Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciências da Computação

Automatização de Testes de Software com Integração contínua na Indústria: um Mapeamento da Literatura

Marluce Aparecida Vitor

JUIZ DE FORA
SETEMBRO, 2024

Automatização de Testes de Software com Integração contínua na Indústria: um Mapeamento da Literatura

Marluce Aparecida Vitor

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento da Ciência da Computação

Bacharelado em Ciências da Computação

Orientador: André Luiz de Oliveira

JUIZ DE FORA

SETEMBRO, 2024

Automatização de Testes de Software com Integração contínua na Indústria: um Mapeamento da Literatura

Marluce Aparecida Vitor

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO.

Aprovada por:

André Luiz de Oliveira
Doutor em Ciência da Computação e Matemática Computacional

Marco Antônio Pereira Araújo
Doutor em Engenharia de Sistemas e Computação

Pedro Henrique Dias Valle
Doutor em Ciência da Computação e Matemática Computacional

JUIZ DE FORA
23 DE SETEMBRO, 2024

Aos meus amigos e irmãos.

Aos pais, pelo apoio e sustento.

Resumo

Contexto: Nos últimos anos houve uma crescente demanda pelo aumento da qualidade de software, independente de plataforma ou sistema operacional. **Motivação**: O motivo disso é que a garantia de qualidade é essencial para a construção de um software funcional, de fácil utilização e manutenção e seguro ao usuário mesmo em caso de falhas. Um equívoco comum é pensar que o Teste de software consiste apenas em executar o software e verificar se as saídas obtidas correspondem às saídas esperadas, considerando apenas os caminhos de sucesso. **Problema**: É necessário identificar as metodologias, técnicas e ferramentas de teste automatizado, bem como as práticas de integração contínua que estão sendo utilizadas na indústria. **Objetivo**: Neste estudo, foi realizada um mapeamento sistemático sobre o uso de técnicas de teste automatizado na indústria no contexto de Integração Contínua (IC). O objetivo foi identificar as metodologias e os estudos empíricos que investigam a influência dos testes desenvolvidos por equipes de qualidade e sua integração na IC. **Metodologia**: O processo de mapeamento sistemático foi conduzido em conformidade com o método proposto por Kitchenham e o método Goals-Questions-Metrics foi utilizado no planejamento do estudo. **Resultado**: Apesar da crescente adoção de testes automatizados na indústria, existem desafios técnicos, organizacionais, de cobertura de testes e de desenvolvimento em tempo de operação (devOps) significativos relacionados à integração efetiva desses testes nas práticas de IC, que impactam diretamente na qualidade do software produzido. As contribuições deste trabalho residem principalmente na análise de como os testes automatizados são incorporados nas práticas de IC, fornecendo uma visão dos benefícios e desafios enfrentados pela indústria. Os resultados desta pesquisa podem contribuir para a melhoria da compreensão dos processos de teste na indústria e da necessidade de maior integração para elevar a qualidade do software.

Palavras-chave: Teste de software. Teste Automatizado. Integração Contínua. Indústria. *DevOps*.

Abstract

Context: In recent years, there has been a growing demand for increased software quality, regardless of platform or operating system.**Motivation**: The reason for this is that quality assurance is essential for building functional software that is easy to use, maintain, and secure for the user, even in the event of failures. A common misconception is that software testing consists solely of executing the software and verifying whether the obtained outputs match the expected outputs, considering only the successful paths.**Problem**: It is necessary to identify the methodologies, techniques, and automated testing tools, as well as the continuous integration practices being used in the industry.**Objective**: In this study, a systematic mapping was conducted on the use of automated testing techniques in the industry within the context of Continuous Integration (CI). The aim was to identify the methodologies and empirical studies that investigate the influence of tests developed by quality teams and their integration in CI.**Methodology**: The systematic mapping process was conducted in accordance with the method proposed by Kitchenham, and the Goals-Questions-Metrics method was used in the planning of the study.**Result**: Despite the increasing adoption of automated testing in the industry, there are significant technical, organizational, test coverage, and operational development (DevOps) challenges related to the effective integration of these tests into CI practices, which directly impact the quality of the produced software. The contributions of this work primarily lie in analyzing how automated tests are incorporated into CI practices, providing insights into the benefits and challenges faced by the industry. The results of this research can contribute to a better understanding of testing processes in the industry and the need for greater integration to enhance software quality.

Keywords: Software Testing. Automated Testing. Continuous Integration. Industry. devOps.

Agradecimentos

A todos os meus parentes, pelo encorajamento e apoio. Ao professor André Luiz de Oliveira, pela orientação, amizade e principalmente pela paciência, sem a qual este trabalho não se realizaria.

Aos professores do Departamento de Ciência da Computação, pelos seus ensinamentos, e aos funcionários do curso, que, durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"Lembra que o sono é sagrado e alimenta de horizontes o tempo acordado de viver".

Beto Guedes (Amor de Índio)

Conteúdo

Lista de Figuras **8**

Lista de Tabelas **9**

Lista de Abreviações **10**

1 Introdução **11**
1.1 Apresentação do Tema e sua contextualização . . . 11
1.2 Problema . . . 12
1.3 Justificativa . . . 14
1.4 Objetivo . . . 15
1.5 Metodologia . . . 15
1.6 Organização do Trabalho . . . 16

2 Fundamentação Teórica **17**
2.1 Teste de Software . . . 17
2.2 Terminologia . . . 19
2.3 Níveis de Testes . . . 19
2.3.1 Tipos de Teste . . . 20
2.4 Integração Contínua . . . 21
2.5 Mapeamento Sistemático da Literatura . . . 22
2.6 Conclusões do capítulo . . . 26

3 Trabalhos Relacionados **27**
3.1 Conclusões do capítulo . . . 33

4 Planejamento do Mapeamento Sistemático **34**
4.1 Questões de Pesquisa . . . 34
4.2 Condução . . . 37
4.2.1 Extração, análise e síntese de dados . . . 38
4.3 Conclusões do capítulo . . . 40

5 Resultados **41**
5.1 Estudos empíricos e as validações aplicadas . . . 41
5.2 Domínios de Aplicação na Integração Contínua . . . 45
5.2.1 Teste de Unidade . . . 49
5.2.2 Teste A/B . . . 51
5.2.3 Teste de Integração . . . 51
5.2.4 Teste de UI, E2E e Sistema Funcional . . . 54
5.2.5 Teste de Regressão . . . 56
5.2.6 Teste de Aceitação . . . 57
5.2.7 Teste de Cobertura de Código . . . 59
5.2.8 Teste de Performance . . . 60
5.2.9 Demais Testes . . . 62
5.3 Ferramentas de Apoio ao Teste Automatizado . . . 64

5.4 Papéis no Processo de Integração Contínua 68
5.5 Problemas Recorrentes na Integração Contínua 71
5.6 Ameaça à Validação 74
5.7 Avaliar a Qualidade da Produção Científica 75
5.8 Conclusões do capítulo 77

6 Conclusão **78**

Bibliografia **80**

Apéndices **87**
.1 Apêndice A: Tabela com Domínio da Aplicação e Técnicas e Critérios de Teste 88
.2 Apêndice B: Tabela com Análise do Qualis CAPES dos artigos 90

Lista de Figuras

2.1 Etapas do processo de execução do mapeamento sistemático da Literatura. Fonte: Kitchenham et al. (2009) 23

3.1 Número de estudos selecionados publicados por ano e sua distribuição por tipos de locais da pesquisa 29
3.2 *Wordcloud* de termos comuns usados no definições de *DevOps* criado pelo Jabbari et al. (2016) 30
3.3 Distribuição de tópicos nos estudos selecionados para revisão (PACHOULY et al., 2022). 32

4.1 Imagem da autora: Processo para seleção de estudos primários relevantes baseado no método de Kitchenham (2004). 38

5.1 Imagem da autora: Número de estudos primários por método de validação. 44
5.2 Imagem da autora: Distribuição do Domínio de Aplicação 46
5.3 Imagem da autora: Distribuição do Domínio de Aplicação para teste de unidade 50
5.4 Imagem da autora: Distribuição do Domínio de Aplicação para Testes de Integração 52
5.5 Imagem da autora: Distribuição do Domínio de Aplicação para Testes de UI/E2E/Sistema/Funcional 54
5.6 Imagem da autora: Distribuição do Domínio de Aplicação para Testes de Regressão 57
5.7 Imagem da autora: Distribuição do Domínio de Aplicação para Testes de Aceitação 58
5.8 Imagem da autora: Distribuição do Domínio de Aplicação para Teste de Cobertura de Código 59
5.9 Imagem da autora: Distribuição do Domínio de Aplicação para Teste de Performance 62
5.10 Imagem da autora: Distribuição do Domínio de Aplicação para Demais Testes identificados 63
5.11 Imagem da autora: Ferramentas utilizadas nos estudos 64
5.12 Imagem da autora: Distribuição de responsáveis pela integração contínua . 68
5.13 Imagem da autora: Número de Publicações por Ano 76
5.14 Imagem da autora: Top 10 *Jornal/Booktitles* com Mais Publicações 77

Lista de Tabelas

2.1 Definição do PICOC para o escopo desta monografia. 24

3.1 Trabalhos Relacionados 28

5.1 Distribuição dos Estudos Primários Identificados 42
5.2 Métricas de Teste por Artigo 45
5.3 Artigos e suas respectivas ferramentas de testes 65
5.4 Artigos e Responsáveis pela Integração Contínua 69

1 Tabela com Domínio da Aplicação é Testes aplicados 90
2 Tabela de classificação Qualis - CAPES 97

Lista de Abreviações

DCC Departamento de Ciência da Computução

UFJF Universidade Federal de Juiz de Fora

1 Introdução

1.1 Apresentação do Tema e sua contextualização

Nos últimos anos, houve uma demanda pelo aumento da qualidade de software, independentemente de plataforma (exs.: *web,* móvel, *desktop*) ou sistema operacional. Qualidade de software é definida como conformidade a requisitos funcionais e de desempenho declarados na especificação, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais (PRESSMAN; MAXIM, 2015).

A Garantia de Qualidade de software é uma subárea da Engenharia de Software, essencial para a construção de um software funcional, que atenda às necessidades dos usuários, de fácil utilização, manutenção e que seja seguro ao usuário mesmo em caso de ocorrência de falhas (BOURQUE et al., 1999) . As falhas podem ser decorrentes de erro humano na programação das aplicações, pela introdução intencional maliciosa de um ataque cibernético, por um erro funcional – no qual o comportamento do software está diferente do esperado –, por um comando ausente ou ainda por um erro de *hardware*. As falhas se apresentam sem avisos, gerando um impacto econômico e social muitas vezes irremediável.

O processo de construção de software não é trivial e pode ser tornar complexo dependendo das características e dimensões do sistema a ser desenvolvido (DELAMARO; JINO; MALDONADO, 2013). Por esses motivos, o processo de desenvolvimento de software está sujeito a problemas, que podem resultar na obtenção de um produto diferente do esperado por seus usuários. Falhas em software podem ser causadas por erros humanos, que acidentalmente podem introduzir defeitos na especificação, no projeto ou no código-fonte do programa. Falhas também podem ser causadas por agentes externos que exploram pontos fracos (vulnerabilidades) da aplicação de forma maliciosa, com a intenção de causar perdas ou danos à confidencialidade, integridade ou disponibilidade dos dados e do sistema.

Para garantir que defeitos introduzidos por engano humano sejam descobertos antes da liberação do software para operação, um conjunto de atividades denominado *Verificação, Validação e Teste* deve ser realizado para garantir que tanto o software que está sendo construído quanto o produto final estejam em conformidade com a sua especificação (BSTQB, 2024). Verificação, Validação e Teste é uma subárea da Engenharia de software que inclui um conjunto de métodos, técnicas e ferramentas de apoio à verificação e validação de software.

Falhas se apresentam sem avisos, gerando um impacto econômico e social muitas vezes irremediável. A ocorrência de uma falha pode ter consequências catastróficas aos usuários do sistema, como lesões, perda de vidas humanas, danos à propriedade ou ao ambiente em sistemas críticos, violações de confidencialidade, integridade e disponibilidade de dados dos usuários e do sistema em conexão com a *Internet*(WOLF; SERPANOS, 2017). O Teste de software é fundamental para a Garantia da Qualidade de aplicações de diferentes domínios, que abrangem desde serviços bancários como *internet banking*, comércio eletrônico, até sistemas automotivos como *Anti-lock Braking System* (ABS) e de controle de centrais multimídia.

1.2 Problema

Um equívoco comum é pensar que o Teste de software consiste apenas em executar a aplicação e verificar se as saídas obtidas correspondem às saídas esperadas considerando apenas os caminhos de sucesso (BURNSTEIN, 2006). Essas são apenas algumas das atividades do processo, que compreende: planejamento, projeto, implementação, execução e avaliação dos resultados da execução de casos de teste[1].

Em virtude da importância da Garantia da Qualidade de software para qualquer sistema computacional, especialmente aplicações cujas atualizações são realizadas de forma contínua e direta nos dispositivos (computadores e *smartphones*) dos usuários durante a operação, torna-se necessário identificar o estado da arte de estudos relacionados, metodologias, técnicas e ferramentas de automatização de testes que vêm sendo utilizados

[1]Caso de Teste: consiste no par formado por um dado de teste e o resultado esperado da execução do programa com este dado de teste (SOMMERVILLE, 2007)

na indústria para apoiar o teste dessas aplicações (DUVALL; MATYAS; GLOVER, 2007).

É importante considerar o envolvimento da equipe de Garantia de Qualidade e a realização de atividades de Teste de software em todas as fases (implementação e integração das unidades de software) do processo de desenvolvimento de um sistema ou aplicação, desde que isso não incorra em prejuízo econômico e não afete o planejamento do projeto (BSTQB, 2024).

