a cenários específicos. Atrelado a isso, a análise dos fatores que atestam essa eficácia, como as especificações e documentação, mostrando que os testes funcionais podem revelar com precisão a completude, ou não, dos requisitos do sistema (Lee *et al.*, 2013).

A definição subjacente de qualidade de software, afirma que os produtos poderiam ser completamente especificados e poderiam ser estabelecidos procedimentos para avaliar um produto fabricado conforme as suas especificações (Sommerville, 2018), assim, o processo de teste de software contribui para a verificação e validação do funcionamento dos sistemas, analisando seus atributos funcionais e especificações. Além de atestarem a usabilidade, confiança e eficiência, estes testes fornecem métricas que buscam reforçar os atributos de segurança, compreensibilidade e portabilidade (Sommerville, 2018). Desse modo, fornecendo uma base sólida para a construção de um software de qualidade, mediante a análise do funcionamento do produto todo, para que os padrões de qualidade resultem na satisfação dos clientes e usuários.

Enfatiza-se que este estudo contribuirá para aumentar a quantidade de materiais didáticos, acerca dos testes de caixa preta, em particular no idioma português. Além disso, busca melhorar o entendimento sobre o funcionamento e aplicação desses testes, oferecendo, assim, ferramentas que podem ser utilizadas durante todo o processo de desenvolvimento, visando orientar práticas mais eficazes de testes e promover a completude dos requisitos no funcionamento de um software.

Portanto, a compreensão aprofundada das falhas detectadas por meio dos testes de caixa preta fornecem a capacidade de reconhecer determinadas áreas do sistema que estão mais suscetíveis a erros, o que permite trilhar um caminho mais direcionado e eficiente nas etapas de desenvolvimento e manutenção. Esta pesquisa pode contribuir, ainda, para revelar padrões de falhas e para uma melhoria contínua geral. Contudo, além de demonstrar a eficácia e a importância dos testes de caixa preta, esta é uma abordagem estruturada acerca de qualidade e confiabilidade.

1.4 Contribuição

O presente trabalho busca contribuir para com a comunidade acadêmica e profissional, a partir da realização de um estudo sobre testes de software, desenvolvido na língua portuguesa. Sabendo da escassez de materiais acerca de teste de software (Marinho, 2010), principalmente em panoramas acerca dos testes funcionais, caixa preta, especialmente no idioma português, este estudo reúne estratégias, técnicas e metodologias que contribuem para um maior conhecimento e melhor entendimento de como podem ser utilizadas para atestar as funcionalidades de um sistema. Bem como, podem se tornar ferramentas para a detecção de defeitos e prevenção de falhas.

Durante a construção do presente estudo, foram analisadas referências bibliográficas e diferentes trabalhos acerca de qualidade, teste de software e testes funcionais. Diante disso, foi percebida a falta de materiais acerca das técnicas de teste caixa preta detalhadamente, envolvendo teoria e prática. A fim de contribuir para as bases de dados e como um recurso metodológico, este estudo reúne diferentes perspectivas e conceitos sobre as técnicas de teste caixa preta, fundamentadas sob abordagens e metodologias que contestam a sua acurácia. Atrelado a isso, foram desenvolvidos sistemas com contextos diversos para as técnicas serem empregadas em diferentes domínios de aplicação, validando o propósito de atestar a sua eficiência para a detecção de falhas em sistemas reais, bem como, examinar o desempenho individual delas, desde o estudo das abordagens até aplicações práticas.

Desse modo, o presente trabalho tem o potencial de se tornar um recurso metodológico importante e acessível para a comunidade acadêmica, um instrumento para estudos e pesquisas. Além de se proporcionar como uma ferramenta para aplicações reais no âmbito profissional, uma vez que haverá um maior domínio e conhecimento sobre as técnicas de teste de software e sua adequação a diferentes situações, que podem ser associadas a problemas reais.

1.5 Organização do Trabalho

Este estudo é organizado como segue. O Capítulo 2, exibe a fundamentação teórica utilizada para sua construção, apresentando ideias e metodologias que serão discutidas ao longo do desenvolvimento, reafirmando a revisão bibliográfica. Depois, são apresentados os trabalhos relacionados (Capítulo 3), que apontam as contribuições dos autores, que se destacaram pelas abordagens argumentadas e somam novas metodologias e empregabilidade das técnicas.

Serão exibidas as metodologias aplicadas (Capítulo 4), bem como atividades realizadas para atestar o objetivo principal deste estudo. Em seguida, o Capítulo 5 exibe como cada técnica foi utilizada, bem como a escolha das mesmas e desempenho de cada uma, obtendo uma abordagem exploratória conforme os requisitos e especificações. Por fim, tem-se o Capítulo 6, em que são apresentadas as conclusões e os trabalhos futuros, demonstrando os achados obtidos pela presente pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo, apresenta a base teórica da pesquisa, bem como contextualizações e elementos utilizados para subvencionar este estudo. A Seção 2.1 trata da Garantia de Qualidade, posteriormente, a Seção 2.2 discorre sobre a granularidade dos requisitos funcionais. Então, as seções 2.3 e 2.4 apresentam o teste de software e as técnicas de teste caixa preta, respectivamente; em seguida, a Seção 2.5 traz os testes de usabilidade e a Seção 2.5.1 apresenta as heurísticas de Nielsen, e, por fim, a Seção 2.6 discute sobre a eficácia dos testes funcionais.

2.1 Garantia de Qualidade

Sommerville (2018) aponta que a qualidade refere-se a um conjunto de atributos que um software possui para satisfazer seus usuários, como a eficiência, confiabilidade e manutenibilidade, tendo os atributos de confiança como os atributos de qualidade mais importantes de um sistema, além do seu desempenho. Além destas, outras características secundárias podem ser apontadas, como a proteção, responsável por garantir a segurança e confiabilidade do sistema; a testabilidade, sendo um atributo importante para detectar possíveis erros; a adaptabilidade, que permite que os sistemas sejam mais modulares, aumentando sua responsividade; e a usabilidade, que reflete a facilidade de entendimento e utilização do sistema. Estas características somam ferramentas para conduzir o processo de garantia de qualidade de software.

Para Koscianski e Soares (2007), para avaliar as métricas de qualidade existem dois elementos fundamentais, a variedade de aspectos a se considerar, e a presença de muitos elementos intangíveis. Assim, considera-se que com um sistema medido mediante uma série de características, é possível obter um retorno mais preciso acerca da sua qualidade, podendo ser usadas também para uma definição mais detalhada dos requisitos, fixando desde o início da produção do projeto os objetivos que devem ser atingidos ao finalizarem o desenvolvimento (Koscianski; Soares, 2007).

A validação da qualidade é definida como o conjunto de atividades planejadas e sistemáticas, que, implementadas no sistema, demonstram-se necessárias para munir a confiança adequada de que a entidade atenderá os requisitos de qualidade especificados (ISO/IEC, 2017). Desse modo, constitui atividades desde revisões dos requisitos, as revisões da análise e do projeto do software, e das inspeções de código até os testes (Koscianski; Soares, 2007). Neste mesmo contexto, para Pressman e Maxim (2021), a qualidade de software é essencial para garantir a

satisfação do cliente e a sustentabilidade do produto no mercado.

2.2 Granularidade dos Requisitos Funcionais

Diante da perspectiva funcional, pode-se decompor o software em componentes que contenham diferentes funcionalidades, bem como as suas próprias responsabilidades (Bourque *et al.*, 2002). Estes módulos contêm diferentes responsabilidades funcionais, dessa forma, as diferenças quanto a granularidade podem dificultar a manutenção, compreensão e modularidade das funcionalidades de um sistema (Spijkman *et al.*, 2021).

Para Spijkman *et al.* (2021), existem cenários com múltiplas camadas, às quais podem ser aplicadas de forma adjacente, classificadas como: a granularidade das especificações, que estão relacionadas ao refinamento e abstração, e possuem requisitos de alto nível somado à decomposição detalhada de seus componentes; sendo assim, estarão dispostos com base em funcionalidades semelhantes e alocados em componentes de arquitetura de alto nível. Já os requisitos de alinhamento, sendo alocação e satisfação, atribuem níveis de granularidade a ambos os níveis dos componentes, sendo essa a relação entre os componentes arquitetônicos e os requisitos de satisfação.

Desse modo, a granularidade dos requisitos funcionais surge como um nivelamento para o detalhamento das funcionalidades que determinado sistema deverá exercer (Spijkman *et al.*, 2021). Assim, quanto mais granulares forem estes requisitos, mais conterão métricas de refinamento, enquanto os que possuem uma menor granularidade partem de uma abstração maior, direcionados para alocação (Spijkman *et al.*, 2021).

2.3 Teste de Software

Testar um software é o processo de executar um programa com a intenção de encontrar erros (Kaner; Falk; Nguyen, 2011), o que reafirma que o processo de teste não garante a sua ausência, somente os revela, sendo imperiosa a aplicação de recursos que os solucionem. Assim, o processo de testes está ligado com a garantia assertiva da confiabilidade e da qualidade (Delamaro; Jino; Maldonado, 2016), visto que as atividades desempenhadas durante esse processo apontam para problemas que podem comprometer o funcionamento do sistema, imprevistos que podem acontecer desde a elaboração da documentação até enganos cometidos entre as linhas de código.

Os testes de software seguem uma série de processos projetados para garantir que os

sistemas desenvolvidos funcionem conforme foram planejados, e que não possua ações não intencionais, devendo ser previsível e consistente, sem apresentar surpresas para os usuários (Myers; Sandler; Badgett, 2011). Para assegurar o bom funcionamento dos sistemas é necessário verificar aspectos de correção, completude e coerência, bem como os requisitos funcionais e não funcionais, incluindo aspectos de segurança, robustez e desempenho (Delamaro; Jino; Maldonado, 2016).

Para os diversos problemas encontrados durante o desenvolvimento de um sistema, a literatura define defeito, erro e falha Delamaro, Jino e Maldonado (2013). O defeito encontrado em um software corresponde a alguma imperfeição do código; normalmente, um padrão incorreto de lógica gerado durante a implementação. O resultado diferente do especificado provoca um erro. Dos resultados ou condições inesperadas causadas por um defeito em não conformidade com os requisitos do software, denomina-se falha (Delamaro; Jino; Maldonado, 2016; Pressman; Maxim, 2016).

Os testes realizados com base na estrutura interna dos sistemas, são chamados testes de caixa branca, também conhecidos como testes estruturais, que possuem uma alta granularidade (Khan; Khan, 2012), sendo projetados com base no código-fonte dos programas, para exercitar os seus caminhos, condições, ciclos e estruturas de dados (Myers; Sandler; Badgett, 2011). Dessa forma, os caminhos lógicos do software são testados resultando em casos de teste que põem à prova os conjuntos de condições pré-definidos, bem como pares de definições e uso de variáveis, sendo classificados com base na sua complexidade (Delamaro; Jino; Maldonado, 2013).

Para Delamaro, Jino e Maldonado (2016) os testes estruturais são classificados mediante critérios baseados na complexidade, que utilizam o grau de complexidade do programa para determinar os casos de teste, utilizando a complexidade ciclomática como métrica quantitativa para exercitar testes linearmente independentes do conjunto básico do programa (Delamaro; Jino; Maldonado, 2016). O fluxo de controle utiliza apenas as características de controle presentes na execução do programa, como comandos e desvios, que podem testar todos os nós, arestas e caminhos dos grafos construídos (Delamaro; Jino; Maldonado, 2016). E o fluxo de dados, que utilizam as interações envolvendo variáveis e subsequentes para derivar os casos de teste, sendo indicados para apontar defeitos computacionais (Delamaro; Jino; Maldonado, 2016). Este processo identifica possíveis inconsistências presentes na lógica de programação, que podem ser levados da fase inicial do desenvolvimento até a implantação, para detectar este tipo de falta, são realizados os processos de testes unitários, de integração e os testes de sistema (Galdino, 2010).

Os testes de unidade podem ser utilizados para testar pequenas porções do sistema por vez, destinando seu foco para os menores blocos de construção do programa (Myers; Sandler; Badgett, 2011). A sua execução resulta em um melhor gerenciamento dos elementos para combinação e criação dos casos de teste, além de facilitar as tarefas de depuração dos programas, facilitando a identificação e correção de um erro, uma vez que se sabe sua localização específica.

Os testes de integração visam detectar as falhas relacionadas às interfaces entre os módulos quando os mesmos estão integrados para a construção da estrutura do software (Neto, 2022). Sendo assim, realizam uma verificação conjunta do funcionamento do sistema, para verificar a implementação das funcionalidades, avaliar o comportamento do programa quando processado com grandes volumes de dados, e os submetendo a grandes cargas, para gerar um estresse e verificar o funcionamento simultâneo dos módulos (Myers; Sandler; Badgett, 2011).

Os testes de sistema estão relacionados à detecção de defeitos em toda a aplicação, que inclui a cobertura das funções do programa e casos de uso do usuário, além de cobrir as páginas e *links* (Galdino, 2010). Assim, analisam o software em busca de falhas com base na sua utilização, executando os casos de teste nos mesmos ambientes, condições e dados de entradas que os usuários finais poderiam inserir na utilização diária quanto a manipulação do software, o que verifica a satisfação quanto a completude dos seus requisitos (Neto, 2022).

Além disso, a realização dos testes de aceitação funcionam como uma verificação, simulando operações rotineiras do sistema, para analisar se o seu comportamento está conforme o solicitado, sendo executados por um grupo restrito de usuários finais do sistema (Neto, 2022). Atrelado a isso, os testes de regressão surgem como uma estratégia para a redução de erros iminentes, executados a cada ciclo uma nova rodada de aplicações de todos os testes aplicados anteriormente, podendo ser aplicado em qualquer um dos níveis de teste (Neto, 2022).

Para a detecção de falhas relacionadas à verificação do funcionamento dos requisitos que definem as funcionalidades que o sistema executará, está a realização dos testes funcionais, chamados de testes de caixa preta, por serem derivados dos requisitos e especificações, analisando o sistema externamente, com base no ponto de vista dos utilizadores (Galdino, 2010). Posto isto, a qualidade dos critérios criados para a realização destes testes correspondem a uma boa especificação dos requisitos. Além da possibilidade de aplicação desde as fases iniciais da produção dos sistemas, mesmo que desenvolvidos em qualquer paradigma de programação (Delamaro; Jino; Maldonado, 2016).