Para Xi (2010), a Integração Contínua (IC) consiste em uma estratégia de unir os artefatos utilizados no processo de desenvolvimento de software de maneira constante em ciclos curtos. A IC permite aos desenvolvedores realizarem alterações no código caso o sistema apresente qualquer inconsistência e serem alertados imediatamente, tendo mais tempo para corrigir qualquer defeito, erro ou falha. Defeito é o código incorreto introduzido por um engano do programador. O erro consiste na execução do código defeituoso. Falha é a propagação do erro para a saída do sistema, ou seja, a situação em que o sistema gerou uma saída diferente da esperada, que é visível para o usuário final.

Podem existir empresas nas quais a criação de testes é automatizada, todavia a Integração Contínua dos testes não é realizada. Os testes são executados separados da aplicação (VINCENT et al., 2002). Com isso, vários problemas podem ocorrer, causando prejuízos significativos. Devido à demora na detecção dos problemas, falhas podem ser frequentes e de maior probabilidade de ocorrência. Um dos motivos para muitas empresas não adotarem a IC é decorrente dos altos custos do teste por módulo. Tais custos referem-se à infraestrutura de *hardware* e de servidor físico ou em nuvem computacional. Além disso, gestores não consideram relevante que os testes implementados pela equipe de Garantia de Qualidade sejam incorporados ao processo de IC devido à falta de preparo da equipe para o processo, à necessidade de mão de obra para implantação, aos prazos de entregas curtos, entre outros fatores. Todavia, a adoção da prática de Integração Contínua contribui para elevar o nível de maturidade dos processos de teste e de desenvolvimento de software.

1.3 Justificativa

A metodologia de desenvolvimento de software mais utilizada atualmente é a ágil, que consiste em um conjunto de técnicas para a gestão de projetos, oferecendo mais rapidez nas entregas das tarefas. Dessa forma, os testes de software são de grande importância. Os testes são automatizados e executados a cada *commit*, que é o ato de enviar o código para um gerenciador de controle de versão, reduzindo significativamente o custo de testes manuais, o tempo, a alocação de esforços, o desenvolvimento, a execução e manutenção do software. Além disso, o teste cria garantias de estabilidade para o desenvolvimento de novos recursos e garante a funcionalidade conforme esperado na especificação do projeto.

A Integração Contínua é uma das práticas mais importantes do desenvolvimento ágil e que através dela, é possível agilizar tarefas demoradas, como a compilação de um projeto e a execução dos seus testes automatizados (HENRIQUE, 2018). Essa prática faz com que os desenvolvedores, gestores e demais envolvidos sejam mais facilmente notificados de erros de compilação ou falhas nos testes automatizados. Além disso, a IC fornece resultados para discutir melhorias no processo de desenvolvimento, na equipe de qualidade, na homologação pela equipe de produtos e na entrega destes para o cliente.

Na indústria de software, os testes mais difundidos são os testes de unidade referentes à análise de componentes. Considerados na hierarquia como um nível mais baixo (próximo ao código-fonte) do projeto, geralmente são realizados pelos desenvolvedores envolvidos. Esses testes têm como objetivo validar as unidades isoladamente, buscando garantir que a lógica de cada uma delas esteja correta e tenha o resultado esperado. O Teste de Integração tem por objetivo garantir a interação entre as unidades. O teste de ponta a ponta (E2E) busca verificar o comportamento do sistema como um todo (BSTQB, 2024). Geralmente simula atividades que o usuário final teria realizado e um ambiente preparado de maneira semelhante ao de produção.

A motivação para realizar este trabalho está na necessidade de identificar estudos relacionados a técnicas de automação de teste e suas aplicações no processo de integração contínua na indústria para obter evidências empíricas sobre as técnicas, tecnologias, seus benefícios e limitações no estado da arte e oportunidades de pesquisa na área de integração contínua de software.

1.4 Objetivo

O objetivo deste trabalho foi realizar um mapeamento sistemático sobre o uso de técnicas de teste automatizado na indústria no contexto de Integração Contínua (IC). O objetivo foi identificar as metodologias e os estudos empíricos que investigam a influência dos testes desenvolvidos por equipes de qualidade e sua integração na IC.Para verificar se o objetivo foi alcançado, questões de pesquisa relacionados aos seguintes temas respondidas com a realização deste estudo:

- Estudos empíricos em testes automatizados na indústria;
- Domínios de aplicação utilizam testes automatizados na Integração Contínua;
- Ferramentas de suporte aos testes automatizados na Integração Contínua;
- Papéis da área de Qualidade de software envolvidos no processo criação e execução de testes automatizados;
- Benefícios e desafios dos testes de qualidade na Integração Contínua.

1.5 Metodologia

Este mapeamento sistemático de literatura foi conduzida em conformidade com o processo proposto por Kitchenham et al. (2009), que inclui as fases de planejemento, condução e sumarização (relato) dos resultados. O método *Goals-Questions-Metrics* (GQM) Caldiera e Rombach (1994) foi utilizado na definição do objetivo e das questões de pesquisa respondidas com a realização deste estudo, bem como na definição das informações do formulário de extração das informações de cada estudo primário analisado.

As seguintes atividades foram desenvolvidas para a realização deste trabalho:

a) levantamento dos trabalhos relacionados: levantamento de trabalhos relacionados a revisões sistemáticas de literatura na área de teste de software com o objetivo de identificar lacunas no estado de arte;

b) planejamento da revisão: definição do protocolo de revisão, que inclui a especificação dos objetivos, questões de pesquisa bem como a *string* de busca, os critérios

de inclusão e exclusão de estudos primários e as bases de dados consideradas na revisão;

c) condução (execução) da revisão: execução da *string* de busca nas bases de dados selecionadas, exclusão de estudos duplicados, seleção de estudos primários por meio da leitura do título e do resumo, seleção de estudos primários por meio da leitura completa, realização de *snowballing* nos estudos selecionados para identificar os que são possivelmente relacionados;

d) coleta de dados: definição do formulário de extração de dados a ser usado para extrair informações de cada publicação; e coleta dos dados de cada estudo primário selecionado;

e) sumarização dos resultados da revisão: análise dos dados coletados e produção de gráficos e tabelas a serem utilizados para responder cada questão de pesquisa definida na revisão.

A ferramenta *Rayyan*[2], foi utilizada para apoiar o gerenciamento do processo de revisão e o armazenamento dos estudos encontrados e selecionados.

1.6 Organização do Trabalho

O presente trabalho está organizado em seis capítulos. No Capítulo 2 é apresentada a fundamentação teórica, em que são descritos os conceitos de teste de software e integração contínua e o processo de mapeamento sistemático da literatura proposto por (KITCHENHAM et al., 2009), necessários para o entendimento do trabalho. No Capítulo 3 são apresentados os trabalhos relacionados. No Capítulo 4 são descritas as fases de planejamento e condução do Mapeamento Sistemático apresentado neste trabalho. No capítulo 5 são apresentados os resultados da revisão. Por fim, no Capítulo 6 é apresentada a conclusão.

[2]https://rayyan.ai/

2 Fundamentação Teórica

Neste capítulo são apresentadas as definições e os conceitos básicos da área de teste de software como item de teste, caso de teste, engano, defeito, erro e falha, o processo de teste e os níveis de teste, a definição de integração contínua e uma visão geral do processo de mapeamento sistemático de literatura proposto por Kitchenham et al. (2009) necessários a compreensão das contribuições deste trabalho.

2.1 Teste de Software

O processo de desenvolvimento de software envolve um conjunto de atividades nas quais, apesar da existência de técnicas, métodos e ferramentas, defeitos e erros no produto podem ocorrer. Atividades de Garantia de Qualidade de software vêm sendo introduzidas ao longo do processo de desenvolvimento para identificar e corrigir defeitos no software. Atividades de Verificação, Validação e Teste têm o objetivo de minimizar a ocorrência de erros e falhas de software e suas consequências (MALDONADO et al., 2004). Verificação consiste em verificar se o software foi desenvolvido em conformidade com a especificação, ou seja, em conformidade os requisitos funcionais e não funcionais. Validação é garantir que o software atende às necessidades e expectativas do cliente. Portanto, verificação é uma atividade de análise estática na qual métodos como inspeção podem ser aplicados para verificar se o software desenvolvido de forma correta, ou seja, em conformidade com sua especificação. Por outro lado, validação é uma atividade de análise dinâmica, que envolve a execução de testes para demonstrar que o software se comporta conforme o esperado.

Teste de software pode ser definido como o processo de análise de um item de software, que pode ser uma função, algoritmo, sistema, ou subsistema, para detectar a diferença entre as condições desejadas (estado esperado) e as condições existentes (estado obtido) e a avaliação de uma característica/atributo de um item de teste. Teste é uma atividade de análise dinâmica do produto relevante para a identificação e eliminação de

erros.

O Teste de software, de acordo com Roman (2018), é uma maneira de avaliar a qualidade do software e reduzir seu risco de falha em operação. Pressman e Maxim (2021) dizem que

> Qualidade de software é a conformidade a requisitos funcionais e de desempenho declarados na especificação, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais.

Roman (2018) define teste de software como uma maneira de avaliar a qualidade do software e reduzir o risco de falha do software em produção, ou seja, disponível para utilização do usuário final.

O processo de teste de produtos de software envolve quatro etapas: planejamento de teste, projeto de casos de teste, execução e avaliação dos resultados do testes (MALDONADO et al., 2004). Essas atividades devem ser realizadas ao longo do processo de desenvolvimento de software e concretizam-se em teste de unidade, teste de integração e teste de sistema (aceitação). O teste de unidade concentra esforço na menor unidade do projeto de software, ou seja, em algoritmos e funções, para identificar erros de lógica e de implementação. O teste de integração é uma atividade sistemática aplicada durante a integração de unidades de um programa para identificar erros associados a interfaces entre módulos. O objetivo é identificar erros de operação em unidades quando executadas em conjunto. O teste de sistema, também denominado teste de aceitação, é realizado após a integração das unidades para identificar erros de funções e em características não funcionais do software como desempenho que estejam em conformidade com a especificação.

O projeto e avaliação da qualidade do conjunto de casos de teste (T) são cruciais para o teste de um produto (P), uma vez que não é possível considerar todos o domínio de entrada para avaliar aspectos funcionais e operacionais de um item de teste. O objetivo da atividade de teste é identificar o conjunto de casos de teste que tenham maior probabilidade de revelar a maioria dos erros no software com o mínimo de tempo e esforço. Critérios de teste de software para identificar os casos teste com maior probabilidade de revelar erros, foram estabelecidos por técnicas de teste funcional, estrutural e baseada em erros (MALDONADO et al., 2004).

2.2 Terminologia

Existem alguns conceitos básicos, associados ao teste de software, essenciais para a compreensão dessa atividade. Esses conceitos são o de Defeitos, Erros e Falhas. Neto e Claudio (2007) os definem assim: defeito: um ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, solucionar um problema ou recorrer a um método ou a uma ferramenta, como, por exemplo, uma instrução ou comando incorreto. Erro: uma exposição concreta de um defeito de um elemento de software, a diferença entre o valor obtido e o valor esperado. Em outras palavras, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa consistem em erros. Por fim, a Falha é o comportamento incorreto de um software. Uma falha pode ter sido causada por diversos erros, todavia erros podem nunca causar uma falha.

O teste de software é um termo amplo que abrange uma variedade de atividades ao longo do ciclo de desenvolvimento. Testes podem não ser a garantia de um software totalmente livre de defeitos, porém são indispensáveis durante todo o desenvolvimento de um software, pois minimizam os defeitos e futuramente ajudam na parte de custos.

2.3 Níveis de Testes

De acordo com o *syllabus* do BSTQB (2024), há cinco níveis de teste no ciclo de vida do software. O primeiro é o Teste de Componente, também conhecido como teste de unidade, que se concentra na verificação de componentes de forma isolada. Esse tipo de teste geralmente requer suporte específico, como *frameworks* ou estruturas de teste, e costuma ser realizado por desenvolvedores em seus próprios ambientes.

O segundo nível é o Teste de Integração de Componentes, cujo foco está nas interfaces e interações entre diferentes componentes do sistema. Esse tipo de teste pode seguir diferentes estratégias de integração, como as abordagens *bottom-up*, *top-down* ou *big-bang*. *Bottom-up*: Integra e testa primeiro os módulos de nível inferior, subindo gradualmente para os superiores. *Top-down*: Começa integrando e testando os módulos de nível superior, substituindo temporariamente os inferiores por stubs. *Big-bang*: Integra todos os módulos de uma vez e testa o sistema completo, mas dificulta a identificação de

problemas.

Em seguida, o Teste de Sistema abrange o comportamento geral e as funcionalidades de todo o sistema ou produto. Ele inclui tanto testes funcionais quanto não funcionais, e é comum que as características de qualidade sejam avaliadas em um ambiente de teste representativo. Simulações de subsistemas também podem ser utilizadas nesse nível, que geralmente envolve uma equipe de teste independente.

O Teste de Integração de Sistema verifica as interfaces entre o sistema principal e outros sistemas ou serviços externos. Esse teste requer um ambiente de teste apropriado, idealmente semelhante ao ambiente operacional real.

Por fim, o Teste de Aceite valida se o sistema está pronto para ser implantado, garantindo que atende às necessidades de negócio dos usuários. As principais formas de teste de aceite incluem Teste de Aceite do Usuário (UAT), Teste de Aceite Operacional, Teste de Aceite Contratual e Normativo, além de testes alfa e beta.

Os níveis de teste são diferenciados por diversos atributos, como o objeto e os objetivos do teste, a base de teste utilizada, o tipo de defeitos e falhas que se pretende identificar, além da abordagem e das responsabilidades de quem os conduz.