A realização dos processos de teste de software corresponde a aspectos críticos que podem

impactar diretamente o funcionamento e a qualidade dos sistemas (Correia; Ferreira; Pires, 2021). Sendo assim, a empregabilidade hábil das metodologias e técnicas de teste facilita a detecção e, consequentemente, a correção de possíveis falhas, que quando descobertas precocemente diminuem os custos de manutenibilidade, promovendo uma verificação dos requisitos funcionais e não funcionais, garantindo a sua conformidade, confiabilidade e segurança (Galdino, 2010).

2.4 Técnicas de Teste Caixa Preta

Os testes de caixa preta são metodologias e técnicas relacionadas à análise externa do sistema (Delamaro; Jino; Maldonado, 2013), são uma investigação no que concerne principalmente à documentação e sua validação, mas também ao que está diretamente relacionado entre a interação do usuário e o produto (Chrisantyo. *et al.*, 2022). Validando esta proposição, o objetivo dos testes caixa preta é a validação das funções do software e comprovação que funcionem como o esperado (Myers; Sandler; Badgett, 2011), sendo utilizados frequentemente para verificar a funcionalidade do sistema, interação com outros sistemas e legitimar seus requisitos funcionais e não funcionais.

Para Khan (2021), existe um conjunto de vantagens a respeito da utilização dos testes caixa preta, como a relação de independência entre o design e o testador, tornando o processo ainda mais importante, visto que os testes são baseados no ponto de vista dos utilizadores dos sistemas. Além disso, pode-se mencionar o fato destes testes serem mais eficazes em unidades de código do que os testes caixa branca, contando ainda com um desenvolvimento mais rápido dos casos de teste. Atrelado a isso, está a frequente falta de acesso ao código-fonte; então, nesses casos, os testadores verificam o sistema com base nas suas especificações e no documento de requisitos, para que assim seja garantida a completude dos requisitos.

O presente trabalho objetifica a utilização das diferentes técnicas de caixa preta para detectar possíveis falhas ou inconsistências nos sistemas. Sendo elas: Particionamento por Equivalência, Análise de Valor Limite, Teste Funcional Sistemático, Grafo de Causa e efeito, Teste Combinatorial, *Error Guessing* e Teste *Fuzzing* (Delamaro; Jino; Maldonado, 2013; Myers; Sandler; Badgett, 2011).

O Particionamento por Equivalência, é uma abordagem que define entradas eficazes e determinadas por válidas e inválidas, dividindo os dados de entrada em uma unidade de software em dados particionados, que, resultantes disso, podem criar casos de teste derivados (Khan, 2021). Durante a aplicação desta técnica, as classes são formadas pelas entradas esperadas pelo

sistema, bem como o comportamento, para ser igual ou semelhante do que foi pré-definido. Dessa forma, uma vez que as classes são definidas, é possível assegurar que qualquer elemento da classe pode ser considerado representante dela, isso se dá ao fato de que eles devem obter um comportamento similar; assim, se um elemento detectar uma falha, os outros também detectam, se não detectar nenhum, os outros também não detectarão (Delamaro; Jino; Maldonado, 2013).

A partir da experiência do usuário, pode-se exemplificar que os casos de testes capazes de explorar os limites das aplicações possuem uma maior probabilidade de detectar possíveis falhas (Myers; Sandler; Badgett, 2011). Nesse contexto, pode-se mencionar a técnica de Análise de Valor Limite, que corresponde aos valores exatamente sobre ou abaixo dos limites pré-definidos no sistema (Delamaro; Jino; Maldonado, 2013).

Os valores limite incluem os máximos, mínimos, somente dentro e fora dos limites, com valores típicos e de erro (Khan, 2021). Com isso, as falhas podem estar presentes nos limites da aplicação, onde o sistema mais costuma falhar, devido aos erros de limite nas classes durante a produção de um sistema (Khan, 2021). Pode ser usado ainda combinado ao Particionamento por Equivalência, mas se difere ao não escolher dados aleatórios da partição, assim, os limites de cada partição são selecionados e explorados, proporcionando uma escolha seletiva dos dados de teste (Delamaro; Jino; Maldonado, 2013).

O Teste Funcional Sistemático, também conhecido como *Edge Testing*, é o nome dado à combinação das técnicas Particionamento por Equivalência e Análise de Valor Limite. Partindo da divisão dos domínios das entradas e saídas, que já haviam sido particionados, essa abordagem necessita de pelo menos dois casos de teste de uma partição para que assim os problemas coincidentes de esconder possíveis falhas sejam minimizados (Delamaro; Jino; Maldonado, 2013). Portanto, o principal objetivo desta técnica torna-se associar os benefícios de uma técnica funcional, independente da implementação, ao bom desempenho de uma técnica de teste funcional, que promove uma maior cobertura, garantindo, assim, uma maior confiabilidade para o sistema (Rocha, 2011).

Além disso, a sistematização pode ser considerada a partir de outras evidências, como: a criação necessária de dois casos de teste de cada partição, minimizando problemas análogos; para os valores numéricos discretos em que devem ser considerados os dados de entrada e saída e casos gerados especificamente para cada classe; e ainda casos de teste que exploram valores ilegais mostram-se necessários para evidenciar ao software o tratamento de desvios de caminhos de sucesso (Rocha, 2011).

Para explorar as ambiguidades e incompletudes das especificações utiliza-se o Grafo de Causa e Efeito (Delamaro; Jino; Maldonado, 2013). Através desse grafo são exploradas as combinações de todos os diferentes tipos de entradas, e assim analisam-se os caminhos que podem ser seguidos e se o sistema se comportará como o esperado, como, por exemplo, a liberação de funcionalidades após a completude do requisito prévio.

Para a construção do grafo, é importante seguir o critério de alguns passos, como: a divisão por partes das especificações do sistema, identificando suas causas e efeitos e analisar a semântica da especificação e transformá-la em um *booleano*, para os relacionar; a conversão para uma Tabela de Decisão, que também se baseia na criação de diferentes cenários e combinações, exemplificando os resultados para cada ação esperada; e, posteriormente, transformá-los em casos de teste (Delamaro; Jino; Maldonado, 2013). Consequentemente, visa a importância da construção de cenários para analisar o sistema e garantir o funcionamento conforme solicitado na documentação.

A técnica de Teste Combinatorial também está relacionada às combinações de cenários e entradas possíveis do sistema, que busca diferentes possibilidades de combinações (Delamaro; Jino; Maldonado, 2013), como ferramenta para encontrar inconsistências. Baseia-se principalmente nos resultados obtidos entre a interface de comunicação e o usuário. O teste baseado na interface gráfica do usuário, tem o objetivo de assegurar que o software irá apresentar determinada interface como resposta para alguma ação, dessa forma, a geração dos dados de entradas para esses testes torna-se uma atividade mais complexa e que obtém diferentes combinações (Delamaro; Jino; Maldonado, 2013).

Utilizando uma técnica *ad hoc* como o *Error Guessing*, é possível analisar e descobrir possíveis falhas por meio da expertise do testador, que mediante seu conhecimento prévio se torna capaz de deduzir alguns erros, da mesma forma que definir casos de teste para detectá-los (Delamaro; Jino; Maldonado, 2013). Essa técnica proporciona uma análise baseada na exploração, em que o testador supõe em que partes o sistema possa conter erros coincidentes e busca solucioná-los.

Para descobrir falhas do processo de implementação, utiliza-se a técnica de Teste *Fuzzing*, que funciona mediante uma injeção de dados que sejam mal formados ou semi-formados, além da sua utilização para detectar problemas relacionados à segurança (Khan, 2021). Esta técnica divide-se entre os *Fuzzers* baseados em mutação, que modificam a amostra de dados existentes para criar os dados de teste, e os *Fuzzers* baseados em geração, que definem novos dados de teste

com base nos modelos de entrada, em que este tipo de teste é utilizado para encontrar erros como falhas de asserção e vazamentos da memória, que podem existir na difusão de um protocolo (Khan, 2021).

Com o embasamento das técnicas abordadas, este estudo promoverá uma discussão acerca da sua utilização, relacionadas à sua metodologia e aplicações.

2.5 Testes de Usabilidade

Ao longo dos últimos anos foram desenvolvidas técnicas enumeradas para facilitar a interação entre os humanos e as suas máquinas, centrados na experiência dos usuários para desenvolver técnicas de design, devendo ser consideradas durante todo o ciclo de vida do software (Seffah; Metzker, 2004). Atrelado a isso, estão os testes de usabilidade, formados por um processo no qual participantes representativos avaliam o grau em que o produto em questão se encontra em relação aos critérios específicos relacionados à usabilidade, evidenciando a preocupação em minimizar custos futuros, uma vez que se tornam mais eficientes quando implementados durante todas as fases de desenvolvimento (Ferreira, 2002).

A implementação destes testes é utilizada para analisar diferentes métricas, que envolvem tipos de tarefas, medidas de desempenho e disposição de escalas, além de entrevistas e inspeções aplicadas ao processo, no intuito de detectar problemas quanto à usabilidade dos sistemas (Ferreira, 2002). A realização ocorre independente do código-fonte, consistindo em análises técnicas para garantir o bom entendimento sobre o funcionamento do sistema. Para Seffah e Metzker (2004), a usabilidade pode ser definida como a capacidade que um software tem de ser compreendido, aprendido, usado e atraente para os usuários, quando usados sob condições específicas. Isso objetifica a extensão na qual o sistema pode ser usado de forma que atinja aos objetivos especificados, mantendo padrões de eficácia, eficiência e satisfação.

Para os usuários finais dos sistemas, a usabilidade é essencial para medir o desempenho, satisfação e produtividade, permitindo que os mesmos executem tarefas de forma mais eficiente (Seffah; Metzker, 2004). Isso destaca a importância destes testes como parte do processo, não devendo ser descartados com vias de redução de custos a curto prazo (Ferreira, 2002); uma vez que isso ocorra, posteriormente, podem ser gerados custos adicionais de manutenção para corrigir defeitos que poderiam ter sido evitados com a priorização dos testes de usabilidade.

A elaboração e realização dos testes de usabilidade aplicados a sistemas são um potente indicador de defeitos e para expor erros em potencial, minimizando os riscos de que seja

disponibilizado ao mercado um produto instável e de difícil entendimento (Ferreira, 2002), tendo em vista que os aspectos de usabilidade são um elemento crítico para o mercado tecnológico, bem como um elemento imprescindível em um projeto.

2.5.1 Heurísticas de Nielsen

Segundo Nielsen (2007), os primeiros testes realizados com usuários em análise da usabilidade foram efetuados em 1994. A partir disso, foram identificados milhares de problemas relacionados a utilização de sistemas e desenvolveram-se diretrizes para evitá-los (Nielsen, 2007). Para isso, é importante destacar que os aspectos de usabilidade não são uma propriedade única e unidimensional das interfaces de usuário (Nielsen, 1994).

Os componentes que a constituem são: **aprendizagem**, em que o sistema deve ser fácil de aprender e utilizar, para o usuário utilizar o sistema de forma mais rápida e eficiente; **a eficiência**, significa que o sistema deve ser eficiente para utilização, para elevar o nível da produtividade; **memorabilidade**, em que o sistema deve ser fácil de lembrar, para os usuários conseguirem lembrar casualmente do seu funcionamento, mesmo após um tempo sem utilizar; **os erros**, em que o sistema deve possuir uma baixa taxa de erros, para serem rapidamente recuperados de forma fácil; **satisfação**, em que o sistema deve ser agradável de usar e os usuários gostem subjetivamente de utilizá-lo (Nielsen, 1994).

Nielsen (1994) desenvolveu heurísticas para avaliar a usabilidade de interfaces, descritas no Quadro 1. Estas heurísticas podem ser redesenhadas para atender às métricas de qualidade relacionadas a usabilidade de sistemas computacionais (Cruz; Neto, 2015). Utilizando esses princípios, é possível obter uma perspectiva mais crítica quanto aos aspectos de interação entre os seres humanos e computadores, dando a devida importância a construção de sistemas que sejam agradáveis, intuitivos e eficientes para os seus usuários.

Os estudos de Nielsen (1994), mostram com perspicácia como os profissionais da computação podem ser capazes de aplicar métodos para avaliar interfaces de usuário, a partir de um treinamento, e que mediante experimentos metodológicos pode-se encontrar muitos problemas relacionados à usabilidade.

Desse modo, utilizando o conjunto de heurísticas designados para avaliar as métricas de usabilidade de interfaces de usuário, garante um impacto direto em relação à qualidade desempenhada pelo software. Assim, a usabilidade se aplica a todos os aspectos de um sistema aos quais um ser humano pode interagir, incluindo os procedimentos de instalação, utilização e

Quadro 1 – Heurísticas de Nielsen para Avaliar Métricas de Usabilidade

Heurística	**Descrição**
Visibilidade do status do sistema	A interface deve informar ao usuário o que está acontecendo, utilizando elementos como barras de progresso, mensagens claras e símbolos universais.
Correspondência entre o sistema e o mundo real	O sistema deve falar a linguagem do usuário, evitando termos e referências mais técnicas. Considerando o modelo mental do público-alvo e, se necessário, oferecer interfaces distintas para diferentes perfis de usuário.
Liberdade e controle do usuário	O usuário deve poder cancelar ações e retornar a estados anteriores quando desejar. Caso isso não seja possível, o sistema deve explicar os motivos da restrição.
Consistência e padronização	Ícones, comandos, cores, posições e termos devem seguir padrões consistentes, preferencialmente alinhados ao sistema operacional, para garantir previsibilidade nas interações.
Prevenção de erros	Devem ser utilizados mecanismos para evitar erros comuns, como mensagens de confirmação, formatos obrigatórios e preenchimento automático.
Reconhecimento em vez de memorização	O sistema deve fornecer elementos visuais e comandos acessíveis que reduzam a necessidade de o usuário lembrar informações específicas.
Flexibilidade e eficiência de uso	Interfaces devem ser intuitivas para usuários iniciantes e permitir atalhos ou comandos mais rápidos para usuários experientes, promovendo ganho de produtividade.
Estética e design minimalista	A interface deve conter apenas os elementos essenciais, evitando informações excessivas ou desnecessárias que possam distrair ou confundir o usuário.
Ajudar usuários a reconhecer, diagnosticar e corrigir erros	Mensagens de erro devem ser claras, indicar o problema e sugerir soluções, sem intimidar ou confundir o usuário.
Ajuda e documentação	Mesmo sendo ideal que o sistema seja autoexplicativo, deve-se disponibilizar documentação acessível e objetiva para auxiliar o usuário em caso de dúvidas.

Fonte: Adapdado de Cruz e Neto (2015)

manutenção (Nielsen, 1994).

2.6 Eficácia dos Testes Funcionais

Segundo Myers, Sandler e Badgett (2011), a eficácia dos testes está relacionada à habilidade de descobrir falhas, que poderiam passar despercebidas e afetar o desempenho do software em desenvolvimento. A eficácia dos testes funcionais pode ser medida pela metodologia e critérios que identificam problemas externamente (Delamaro; Jino; Maldonado, 2013).

Para avaliar o quão eficaz pode ser a implementação dos mesmos, é necessário realizar uma combinação de etapas envolvendo a garantia da completude dos requisitos e das especificações, somada à capacidade de identificar problemas, especialmente na interação entre o usuário e o sistema (Pressman; Maxim, 2021). Assim, faz-se indispensável a utilização de métricas para garantir seu bom funcionamento, como a cobertura de requisitos e dos caminhos, e a validação da documentação, a base principal destes tipos de testes (Delamaro; Jino; Maldonado, 2013).

O presente estudo fundamenta-se principalmente no trabalho de Delamaro, Jino e Maldonado (2013), que dedicam um capítulo aos testes funcionais em que expõem a ideia central e destacam suas vertentes. Os autores destacam como os testes funcionais impactam no processo de detecção de falhas e, assim, são fundamentais para assegurar que o sistema cumpre seus requisitos especificados (Delamaro; Jino; Maldonado, 2013).

Mediante a contribuição de Myers, Sandler e Badgett (2011), que enfatiza a importância da aplicação das técnicas de teste caixa preta diante do estudo e exploração das melhores estratégias e técnicas de teste de software. Os autores também definem os testes funcionais como peça essencial para a construção de uma base sólida para garantir a validação dos sistemas, destacando o aspecto do atributo de qualidade.

Delamaro, Jino e Maldonado (2013) abordam desde os conceitos iniciais de testes de software até seu desempenho na aplicação; para este estudo, ressaltam-se os aspectos teóricos no que se refere à contextualização de termos para entendimento e diferenciação de erros, falhas e problemas, além da validação do uso de metodologias de teste. Este estudo traz também destaque aos testes funcionais, descrevendo-os como um aspecto crítico para atestar um produto, salientando sua importância para a cobertura dos requisitos e especificações essenciais para validar o comportamento do sistema.

Sommerville (2018) explora a necessidade do teste extensivo das funcionalidades do sistema com base nos seus requisitos e especificações, enfatizando a aplicabilidade dos testes funcionais desde o início do desenvolvimento, dos testes de unidade até os testes de aceitação, apresentando seu andamento ao longo do processo.

A norma ISO/IEC (2017) fornece a compreensão da estrutura dos processos de ciclo do software, o que a torna fundamental para a padronização e melhoria contínua da qualidade desses processos, garantindo a realização eficiente das atividades.

Pressman e Maxim (2021), abordam o estudo das métricas de testes, que incluem a cobertura dos requisitos e caminhos que avaliam a eficácia e o desempenho do funcionamento

do sistema, bem como dos testes realizados, enfatizando a importância da documentação e sua validação como garantia de que o sistema atenda a todos os requisitos pré-definidos, evidenciando os funcionais que são os contribuintes com o presente estudo. Trazendo uma descrição dos testes funcionais baseados nos requisitos e especificações, ou seja, analisando a estrutura externa do programa e a interação com o usuário. Ressaltam a importância de validar as funcionalidades conforme os diferentes níveis de verificação; para eles, primordial para o software funcionar como o esperado.

3 TRABALHOS RELACIONADOS

Este capítulo, exibe os trabalhos acerca de teste de software que contribuíram de alguma forma para esta pesquisa. Estes estudos se destacaram pela sua metodologia e conteúdos abordados, sabendo da existência de uma ampla gama de vertentes da área de testes, apresentando argumentações e abordagens no que diz respeito à garantia de qualidade dos testes funcionais e assim abordar a sua eficácia na detecção de falhas. Os trabalhos foram categorizados conforme as linhas de pesquisa.

3.1 Processo de Teste de Software

Myers, Sandler e Badgett (2011) exploram as melhores estratégias e abordagens de teste de software, incluindo a aplicação de técnicas de teste caixa preta. Os autores enfatizam a importância dos testes funcionais para construir uma base sólida para concluir a validação dos sistemas conforme os requisitos especificados, a fim de garantir sua completude.

O processo de teste de software torna-se ainda mais necessário durante as fases de reparo do sistema, para que assim as mudanças e melhorias não afetem as funcionalidades já definidas (Chrisantyo. *et al.*, 2022). Em sua pesquisa do "Sistema de Informação Agrícola", que inclui aplicações de coleta de dados para agricultores e atividades agrícolas, Chrisantyo. *et al.* (2022), apontam que as tecnologias testadas durante o desenvolvimento podem se mostrar divergentes quando testadas com os usuários, nesse sentido, a implementação dos testes caixa preta surge como uma solução para satisfazer a completude das funcionalidades e gerar uma boa experiência para quem os utiliza. Com isso, a taxa de sucesso aumenta e, ao considerar o trabalho conjunto entre a equipe de desenvolvimento e os usuários, como forma de interação, os testes tendem a aumentar a taxa de sucesso para cerca de $11,79\%$, em que a dinâmica do desenvolvimento pode revelar problemas, que podem estar relacionados à programação ou ferramentas de desenvolvimento.

A visão geral e as estratégias das diferentes técnicas de teste de software também são exploradas por AbuSalim, Ibrahim e Wahab (2020), especificamente para aplicativos móveis. Comparativamente, o trabalho em questão contribui para este estudo ao analisar as técnicas de teste caixa preta, somado a um resumo de aplicação das mesmas e suas respectivas descrições, objetivando a utilidade dos testes funcionais em sistemas maiores devido ao seu atendimento para com a completude dos requisitos do aplicativo ou software do usuário.

A realização dos testes de software faz uma verificação por meio das suas técnicas e estratégias, visando simular a aplicação das funcionalidades, exibindo o máximo de situações possíveis, para determinar se um sistema funciona conforme as suas especificações (Galdino, 2010). Estas atividades englobam duas atividades principais para desenvolver os testes, a verificação, para garantir que a aplicação corresponde às suas funções específicas, e a validação, que assegura que os requisitos implementados sejam correspondentes aos definidos pelos clientes (Galdino, 2010).

Para consolidar o seu estudo, Galdino (2010) utiliza os testes de caixa preta, especificamente os testes de desempenho, de carga, estresse, compatibilidade, usabilidade e segurança, e acessibilidade. Além disso, há a utilização dos testes caixa branca, que são estruturais e se baseiam na estrutura interna do sistema, implementando testes unitários, em pequenas porções de código por vez; os testes de integração para verificar um conjunto de páginas da aplicação, para verificar que funcionam em conjunto; e os testes de sistema, que buscam detectar defeitos em toda a aplicação (Galdino, 2010).

Desse modo, ocasionando uma nova perspectiva acerca da utilização de ferramentas para testes, que atestem a qualidade e facilitem o processo de implementação. Objetificando a automação de testes, como uma alternativa para a redução de custos e para gerar um aumento na produtividade ao longo do desenvolvimento, resultando em uma maior qualidade do produto final (Galdino, 2010).

3.2 Testes Funcionais

Kaner, Falk e Nguyen (2011) ressaltam a descrição dos testes funcionais como sendo baseados nos requisitos e especificações, reforçando a verificação da documentação a fim de garantir que as funções esperadas estão sendo executadas. Os autores argumentam, ainda, que os testes funcionais são um ponto crucial para garantir a confiabilidade e a qualidade de um software, contribuindo também para o aproveitamento correto das práticas de verificação e validação na análise de softwares em desenvolvimento.

Rocha (2011) traz aspectos que objetivam uma melhor seleção de casos de teste, aumentando assim a qualidade dos produtos e o desempenho de roteiros de teste. Aborda também conceitos importantes acerca de teste de software, testes funcionais e qualidade de software, ressaltando a importância desses tópicos para assegurar o funcionamento de um sistema e seu desempenho.

Jacob e Prasanna (2016) analisam as metodologias de teste caixa preta, como análise de valor limite e particionamento de classes por equivalência, aplicadas em uma página Web, identificando pontos positivos e negativos de cada técnica. Os autores identificam limitações em ambas as metodologias: a análise de valor limite deve ser executada em ambientes que tenham limites bem definidos, enquanto, o particionamento é mais eficiente em ambientes com limites variáveis. Dessa forma, a utilização em conjunto das abordagens trará casos de testes com maiores índices de eficiência. Assim, o trabalho contribui para o presente estudo com respeito à base sobre as técnicas diante das limitações já reconhecidas.

A análise dos testes de caixa preta para Khan (2021) contribui para essa pesquisa diante da análise dos diferentes tipos de técnicas, como o particionamento por equivalência, análise de valor limite, grafo de causa e efeito, teste de matriz ortogonal, testes de todos os pares e teste de transição de estado. O autor parte de uma contextualização e, posteriormente, discute a aplicação individual das técnicas, atrelada a exemplos de possíveis entradas e saídas de dados a cada utilização. Assim, promove um estudo sobre técnicas de teste caixa preta e os cenários aos quais podem estar inseridas.

Mahendra e Asmarajaya (2022) fornecem uma investigação das técnicas de teste caixa preta no sistema “Kidung Dharma Gita Sekar Madya”, no qual utilizam 16 casos de teste para colocá-lo em prova. Baseando-se em estudos de caso e instrumentos de usabilidade, seu estudo possibilita a análise de aplicações reais das técnicas funcionais, em que são mostradas as descrições e resultados esperados de cada um dos casos de teste. Os autores apresentam as expectativas para o sistema funcionar adequadamente e assim exibir a contribuição e o nível de eficácia dos testes caixa preta para detectar erros e melhorar a qualidade de software, e promover a facilidade da sua manutenção.

O estudo centralizado em uma amostra estratégica dos testes caixa preta através da criação de modelos com diferentes entradas para os serviços, e funcionamento da modelagem do usuário, é promovido por Lee *et al.* (2013). Os autores realizam uma comparação dos resultados das recomendações de filmes, mostrando que as abordagens propostas podem revelar com precisão os esquemas de recomendações de sua pesquisa, sintetizando vários modelos de usuários, objetificando a importância da análise de um sistema a partir da perspectiva de quem está utilizando, operando comparações mediante dados externos, devido à falta das informações internas do sistema.

Sutiah. e Supriyono (2021) utilizam em sua pesquisa os testes funcionais executados no

aplicativo de ensino à distância “E-Learning Madrasah”, o qual é um aplicativo para produtos que possuem o objetivo de incentivar a aprendizagem. Os autores empregaram as técnicas de caixa preta: particionamento por equivalência, análise de valor limite, teste de comparação, amostra e robustez, testes de comportamento e desempenho, de requisitos e resistência, e o grafo de causa e efeito, utilizados para testar o nível de funcionalidades de software com necessidade de atingir 90% de cobertura, a fim de melhorar a sua capacidade. Os autores incluíram cenários de testes como os de *login* e configuração de classe, além de apresentar a metodologia de análise de identificação dos requisitos do sistema, em que há determinação dos cenários de teste. Com os esforços realizados, os testes no aplicativo tornaram-se referência na utilização e aplicação dos testes funcionais, bem como na garantia em atestar a sua precisão.

Entre o conjunto de técnicas de teste existentes, voltando-se ao âmbito dos testes caixa preta, Chen *et al.* (2018) realizam uma análise da melhoria da técnica de *fuzzing* desde a sua criação, na tentativa de descobrir *bugs* críticos ou vulnerabilidades de software. Dessa forma, o estudo aborda as melhorias na eficiência do *fuzzing* a partir do aumento do número de sistemas desde a primeira ferramenta de *fuzzing* desenvolvida por Miller em 1990. Assim, Chen *et al.* (2018) reúne um conjunto de técnicas úteis aplicadas ao teste *fuzzing*, no intuito de evidenciar os princípios, vantagens e desvantagens do método referenciado. Com isso, destaca-se o desenvolvimento do *fuzzing* de caixa preta original para o *fuzzing* de caixa branca e o *fuzzing* de caixa cinza, assim como é retratada a combinação de novas técnicas na superação de dificuldades iniciais que contribuem para a detecção de *bugs* de segurança e vulnerabilidades nos softwares.

A partir da elaboração dos primeiros conceitos de qualidade, os desenvolvedores e testadores trabalham juntos do início da produção do produto até a sua conclusão, em que ao final das fases de desenvolvimento eram realizadas atividades de conferência. Correia, Ferreira e Pires (2021) definem que erros geram falhas, resultando em comportamentos inesperados e afetam diretamente o usuário final, o que pode inviabilizar a utilização de um software.

Em seu estudo, determinam elementos essenciais para a realização dos testes de software como garantia de qualidade em um sistema, sendo elas: a criação de casos de teste para estabelecer condições particulares, os procedimentos de teste, que descrevem o passo a passo da sua execução; e os critérios de teste, que selecionam e avaliam os casos de teste para que as possibilidades de geração de falhas sejam ampliadas (Correia; Ferreira; Pires, 2021).

Dessa forma, Correia, Ferreira e Pires (2021) utilizam testes de usabilidade para identificar falhas funcionais, que não estão conforme os requisitos especificados durante o planejamento.

Mediante o embasamento teórico e a execução técnicas de verificação e validação, os autores contribuem para este estudo através da evidenciação da importância de garantir a completude dos requisitos, utilizando métodos de melhorias para os processos de software em busca pela qualidade. Além da ênfase na necessidade da aprendizagem em particular acerca dos testes de caixa preta, que podem contribuir para melhorar a qualidade de um software.

3.2.1 Teste Caixa Preta e Teste Caixa Branca

Khan e Khan (2012) fazem um estudo comparativo entre estratégias caixa branca e caixa preta, além de técnicas de testes de caixa cinza. Os autores ressaltam a importância da utilização dos testes funcionais, que, quando comparados com os demais, possuem diferentes atribuições, que podem ser menos demoradas e detectar erros coincidentes com mais agilidade, além da baixa granularidade, sendo fatores contribuintes para um bom desempenho do processo de teste.