2.3.1 Tipos de Teste

Roman (2018) propõem que um tipo de teste é um grupo de atividades destinadas a verificar características específicas de um sistema de software ou parte de um sistema. Os objetivos podem incluir: avaliar características de qualidade funcional, como integridade, correção e adequação; e avaliar características de qualidade não funcionais, como confiabilidade, eficiência de desempenho, segurança, compatibilidade e usabilidade.

Testes manuais, segundo Neto (2019), são aqueles cujo fluxo é realizado passo a passo, de uma forma em que é normalmente guiado por um roteiro do que deve ser realizado e do comportamento resultante das ações realizadas. Nesse contexto, os testes manuais são úteis quando se precisa de agilidade na criação do teste e de certa subjetividade ou análise de um ser humano.

Testes automatizados, para Neto (2019), são as possibilidades de simular atividades de usuários ou humanos de forma que não exijam procedimentos manuais no processo

de execução dos testes. Os Métodos Ágeis, como *Extreme Programming* (XP), defendem explicitamente o uso da automação de testes para garantir a qualidade dos sistemas desenvolvidos. Um teste automatizado tem a facilidade de poder ser executado várias vezes sem a interação humana no processo.

2.4 Integração Contínua

IBM (2024) define a integração contínua é uma prática de *DevOps* em que os desenvolvedores integram regularmente novas alterações e códigos ao repositório do software em desenvolvimento. Esse processo auxilia nas fases subsequentes de criação e testes, com o objetivo de identificar e corrigir erros de forma rápida. Os testes automatizados são executados em cada ciclo de desenvolvimento para detectar problemas de integração antecipadamente, quando ainda são mais fáceis de corrigir. Essa abordagem também ajuda a evitar complicações na fusão final da versão. De maneira geral, a integração contínua simplifica o processo de desenvolvimento, resultando em software de maior qualidade e prazos de entrega mais previsíveis.

Para Hirano (2019) A integração contínua(CI) é uma prática fundamental no processo de *DevOps*, onde os desenvolvedores frequentemente integram novas alterações ao repositório de código, facilitando as fases subsequentes de construção e testes. A cada alteração confirmada no repositório, uma compilação e um teste automatizados são acionados, garantindo que eventuais erros ou conflitos sejam detectados rapidamente. Esse processo é crucial para evitar que as cópias locais do código fiquem desatualizadas em relação ao ramo principal, o que poderia causar problemas complexos de integração no futuro.

Uma característica central da CI é a realização de testes automatizados, que ocorrem a cada integração. Esses testes são essenciais para identificar problemas de integração logo no início do ciclo de desenvolvimento, quando ainda são mais fáceis de resolver. Detectar erros antecipadamente evita falhas durante a fase final de mesclagem e ajuda a garantir que a qualidade do código seja mantida ao longo de todo o processo de desenvolvimento. Essa abordagem também permite simplificar o processo de criação, o que resulta em cronogramas de entrega mais previsíveis e um software de maior qualidade.

Além disso, para Martins (2024) a integração contínua cobre os estágios de construção e execução de teste de unidade no processo de lançamento do software. Cada nova revisão do código dispara uma nova rodada de compilação e testes automatizados. Com a introdução da entrega contínua(CD), as alterações de código são automaticamente criadas, testadas e preparadas para a liberação em produção, proporcionando uma transição mais suave entre as fases de desenvolvimento e lançamento. No entanto, os testes automáticos realizados frequentemente envolvem análise estática de código, que busca detectar *bugs*(falha que impede o funcionamento adequado de um hardware e software) e identificar *code smells*(indicadores de possíveis problemas mais profundos no código), mas podem não ser suficientes para cobrir todos os cenários.

Portanto, a inclusão de testes automatizados desenvolvidos pelas equipes de qualidade é de fundamental importância para garantir que o ciclo de vida do software ocorra sem problemas. Esses testes complementam a análise estática, abrangendo diferentes cenários e comportamentos do software, o que reduz significativamente a probabilidade de problemas críticos surgirem nas fases finais do desenvolvimento. Como o software pode ser compilado e testado em vários momentos ao longo do processo, a CI assegura que a equipe esteja constantemente ciente do estado do projeto, facilitando a correção imediata de eventuais erros e mantendo um fluxo contínuo de integração e entrega de código.

Esse processo não só melhora a qualidade do código, mas também promove uma cultura colaborativa e integrada dentro das equipes de desenvolvimento e operação, ajudando a eliminar silos entre os grupos e a acelerar o ciclo de lançamento. A integração contínua, quando bem implementada, torna-se um pré-requisito para as fases seguintes, como a entrega contínua e a liberação final do software em produção, garantindo que o software esteja sempre em um estado pronto para ser entregue com segurança.

2.5 Mapeamento Sistemático da Literatura

Um dos pilares da engenharia de software baseada em evidência, inspirado na medicina baseada em evidência, é a condução do Mapeamento Sistemático da Literatura como método para realizar o levantamento do estado da arte (DERMEVAL; COELHO; BITTENCOURT, 2020). O mapeamento sistemático busca minimizar erros sistemáticos e

aleatórios e definir claramente o procedimento a ser adotado na realização do levantamento do estado da arte de um tema de pesquisa. Um mapeamento sistemático sintetiza a literatura existente de forma justa e parece justa para outros pesquisadores.

Nos processos do mapeamento sistemático baseados nas diretrizes de Kitchenham et al. (2009), o protocolo é fundamentado em outros protocolos amplamente utilizados na pesquisa médica baseada em evidência. Traduzindo para a área da computação, essa mesma definição é amplamente utilizada nos trabalhos sistemáticos de levantamento da literatura da área de Informática. Esse processo é o *start* de pesquisa, que deve ser evidentemente relatado antes da execução da revisão literária.

Na Figura 2.1 estão presentes os elementos que são itens obrigatórios do protocolo a respeito do mapeamento sistemático da literatura. Nela se encontram: a definição, a questão de pesquisa, a busca e seleção dos estudos, a avaliação de qualidade, a extração de dados, a síntese, a análise dos dados e o resumo.

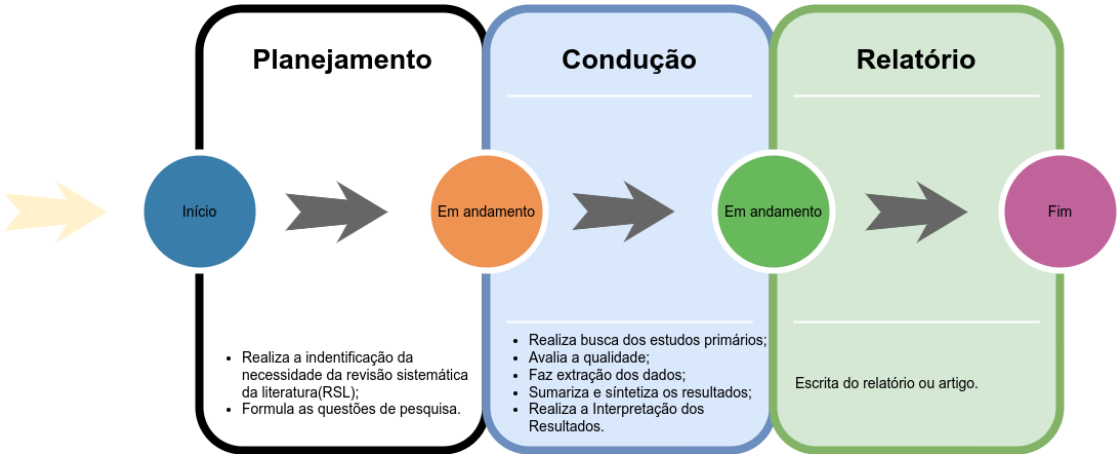


Figura 2.1: Etapas do processo de execução do mapeamento sistemático da Literatura. Fonte: Kitchenham et al. (2009)

Petticrew e Roberts (2008) fornece uma visão geral abrangente dos métodos de mapeamento sistemático nas ciências sociais. Ele inclui uma discussão detalhada da estrutura PICOC, que é uma ferramenta amplamente usada para formular perguntas de pesquisa e identificar estudos relevantes. PICOC significa:

- População: O grupo de pessoas ou coisas sendo estudado.
- Intervenção: O tratamento ou exposição sendo avaliado.
- Comparação: O grupo de controle ou tratamento alternativo sendo comparado.

- Resultado: O efeito ou resultado que está sendo medido.

- Contexto: O cenário ou as circunstâncias em que a intervenção está ocorrendo.

A estrutura PICOC é útil para esclarecer a questão da pesquisa e garantir que a revisão se concentre em estudos relevantes. Ao definir explicitamente a população, intervenção, comparação, resultado e contexto, os pesquisadores podem pesquisar e avaliar a literatura de forma mais eficaz. Com base nas indagações e no mapeamento, adotou-se o método PICOC proposto por Petticrew e Roberts (2008), o qual visa delinear a abrangência de uma questão de pesquisa. O quadro contendo o modelo de PICOC elaborado para este estudo está apresentado na tabela 2.1 :

Tabela 2.1: Definição do PICOC para o escopo desta monografia.

PICOC	Descrição
Population (P)	Integração contínua na indústria
Intervention (I)	Técnicas de teste de software aplicadas de Integração Contínua
Comparison (C)	Não se aplica
Outcome (O)	Estudos empíricos que descrevem os testes de software aplicados na Integração Contínua
Context (C)	Indústria

A análise da Tabela 2.1 no contexto dos testes de software na Integração Contínua (CI) revela uma abordagem metodológica clara e objetiva para investigar práticas críticas na indústria de software. O *framework* PICOC permite estruturar a pesquisa de forma precisa, proporcionando uma visão abrangente sobre como as técnicas de teste são integradas aos processos de desenvolvimento contínuo. Nesse sentido, é fundamental argumentar que a adoção de testes de software bem planejados na CI tem implicações diretas na qualidade e agilidade do desenvolvimento, beneficiando as empresas que implementam essas práticas.

Primeiramente, o elemento População (Population) da tabela 2.1 define o escopo da pesquisa como os projetos e empresas que já adotam a CI. A integração contínua é uma prática amplamente utilizada na indústria de software moderna devido à necessidade de ciclos de desenvolvimento ágeis e iterações rápidas. Nessa abordagem, os testes de software desempenham um papel essencial na garantia de que novos incrementos de código não introduzam falhas ou comprometam a estabilidade do sistema. Assim, a pesquisa se

justifica ao investigar como diferentes empresas utilizam testes automatizados para assegurar a qualidade em ambientes de CI, abrangendo uma variedade de setores e contextos industriais.

Em seguida, o componente Intervenção*Intervention*) concentra-se nas técnicas de teste de software aplicadas no processo de CI. Diversas práticas, como testes unitários, testes de integração, testes de ponta a ponta e testes de desempenho, são fundamentais para identificar rapidamente erros no código e garantir que as funcionalidades sejam mantidas. A importância desse ponto reside no fato de que, com a adoção de CI, o tempo entre as iterações é reduzido, tornando o *feedback* contínuo e imediato um fator crucial para o sucesso do projeto. Argumenta-se que a implementação eficaz dessas técnicas de teste pode reduzir o retrabalho, aumentar a eficiência e proporcionar maior confiança nos lançamentos frequentes de software.

Entretanto, o componente Comparação(*Comparison*) não se aplica a essa pesquisa, o que, de certo modo, fortalece o foco na descrição e análise das práticas de teste. Ao não envolver comparações entre grupos ou métodos, o estudo ganha profundidade ao se concentrar em entender o impacto direto das práticas de teste em CI. A ausência de um grupo de controle não diminui o rigor da pesquisa, uma vez que o objetivo não é determinar a superioridade de uma abordagem, mas sim compreender as práticas que estão funcionando na indústria e seus resultados concretos.

O componente Resultado(*Outcome*), por sua vez, destaca a necessidade de identificar e analisar estudos empíricos que descrevem a aplicação de testes de software em CI. A busca por evidências científicas sobre a eficácia das práticas de teste no contexto da Integração Contínua é um dos pontos centrais dessa investigação. Estudos empíricos oferecem uma base sólida para avaliar os benefícios e desafios associados à implementação de testes no desenvolvimento contínuo, fornecendo *insights* valiosos sobre como essas práticas podem ser otimizadas. Nesse ponto, é argumentado que a pesquisa pode auxiliar tanto empresas que já adotam CI quanto aquelas que estão em processo de transição, ao identificar as melhores práticas e soluções para problemas comuns.

Finalmente, o Contexto (*Context*) da pesquisa é definido como o ambiente industrial em geral. Considerando as especificidades e desafios da indústria, como prazos

apertados, orçamentos limitados e a necessidade de lançamentos frequentes de software, os testes de software tornam-se ainda mais críticos. O contexto industrial traz à tona a realidade de que muitas empresas, especialmente as de grande porte, enfrentam dificuldades na integração de testes com seus processos de CI. Portanto, é essencial argumentar que pesquisas que abordem essa questão podem fornecer orientações práticas para a melhoria contínua e adoção mais eficiente dessas práticas no setor.

Em resumo, a análise da Tabela PICOC oferece uma base metodológica robusta para investigar a aplicação de testes de software no contexto da Integração Contínua. Ao delimitar claramente o escopo, a pesquisa contribui para uma compreensão mais aprofundada de como as práticas de teste impactam a qualidade do software e a agilidade dos processos de desenvolvimento. Além disso, ao focar em estudos empíricos, a pesquisa pode fornecer *insights* práticos e aplicáveis, que são de grande valor para empresas que buscam melhorar seus processos de CI.

Possíveis direcionamentos para essa investigação incluem a realização de um mapeamento sistemático da literatura, a análise de estudos de caso em empresas que adotam CI e *surveys* que busquem coletar dados sobre as práticas de teste em diferentes organizações. Esses métodos complementares permitiriam uma visão holística e fundamentada sobre o tema, trazendo contribuições significativas tanto para a academia quanto para o mercado.