Assim, promovem um estudo sobre técnicas de teste caixa preta e os cenários aos quais podem estar inseridos, como uma afirmativa da qualidade do sistema. Os autores destacam que estes testes buscam testar sistematicamente o software em cenários controlados, visando identificar a integridade, bem como a correção do software, e como consequência detectar erros que passaram despercebidos. Para isso, utilizam as técnicas funcionais para examinar os aspectos fundamentais do sistema, as quais são mais eficientes para grandes segmentos de código, e soma uma perspectiva para além da de desenvolvimento (Khan; Khan, 2012).

3.3 Análise da Usabilidade de Sistemas

Em seu trabalho, Cruz e Neto (2015) propõem uma análise sob a perspectiva de um redirecionamento acerca das heurísticas de Nielsen (1994), fazendo um redesenho dos seus princípios para avaliar as interfaces de sistemas computacionais, bem como o seu comportamento quando postos em interação com os usuários finais. Isso promove um estudo atualizado acerca da aplicabilidade das heurísticas em cenários mais atuais, nesse caso, para jogos. Submetendo a avaliações para os artefatos serem analisados em completude com os princípios da engenharia de requisitos, dessa forma, se tornam uma alternativa caixa preta para a avaliação dos sistemas, visto que não há a necessidade de acesso ao código-fonte (Nielsen, 2007).

Para Martins *et al.* (2013), existem diferentes formas de avaliar a usabilidade, em que alguns modelos se baseiam na análise do sistema ou produto por especialistas da área, e outros

modelos são analisados de forma empírica, mediante dados coletados dos utilizadores reais de um sistema. Em seu estudo, visam a avaliação da usabilidade dividida em três partes principais, são elas: o teste, o inquérito, a experiência controlada e inspeção. Assim, é construída uma visão mais global acerca dos modelos, métodos, técnicas e avaliações de usabilidade. Em que o modelo empírico, que contempla o teste, inquérito e experiência são frequentemente mais utilizados como fonte para reconhecimento de métricas de usabilidade; sendo o teste, a técnica mais utilizada o âmbito de avaliação de desempenho.

3.4 Comparação Entre a Aplicação das Técnicas nos Trabalhos

Comparando os trabalhos relacionados com o presente estudo, observa-se que os artigos em questão fornecem visões gerais quanto à qualidade e à confiabilidade de software, mas também oferecem metodologias e aplicações detalhadas dos testes de caixa preta. Este trabalho aspira expandir essas abordagens das técnicas a serem utilizadas, aplicando as mesmas a um estudo de caso real em que o objetivo é estabelecer um panorama das estratégias e metodologias e seu comportamento em diferentes situações.

Como consequência da análise das técnicas utilizadas nos trabalhos relacionados, está a criação do Quadro 2 que ressalta o nível de aplicação e utilização das técnicas como sendo baixo, médio e alto que descreve a classificação da realização das técnicas. Por fim, também demonstra as vantagens e desvantagens da utilização das técnicas.

Quadro 2 – Comparação entre as técnicas mais utilizadas pelos autores nos trabalhos relacionados

Abordagem	**Aplicação das técnicas nos Estudos**	**Vantagens**	**Desvantagens**
Particionamento por Equivalência	Alto	Reduzir o número de casos de teste	Pode não cobrir todos os casos
Análise de Valor Limite	Alto	Foca nas áreas críticas	Pode ignorar casos internos
Error Guessing	Médio	Simplicidade e flexibilidade	Subjetivo, sem formalização
Grafo de Causa e Efeito	Médio	Identifica causas e efeitos	Pode ser complexo de desenhar
Teste *Fuzzing*	Baixo	Encontra falhas de implementação	Requer recursos computacionais elevados
Teste de Usabilidade	Médio	Foca na experiência e interação do usuário com o sistema	Difícil de automatizar
Presente trabalho	Alto	Contempla a análise das principais técnicas de teste caixa preta	Processo não automatizado

Fonte: Autoria própria (2025)

4 METODOLOGIA

O resultado desta pesquisa fundamenta-se em levantamentos bibliográficos sobre testes funcionais e qualidade de software. Utilizando os referenciais teóricos, esta pesquisa abrange a investigação de diferentes metodologias, técnicas e abordagens de testes funcionais. Discutir as estratégias e metodologias dos testes de caixa preta e sua adequação a diferentes tipos de sistemas e especificações, significa contribuir com ferramentas hábeis para solucionar problemas e garantir qualidade e confiabilidade aos sistemas.

Atrelado a isso, quando bem executados, esses testes podem conseguir identificar uma multiplicidade de falhas, em especial, as que estão relacionadas à interação com os usuários e à integração de sistemas (Kaner; Falk; Nguyen, 2011). Para consolidar estes resultados, foi desenvolvido um conjunto de atividades que reafirmam o embasamento teórico e prático. Sendo elas: o levantamento bibliográfico (Seção 4.1); o desenvolvimento dos sistemas para análise, (Seção 4.2); o desenvolvimento dos casos de teste (Seção 4.3); e a aplicação das técnicas caixa preta (Seção 4.4).

4.1 Levantamento Bibliográfico

Durante esta atividade, foram analisados trabalhos que abordassem conceitos de teste de software e, especialmente, os testes funcionais. Além disso, buscou-se investigar linhas de pesquisa com técnicas e metodologias já utilizadas e os resultados apresentados da sua atuação, para que assim esta pesquisa contribuísse com novos dados e aplicações específicas.

Para buscar artigos e trabalhos relacionados, foi conduzida uma análise em bases de dados tais como *Google Scholar*[1], Portal de Periódicos da CAPES[2] e *IEEE Explore*[3]. As strings de busca utilizadas foram buscadas individualmente: “Black Box”, “Black Box Testing”, “Failures”, “Software Testing” e “Teste de Software”. Para a escolha dos artigos foi considerado o resumo do trabalho e sua coesão com o tema da presente pesquisa, de forma que abordassem aspectos de qualidade e teste de software, especialmente, os testes funcionais.

Com essa etapa foi possível, além de explorar e reunir assuntos relevantes para a linha de pesquisa, definir o andamento e necessidade de cada metodologia de teste, lidando com os prós e contras identificados. Desse modo, há a geração de artefatos constituídos pelos casos de

[1] Disponível em: https://scholar.google.com. Acesso em: 01 jul. 2025
[2] Disponível em: https://www-periodicos-capes-gov-br.ez13.periodicos.capes.gov.br. Acesso em: 01 jul. 2025
[3] Disponível em: https://ieeexplore.ieee.org. Acesso em: 01 jul. 2025

testes que têm como propósito a disponibilidade e demonstração dos testes para futuros estudos e pessoas que desejam visualizar os pontos apresentados no presente estudo.

Em linhas gerais, esta etapa visou identificar trabalhos e metodologias que fossem atrativas para o resultado positivo do trabalho. Dessa forma, foi possível, subsequente a essa etapa, dar início ao desenvolvimento dos casos de teste.

4.2 Desenvolvimento dos Sistemas Analisados

Nesta etapa, foram desenvolvidos os sistemas para submeter para avaliação utilizando as técnicas de teste caixa preta, utilizando as linguagens de programação Java, *JavaScript*, além da linguagem de marcação HTML e o CSS, Para a construção dos sistemas houve uma reflexão sobre contextos de utilização cotidiana, em situações comumente executadas no dia a dia das pessoas em contato com recursos tecnológicos.

A partir disso, foram desenvolvidos sistemas que simulassem necessidades cotidianas dos usuários, como a capacidade de inserção de dados corretos para validar e-mail e senha, a realização de cálculos matemáticos, e ainda a utilização de sistemas que prestam serviços, como bibliotecas e aeroportos. Com isso, podemos ilustrar a utilização das técnicas de teste caixa preta, bem como o seu desempenho individual e combinado. Isso propõe uma análise mais crítica acerca sobre como o mau funcionamento dos sistemas pode afetar a utilização por parte de usuário, investigando os possíveis defeitos quanto à garantia da completude da métrica de qualidade, de adequação funcional e de proteção a erros do usuário.

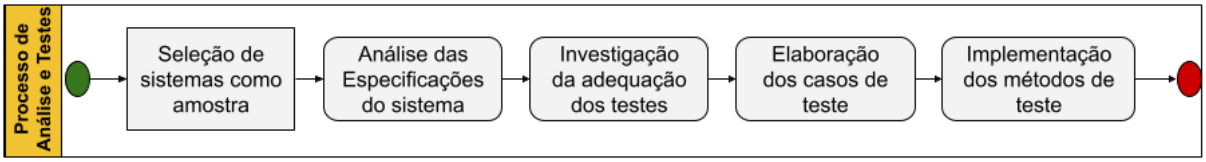
4.3 Desenvolvimento dos Casos de Teste

Esta atividade consistiu na criação de diferentes casos de teste, baseados nos requisitos e especificações dos sistemas selecionados como amostras, e utilizados na pesquisa como material de apoio para uma análise panorâmica na qual as estratégias sejam colocadas em prática. Este processo promoveu uma exploração quanto à adequação dos diferentes testes no cenário imposto, investigando como cada método pode ser implementado individualmente, ilustrados na Figura 1.

4.4 Aplicação das Técnicas Caixa Preta

Os métodos de pesquisa utilizados neste estudo envolvem a aplicação das técnicas de teste caixa preta em sistemas previamente selecionados. Os sistemas selecionados foram submetidos

Figura 1 – Processo de Análise e Testes



Fonte: Autoria própria (2025)

a um processo de análise de desempenho de qualidade por meio dos testes funcionais, ou seja, sem acesso ao código-fonte, realizados somente mediante a interface de comunicação. A partir dessa avaliação, foi possível visualizar qual técnica é mais adequada para cada tipo de situação específica, para que assim exista a completude dos seus requisitos, bem como o seu funcionamento.

A respeito das técnicas utilizadas, o Particionamento por Equivalência divide o domínio de entradas em classes de equivalência que possuem entradas válidas e inválidas, sendo tradadas da mesma maneira, segundo as especificações; portanto, se um elemento detectar um defeito, o outro também irá (Delamaro; Jino; Maldonado, 2013). A Análise de Valor Limite, por sua vez, visa casos de teste com maior probabilidade de encontrar defeitos, utilizando valores que podem estar acima ou abaixo do limite pré-definido (Delamaro; Jino; Maldonado, 2013). A combinação dessas técnicas gera o Teste Funcional Sistemático (Delamaro; Jino; Maldonado, 2013), que utiliza duas metodologias para detectar erros.

Atrelado a elas, está o *Error Guessing* que consiste em uma análise minuciosa com base na experiência do testador (Delamaro; Jino; Maldonado, 2013). A técnica de Grafo de Causa e Efeito, explora possíveis inconsistências do sistema; dessa forma, indicando as ações conforme as especificações do programa, assim trilhando um caminho de decisões a serem tomadas (Delamaro; Jino; Maldonado, 2013). A técnica de Tabela de Decisão permite a exibição do comportamento do sistema perante a combinação de diferentes condições pré-definidas e entradas (Delamaro; Jino; Maldonado, 2013). O Teste Combinacional também se mostra útil para realizar testes em sistemas com múltiplos cenários de diferentes entradas e ações, colocando em prova as combinações possíveis (Kaner; Falk; Nguyen, 2011).

O teste *fuzzing* diz respeito a aplicação de uma injeção de dados no sistema, que podem ser mal formados ou semi-formados. Esta técnica pode detectar problemas relacionados a implementação do código-fonte ou de questões relacionadas a segurança, além da percepção de falhas de asserção e vazamentos de memória.

Como técnica complementar, está a realização dos Testes de Usabilidade, baseadas nas

heurísticas de Nielsen (1994), que constituem princípios para a garantia de qualidade em relação à interação entre os usuários e o sistema, em que as interfaces devem ser agradáveis, de fácil utilização e eficientes. Contribuindo para detectar possíveis defeitos que violem as métricas de qualidade atribuídas ao sistema, para que atenda com eficiência os seus requisitos funcionais e não funcionais.

O quadro 3, representa de forma mais geral, mediante uma ilustração, a descrição para cada uma das técnicas utilizadas neste estudo.

Quadro 3 – Técnicas de Teste Caixa Preta

Técnica	**Objetivo Geral**
Particionamento por Equivalência	Divide entradas em classes de equivalência, dividindo-as em valores válidos e inválidos.
Análise de Valor Limite	Testa valores nos limites mínimos e máximos das entradas, nas extremidades dos sistemas
Teste Funcional Sistemático	Combina particionamento por equivalência e análise de valor limite para potencializar a análise de acordo com partições e limites.
Error Guessing	Baseia-se na experiência do testador para identificar pontos em que o sistema pode falhar.
Grafo de Causa e Efeito	Modela a lógica do sistema com base em causas (entradas) e efeitos (saídas), mapeando as tomadas de decisão.
Tabela de Decisão	Representa combinações de condições e as ações correspondentes.
Teste Combinacional	Verifica o comportamento do sistema sob diferentes combinações de entradas múltiplas.
Fuzzing	Injeta dados semi-formados ou malformados para descobrir falhas, especialmente de asserção e segurança.
Teste de Usabilidade	Avalia o nível de satisfação e a experiência do usuário com base em heurísticas de usabilidade.

Fonte: Adaptado de Delamaro, Jino e Maldonado (2016) e Nielsen (1994).

5 RESULTADOS

A investigação e aplicação das técnicas de teste caixa preta nos diferentes cenários descritos neste capítulo, exploram, além da sua capacidade na detecção de falhas, a capacidade de uma visualização panorâmica sobre como cada técnica se adequará a sistemas de diferentes naturezas. Neste estudo foram realizados testes em sistemas com diferentes características e complexidades, sendo eles: um sistema para validação de senhas; sistema para validação de e-mail; sistema de cálculo de matrizes; sistema de reserva de voos; sistema para verificação de palavras palíndromas; sistema para agenda de contatos; sistema para utilitário de calculadora; e um sistema de empréstimo de livros, além da análise da usabilidade como técnica caixa preta, nos sistemas de: formulário, biblioteca digital e plataforma de cursos on-line.

Os sistemas foram construídos exclusivamente para a realização deste estudo, podem ser acessados no repositório da pesquisa[4], o qual contém os códigos utilizados para a avaliação das técnicas durante o desenvolvimento deste estudo. Esses sistemas, foram organizados conforme a utilização das técnicas aos quais foram submetidos para investigação da sua eficácia.

5.1 Particionamento por Equivalência

Nesta seção, estão dispostos os sistemas em que foram aplicadas a técnica de particionamento por equivalência, determinando as suas classes válidas e inválidas para análise.