2.6 Conclusões do capítulo

Este capítulo apresentou uma breve revisão dos temas que serão aprofundados ao longo deste trabalho, destacando a importância dos testes de software. Testar um software vai além da simples execução e verificação de resultados; é um processo estruturado que envolve planejamento, análise, modelagem e implementação, além de gerar relatórios de progresso e avaliação. A presença do testador em todas as fases do desenvolvimento é essencial para detectar problemas precocemente, o que acelera sua correção e reduz custos, aumentando, assim, as chances de sucesso dos projetos.

3 Trabalhos Relacionados

Este capítulo visa a apresentar alguns dos trabalhos relacionados ao objetivo de pesquisa, que é automatização de testes de software na indústria: um mapeamento sistemático. Os artigos selecionados previamente serviram como base para o desenvolvimento deste trabalho. Abaixo estão alguns trabalhos que mais se aproximam ou trazem direções para o desenvolvimento do mapeamento sistemático sobre teste de software aplicado na indústria e integração contínua. Informações importantes sobre os artigos podem ser encontradas no Quadro 3.1.

Muitos dos trabalhos encontrados durante a fase de pesquisa buscaram de alguma forma refinar os resultados obtidos referentes ao mapeamento sistemático para responder às perguntas em questão.

Isso pode ser visto no trabalho apresentado por Shahin, Babar e Zhu (2017), que visa à revisão sistemática da literatura tomando como base a técnica de Engenharia de software Baseada em Evidências escrita por (PETERSEN; VAKKALANKA; KUZNIARZ, 2015). O trabalho foi realizado considerando o período de publicação de 2004 até 1 de junho de 2016. Nessa pesquisa foram analisados 69 artigos relacionados à temática. As abordagens de 30 desses artigos estão associadas a alguma ferramenta que facilita o processo de Integração Contínua, como redução do tempo de construção e teste de Integração Contínua, aumento da visibilidade e conscientização sobre os resultados de teste de IC, suporte aos testes contínuos (semi) automatizados, detecção de violações, falhas na IC, abordagem de questões de segurança e escalabilidade na implantação do produto, melhora na segurança e confiabilidade do processo de implantação.

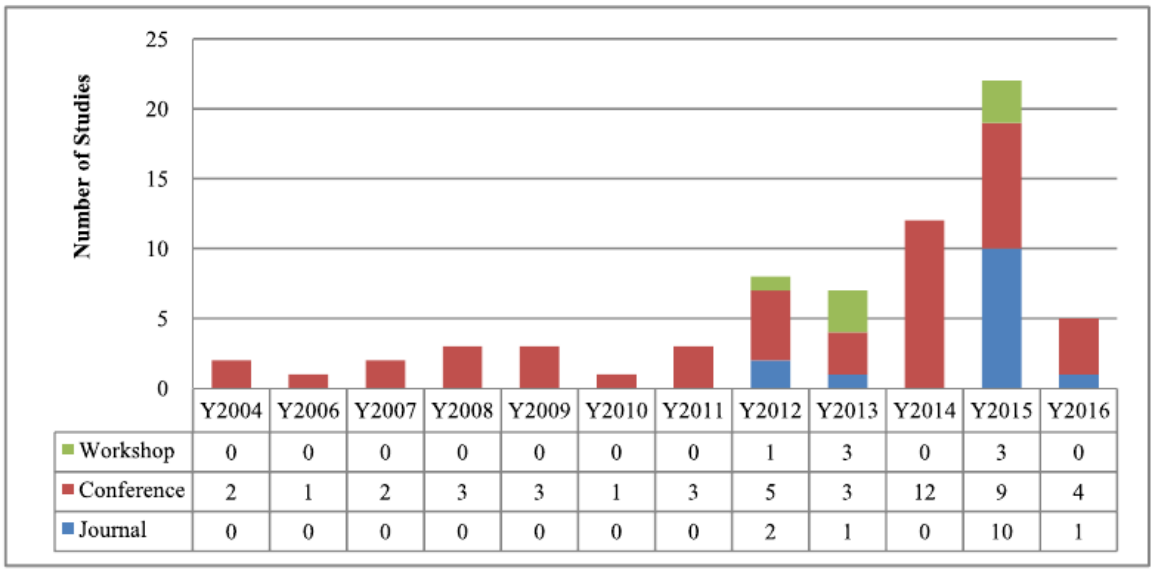
Dada a crescente importância da segurança nos *pipelines* de implantação, há necessidade de mais pesquisas para explorar como os *pipelines* de implantação devem ser projetados e implementadas para mitigar problemas de segurança.

Na figura 3.1 é possível observar que houve um crescimento maior sobre o assunto a partir de 2012, com um pico em 2015 referente ao período da pesquisa. A pesquisa revelou que as práticas contínuas têm sido aplicadas com sucesso em projetos novos e de

Tabela 3.1: Trabalhos Relacionados

Artigo	**Pontos Positivos**	**Pontos Negativos**
Shahin, Babar e Zhu (2017)	Detecção de ferramentas que facilitam o processo de Integração Contínua.	Faltam maiores detalhes sobre a segurança na pipeline de IC com relação às ferramentas.
Jabbari et al. (2016)	DevOps como metodologia de desenvolvimento voltada a preencher lacuna entre desenvolvimento e operações, atuação na comunicação e colaboração, integração contínua, garantia da qualidade e entrega com implantação automatizada.	Conduzir levantamento com maior nível de detalhes sobre assunto correlato de profissionais.
Kumaresen, Frasheri e Enoiu (2020)	O resultado da análise constatou que a maioria das abordagens são testes de software baseados em agente (ABST), implementados e elaborados usando o framework JADE.	escopo limitado do estudo.
Pachouly et al. (2022)	Arquitetura capaz de não só realizar a previsão de defeitos de software, mas conseguir prever a gravidade do defeito, estimativas de defeitos, referências de código, alocação de recursos e tipos de defeitos.	Existe a necessidade da previsibilidade do que ocorrer na produção.
Bosch (2017)	Testes exploratórios na pipeline de integração contínua, além de serem eficientes para produtos.	Necessidade de casos de usos para validação da hipótese sobre testes exploratórios.
Martensson, Stahl e Bosch (2017)	Modelo EMFIS permite que as empresas esclareçam a apresentação do estado atual da organização de obstáculos à integração contínua e visualizem onde a organização precisa se concentrar para aumentar a integração de software.	Examinar mais detalhadamente o comportamento de integração contínua dos desenvolvedores, afetado por fatores como organização e liderança.

manutenção, um ponto positivo do estudo. Além disso, as práticas de IC tornaram-se importantes para a área de pesquisa e para a prática na engenharia de software. As limitações relatadas se concentram nas abordagens, ferramentas e práticas. Pode-se citar a restruturação do sistema para suportar as práticas de IC.



	Y2004	Y2006	Y2007	Y2008	Y2009	Y2010	Y2011	Y2012	Y2013	Y2014	Y2015	Y2016
Workshop	0	0	0	0	0	0	0	1	3	0	3	0
Conference	2	1	2	3	3	1	3	5	3	12	9	4
Journal	0	0	0	0	0	0	0	2	1	0	10	1

Figura 3.1: Número de estudos selecionados publicados por ano e sua distribuição por tipos de locais da pesquisa

Já em Jabbari et al. (2016), a pesquisa se concentrou em compreender o papel do *DevOps*, que é a combinação de desenvolvimento e operações em engenharia de software, uma vez que um novo conceito é introduzido, e pouco se compreende do que implica. Com isso, foi realizado uma revisão sistemática da literatura com o objetivo de caracterizar os *DevOps* explorando os componentes centrais das definições e relatos existentes.

O resultado da pesquisa de Jabbari et al. (2016) se concentrou na análise de 44 estudos selecionados, que relatam uma definição de *DevOps*. Desses estudos, 15 explicitaram as práticas dessa profissão. Além disso, 15 estudos as relacionaram com outras funções existentes. O número total de artigos pré-selecionados para a pesquisa era 49. Os pontos positivos dessa pesquisa foram a identificação dos *DevOps* como metodologia de desenvolvimento voltada a preencher lacuna entre desenvolvimento e operações; atuação na comunicação e colaboração; integração contínua; garantia da qualidade e entrega com implantação automatizada; e a utilização de conjunto de práticas de desenvolvimento. Na figura 3.2 estão representadas as definições mais comuns de *DevOps*. Entre os pontos

a serem aprimorados, está conduzir levantamento com maior nível de detalhes sobre o assunto com relatos de profissionais.

Figura 3.2: *Wordcloud* de termos comuns usados no definições de *DevOps* criado pelo Jabbari et al. (2016)

Em Kumaresen, Frasheri e Enoiu (2020), os autores buscaram realizar um mapeamento sistemático da adoção de agentes em teste de software de forma contínua, na qual várias metodologias, arquiteturas e ferramentas de agentes são empregadas para melhorar diferentes problemas de teste. Como a pesquisa é uma área que abrange vários tópicos, os autores definiram o trabalho como teste de software baseado em agente (ABST).

Os autores consideraram o período de 1999 a 2020 para a revisão. Nesse intervalo de tempo, foram encontrados 41 trabalhos relacionados. O ABST é a aplicação de agentes (exs.: agentes de software, agentes inteligentes, agentes autônomos e sistemas multiagen-

tes) para problemas de testes de software, automatizando tarefas de testes complexos. O resultado da análise constatou que a maioria das abordagens ABST implementadas são elaboradas usando o *framework* JADE e a linguagem JAVA. Além disso, os resultados evidenciaram que os testes são em nível de sistema para testes funcionais, não funcionais e de caixa branca. Pontos a serem explorados são os testes de regressão, cujo assunto foi encontrado em 4 resultados relacionados.

Pachouly et al. (2022) realizaram uma revisão sistemática da literatura sobre a previsão de defeitos de software usando inteligência artificial. Os autores enfatizam que a entrega de produtos de software com alta qualidade é uma tarefa desafiadora, pois para isso existe a necessidade de uma coordenação com bom planejamento, boa execução e teste. Além disso, muitos *softwares* apresentam altos números de defeitos em ambiente de produção, gerando custos altos para correção, tempo e reputação da empresa.

O trabalho dos autores consistiu em analisar artigos de 2010 a 2021 que tenham como assunto os temas distribuídos em tópicos representados na Figura 3.3. Para a análise, 146 publicações foram identificadas e selecionadas por abordarem as questões de pesquisa propostas no trabalho.

Conforme os resultados da pesquisa de Pachouly et al. (2022), vários estudos se concentram em técnicas baseadas em Inteligência Artificial, como *Machine Learning* e *Deep Learning*, para prever defeitos de software devido à sua eficácia em fornecer resultados corretos com alta confiança. Os autores propuseram uma arquitetura capaz de não só realizar a previsão de defeitos de software, mas também da gravidade do defeito, estimativas de defeitos, referências de código, alocação de recursos e tipos de defeitos. A limitação existente é que os sistemas são baseados em previsão de problemas que podem ser corrigidos antes da entrega, porém há a necessidade da previsibilidade do que ocorre na produção.

Bosch (2017) propôs um método de teste que incorpora o teste exploratório como uma atividade no *pipeline* de integração e entrega contínua, baseada em elementos de outras técnicas de teste, como a baseada em cenário, em equipes e teste em sessões de *time-box*. A pesquisa consistiu em uma revisão sistemática da literatura sobre Testes exploratórios de sistemas em larga escala - Testes no pipeline de integração e entrega

Figura 3.3: Distribuição de tópicos nos estudos selecionados para revisão (PACHOULY et al., 2022).

contínua.

O trabalho consistiu na análise de 39 publicações relevantes para a pesquisa. Além disso, os autores realizaram uma validação recorrendo à entrevista com 18 engenheiros e 7 pilotos de teste de voo, questionando qual método consideram o mais eficiente para os testes de estudo de caso. A conclusão foi de que o teste exploratório foi o mais apreciado.

Ponto positivo da pesquisa: o método de testes apresentado consegue incorporar testes exploratórios no *pipeline* de integração contínua, além de ser eficiente para produtos de software complexos e de grande escalabilidade. Um ponto a ser melhor avaliado é explorar mais casos de usos para a validação da hipótese levantada.

Mårtensson, Ståhl e Bosch (2017) realizaram uma revisão sistemática da literatura para investigar os impedimentos da integração contínua em projetos de grande escala e a validação do modelo EMFIS (*Enable More Frequent Integration of Software*), desenvolvido pelos próprios autores.

A RSL consistiu na análise de 74 artigos e 4 livros pré-selecionados do período de 1996 a 2017. Como resultado da avaliação da literatura, foi analisada a possibilidade de desenvolvimento de um modelo que pudesse ser usado como uma representação da situação atual de uma organização em relação aos impedimentos da integração contínua. Em suma, chegaram às limitações e aos desafios relacionados à IC em larga escala.

Os autores desenvolveram o modelo EMFIS, que permite às empresas esclarecerem a apresentação do estado atual da organização de obstáculos à integração contínua e visualizarem onde a organização precisa se concentrar para aumentar a integração de software. O modelo é utilizado para realizar a avaliação de 12 fatores. Nele, as avaliações dos participantes representando os desenvolvedores são resumidas separadamente das avaliações dos participantes representando os processos. Um ponto a ser desenvolvido é examinar mais detalhadamente como o comportamento de integração contínua dos desenvolvedores é afetado por fatores como organização e liderança.

3.1 Conclusões do capítulo

A análise dos artigos para este capítulo apenas considerou artigos escritos na língua inglesa. A seleção dos artigos foi manualmente, sem a utilização de um classificador, pois a busca não teve um resultado expressivo. O período considerado para os resultados foi de 2015 a 2023. Contudo, os artigos selecionados e supramencionados demonstram a riqueza de informações e detalhes para a continuação e o desenvolvimento deste trabalho. Ficou explícito, em alguns resultados da RSL, que existem problemas da IC em larga escala. Além disso, alguns autores usam validação da RSL através de estudos de caso, para confirmar a hipótese e propor arquiteturas e modelos. Com todos os trabalhos dos autores avaliados, a RSL nos proporciona diferentes visões e opções de linha de pesquisa. Caberá escolher criteriosamente a linha de pesquisa mais adequada às perguntas que pretendemos responder.