5.1.1 Validação de Senhas

No sistema referente a um validador de senhas, foi aplicada a técnica de Particionamento por Equivalência (Delamaro; Jino; Maldonado, 2013). Esta técnica se mostra eficaz na divisão das entradas entre as classes válidas e inválidas, que possuem algumas restrições, são elas: incluir letras maiúsculas, minúsculas, dígitos e caracteres especiais. Com isso, foram definidas cinco classes baseadas nos critérios de aceitação pré-definidos, permitindo uma exploração mais rápida de erros que poderiam estar relacionados às senhas violando os padrões definidos nos requisitos. Para realizar a verificação, foi desenvolvido o laço disponível no Código-fonte 1.

Código-fonte 1 – Laço de Repetição para Validação de Senhas

```
1 for (char c : senha.toCharArray()) {
```

[4] Disponível em: https://github.com/clarameira/BlackBoxTestingOverview. Acesso em: 23 jun. 2025

```
2       if (Character.isUpperCase(c)) {
3           temLetraMaiuscula = true;
4       } else if (Character.isLowerCase(c)) {
5           temLetraMinuscula = true;
6       } else if (Character.isDigit(c)) {
7           temDigito = true;
8       } else {
9           if ("!@#$%^&*()-+".indexOf(c) != -1) {
10              temCaractereEspecial = true;
11          }
12      }
13  }
14  return temLetraMaiuscula && temLetraMinuscula && temDigito
    && temCaractereEspecial;
```

Utilizando a técnica de particionamento por equivalência, para este cenário, têm-se as classes de equivalência apresentadas a seguir, sendo em negrito o tipo de entrada, em seguida a quantidade de caracteres para uma entrada válida e a quantidade para uma entrada inválida, ilustrados na Tabela 1.

Tabela 1 – Tabela de Validação de Senhas

Entradas	**Classes Válidas**	**Classes Inválidas**
#Quantidade de caracteres	≥ 8	< 8
#Letras maiúsculas	≥ 1	< 1
#Letras minúsculas	≥ 1	< 1
#Caracteres especiais	≥ 1	< 1
#Dígitos	≥ 1	< 1

Fonte: Autoria Própria (2025)

Essa abordagem é adequada para agrupar as combinações de entradas e representá-las em classes, o que pode reduzir a necessidade de mais testes, uma vez que atestem a sua cobertura. Além disso, é possível combiná-la com a técnica de Análise de Valor Limite para testar as entradas próximas aos limites, garantindo uma maior precisão e reforçando a importância da integração dos diferentes tipos de técnicas.

Nesse sentido, utilizar esta técnica é indispensável para obter uma maior cobertura

dos requisitos determinados pelo sistema. Com isso, cada partição representa um conjunto de entradas possíveis, que, como consequência, facilita a detecção de falhas específicas de verificação, que podem passar despercebidas durante o desenvolvimento.

5.1.2 Verificação de Palavras Palíndromas

No sistema desenvolvido para verificar palavras palíndromas, foi aplicada a técnica de Particionamento por Equivalência, que identificou problemas no tratamento de frases e na distinção de letras maiúsculas e minúsculas. Dessa forma, pode-se definir uma exploração mais aprofundada em relação às entradas e à interação entre as palavras, ampliando a cobertura dos testes.

Código-fonte 2 – Método para Verificação de Palíndromo

```
for (int i = 0; i < n / 2; i++) {
    if (word.charAt(i) != word.charAt(n - i - 1)) {
        return false;
    }
}
return true;
```

Para a verificação e análise prática, a Figura 2 ilustra dois casos de teste para validar o funcionamento correto do laço de repetição que os verifica. Durante a análise deste exemplo, foram colocadas em teste as palavras palíndromas (que podem ser lidas da esquerda para a direita e não há alteração em seu sentido). Utilizando a técnica de Particionamento por Equivalência, pode-se dividir as entradas segundo o apresentado no Quadro 4.

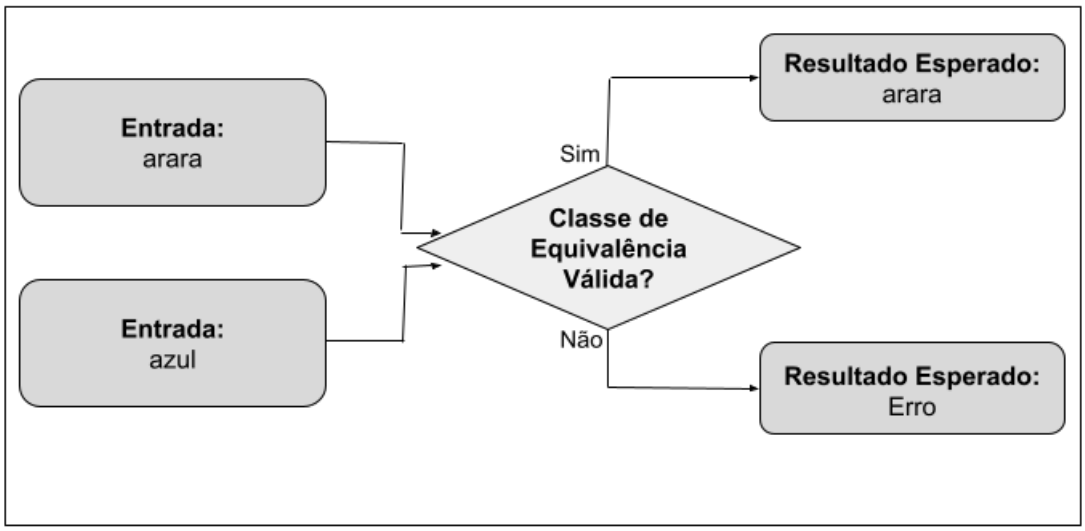
Quadro 4 – Classes de Equivalência para Validar Palíndromos

Classes Válidas	**Classes Inválidas**
Palavras que são Palíndromos	Palavras que não são Palíndromos
Frases com espaço	Entrada vazia
Frases com letras maiúsculas e minúsculas	Caracteres especiais

Fonte: Autoria Própria (2025)

Dessa forma, a utilização do Particionamento por Equivalência permite que as classes sejam analisadas individualmente, o que contribui para aumentar a cobertura das diferentes

Figura 2 – Validação de Palavras Palíndromas



Fonte: Autoria Própria (2025)

entradas, trazendo melhoras para a robustez do sistema. Com isso, a categorização desses dados podem evidenciar erros que poderiam passar despercebidos, trazendo atenção para o espaçamento, acentuação e sensibilidade a letras maiúsculas e minúsculas, e a partir disso garantir que a verificação funcione corretamente.

5.2 Teste Funcional Sistemático

Esta seção, está direcionada a aplicação da técnica de teste funcional sistemático, utilizando as técnicas de forma combinada, para potencializar o efeito dos testes.

5.2.1 Validação de E-mail

No caso de uma validação de e-mails, existe uma maior combinação de entradas válidas e inválidas possíveis. Por exemplo, têm-se os diferentes formatos de domínio, subdomínios e a extensão do *Top Level Domain* (TLD) (Marianne, 2024).

Os domínios no e-mail são os identificadores registrados após o símbolo @, já o subdomínio é a extensão do domínio, por fim, a extensão do (TLD) é a extensão posterior ao ponto. Assim, realizando uma integração das técnicas mediante o Teste Funcional Sistemático, e aplicando o *Error Guessing* (Delamaro; Jino; Maldonado, 2013), que utiliza a expertise do testador para definir os casos de teste e caminhos a serem seguidos. Dessa forma, é possível explorar os cenários já pré-definidos como casos de teste, além da capacidade de detectar inconsistências mais ocultas, que podem ser observados no Código-fonte 3.

Código-fonte 3 – Validação de Email

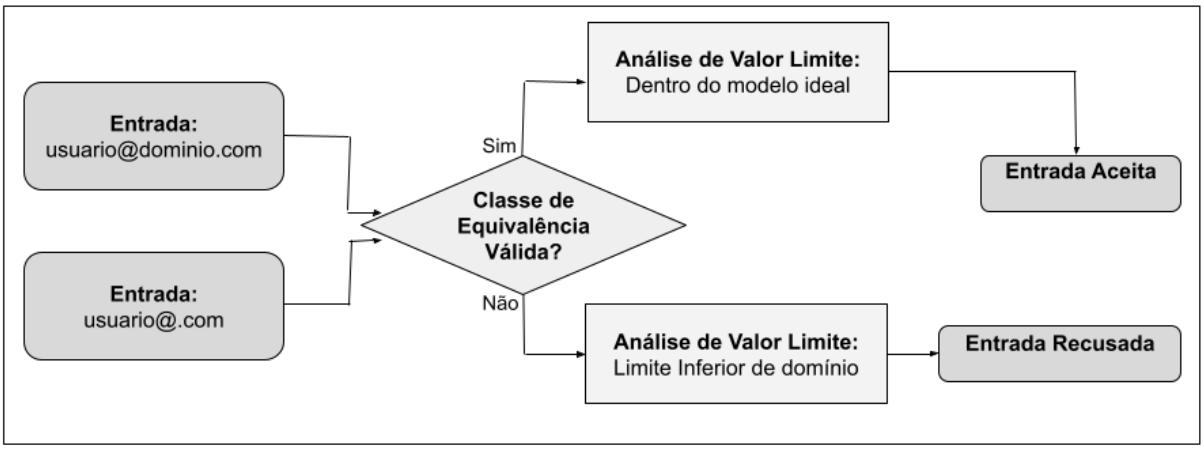
```
public class ValidacaoEmail {

    private static final Pattern EMAIL_PATTERN = Pattern.
       compile(
         "^[A-Za-z0-9+_.-]+@[A-Za-z0-9.-]+$"
    );

    public static boolean EmailEhValido(String email) {
        return EMAIL_PATTERN.matcher(email).matches();
    }
}
```

A aplicação combinada das técnicas de Particionamento Por Equivalência e Análise de Valor Limite revelam falhas no tratamento de e-mails que passariam despercebidas durante a fase de verificação. Estas falhas têm relação, por exemplo, com o uso de caracteres especiais inadequados. Isto pode ser ilustrado a partir da análise da Figura 3.

Figura 3 – Fluxograma para Análise de Casos de Teste



Fonte: Autoria Própria (2025)

Assim, fica evidente que para a cobertura mais completa em um sistema com múltiplas entradas, a combinação de abordagens torna-se um elemento importante para cobrir a sua alta variação, conseguindo testar todos os casos e garantir seu bom funcionamento. Dessa forma, a combinação de diferentes tipos de teste caixa preta é eficaz na identificação de cenários que

podem ser críticos que surgem durante o processo de validação, apontando limites e partes defeituosas. Para meio de exemplificação, algumas das funções aplicadas às entradas durante o cadastro ou log in de e-mail estão representadas no Quadro 5.

Quadro 5 – Teste Funcional Sistemático

Casos de teste	**Entradas**	**Classes de equivalência**	**Análise de valor limite**
CT01	usuario@dominio.com	Válida	Limite em modelo ideal
CT02	u@d.c	Inválida	Limite de comprimento inferior
CT03	@dominio.com	Inválida	Limite de comprimento inferior
CT04	usuario@.com	Inválida	Limite Inferior de domínio
CT05	usuario@dominio.	Inválida	Limite inferior de TLD
CT06	usuario@dominio.commm	Inválida	Limite superior de TLD
CT07	usuariooo+@dominio.com	Válida	Limite superior de nome

Fonte: Autoria própria (2025)

5.2.2 Cálculo de Matriz 3x3

No sistema desenvolvido para realizar cálculos de determinantes de uma matriz 3x3, foi empregada a técnica de Teste Funcional Sistemático. Mediante a aplicação da Análise de Valor Limite e do Particionamento por Equivalência foram estabelecidas classes considerando os diferentes tipos de entradas, sendo eles, números inteiros positivos ou negativos, decimais e zeros. Como mostra o Código-fonte 4, que exibe o cálculo das posições dos vetores, que serão definidos pelos usuários, para realizar calcular o valor do seu determinante.

Código-fonte 4 – Cálculo do Determinante de uma Matriz 3x3

```
1  public float calcularDeterminante() {
2      float a = matriz[0][0];
3      float b = matriz[0][1];
4      float c = matriz[0][2];
5      float d = matriz[1][0];
6      float e = matriz[1][1];
7      float f = matriz[1][2];
8      float g = matriz[2][0];
9      float h = matriz[2][1];
10     float i = matriz[2][2];
```

```
11
12      return a * (e * i - f * h)
13             - b * (d * i - f * g)
14             - c * (d * h - e * g);
15  }
```

Avaliando este cenário, para identificar possíveis falhas, utilizou-se o Teste Funcional Sistemático, que é a combinação das técnicas de Análise de Valor Limite com o Particionamento de Equivalência, dividindo um conjunto de possíveis entradas válidas e inválidas, conforme disponibilizado no Quadro 6.

Quadro 6 – Teste Funcional Sistemático Para Cálculos de Matrizes

Casos de teste	**Entradas**	**Classe de equivalência**	**Análise de valor limite**
CT01	{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}	Válida	Limite em modelo ideal
CT02	{{1, 2, 3}, {4, 5}}	Inválida	Valor da matriz incompleto
CT03	{{1, 2, "a"}, {4, 5, 6}, {7, 8, 9}}	Inválida	Entradas não numéricas
CT04	{{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}	Válida	Limite com determinante igual a zero
CT05	{{1.5, 2, 3}, {4, 5.5, 5}, {7, 8, 5.5}}	Válida	Valores decimais positivos
CT06	{{-1, -2.5, -3}, {-4.5, -5, -6}, {-7, -8, -9.5}}	Válida	Valores decimais negativos
CT07	{{1, 0, -1}, {0, 1, -1}, {-1, 0, 1}}	Válida	Valores positivos, negativos e zero

Fonte: Autoria própria (2025)

Para isso, os elementos determinados para a criação dos casos de teste são: os valores válidos (positivos, negativos, decimais e inteiros) e inválidos (matrizes de outros tamanhos e entradas não numéricas); a aplicação dos casos de teste conforme as suposições que cada classe deva possuir o mesmo comportamento, combinado à Análise de Valor Limite, que determina e testa os limites das entradas possíveis, como, por exemplo, valores muito grandes ou pequenos, e testes em matrizes que possuem zeros em posições que podem afetar o cálculo do determinante.

Dessa forma, os casos de teste são compostos por diferentes conjuntos numéricos que verificam possíveis alterações e falhas na precisão de seus cálculos. Durante a aplicação, observa-se que a divisão das classes de equivalência facilitam a identificação das falhas que poderiam surgir durante o processamento de matrizes com zeros em posições específicas, além dos testes

conforme o limite de suas extremidades, para garantir a precisão dos cálculos.

5.3 Grafo de Causa e Efeito e Tabela de Decisão

Nesta seção, foram aplicadas as técnicas de grafo de causa e efeito e de tabela de decisão, analisadas a partir das possíveis entradas e saídas que poderão ser executadas pelos usuários dos sistemas.

5.3.1 Reserva de Voos

Para analisar o sistema de reserva de voos, primeiramente, foi utilizada a técnica de Grafo de Causa e Efeito. Com esta técnica foram definidos os casos e os caminhos a serem seguidos, bem como a implicação dos passos seguintes após cada decisão. Dessa forma, foram mapeadas as condições e ações (Código-fonte 5), sobre o que o sistema deveria realizar com base nas entradas fornecidas pelos usuários. Para complementar, foi utilizada a técnica de Tabela de Decisão, para facilitar a organização das possíveis combinações de entrada e saída.

Código-fonte 5 – Métodos para Confirmação de Reserva

```
1  String confirmacao;
2  do {
3   System.out.print("\nDeseja confirmar a reserva? (S/N): ");
4   confirmacao = sc.nextLine().trim().toUpperCase();
5  } while (!confirmacao.equals("S") && !confirmacao.equals("N
       "));
6
7  if (confirmacao.equals("S")) {
8      System.out.println("Reserva confirmada.");
9  } else {
10     System.out.println("Reserva cancelada.");
11 }
12
13 String novaReserva;
14 do {
15  System.out.print("\nDeseja realizar outra reserva? (S/N):
```

```
    ");
16  novaReserva = sc.nextLine().trim().toUpperCase();
17 } while (!novaReserva.equals("S") && !novaReserva.equals("N
    "));
18
19 if (!novaReserva.equals("S")) {
20     System.out.println("Encerrando o sistema.");
21     sc.close();
22     return;
23 }
```

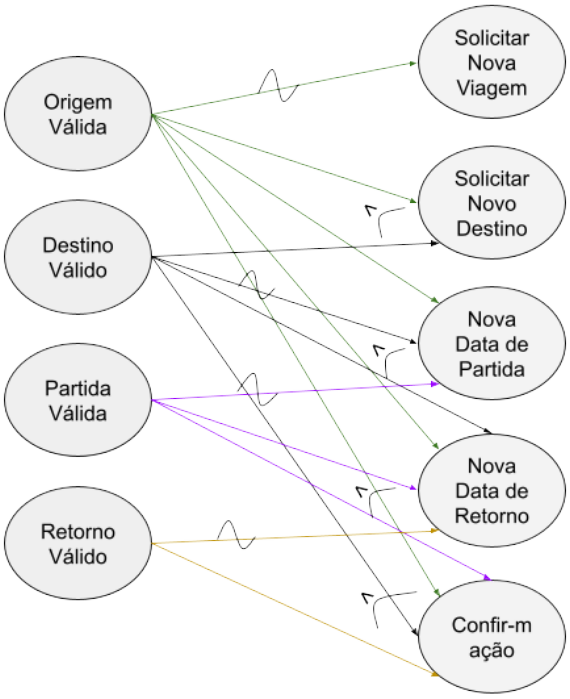
O uso do Grafo de Causa e Efeito auxilia a identificar possíveis falhas em interações mais complexas, como o tratamento das datas de origem e retorno (Figura 4). Com o auxílio da Tabela 2, as entradas foram testadas de forma sistemática, garantindo maior cobertura dos cenários possíveis para a elaboração dos casos de teste. Ao utilizar esta técnica, são definidos um conjunto de casos de teste baseados na divisão das especificações, identificando as entradas e saídas possíveis. Além disso, a utilização da técnica de Tabela de Decisão permite a visualização de como o sistema irá se comportar com base em diferentes combinações de entradas e condições.

Assim, é possível estabelecer rotas de teste a serem seguidas, determinando condições e ações que ilustram o fluxo seguido pelo sistema com base nas escolhas de combinações de entradas. Neste contexto, possibilita-se identificar possíveis falhas ou inconsistências no sistema devido ao mapeamento das interações e condições de entrada e da representação das diferentes combinações de cenários que podem ser seguidos.

5.4 Testes Combinatoriais e *Error Guessing*

A aplicação das técnicas de teste combinatorial e *error guessing* abordadas nesta seção, combina a análise baseada na expertise dos testadores, bem como a combinação de diferentes ações e parâmetros.

Figura 4 – Grafo de Causa e Efeito para o Sistema de reserva de Voos



Fonte: Autoria Própria (2025)

Tabela 2 – Tabela de Decisão para o Sistema de Reservas de Voos

Condições	**C1**	**C2**	**C3**	**C4**
Origem válida	F	V	V	V
Destino válido	F	F	V	V
Data de partida válida	-	-	F	V
Data de retorno válida	-	-	-	V
Ações				
Solicitar nova origem	X			
Solicitar novo destino		X		
Solicitar nova data de partida			X	
Solicitar nova data de retorno				X
Confirmar reserva				X

Fonte: Autoria Própria (2025)

5.4.1 Agenda de Contatos

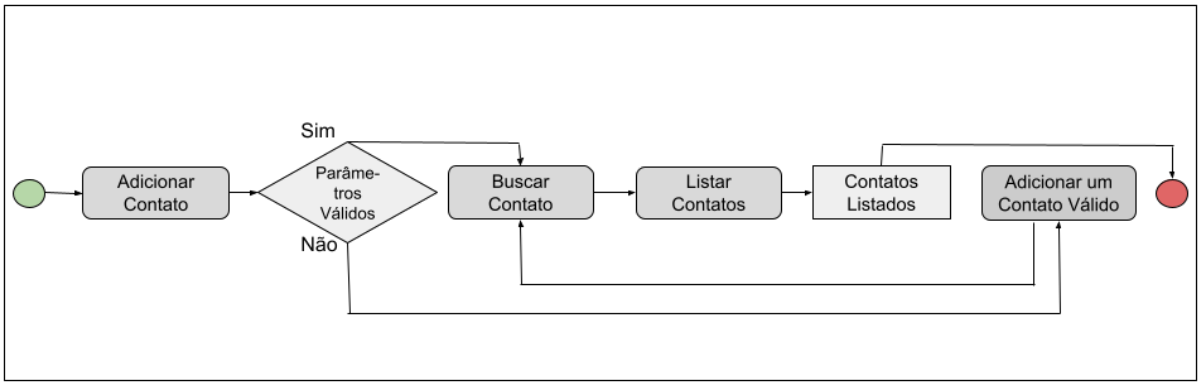
Colocando em análise o sistema para uma agenda de contatos, foi utilizada a técnica de Testes Combinacionais, que explora as interações e caminhos entre as funcionalidades de adicionar, remover, buscar, listar e exportar contatos. Os requisitos para o sistema funcionar em completude ao gerenciar uma agenda de contatos são as funções de adicionar, buscar, remover, listar e exportar contatos. Utilizando esta técnica, pode-se obter cenários com diversas entradas e ações (Quadro7). A aplicação dos casos de teste para que a adição dos contatos possam ser bem sucedidas, e consequentemente sejam buscados e listados, está ilustrado na Figura 5.

Quadro 7 – Casos de Teste Combinatoriais para o Sistema de Agenda de Contatos

Caso de Teste	**Ação 1**	**Ação 2**	**Parâmetros**
CT01	Adicionar contato	Buscar contato	Nome válido, telefone e e-mail válidos
CT02	Adicionar contato	Remover contato	Nome e telefone válidos
CT03	Adicionar contato	Listar contatos	Nome e telefone válidos
CT04	Buscar contato	Listar contatos	Nome existente
CT05	Exportar contatos	Listar contatos	Nome existente
CT06	Remover contato	Buscar contato	Nome existente

Fonte: Autoria Própria (2025)

Figura 5 – Fluxograma para a Inserção e Listagem de um Contato Válido



Fonte: Autoria Própria (2025)

O uso dessa técnica auxiliou a identificar as falhas entre as funcionalidades e atestar a completude dos requisitos do sistema. Assim, revelam-se diferentes cenários e como os mesmos estão sendo tratados. Além disso, a utilização da técnica *Error Guessing* também seria necessária, uma vez que permitiria a antecipação de possíveis falhas baseadas na experiência do testador.

5.5 Empréstimo de Livros

O sistema para a realização de empréstimo de livros teve a aplicação combinada das técnicas de *Error Guessing* e Teste Combinacional. Estas técnicas foram eficazes para garantir que as interações entre as diferentes funcionalidades fossem testadas, além de identificar precisamente falhas que poderiam passar despercebidas, mas que com a análise prática baseada na expertise têm-se coberturas diferentes.

Ao utilizar a técnica de teste combinacional, obtém-se a cobertura das combinações possíveis no sistema, representando as múltiplas ações mediante tabelas, que dependem de decisões para seguir diferentes percursos. A utilização da técnica de *Error Guessing* também torna-se útil para considerar os caminhos e ações para a construção da tabela, além de conseguir identificar erros antes de seguir o fluxo das combinações, declaradas no Quadro 8.

Dessa forma, pode-se observar uma adequação diferente para cada técnica de teste caixa preta aos sistemas analisados, permitindo uma análise sobre o desempenho de cada metodologia em cenários específicos. Essa abordagem gera uma ampliação da perspectiva sobre a eficácia das técnicas, evidenciando em quais contextos elas podem ser mais eficientes para detectar os erros, seja de forma individual ou combinadas, para que assim garantam a completude dos requisitos funcionais e os sistemas funcionem conforme as suas especificações.

Quadro 8 – Casos de Teste Combinatoriais para o Sistema de Biblioteca

Caso de Teste	**Ação 1**	**Ação 2**	**Parâmetros**
CT01	Registrar livro	Listar livros registrados	Título, autor, número de páginas, ano e localização válidos
CT02	Registrar livro	Emprestar livro	Título vazio, autor e ano válidos
CT03	Emprestar livro	Conferir livros alugados	Título existente e disponível
CT04	Emprestar livro	Emprestar livro	Título existente já emprestado
CT05	Devolver livro	Emprestar livro	Título existente, livro já emprestado
CT06	Devolver livro	Conferir livros alugados	Título não emprestado
CT07	Conferir livros alugados	Listar livros registrados	Nenhum parâmetro
CT08	Listar livros registrados	Sair do sistema	Nenhum parâmetro

Fonte: Autoria Própria (2025)

5.6 Teste Fuzzing

A aplicação da técnica de teste *fuzzing* nesta seção, ilustra a injeção de dados para detectar falhas de sistemas, objetificados através de uma exploração das funcionalidades mediante ao seu desempenho.

5.6.1 Utilitário de Calculadora

No utilitário de linha de comando que realiza operações aritméticas, foram realizados testes exploratórios procurando investigar a aleatoriedade das variações dos casos gerados a partir de uma entrada inicial válida, semelhante aos testes de *fuzzing* (Chen *et al.*, 2018). Com isso, tomando como base a natureza do programa, as variações de entradas criadas a partir da entrada inicial evidenciaram mutabilidade em relação às saídas fornecidas pelo programa (Quadro 9).

Quadro 9 – Casos de Teste para um Utilitário de Calculadora

Caso de teste	**Entradas**	**Ação**	**Saídas**
CT01	$sqrt(3.14*5)^2$	Operação aritmética válida	15.68
CT02	$(sqrt(3.14*5)^2)/9$	Acréscimo da operação de divisão	Omissão da casa decimal
CT03	$sqrt(-3.14*5)^2$	Raiz quadrada de número negativo	Erro de tempo de execução
CT04	$sqrt(3.14*5)^2$	Exponenciação sem circunflexo	Caractere ilegal
CT05	$(sqrt(3.14*5)^2)/0$	Divisão por zero	Erro de tempo de execução
CT06	$sqrt(3,14*5)^2$	Substituição do ponto por vírgula	Erro de sintaxe
CT07	$sqrt(3.14*5)^{2+}$	Operador sem valor associado	Erro de sintaxe
CT08	$sqrt(3.14*5)^{1234567890}$	Expoente com valor elevado	Cálculo não realizado

Fonte: Autoria Própria (2025)

As entradas potencialmente válidas geradas com a modificação do valor inicial fornecido estiveram envoltas na operação matemática de radiciação. A partir disso, foram selecionados mais sete casos de teste correspondentes a variações da fórmula inicial. Assim, as alterações criadas foram enviesadas a fim de analisar algum comportamento inesperado transmitido pelo programa. Diante disso, a demonstração da técnica de testes exploratórios procurou explorar a dinamicidade em projetar, executar e modificar casos de teste no processo de identificação de defeitos, além de fornecer maior liberdade ao testador (Silva, 2009).

Dessa forma, no caso onde a entrada apresenta um valor elevado, o desempenho do programa é comprometido, visto a quantidade de algarismos especulados para a saída. Além disso, variações de comportamento também foram observadas conforme o tipo do problema. Notou-se, portanto, que erros e omissões quanto à sintaxe provocam comumente saídas distintas. Com isso, reafirma-se a capacidade que os testes exploratórios possuem em encontrar variações de problemas e comportamentos em um tempo de execução relativamente menor, uma vez que a experiência e conhecimento do testador sobre o programa otimizam o tempo em relação ao processo de execução de testes e identificação de defeitos (Castro, 2018).

5.7 Aplicação dos Testes Caixa Preta em Relação à Usabilidade

A realização dos testes de usabilidade surgem como uma técnica de teste caixa preta, de modo que a interação entre os usuários e o computador é analisada conforme as heurísticas de Nielsen (1994), para os softwares atenderem às métricas de qualidade necessárias para serem agradáveis e eficientes. Para atestar a eficácia dos testes de usabilidade relacionados a detecção de falhas em interfaces, foram analisados os sistemas na seção 5.7.

A implementação dos Testes de Usabilidade como uma técnica de teste caixa preta, devido à falta de acesso ao código-fonte, é utilizada para analisar diferentes métricas, que envolvem tipos de tarefas, medidas de desempenho e disposição de escalas. Baseia-se na interação entre os seres humanos e os computadores, utilizando de entrevistas e inspeções aplicadas ao processo, no intuito de detectar problemas quanto à usabilidade dos sistemas (Ferreira, 2002). Para que os sistemas estejam funcionando em conformidade ao esperado, de forma eficiente e intuitiva.