4 Planejamento do Mapeamento Sistemático

No planejamento desta pesquisa, foram estabelecidos os propósitos e um protocolo para realizar o método de revisão, visando a analisar a aplicação de testes automatizados no contexto da integração contínua e a reduzir qualquer viés no procedimento. Como resultado, todos os achados são replicáveis e estão detalhados nesta seção do estudo.

Foi aplicado mapeamento sistemático da literatura, conforme descrito por (KITCHENHAM et al., 2009), neste estudo. A pesquisa foi conduzida em sete fases principais: 1) Planejamento da pesquisa; 2) Pilotagem da busca; 3) Pesquisa de artigos; 4) Inclusão e exclusão de artigos; 5) Avaliação da qualidade dos estudos primários; 6) Análise e agrupamento de estudos primários; e 7) Relato dos resultados da RSL.

4.1 Questões de Pesquisa

As perguntas de pesquisa devem ser elaboradas com o objetivo de coletar informações que possam ser organizadas e avaliadas. No mapeamento sistemático, essas perguntas tendem a ser mais abrangentes, visando reunir os estudos em subáreas ou subtópicos.Usando o método GQM (Goal/Question/Metric) proposto por Basili e Rombach (1988) foi definido o seguinte objetivo do mapeamento:

Analisar *a integração de testes de software na Integração Contínua (CI), com o propósito de melhorar a qualidade e agilidade no desenvolvimento de software, sob o ponto de vista de* ***empresas que adotam CI****, no contexto da* ***indústria de software****.*

O objetivo deste trabalho é explorar metodologias e estudos empíricos sobre a aplicação de testes automatizados na indústria, bem como a influência desses testes desenvolvidos pela equipe de qualidade. A partir deste objetivo, foram derivadas as seguintes questões de pesquisa (QP):

- QP1: Quais são os estudos empíricos relacionados à aplicação ou avaliação de técnicas de teste automatizado, com enfoque em integração contínua, na indústria?

  - *Justificativa:* obter um panorama dos estudos empíricos sobre a aplicação de técnicas de automação de teste baseadas em integração contínua na indústria.

    Esses estudos empíricos fornecem *insights* valiosos sobre a aplicação real, desafios enfrentados e benefícios alcançados com a implementação de técnicas de teste automatizado na integração contínua, contribuindo para a compreensão e aprimoramento dessas práticas na indústria de software.

  - *Métricas:* conjunto de estudos que envolvem a aplicação de técnicas de integração contínua na indústria e os critérios e métricas de teste utilizadas nesses estudos (exs.: cobertura de código, tempo de execução de casos de teste, taxa de falhas).

- QP2: Quais domínios de aplicação de teste automatizado foram aplicados na integração contínua?

  - *Justificativa:* identificar os domínios de aplicação nos quais técnicas de teste automatizado foram utilizadas na integração contínua.

  - *Métricas:* conjunto de domínios de aplicação nos quais técnicas de teste automatizado foram aplicadas na integração contínua na indústria.

- QP3: Quais ferramentas de apoio ao teste automatizado, com ênfase na integração contínua, vêm sendo utilizadas em contexto industrial?

  - *Justificativa:* levantar o conjunto de ferramentas utilizadas no processo de integração contínua na indústria.

  - *Métricas:* conjunto de ferramentas utilizadas na integração contínua na indústria.

- QP4: Quais papéis são responsáveis pelo processo de integração contínua?

  - *Justificativa:* identificar os papéis responsáveis pela integração contínua.

  - *Métricas:* os papéis envolvidos na integração contínua e número de estudos por papel.

- QP5: Quais os problemas recorrentes na integração contínua?

– *Justificativa:* identificar quais são os problemas recorrentes na automação de testes com enfoque na integração contínua.

– *Métricas:* os problemas recorrentes na integração contínua e os domínios de aplicação.

Considerando que técnicas de integração contínua vêm sendo amplamente utilizas pela indústria de software em atividades de verificação e validação, essas técnicas constituem a população investigada neste mapeamento sistemático da literatura. Dessa forma, a *string* de busca foi definida com base em três palavras chave principais: (i) Teste de software, como o termo mais abrangente, para obter o maior número possível de estudos relacionados ao tema pesquisado; (ii) Integração Contínua, para restringir a busca a sistemas de software e evitar estudos que abordem outros tópicos dentro da área de engenharia de software, e (iii) Indústria, para excluir estudos acadêmicos que ainda não foram validados na indústria, que estão fora do escopo deste estudo. Dessa forma, a seguinte *string* de busca foi definida: ***(Software Testing OR Automated Testing) AND (Continuous Integration OR Continuous Delivery) AND Industry***.

A respeito às bases de dados de publicações, foram selecionadas as bases *Science Direct*[3] e *IEEE Xplore*[4]. Essas bases de dados indexam as publicações das mais importantes conferências e periódicos relacionados às técnicas de verificação e validação de software, incluindo integração contínua. A *string* e os procedimentos de busca foram adaptados de acordo com as características de cada base de dados.

Neste estudo foram definidos dois critérios de inclusão (CI) e quatro critérios de exclusão (CE):

- CI1: O estudo propõe uma técnica de integração contínua com validação na indústria;
- CI2: O estudo relata a aplicação de testes de software com técnicas de integração contínua na indústria;
- CE3 : Estudo secundário, mapeamento ou revisão sistemática de literatura;

[3]⟨https://www.sciencedirect.com/⟩
[4]⟨https://ieeexplore.ieee.org/⟩

- CE4: O estudo apresenta aplicação de técnicas de integração contínua em contexto acadêmico;
- CE5: O estudo foi escrito em outro idioma que não o inglês;
- CE6: O estudo não foi revisado por pares;

4.2 Condução

Este mapeamento sistemático da literatura foi conduzida no período de agosto de 2023 a abril de 2024 por dois pesquisadores que possuem experiência acadêmica e profissional na área de Qualidade de software. Com o aumento da interconectividade, do poder computacional dos dispositivos e do número de aplicações *web* e móveis a partir do final dos anos 2000, houve um aumento da preocupação com a Qualidade de software em tempo real (contínua). Por esse motivo, neste estudo foram considerados estudos primários publicados nos últimos quinze anos: de 2009 até 2024.

Na Figura 4.1 são descritas as etapas do processo de seleção. Adaptando a *string* de busca para cada base de dados e considerando a busca

por título, resumo e palavras-chave, foram obtidos um total de 109 estudos, sendo 90 retornados pela Science Direct e 19 pelo IEEE. Após a remoção de duplicatas, o número total de estudos foi reduzido para 105. Após a primeira seleção, na qual os critérios de inclusão e de exclusão foram aplicados a partir da leitura do título, do *abstract* e das palavras-chave, foram selecionados 63 estudos.

Na segunda etapa de seleção, após a leitura completa de cada um desses estudos, foram selecionados 23 estudos primários.

Em seguida, a técnica de inspeção por *snowballing* (WOHLIN, 2014) foi aplicada na análise das referências de cada estudo primário selecionado na etapa anterior, acrescentando 32 novos estudos, resultando em um total de 55 estudos. Como resultado, foram obtidas técnicas de integração contínua com aplicação na indústria e 23 estudos de caso de aplicação de técnicas de IC, listadas no Apêndice A (Tabela 1).

A ferramenta *RAYYAN*[5] foi utilizada para gerenciar cada etapa do processo de

[5]⟨https://rayyan.ai/⟩

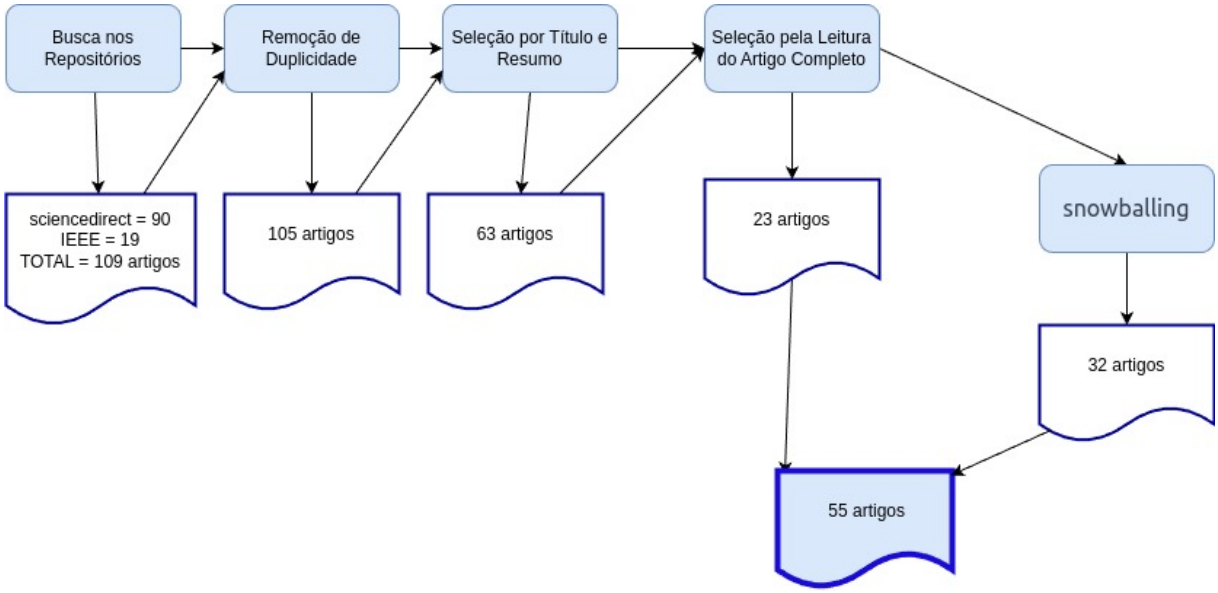


Figura 4.1: Imagem da autora: Processo para seleção de estudos primários relevantes baseado no método de Kitchenham (2004).

mapeamento sistemático: planejamento, condução e sumarização dos resultados. Dois pesquisadores participaram do processo de revisão. A revisão por pares envolveu uma comparação das decisões e um processo de resolução de conflitos, alcançado por meio de discussões sobre o conteúdo de cada artigo e sobre os critérios de inclusão, exclusão e qualidade.

4.2.1 Extração, análise e síntese de dados

Para responder às Questões de Pesquisa (QPs), foi utilizado o formulário de extração de dados, disponível como material complementar no repositório, conforme referenciado em (VITOR, 2024). Esse formulário inclui propriedades que abordam as cinco questões de pesquisa relacionadas a este trabalho: estudos empíricos, domínios de aplicação, ferramentas de apoio ao teste automatizado, os papés envolvidos na integração contínua (IC) e os problemas recorrentes associados à IC.

Inicialmente, foi analisada a qualidade das publicações considerando o fator de impacto e quartil para publicações em periódicos juntamente com o índice QUALIS da CAPES, veículo (exs.: ScienceDirect e IEEE). A avaliação da qualidade das publicações em conferências, simpósios e *workshops* foi realizada com base no índice QUALIS da CAPES e no *h-index*. Essa análise possibilitou uma avaliação da qualidade dos veículos

nos quais os trabalhos selecionados foram publicados. Posteriormente, foi realizado o processo de extração de dados de cada estudo primário, a fim de fornecer respostas a cada questão de pesquisa:

Estudos empíricos: referenciar os conjuntos de estudos que envolvem as aplicações de técnicas de IC na indústria e os critérios de testes utilizados.

Domínio de aplicação: aludir os conjuntos de domínios de aplicação nos quais técnicas de teste automatizado foram aplicadas.

Ferramentas de apoio ao teste automatizado: indicar o conjunto de ferramentas utilizadas na integração contínua na indústria.

Responsáveis pelo processo de integração contínua: mencionar os papéis dos envolvidos na integração contínua.

Problemas recorrentes na integração contínua: referenciar os tipos de problemas associados à integração contínua na indústria.

Em uma etapa posterior, os dados extraídos foram cruzados para garantir a extração correta e minimizar o erro humano. Os dados do conjunto final de estudos incluídos foram armazenados em um banco de dados e gerenciados por meio das bibliotecas da linguagem de programação Python como Plotly[6] e Pandas[7].

Para a análise dos dados, foram empregados métodos qualitativos e de síntese narrativa, conforme recomendado em (OLIVEIRA et al., 2010). Os resultados são descritos no Capítulo 6.

Além disso, para analisar campos do formulário de extração de dados com texto aberto (ex.: domínio de aplicação), utilizou-se a análise temática (HAMEL et al., 2021). A análise temática é um método qualitativo que identifica padrões significativos nos dados, começando pela leitura detalhada e pelas anotações iniciais. O processo envolve a codificação dos dados em categorias relevantes e a formação de temas emergentes, que são refinados e definidos. O método culmina na elaboração de um relatório que destaca os temas com exemplos e discute suas implicações, proporcionando uma análise profunda dos dados qualitativos.

[6] ⟨https://plotly.com/python/⟩
[7] ⟨https://pandas.pydata.org/⟩

4.3 Conclusões do capítulo

Neste capítulo, foi descrito o planejamento do mapeamento sistemático da literatura, que envolveu a definição de um protocolo rigoroso para reduzir possíveis vieses na condução da pesquisa. As questões de pesquisa foram elaboradas para garantir a extração de dados relevantes, abrangendo tópicos como a aplicação de testes automatizados na Integração Contínua (CI), as ferramentas utilizadas, os papéis envolvidos e os problemas recorrentes. O uso de uma abordagem estruturada, como o método GQM, garantiu que os objetivos fossem bem definidos e direcionados. Por fim, a condução do mapeamento seguiu procedimentos bem delineados, desde a busca e seleção de estudos até a análise e síntese dos dados, resultando em informações replicáveis e de alta qualidade.

5 Resultados

Neste capítulo é fornecida uma visão detalhada sobre os teste de software e técnicas de integração contínua aplicadas na indústria e estudos de caso da aplicação dessas técnicas em diferentes domínios, que respondem a cada questão de pesquisa proposta neste mapeamento sistemático de literatura. Na Seção 5.1 são apresentados os estudos empíricos relacionados à aplicação ou avaliação de técnicas de teste automatizado com enfoque na integração contínua na indústria — identificados na literatura como resposta à **QP1**. Na Seção 5.2 são apresentados os domínios de aplicação nos quais técnicas de teste automatizado foram utilizadas na integração contínua, como resposta à questão de pesquisa **QP2**. Na Seção 5.3 são listadas as ferramentas de apoio ao teste automatizado, com ênfase na integração contínua, que têm sido utilizadas na indústria, respondendo a **QP3**. Na Seção 5.4, são descritos os papéis envolvidos no processo de integração contínua (**QP4**). Na, na Seção 5.5, são apresentados problemas recorrentes no processo de integração — identificados com base na análise dos estudos primários como resposta à questão **QP5**. Além disso, na Seção 5.6 são relatadas as Ameaças à Validação. Por fim, na seção XXX são demonstrados as análises com relação a classificação Qualis da CAPES[8].

5.1 Estudos empíricos e as validações aplicadas

Com respeito à ***QP1****: Quais são os estudos empíricos relacionados à aplicação ou avaliação de técnicas de teste automatizado, com enfoque em integração contínua, na indústria?*, na Figura 5.1 é ilustrada a distribuição dos estudos primários identificados. Os estudos foram categorizados por método de validação em: Estudos de Caso, Estudos de Caso Múltiplos, Estudos de Caso Descritivos, Experimento, Estudo/Pesquisa Exploratório, Pesquisa-Ação e Pesquisa (*Survey*).

A tabela 5.1 resume a distribuição dos tipos de estudos primários identificados em uma revisão sistemática da literatura sobre testes automatizados e integração contínua.

[8]Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

Tabela 5.1: Distribuição dos Estudos Primários Identificados

Tipo de Estudo	%	Referências
Estudos de Caso	36,36%	(BARON; LOUIS, 2023; LUZ; PINTO; BONIFÁCIO, 2019; CLAPS; SVENSSON; AURUM, 2015; BRUNELIERE et al., 2022; STURM; POLLARD; CRAIG, 2017; SNEED; VERHOEF, 2019; PRADHAN; NANNIYUR, 2021; RUBERT; FARIAS, 2022; PAULE; D¨ULLMANN; HOORN, 2019; STÅHL; BOSCH, 2013; VIRMANI, 2015; MELLADO et al., 2010; FEITELSON; FRACHTENBERG; BECK, 2013; SHAHIN; BABAR, 2020; RAMLER; PUTSCH¨OGL; WINKLER, 2014; D¨ULLMANN; PAULE; HOORN, 2018; MARIJAN; GOTLIEB; SEN, 2013; MARIJAN; LIAAEN, 2016; HEMON-HILDGEN; ROWE; MONNIER-SENICOURT, 2020)
Experimentos	16,36%	(LANGE et al., 2023; HAGHIGHATKHAH et al., 2018; MARIJAN; LIAAEN; SEN, 2018; BATRA; JATAIN, 2020; PIANINI; NERI, 2021; LACOSTE, 2009; SPIEKER et al., 2017; CHEN, 2015; RODRIGUES et al., 2017)
Pesquisa-Ação	12,73%	(HILTON et al., 2017; WANG et al., 2022; RAHMAN et al., 2015; STOLBERG, 2009; MEMON et al., 2017; ELBAUM; ROTHERMEL; PENIX, 2014; LEITE et al., 2021)
Pesquisa-Ação (outros)	12,73%	(BARON; LOUIS, 2021; T¨UZ¨UN et al., 2019; SIQUEIRA et al., 2018; NEELY; STOLT, 2013; COLLINS; LUCENA, 2012; RAUSCH et al., 2017; V¨OST; WAGNER, 2016)
Surveys	7,27%	(STRADOWSKI; MADEYSKI, 2023; TANZIL et al., 2023; WANG; PYH¨AJ¨ARVI; M¨ANTYL¨A, 2020)
Estudos de Caso Múltiplos	5,45%	(LWAKATARE et al., 2019; M¨AKINEN et al., 2016; OLSSON; ALAHYARI; BOSCH, 2012)
Pesquisa Exploratória	3,64%	(SHAHIN et al., 2017; ERICH; AMRIT; DANEVA, 2017)
Outros (Estudo Exploratório, Estudos de Caso Descritivo, Prova de Conceito)	1,82%	(LEPP¨ANEN et al., 2015; LEPP¨ANEN; KILAMO; MIKKONEN, 2015; SONI, 2015)

Ela apresenta os seguintes pontos principais:

1. Estudos de Caso (36,36%): A maior parte dos estudos primários analisados são estudos de caso, que investigam a aplicação prática de técnicas de testes automatizados e integração contínua em ambientes industriais específicos.

2. Experimentos (16,36%): Uma porção significativa dos estudos são experimentos

controlados, nos quais as técnicas de teste são avaliadas em cenários experimentais para medir sua eficácia.

3. Pesquisa-Ação (12,73%): Esses estudos envolvem a colaboração ativa entre pesquisadores e profissionais da indústria para aplicar e melhorar as técnicas de teste em contextos reais.

4. Pesquisa-Ação (outros) (12,73%): Similar ao anterior, mas com ênfase em diferentes abordagens e contextos dentro da integração contínua.

5. Surveys (7,27%): Pesquisas realizadas com profissionais da área para coletar dados e opiniões sobre o uso de ferramentas e técnicas de testes automatizados na integração contínua.

6. Estudos de Caso Múltiplos (5,45%): Investigações que envolvem múltiplos estudos de caso, comparando a aplicação de técnicas de integração contínua em diferentes contextos.

7. Pesquisa Exploratória (3,64%): Estudos que exploram novas ideias e abordagens para a integração de testes automatizados, sem uma metodologia fixa ou estruturada.

8. Outros (1,82%): Estudos que não se enquadram nas categorias principais, como estudos descritivos e provas de conceito.

   Essa distribuição fornece uma visão geral dos diferentes tipos de metodologias utilizadas na pesquisa sobre testes automatizados em integração contínua, com destaque para a predominância de estudos de caso e experimentos práticos.

Em relação à Figura 5.1, as categorias mais relevantes são discutidas brevemente a seguir. Os artigos classificados como estudos de caso caracterizam-se pela validação de ferramentas desenvolvidas pelos próprios autores. Esses artigos detalham todas as etapas do processo de validação, incluindo os testes realizados, e também a utilização de *softwares* de terceiros, que ajudam a corroborar os resultados obtidos. Já os trabalhos classificados como experimentos têm como objetivo validar a eficiência do método escolhido para a pesquisa, fazendo uso de conjuntos de dados provenientes da indústria. Esses

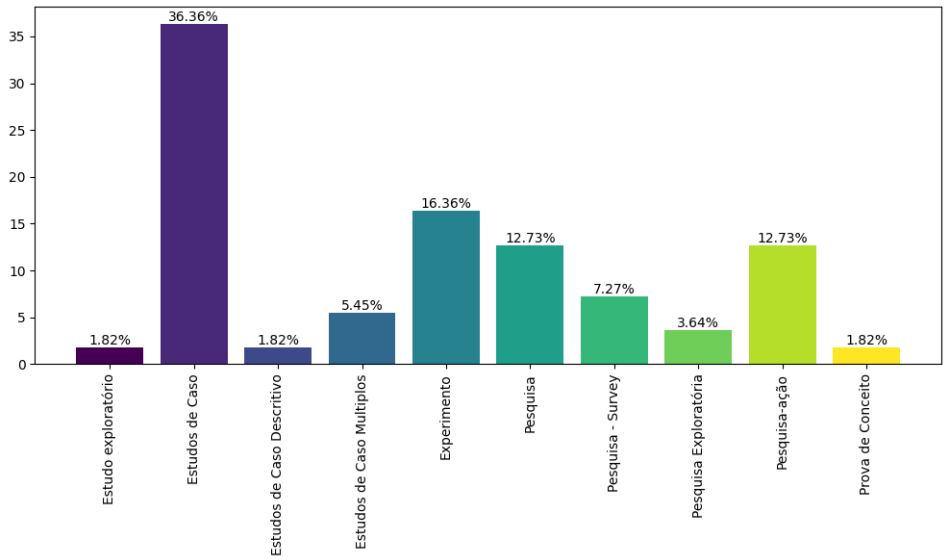


Figura 5.1: Imagem da autora: Número de estudos primários por método de validação.

estudos oferecem informações minuciosas sobre o tempo de execução e sobre a seleção dos parâmetros utilizados.

Os artigos categorizados como Pesquisa distinguem-se pelo uso de entrevistas semiestruturadas e questionários com múltiplas escolhas, que também incluem espaços para textos abertos destinados a informações adicionais. Essas pesquisas têm como objetivo identificar os benefícios, as barreiras e necessidades satisfeitas ou não pelo desenvolvimento ou pela implementação da integração contínua. Por outro lado, nos trabalhos de Pesquisa-ação, os autores coletam relatos dos participantes após a implementação de melhorias ou a criação de sistemas de integração contínua na indústria.

Com base na análise dos estudos primários, foi constatado um conjunto de métricas de teste utilizadas na avaliação de processos e produtos de software listadas na Tabela 5.2. As métricas identificadas abrangem desde "Qualidade, desempenho e Revisão de código"até métricas de análise mais específicas como "nível de cobertura de teste de código"e "tempo de execução de teste". Alguns estudos priorizam métricas que avaliam o desempenho e a eficácia dos testes, como a "porcentagem de falhas detectadas (APFD)"e a "avaliação da cobertura do teste", indicando um foco robusto na melhoria contínua da qualidade do software. Essas métricas são cruciais para compreender a eficiência dos testes automatizados e sua capacidade de detectar regressões e outros problemas potenciais, de forma precoce, no ciclo de desenvolvimento.

Tabela 5.2: Métricas de Teste por Artigo

Artigo	Métricas de Teste
M¨akinen et al. (2016)	Qualidade e desempenho, Revisão de código
Stradowski e Madeyski (2023)	Desempenho. CRT - Teste de regressão contínua. CIT - Teste de integração Contínua. CDRT - Teste de Regressão de entrega contínua
Baron e Louis (2023)	Teste de cobertura de código
Luz, Pinto e Bonifácio (2019)	Análise estática de código-fonte
Haghighatkhah et al. (2018)	Porcentagem de falhas detectadas (APFD), priorização de testes baseada na diversidade (DBTP),a priorização de testes baseada no histórico (HBTP)
Sneed e Verhoef (2019)	e.g.
Baron e Louis (2021)	Nível de rigor de teste
T¨uz¨un et al. (2019)	Análise de tempo de execução
Pianini e Neri (2021)	Análise estática, teste de tempo de execução
Mellado et al. (2010)	Nível de cobertura
Hilton et al. (2017)	Avaliação da cobertura do teste
Marijan e Liaaen (2016)	Tempo de execução de teste
Rouf et al. (2021)	Tempo de execução de teste

5.2 Domínios de Aplicação na Integração Contínua

Com relação à QP2 da seção 4.1, na Figura 5.2 são listados os domínios de aplicação nos quais técnicas de teste automatizado foram aplicadas na integração contínua. O gráfico 5.2 ilustra a frequência de diferentes áreas de aplicação. A área **Tecnologia** tem a maior frequência, com aproximadamente 25 ocorrências. As áreas subsequentes, com frequências menores e decrescentes, incluem **Indústria**, **Telecomunicações**, **Segurança**

de Software, **Automotivo** ,**Serviços**, até chegar a áreas com uma frequência de uma ocorrência, como **Administração Pública**,**Gerenciamento de Serviços de TI**, **Seguros**, **Rede de Comunicação**, **Processamento de Dados**, **Jogos**, **Infraestrutura de Serviços Computacionais**, **Governança**, **Mídia**, **Educação**, **Financeira**, **Geoprocessamento**, **Comunicação**, **Logística**, **Ferroviário**, **Aprendizado de Máquina**, **Agência de Pesquisa** e **Videoconferência**. Uma observação importante é que vários artigos se concentraram no desenvolvimento do software, não deixando explicita qual seria a área de aplicação do software. Além disso, nem todos os trabalhos citam o nome das empresas em que foi aplicada a validação.