Para ilustrar a importância da aplicação desta abordagem, foram utilizados os sistemas aplicados durante a realização do curso “Introdução ao Teste de Software”, então colocados sob análise, baseados em heurísticas que assegurem a qualidade do seu desenvolvimento; sendo este um aspecto importante ao longo de todo o ciclo de vida dos sistemas (Neto, 2022).

5.7.1 Sistema de Formulário

Para a primeira análise em relação à usabilidade, está um sistema para a realização de envio de formulários, ilustrado na Figura 6.

Figura 6 – Análise de Usabilidade de Sistema de Formulários

Fonte: Autoria Própria (2025)

Analisando a completude das funcionalidades relacionadas com as heurísticas de Nielsen (1994), é possível identificar mediante a aplicação dos testes de usabilidade na interface, como estão dispostas as funcionalidades do sistema, e se estão conforme o que é determinado para o sistema atender aos parâmetros de qualidade, comparados no Quadro 10.

Dessa forma, a partir da análise deste sistema, é notável que ele segue padrões visuais confortáveis, utilizando linguagens e símbolos simples e intuitivos. Porém, algumas funcionalidades não estão conforme os princípios que garantiriam a sua eficiência, como a falta de retorno dos botões, e a falta de mensagens de exibição de erros e instruções objetivas. Violando principalmente o atributo de qualidade de proteção a erros do usuário, devido à introdução de defeitos, que dificultam a sua utilização, exibindo maior complexidade acerca da sua utilização, tornando-o pouco flexível e apresentando dificuldades em corresponder às entradas esperadas atribuídas às suas funcionalidades; gerando desconforto funcional devido à má adequação funcional, que também é uma métrica importante para a qualidade de software.

Quadro 10 – Avaliação das Heurísticas de Nielsen no sistema de Formulário

Heurística de Nielsen	**Princípio Contemplado**	**Justificativa**
Visibilidade do status do sistema	Parcialmente	O botão “Enviar” não dá retorno imediato.
Correspondência entre o sistema e o mundo real	Sim	Usa linguagem natural e amigável, com campos compreensíveis.
Controle e liberdade do usuário	Sim	Há como para apagar dados inseridos ou retroceder caso algo seja preenchido incorretamente.
Consistência e padronização	Sim	Os campos seguem padrões comuns de formulários.
Prevenção de erros	Não	Não há validação visível de campos, nem mensagens que alertem sobre preenchimento incorreto.
Reconhecimento em vez de memorização	Sim	Os campos têm lembretes, que ajudam o usuário a preencher sem memorizar.
Flexibilidade e eficiência de uso	Não	Não há atalhos, preenchimento automático ou outro recurso que beneficie usuários avançados.
Estética e design minimalista	Sim	Design organizado, uso adequado de cores e espaçamento.
Ajudar a reconhecer, diagnosticar e corrigir erros	Não	Não aparecem mensagens de erro claras nem instruções para correção na falha no envio.
Ajuda e documentação	Não	Não há nenhum link ou informação de ajuda disponível na interface.

Fonte: Autoria Própria (2025)

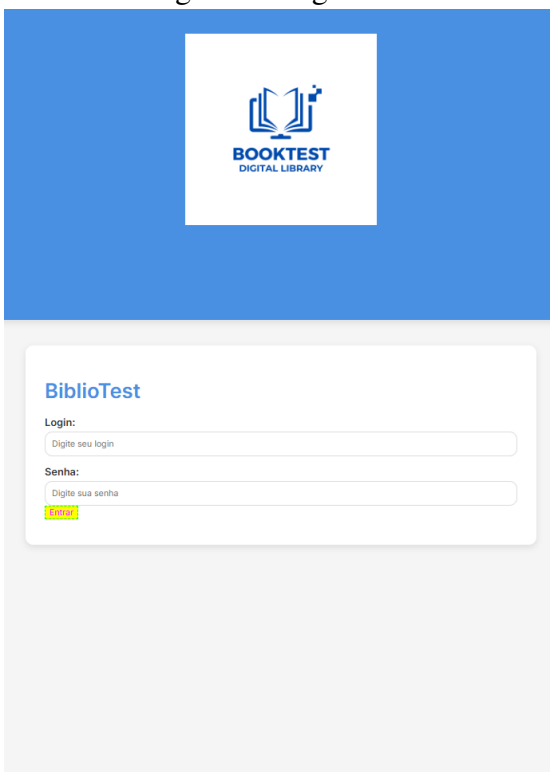
5.7.2 Sistema de Biblioteca Digital

Durante a realização da segunda análise relacionada à completude das heurísticas como uma garantia da completude das métricas de qualidade, foi utilizado um sistema de Biblioteca Digital (Figura 7). Este sistema é capaz de realizar reservas, armazenar o histórico de leituras e marcação de livros favoritos, bem como a interação entre leitores, baseados na avaliação das obras literárias (Figura 8). Analisando o funcionamento deste sistema, possibilita a criação do Quadro 11, para ilustrar comparativamente o nível de obtenção das métricas de qualidade, segundo as heurísticas de Nielsen (1994).

Após a criação do Quadro 11, e analisando o cumprimento das heurísticas, pode-se perceber que a interface apresenta problemas quanto à confortabilidade visual, fora do padrão das demais cores. Além disso, possui botões e funcionalidades rotuladas, que auxiliam na sua utilização, mediante o emprego de termos naturais e simples, exibindo mensagens de confirmação para as funcionalidades bem sucedidas, mas contrárias para a proteção aos erros do usuário.

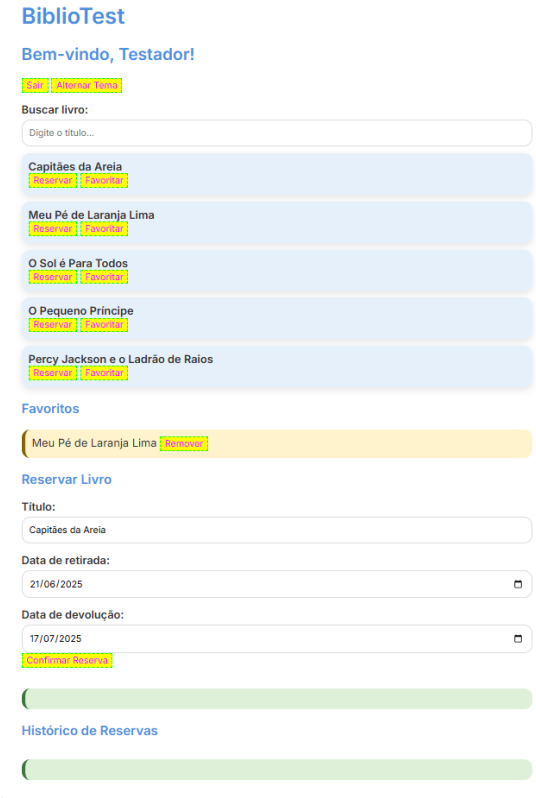
Porém, este sistema viola alguns dos princípios que o fariam estar conforme as principais métricas de garantia de qualidade, algumas cores causam desconforto e poluição visual, comprometendo o padrão da interface. Além de não estar bem preparado para lidar com os possíveis

Figura 7 – Sistema de Biblioteca Digital: Página de Login



Fonte: Autoria Própria (2025)

Figura 8 – Sistema de Biblioteca Digital: Página Principal



Fonte: Autoria Própria (2025)

erros de utilização cometidos pelos usuários, em que a falta de mensagens de erros e carência de manuais ou documentação comprometem a eficiência da sua utilização. Além da falta de conformidade no atributo de qualidade de adequação funcional, em que as funcionalidades implementadas possuem defeitos quanto à sua execução e resultados.

5.7.3 Plataforma de Cursos Digital

Esta análise diz respeito à avaliação da usabilidade de um sistema para a realização de cursos on-line. Foram utilizadas as heurísticas de Nielsen (1994) para atestar a conformidade dos seus princípios com a interface em questão, para estar conforme os padrões visuais e funcionais, exibindo sua eficiência e a relatividade do seu entendimento. Apresentando a sua página de log in (Figura 9) e a página principal (Figura 10). Para consolidar a conformidade do sistema com os princípios de Nielsen (1994), é possível analisá-las a partir da construção do Quadro 12.

É possível perceber que o sistema atua em conformidade aos padrões de qualidade quanto à disposição visual das cores e elementos, sendo confortáveis aos olhos, além de possuir

Quadro 11 – Avaliação das Heurísticas de Nielsen no Sistema BiblioTest

Heurística de Nielsen	**Princípio Contemplado**	**Justificativa**
Visibilidade do status do sistema	Sim	A interface mostra mensagens de confirmação após a realização de ações.
Correspondência entre o sistema e o mundo real	Sim	Os termos utilizados são naturais ao usuário, sendo a estrutura da tela simples e intuitiva.
Controle e liberdade do usuário	Parcialmente	O usuário pode sair do sistema e alternar o tema, mas não deixa claro se há opção para desfazer reservas ou avaliações.
Consistência e padronização	Parcialmente	Os botões de ação não seguem padrão visual e textual, comprometendo a identidade visual.
Prevenção de erros	Parcialmente	O sistema evita ações incomuns, mas não exibe mensagens de confirmação antes de realizar as ações.
Reconhecimento em vez de memorização	Parcialmente	Os botões são rotulados, com fluxos intuitivos, mas fora dos padrões de cores e formato confortável à visão.
Flexibilidade e eficiência de uso	Sim	Há atalhos visuais simples para funções repetitivas, promovendo agilidade no uso.
Estética e design minimalista	Parcialmente	Interface simples, mas contendo cores fora do padrão, e disposição acessível das informações.
Ajudar a reconhecer, diagnosticar e corrigir erros	Parcialmente	A interface mostra mensagens de sucesso, mas as mensagens de erro não são claras.
Ajuda e documentação	Não	Não há elementos de ajuda e suporte visíveis.

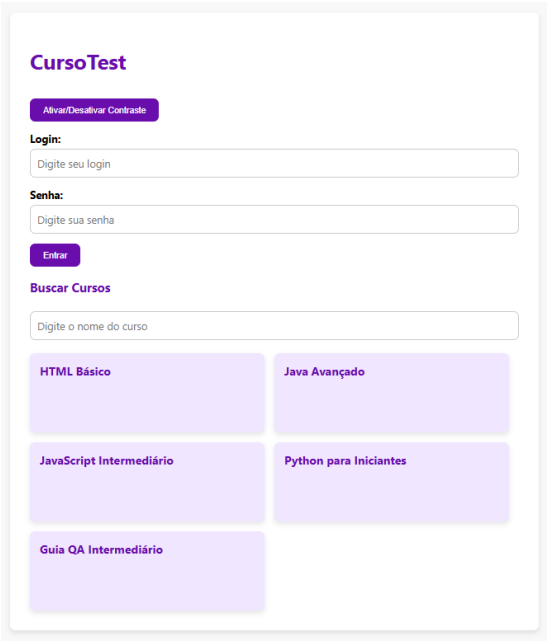
Fonte: Autoria própria (2025)

mensagens acerca da execução das ações, descritas de forma simples e claras, não exigindo memorização e, consequentemente, um rápido acesso. Porém, não existem instruções de uso ou mensagens de confirmação após a execução das funcionalidades, as quais apresentam problemas quanto ao seu funcionamento, objetificando a incompletude de alguns dos princípios, comprometendo a sua eficiência.

A análise das técnicas diante dos cenários em que foram empregadas oferece suporte às aplicações reais, promovendo o desenvolvimento de novas metodologias de teste que possuem impacto direto e significativo na qualidade e confiabilidade dos sistemas em que forem colocadas em prática. Essa exploração permite que as abordagens sejam ajustadas conforme as necessidades de cada sistema especificamente, resultando em soluções mais eficazes para revelar erros.

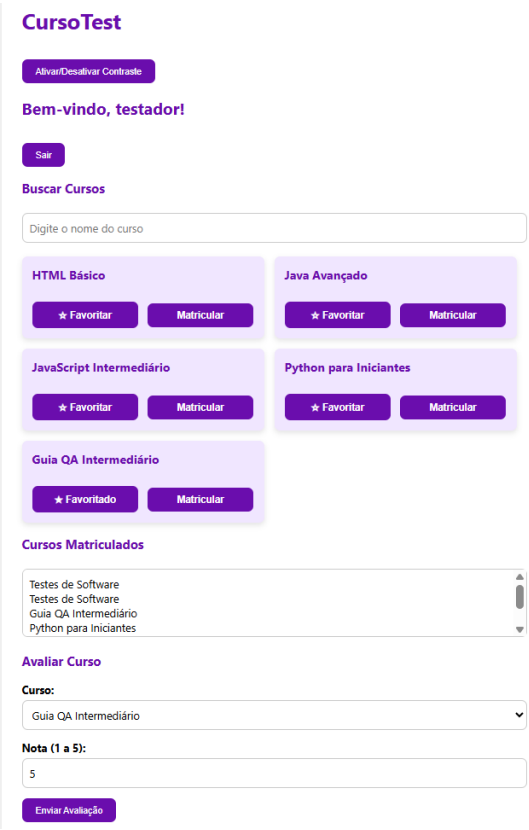
Atrelado a isso, promovendo a aplicação das metodologias utilizadas, fundamentadas mediante técnicas de teste caixa preta, obtêm-se uma abordagem orientada aos requisitos e especificações, adotando uma avaliação mais crítica, garantindo que os produtos finais atendam às expectativas dos usuários.

Figura 9 – Plataforma de Cursos Digital: Página de Login



Fonte: Autoria própria (2025)

Figura 10 – Plataforma de Cursos Digital: Página Principal



Fonte: Autoria própria (2025)

5.8 Discussões

Diante destas aplicações, é possível objetificar com mais profundidade a adequação de cada uma das técnicas caixa preta a cada tipo de sistema. Nos sistemas de Validação de Senha e Validação de E-mail, foram aplicadas as técnicas de Particionamento Por Equivalência e Análise de Valor Limite. Elas permitiram a divisão dos códigos em partes que ainda mantivessem a cobertura dos requisitos, nas quais as classes e entradas foram definidas conforme os seus limites, agrupando caminhos e combinações distintos, que facilitaram principalmente a detecção de falhas coincidentes.

No cenário para o Cálculo de Matriz, foram empregadas as técnicas de Teste Funcional Sistemático, Particionamento por Equivalência e Análise de Valor Limite, surgindo como solução para possíveis problemas com o posicionamento de números negativos, decimais, ou até entradas não numéricas, o que contribuiria para um resultado incorreto. Com essa divisão de cenários, durante o processamento do cálculo, é possível analisar as extremidades do sistema e suas diferentes entradas.