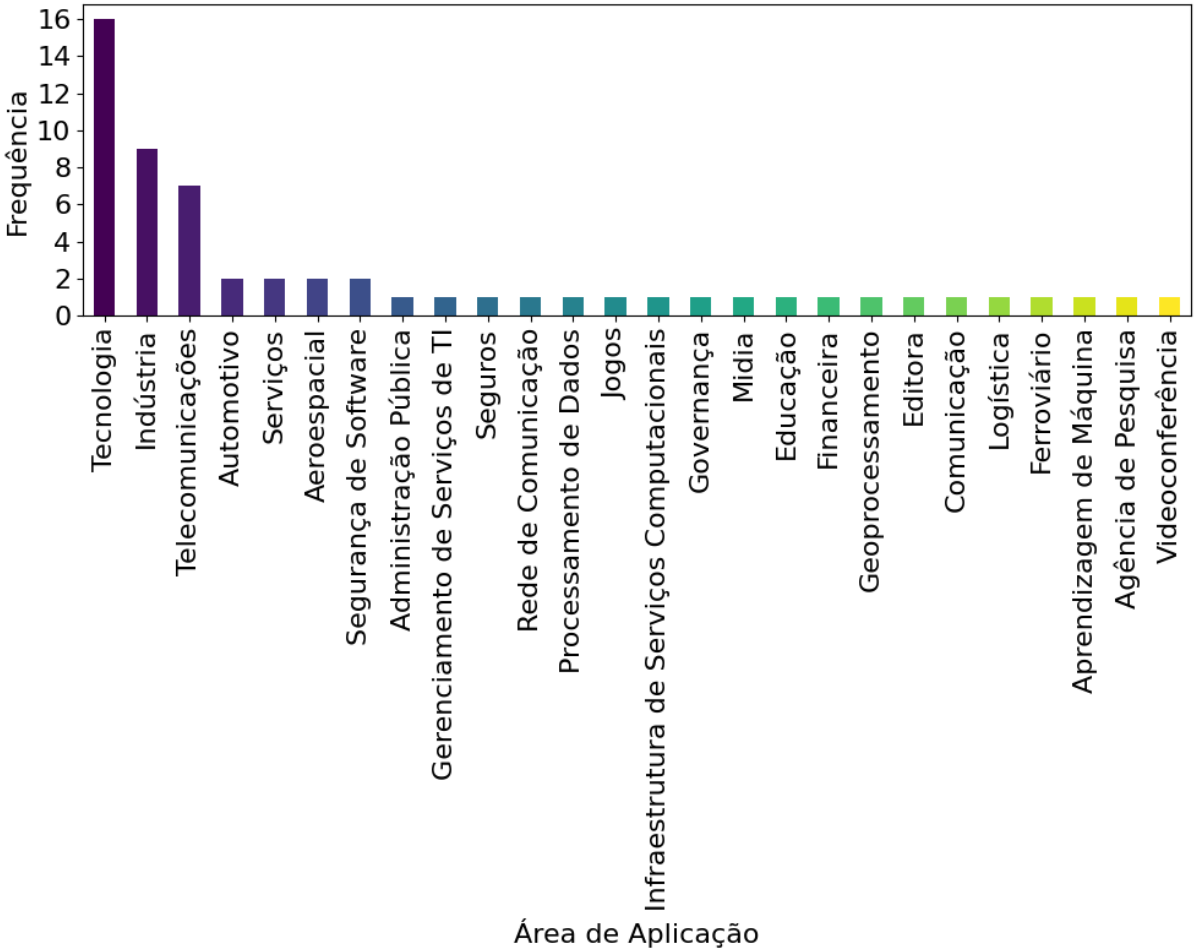


Figura 5.2: Imagem da autora: Distribuição do Domínio de Aplicação

Abaixo, estão os artigos referente ao gráfico 5.2 com suas respectivas áreas de domínio de aplicação, organizando-os de acordo com as categorias mais frequentes. Essa relação ajuda a visualizar como cada artigo contribui para a pesquisa em suas áreas específicas:

Na área de **Tecnologia**, os artigos incluem (CLAPS; SVENSSON; AURUM, 2015), (HAGHIGHATKHAH et al., 2018), (SNEED; VERHOEF, 2019), (WANG et al., 2022), (BATRA; JATAIN, 2020), (VIRMANI, 2015), (RAHMAN et al., 2015), (FEITELSON; FRACHTENBERG; BECK, 2013), (LACOSTE, 2009), (MEMON et al., 2017), (ELBAUM; ROTHERMEL; PENIX, 2014), (ERICH; AMRIT; DANEVA, 2017), (OLSSON; ALAHYARI; BOSCH, 2012), (NEELY; STOLT, 2013), e (RAUSCH et al., 2017), além de (RODRIGUES et al., 2017).

Na área de **Indústria**, destacam-se os artigos (PRADHAN; NANNIYUR, 2021), (BRUNELIERE et al., 2022), (TANZIL et al., 2023), (LWAKATARE et al., 2019), (STURM; POLLARD; CRAIG, 2017), (RAMLER; PUTSCH¨OGL; WINKLER, 2014), (STOLBERG, 2009), e (SHAHIN; BABAR, 2020).

Para a área de **Telecomunicações**, os artigos relevantes incluem (HILTON et al., 2017), (STÅHL; BOSCH, 2013), (COLLINS; LUCENA, 2012), (MARIJAN; GOTLIEB; SEN, 2013), (MARIJAN; LIAAEN, 2016), e (MARIJAN; LIAAEN; SEN, 2018).

Na área de **Serviços**, encontramos os artigos (ROUF et al., 2021) e (HEMON-HILDGEN; ROWE; MONNIER-SENICOURT, 2020).

A área de **Educação** é representada pelo artigo (RUBERT; FARIAS, 2022).

Para **Segurança de Software**, temos os artigos (PAULE; D¨ULLMANN; HOORN, 2019) e (D¨ULLMANN; PAULE; HOORN, 2018).

Na área de **Comunicação**, destaca-se o artigo (LEPP¨ANEN et al., 2015).

A área de **Finanças** é representada pelo artigo (LUZ; PINTO; BONIFÁCIO, 2019).

Na **Aeroespacial**, os artigos relevantes são (BARON; LOUIS, 2023) e (BARON; LOUIS, 2021).

Na área de **Aprendizagem de Máquina**, temos o artigo (SPIEKER et al., 2017).

Para a área **Automotivo**, destacam-se os artigos (V¨OST; WAGNER, 2016) e (PONSARD; RAMON, 2022).

Na área de **Geoprocessamento**, temos o artigo (MELLADO et al., 2010).

Na área de **Gerenciamento de Serviços de TI**, o artigo relevante é (LEPP¨ANEN;

KILAMO; MIKKONEN, 2015).

Para a área de **Governança**, temos o artigo (T¨UZ¨UN et al., 2019).

Na área de **Jogos**, destaca-se o artigo (CHEN, 2015).

Para a área de **Rede de Comunicação**, o artigo relevante é (STRADOWSKI; MADEYSKI, 2023).

Por fim, na área de **Seguros**, encontramos o artigo (SONI, 2015).

Essa relação ajuda a entender como as diferentes áreas estão sendo abordadas na pesquisa e qual é a relevância dos artigos em cada contexto. A diversidade de temas reflete a complexidade do campo de testes de software e a necessidade de metodologias adaptativas para cada domínio.

Embora o Gráfico 5.2 e a Tabela 1 mostrem uma prevalência de testes automatizados em integração contínua na área de *Tecnologia*, outros domínios também podem explorar essas técnicas, mas em menor escala. Esses outros domínios se incluem, mas não estão visíveis ou são minimamente representados no gráfico, o que pode indicar áreas de crescimento potencial ou sub-representação na amostra de dados coletada. A análise sugere que Fábrica de software continua a ser um campo líder na adoção de práticas avançadas de teste, refletindo a necessidade contínua de desenvolvimento ágil e eficiente de software. Todavia, não é possível em todos os artigos identificar de forma clara o domínio da aplicação, pois muitos dos artigos são relatos de desenvolvimento de software e as aplicações da integração contínua na indústria, com o foco no processo de desenvolvimento, em testes e na manutenção de software.

Nos artigos citados (SHAHIN et al., 2017; PRADHAN; NANNIYUR, 2021; SPIEKER et al., 2017; ROUF et al., 2021; LEPP¨ANEN et al., 2015; CLAPS; SVENSSON; AURUM, 2015; STURM; POLLARD; CRAIG, 2017; ERICH; AMRIT; DANEVA, 2017; OLSSON; ALAHYARI; BOSCH, 2012; LANGE et al., 2023, 2023; WANG; PYH¨AJ¨ARVI; M¨ANTYL¨A, 2020), a pesquisa não deixou claras quais técnicas de testes foram de fato aplicadas. Devido a essa falta de evidência sobre o uso efetivo de técnicas de teste, esses artigos foram excluídos das análises realizadas sobre o domínio de aplicação *versus* técnicas e critérios de teste.

5.2.1 Teste de Unidade

Com base nos dados da Figura 5.3 (teste de unidade), O gráfico ilustra a distribuição da quantidade de testes unitários aplicados em diversas áreas de domínio, destacando a **Tecnologia** como a mais representativa, com **34.4%** dos testes, refletindo a ênfase na qualidade e funcionalidade em software e aplicativos. A **Telecomunicações** também se destaca com **9.4%**, evidenciando a necessidade de garantir a eficiência dos serviços de comunicação. Outras áreas como **Indústria**, **Serviços** e **Segurança de Software** apresentam **3.1%** cada, indicando uma adoção crescente, embora menor, de testes unitários. Setores como **Educação, Finanças, Geoprocessamento** e **Infraestrutura de Serviços Computacionais** possuem representações semelhantes, sugerindo que a implementação de testes ainda é uma prática em desenvolvimento nessas áreas. Por outro lado, a categoria **Automotivo, Ferroviário, Logística e Telecomunicações**, com **6.2%**, indica que setores multifacetados estão reconhecendo a importância dos testes unitários. Essa análise ressalta a centralidade da Tecnologia e aponta o desafio para áreas com menor adoção em integrar eficazmente metodologias de teste para garantir a qualidade das soluções oferecidas.

Na área de **Administração Pública**, o artigo (SIQUEIRA et al., 2018), intitulado *Continuous delivery: Building trust in a large-scale, complex government organization*, destaca a aplicação de testes de unidade, evidenciando sua importância na garantia de qualidade em ambientes governamentais complexos.

No domínio **Aeroespacial**, os artigos (BARON; LOUIS, 2023) e (BARON; LOUIS, 2021) abordam a aplicação de testes de unidade em sistemas críticos, enfatizando sua relevância para a certificação de software embarcado. Esses estudos demonstram que a implementação de testes de unidade é fundamental para assegurar a segurança e a confiabilidade dos sistemas utilizados na aviação.

A área **Indústria** é representada por (BRUNELIERE et al., 2022), que discute a aplicação de testes de unidade em um framework para desenvolvimento contínuo, e por (TANZIL et al., 2023), que analisa os desafios enfrentados na adoção de práticas de DevOps, onde os testes de unidade são destacados como uma técnica crucial para a manutenção da qualidade do software. O artigo (LWAKATARE et al., 2019) também enfatiza

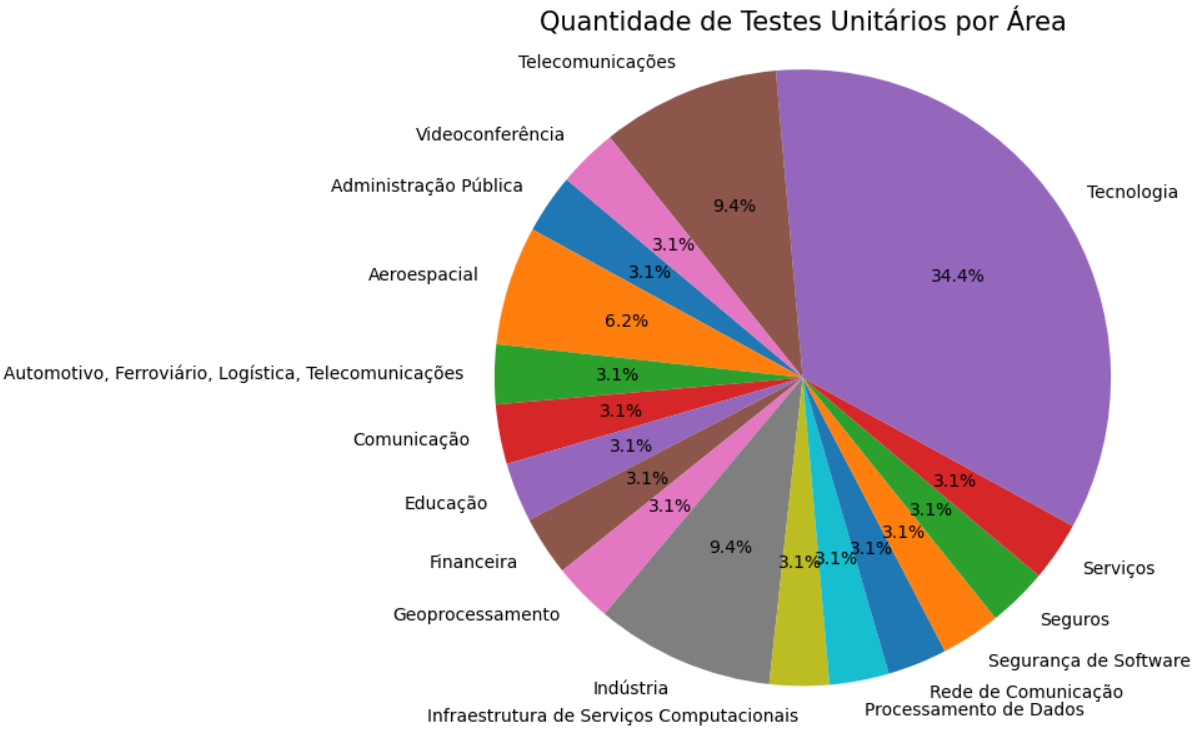


Figura 5.3: Imagem da autora: Distribuição do Domínio de Aplicação para teste de unidade

a relevância dos testes de unidade em práticas de DevOps em empresas industriais.

Na área de **Automotivo, Ferroviário, Logística e Telecomunicações**, o artigo (PONSARD; RAMON, 2022) menciona a utilização de testes de unidade como parte das práticas de automação em desenvolvimento orientado a modelos, o que ressalta sua importância em setores que exigem alta confiabilidade.

Na **Tecnologia**, a diversidade de artigos que aplicam testes de unidade é notável. Por exemplo, (M¨AKINEN et al., 2016) discute a integração de testes de unidade no ciclo de entrega em empresas de software intensivo, enquanto (SNEED; VERHOEF, 2019) reflete sobre a reimplementação de sistemas legados, onde os testes de unidade são essenciais para validar a funcionalidade. O artigo (WANG et al., 2022) investiga como a maturidade na automação de testes, incluindo testes de unidade, melhora a qualidade do produto, enquanto (RUBERT; FARIAS, 2022) analisa o impacto da entrega contínua na qualidade do código.