Quadro 12 – Avaliação das Heurísticas de Nielsen da Plataforma de Cursos Digital

Heurística de Nielsen	**Princípio Contemplado**	**Justificativa**
Visibilidade do status do sistema	Sim	O sistema mostra cursos favoritos, matriculados e avaliações de forma clara. Havendo retorno visual imediato, após as ações.
Correspondência entre o sistema e o mundo real	Sim	Usa termos comuns e compreensíveis e interface familiar.
Controle e liberdade do usuário	Parcialmente	Há botão de sair e alternar contraste, mas não é possível desmatricular ou alterar uma avaliação.
Consistência e padronização	Sim	A interface mantém padronização de cores, botões e estruturação das telas.
Prevenção de erros	Parcialmente	Não há confirmação antes da execução das ações.
Reconhecimento em vez de memorização	Sim	O sistema usa rótulos e padronização, não exigindo memorização.
Flexibilidade e eficiência de uso	Sim	O sistema facilita acesso rápido às ações principais.
Estética e design minimalista	Sim	Interface é simples, e possui elementos visuais confortáveis.
Ajudar a reconhecer, diagnosticar e corrigir erros	Parcialmente	Não há mensagens de erro visíveis nem confirmações para ações.
Ajuda e documentação	Não	Não há ícones ou seções de ajuda para os usuários.

Fonte: Autoria própria (2025)

No sistema para Reserva de Voos, ao utilizar a técnica de Grafo de Causa e Efeito, foi possível observar se existem desvios nas ações dos caminhos previamente descritos nos casos de teste, com isso, fazendo um mapeamento das condições para as ações serem executadas corretamente. Para aumentar o desempenho do teste, a utilização de uma tabela de decisão estabelece rotas que ilustram o fluxo de ações a ser seguido, possibilitando revelar inconsistências com base nas diferentes combinações de entradas.

Nos testes do sistema de Verificação de Palavras Palíndromas, o uso da técnica de Particionamento por Equivalência expõe a interação entre as palavras divididas em classes, válidas e inválidas, colaborando para a detecção de problemas em relação à distinção das letras, bem como o mau uso do espaçamento e até caracteres especiais. Através dessa divisão, o teste torna-se mais objetivo e eficaz, uma vez que determinadas as entradas inválidas, essa técnica busca erradicar esse problema preexistente de sistemas que lidam com a verificação de caracteres.

Durante a análise do sistema de Agenda de Contatos, foi utilizada a técnica de Teste Combinacional. Esta técnica revelou problemas em relação às funcionalidades do sistema, usando cenários com diferentes ações e parâmetros, facilitando o encontro de inconsistências no seu funcionamento. Além disso, a técnica de *Error Guessing* permitiu prever possíveis falhas no tratamento das ações. Portanto, a soma dessas metodologias permite a verificação e validação

dos requisitos funcionais do sistema.

Os testes realizados com o Utilitário de Calculadora reafirmam a importância dos testes exploratórios, utilizados neste trabalho como uma forma dinâmica de projetar e executar os casos de teste. A Técnica *Fuzzing*, permite a revelação de problemas relacionados à variação dos comportamentos durante o tempo de execução, como erros de sintaxe e valores elevados.

O sistema de Empréstimo de Livros foi verificado mediante a combinação do *Error Guessing* com o Teste Combinacional. Estas técnicas foram empregadas durante o processo de teste das diferentes interações das funcionalidades, permitindo maior cobertura dos requisitos e identificando problemas na implementação das funcionalidades. O uso dessas metodologias em cenários específicos permite prever quais partes do sistema estão mais propensos a falhar, além de avaliar de forma mais crítica as ações e os parâmetros para cada um dos seus caminhos.

Analisando a completude das métricas de qualidade baseados nas heurísticas e princípios de Nielsen (1994), é possível objetificar pontos nos quais os sistemas falham quanto à garantia dos aspectos que tornariam o sistema fácil, intuitivo e eficaz. Analisando os sistemas de Formulário, Biblioteca Digital e a Plataforma de Cursos Digital, são exibidas características dispostas de formas inconsistentes, que pode levar ao mau entendimento e incidência de erros dos usuários.

Tais problemas poderiam ser solucionados por meio da construção de uma base sólida dos princípios relacionados à usabilidade, quando relacionados às heurísticas apresentadas, para assegurar o sistema qualitativamente. Analisando quantitativamente as heurísticas violadas nos sistemas em discussão, observa-se uma ênfase maior, nos princípios de “Controle e Liberdade do Usuário”, “Prevenção de erros”, “Ajuda a Reconhecer, Diagnosticar e Prevenir Erros”, e “Ajuda e Documentação”. Em consequência disto, é notória a incompletude da métrica de qualidade de proteção a erros do usuário, implicando no mau desempenho do sistema quando está suscetível a erros quanto ao seu entendimento, prejudicando a sua eficiência, resultando em insatisfações acerca do seu funcionamento.

6 CONCLUSÕES E TRABALHOS FUTUROS

A análise das metodologias e estratégias dos testes de caixa preta em diferentes cenários realizada neste estudo visou ressaltar a importância da criação de um panorama de diferentes abordagens acerca das técnicas. Dessa forma, destacou-se a importância da realização dos testes funcionais para garantir a completude dos requisitos e especificações dos sistemas, impactando diretamente na qualidade e confiabilidade de um software.

Ao longo do desenvolvimento, destaca-se também a utilização combinada das técnicas, tornando o processo mais eficiente, aumentando a cobertura dos casos de teste e detectando erros que poderiam passar despercebidos se testados de maneira isolada. Assim, tem-se aumentada a qualidade dos testes, que se mostram eficazes em contextos específicos e que devem ser selecionados conforme as necessidades e complexidade do sistema em análise. Essa exploração metodológica permite a adequação de cada abordagem conforme o problema que vai resolver, como uma forma de prevenir ou revelar possíveis falhas, avaliando conjuntamente e, consequentemente, mais criticamente as funcionalidades conforme as especificações de cada um dos cenários construídos para fundamentar esta pesquisa.

A exploração detalhada das estratégias dos testes de caixa preta para a criação de um panorama destaca a importância de uma abordagem baseada nos requisitos e especificações, para que os softwares atendam às suas funcionalidades e garantam uma boa experiência e satisfação para o usuário. Atrelado a isso, este trabalho de conclusão de curso traz contribuições como um recurso metodológico para aprendizagem e aplicações reais, garantindo, mediante execuções em sistemas, um melhor entendimento acerca do funcionamento prático dos testes funcionais como uma ferramenta para avaliar a qualidade e confiabilidade de um software.

Além disso, é importante reafirmar que o presente trabalho é relevante pela sua disponibilidade e contribuição para a linha de pesquisa, escrito em português, contribuindo para a literatura nacional. Esse trabalho contribui, então, para suprir essa lacuna, e, também, serve como uma ferramenta para estudo metodológico e aplicações práticas para a implementação das técnicas de teste caixa preta em diferentes tipos de sistemas.

Como perspectiva para trabalhos futuros, destaca-se a necessidade de promover um estudo utilizando a participação de testadores, colocando em prática a sua experiência durante o processo de testabilidade de um sistema. A partir de uma análise quantitativa acerca da utilização das metodologias de teste caixa preta para a detecção dos defeitos baseados na sua utilização cotidiana, além do experimento qualitativo mediante a aplicação das técnicas em

sistemas pré-definidos como amostra, para avaliar o desempenho das técnicas na revelação de falhas, bem como explicitar a sua utilização como boa prática para que os sistemas funcionem em conformidade e consequentemente reduzam custos de correções. Dessa forma, promovendo um panorama sobre a utilização das técnicas caixa preta, bem como a importância da sua aplicabilidade, fundamental para aprimorar a cobertura dos requisitos do sistema.

Analisando estas técnicas sob outras perspectivas, quando são colocadas em prática, promove-se uma investigação quanto à adequação a cada cenário, bem como a motivação das estratégias utilizadas para detectar falhas em sistemas. Como consequência disso, contribui-se para atestar a importância dos testes caixa preta para garantir a completude dos requisitos e especificações, para o sistema funcionar conforme o esperado. Assim, assegurando as métricas de qualidade e confiabilidade, validando a sua adequação funcional. Com isso, o estudo substancia a importância de realizar os testes de caixa preta, e a melhora significativa na descoberta de defeitos quando utilizadas como uma metodologia para assegurar a qualidade e a confiabilidade.

REFERÊNCIAS

ABUSALIM, S. W. G.; IBRAHIM, R.; WAHAB, J. A. Comparative analysis of software testing techniques for mobile applications. **Journal of Physics: Conference Series**, 2020.

BOURQUE, P. *et al.* The guide to the software engineering body of knowledge. **IEEE Software**, IEEE, v. 16, n. 6, p. 35–44, 2002.

CASTRO, A. K. S. d. **Testes Exploratórios: Características, Problemas e Soluções**. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Norte, 2018.

CHEN, C. *et al.* A systematic review of fuzzing techniques. **Computers & Security**, Elsevier, v. 75, p. 118–137, 2018.

CHRISANTYO., L. *et al.* Teste de caixa preta nos resultados do revamp do sistema de informação agrícola dutatani. **Tokuro Matsuo. Anais do 11º Congresso Internacional de Informática Aplicada Avançada**, 2022.

CORREIA, J. M. G.; FERREIRA, J. V. C.; PIRES, D. F. Importância dos testes para a qualidade do software. **Revista Eletrônica de Computação Aplicada**, v. 2, n. 2, 2021.

CRUZ, A. K. B. S. d.; NETO, C. d. S. S. Revisitando as heurísticas de avaliação de nielsen para análise de usabilidade em jogos de tabuleiro não virtuais. **Human Factors in Design**, v. 3, n. 06, 2015.

DELAMARO, M.; JINO, M.; MALDONADO, J. **Introdução ao Teste de Software**. [S.l.]: Elsevier Brasil, 2013.

DELAMARO, M.; JINO, M.; MALDONADO, J. **Introdução ao Teste de Software**. 2. ed. [S.l.]: Elsevier Brasil, 2016.

DWIVEDI, Y. *et al.* Research on information systems failures and successes: Status update and future directions. **Information Systems Frontiers**, v. 17, p. 143–157, 02 2014.

FERREIRA, K. G. **Teste de Usabilidade**. Monografia (Monografia de Final de Curso: Especialização em Informática) — Universidade Federal de Minas Gerais, 2002.

GALDINO, T. S. **A importância do teste no processo de produção do software**. Monografia (Tecnólogo em Processamento de Dados), Americana, São Paulo, 2010.

ISO/IEC, I. . **Software Life Cycle Processes**. [S.l.]: IEC, 2017.

JACOB, P. M.; PRASANNA, M. A comparative analysis on black box testing strategies. In: IEEE. **2016 International Conference on Information Science (ICIS)**. [S.l.], 2016. p. 1–6.

KANER, C.; FALK, J.; NGUYEN, H. **Testing Computer Software**. [S.l.]: Wiley, 2011. ISBN 9781118080689.

KHAN, M. E. Different approaches to black box testing technique to find errors. **International Journal of Software Engineering & Applications (IJSEA)**, 2021.

KHAN, M. E.; KHAN, F. A comparative study of white box, black box and grey box testing techniques. **(IJACSA) International Journal of Advanced Computer Science and Applications**, 2012.

KOSCIANSKI, A.; SOARES, M. dos S. **Qualidade de Software - 2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software**. Novatec, 2007. ISBN 9788575221129. Disponível em: https://books.google.com.br/books?id=O9aWoUq6L88C.

LEE, N. *et al.* Black-box testing of practical movie recommendation systems: A comparative study. **Computer Science and Information Systems**, 2013.

MAHENDRA, G. S.; ASMARAJAYA, I. K. A. Evaluation using black box testing and system usability scale in kidung sekar madya application. **Journal and Research in Computer Engineering**, 2022.

MARIANNE. **Understanding Email Domains, Subdomains and Aliases**. 2024. Disponível em: https://help.cognism.com/hc/en-gb/articles/4404423775634-Understanding-Email-Domains-Subdomains-and-Aliases. Acesso em: 19 dez 2024.

MARINHO, M. L. M. **Avaliação de Desempenho de Processos de Testes de Software**. [S.l.]: https://repositorio.ufpe.br/handle/123456789/2321, 2010.

MARTINS, A. I. *et al.* Avaliação de usabilidade: uma revisão sistemática da literatura. **RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação**, v. 11, n. 1, 2013.

MYERS, G.; SANDLER, C.; BADGETT, T. **The Art of Software Testing**. [S.l.]: Wiley, 2011. (ITPro collection). ISBN 9781118133156.

NETO, A. C. D. **Introdução a Teste de Software**. [S.l.: s.n.], 2022.

NIELSEN, J. **Usability engineering**. [S.l.]: Morgan Kaufmann, 1994.

NIELSEN, J. **Usabilidade na web**. [S.l.]: Elsevier, 2007. ISBN 9788535221909.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software**. 8. ed. [S.l.]: McGraw Hill Brasil, 2016.

PRESSMAN, R.; MAXIM, D. B. R. **Engenharia de software: Uma abordagem profissional**. 9a. ed. [S.l.]: McGraw-Hill Education, 2021. ISBN 978-6558040101.

ROCHA, A. V. **Teste funcional sistemático estendido: uma contribuição na aplicação de critérios de teste caixa-preta**. Dissertação (Mestrado) — Universidade Federal de Goiás, Goiânia, 2011.

SEFFAH, A.; METZKER, E. The obstacles and myths of usability and software engineering. **Communications of the ACM**, Association for Computing Machinery, v. 47, n. 12, 2004. ISSN 1557-7317.

SILVA, T. D. d. **Ferramenta de suporte a uma Metodologia Para Testes Exploraórios**. Monografia (Bacharel em Ciência da Computação) — Universidade Federal de Pernambuco, 2009.

SOMMERVILLE, I. **Software Engineering, 9th Edition**. [S.l.]: Addison-Wesley, 2018.

SPIJKMAN, T. *et al.* Alignment and granularity of requirements and architecture in agile development: A functional perspective. **Information and Software Technology**, v. 133, p. 106535, 2021. ISSN 0950-5849.

SUTIAH., S.; SUPRIYONO, S. Software testing on e-learning madrasahs using blackbox testing. **IOP Conference Series: Materials Science and Engineering**, 2021.