Na área de **Segurança de Software**, o artigo (BATRA; JATAIN, 2020) discute a utilização de testes de unidade como parte da avaliação de desempenho em práticas de DevOps, evidenciando sua contribuição para pipelines de entrega contínua confiáveis.

A **Infraestrutura de Serviços Computacionais** é abordada pelo artigo (RAMLER; PUTSCH¨OGL; WINKLER, 2014), que apresenta experiências práticas de testes de unidade em software de automação industrial, destacando a importância de garantir a qualidade do software em ambientes críticos.

Na área de **Telecomunicações**, o artigo (COLLINS; LUCENA, 2012) relata práticas de automação de testes em ambientes ágeis, com ênfase na execução de testes de unidade para assegurar a funcionalidade e a qualidade do software desenvolvido.

Por fim, o artigo (MARIJAN; LIAAEN; SEN, 2018) discute melhorias no DevOps com foco na redução de ciclos de desenvolvimento, ressaltando a importância dos testes de unidade como parte de um processo de integração contínua eficiente.

Esta lista inclui as áreas de aplicação, as técnicas utilizadas e a confirmação de que os testes unitários foram aplicados em cada artigo.

5.2.2 Teste A/B

Em relação ao Teste A/B, Tecnologia foi a única área dominante, sendo quatro os trabalhos que fizeram seu uso (RAHMAN et al., 2015; FEITELSON; FRACHTENBERG; BECK, 2013; NEELY; STOLT, 2013; D¨ULLMANN; PAULE; HOORN, 2018). A Engenharia de software utiliza uma variedade ampla de testes automatizados integrados para garantir a qualidade e eficiência no desenvolvimento de software.

5.2.3 Teste de Integração

A prática de integração contínua (CI) tem se tornado um pilar fundamental no desenvolvimento de software, permitindo que equipes entreguem produtos de alta qualidade de forma ágil e eficiente. Esta abordagem é especialmente pertinente em setores críticos, como o aeroespacial, automotivo, industrial, de serviços, tecnologia, telecomunicações e videoconferência. Cada um desses domínios apresenta desafios únicos que a integração contínua pode mitigar, promovendo uma melhoria significativa nos processos de desenvolvimento.Abaixo uma análise detalhada do Gráfico 5.4.

No setor **Aeroespacial**, o trabalho de Baron e Louis (BARON; LOUIS, 2023) destaca a necessidade de um framework robusto para a certificação ágil de software em-

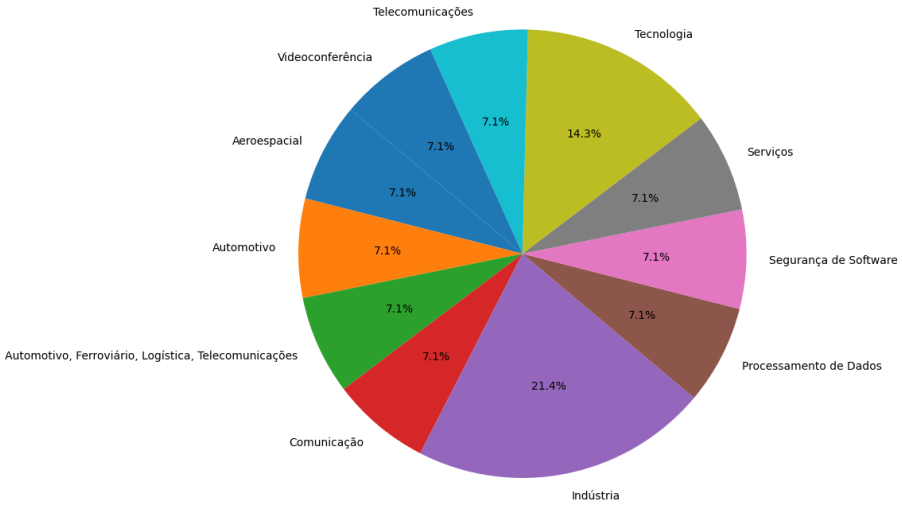


Figura 5.4: Imagem da autora: Distribuição do Domínio de Aplicação para Testes de Integração

barcado em sistemas críticos. A segurança é a principal preocupação neste domínio, e a integração contínua proporciona um mecanismo para verificar constantemente a funcionalidade e a segurança dos sistemas, garantindo que cada nova atualização não comprometa a integridade do software.

Na área **Automotiva**, Vöst e Wagner (V¨OST; WAGNER, 2016) discutem a transição para práticas de integração contínua e entrega contínua. Este movimento é vital para a indústria automotiva, que está se adaptando rapidamente às novas tecnologias e requisitos de mercado. Além disso, o estudo de Ponsard e Ramon (PONSARD; RAMON, 2022) explora as práticas de automação em desenvolvimento orientado a modelos, ressaltando como a integração contínua se integra perfeitamente a essas práticas, proporcionando um ciclo de desenvolvimento mais eficiente.

No contexto da **Indústria**, a pesquisa de Tanzil et al. (TANZIL et al., 2023) apresenta um estudo sobre os desafios do DevOps, enfatizando que a integração contínua é uma prática essencial para enfrentar as complexidades do desenvolvimento moderno. Da mesma forma, Tüzün et al. (T¨UZ¨UN et al., 2019) analisam a adoção da gestão integrada do ciclo de vida de aplicações em uma grande empresa de software, mostrando como a CI pode ser um facilitador em ambientes complexos. Lwakatare et al. (LWAKATARE et al., 2019) também destacam a importância da CI em suas práticas de DevOps, evidenciando que a integração contínua permite a detecção precoce de falhas e a ma-

nutenção da qualidade do software. Marijan et al. (MARIJAN; LIAAEN; SEN, 2018) complementam essa visão, apresentando melhorias no DevOps que reduzem os tempos de ciclo com otimizações de teste integradas para a integração contínua.

O setor de **Serviços** é igualmente impactado pela integração contínua. Hilton et al. (HILTON et al., 2017) discutem os trade-offs entre segurança, flexibilidade e garantia em práticas de CI. Este equilíbrio é crucial para garantir que as soluções atendam às necessidades dos clientes sem comprometer a segurança.

Na área de **Tecnologia**, a integração contínua é essencial para a evolução das práticas de desenvolvimento. Batra e Jatain (BATRA; JATAIN, 2020) realizam uma avaliação de desempenho das práticas de DevOps, destacando como a CI contribui para a eficiência e eficácia dos processos de desenvolvimento. Elbaum et al. (ELBAUM; ROTHERMEL; PENIX, 2014) exploram técnicas para melhorar os testes de regressão em ambientes de CI, reafirmando que a qualidade do software é uma prioridade que deve ser mantida em um ciclo de desenvolvimento ágil. Rahman et al. (RAHMAN et al., 2015) também enfatizam a importância da CI ao sintetizar práticas de implantação contínua, demonstrando como a integração de testes pode melhorar a confiança na entrega de software.

Na área de **Telecomunicações**, Collins e De Lucena (COLLINS; LUCENA, 2012) relatam as práticas de automação de testes em ambientes ágeis, mostrando que a CI permite a rápida detecção e correção de falhas, aumentando a qualidade do software entregue.

Por fim, na **Videoconferência**, Marijan et al. (MARIJAN; LIAAEN; SEN, 2018) discutem como as melhorias no DevOps, com foco na integração contínua, são essenciais para reduzir os ciclos de desenvolvimento, permitindo uma entrega mais ágil e eficiente de soluções.

Portanto, os testes de integração emergem como uma prática essencial nas indústrias representadas no gráfico é na tabela 1. Sua implementação não apenas assegura a qualidade e a confiabilidade do software, mas também otimiza o processo de desenvolvimento, reduzindo o tempo e os custos associados a falhas em produção. À medida que as organizações enfrentam a crescente complexidade de seus sistemas, a adoção de testes de

integração se torna não apenas uma boa prática, mas uma necessidade estratégica para o sucesso em um mercado competitivo.

5.2.4 Teste de UI, E2E e Sistema Funcional

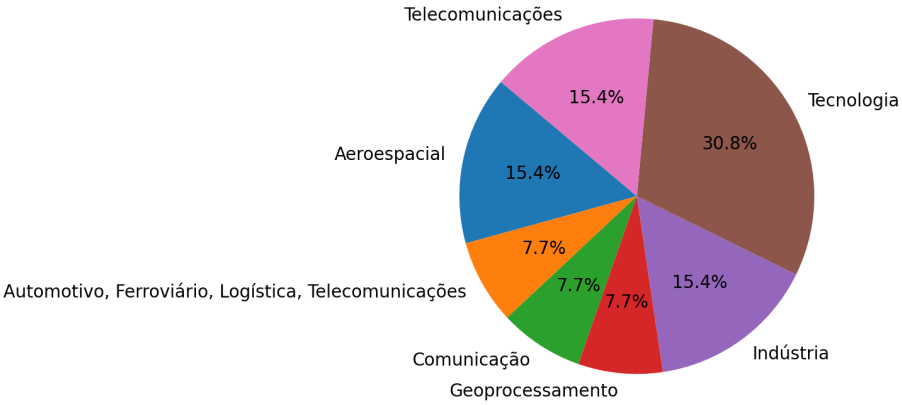


Figura 5.5: Imagem da autora: Distribuição do Domínio de Aplicação para Testes de UI/E2E/Sistema/Funcional

A crescente complexidade dos sistemas de software atuais exige práticas de teste robustas e eficazes. Os testes de UI (Interface do Usuário), E2E (*End-to-End*), Sistema e Funcional são fundamentais para garantir que as aplicações atendam às expectativas dos usuários e funcionem corretamente em todos os cenários de uso. Diversos artigos em diferentes domínios refletem a adoção dessas práticas, destacando sua importância para a qualidade do software.Abaixo uma análise detalhada do Gráfico 5.5.

No setor **Aeroespacial**, representando **15.4%**, os trabalhos de Baron e Louis (BARON; LOUIS, 2023; BARON; LOUIS, 2021) abordam a necessidade de certificação ágil de software embarcado em sistemas críticos. A segurança é uma preocupação primordial neste domínio, e a implementação de testes de UI e E2E é essencial para assegurar que o software funcione conforme esperado em condições reais de operação. A certificação contínua desses sistemas não apenas melhora a segurança, mas também acelera o processo de desenvolvimento.

Na área **Automotiva, Ferroviário, Logística e Telecomunicações**, representando **7.7%**, o estudo de Ponsard e Ramon (PONSARD; RAMON, 2022) analisa práticas de automação em desenvolvimento orientado a modelos, ressaltando como os testes de UI

e E2E são cruciais para a entrega de software de qualidade. Esses testes garantem que as diferentes partes do sistema funcionem em conjunto, algo vital em um setor que está em constante evolução e adaptação.

No contexto da **Indústria**, com **15.4%**, o artigo de Tanzil et al. (TANZIL et al., 2023) apresenta um estudo sobre os desafios do DevOps e enfatiza que a integração contínua e os testes de UI e E2E são essenciais para manter a qualidade em ambientes dinâmicos. Além disso, o trabalho de Lacoste (LACOSTE, 2009) mostra como a introdução de sistemas de integração contínua pode transformar a forma como as empresas desenvolvem e testam seus produtos.

A **Comunicação**, representando **7.7%**, também se beneficia dessas práticas. O artigo de Pianini e Neri (PIANINI; NERI, 2021) discute como a adoção de microserviços e DevOps pode facilitar a implementação de testes de UI e E2E, resultando em uma maior eficiência e flexibilidade nos processos de desenvolvimento.

Na área de **Tecnologia**, que ocupa **30.8%**, diversos estudos, como o de Lwakatare et al. (LWAKATARE et al., 2019), Mäkinen et al. (M¨AKINEN et al., 2016), Sneed e Verhoef (SNEED; VERHOEF, 2019), e Rahman et al. (RAHMAN et al., 2015), mostram como a implementação de testes de UI, E2E e Sistema é fundamental para garantir a qualidade em ambientes de desenvolvimento ágil e DevOps. Esses testes ajudam a detectar falhas precoces, permitindo que as equipes respondam rapidamente a problemas antes que eles afetem os usuários finais.

Na área de **Telecomunicações**, representando **15.4%**, as práticas de automação de testes em ambientes ágeis, conforme descrito por Collins e De Lucena (COLLINS; LUCENA, 2012), são essenciais para garantir que o software atenda às necessidades dos clientes e mantenha um padrão elevado de qualidade.

Por fim, no campo do **Geoprocessamento**, que ocupa **7.7%**, Rodrigues et al. (RODRIGUES et al., 2017) discutem a adoção do DevOps em uma Junior Enterprise, destacando a importância dos testes de UI e E2E para assegurar a funcionalidade das soluções desenvolvidas.

Em conclusão, a adoção de testes de UI, E2E, Sistema e Funcional é vital para o sucesso no desenvolvimento de software em diversas indústrias. Esses testes não apenas

garantem a qualidade e segurança das aplicações, mas também promovem uma cultura de melhoria contínua e agilidade nas organizações. À medida que a tecnologia avança, a implementação dessas práticas se torna cada vez mais imprescindível.

5.2.5 Teste de Regressão

A crescente complexidade dos sistemas de software atuais exige práticas de teste robustas e eficazes. Os testes de regressão são fundamentais para garantir que as aplicações continuem a funcionar corretamente após alterações no código, como correções de bugs ou implementações de novas funcionalidades. Diversos artigos em diferentes domínios refletem a adoção dessas práticas, destacando sua importância para a qualidade do software. Abaixo uma análise detalhada do Gráfico 5.6.

No setor **Serviços**, representando **15.4%**, o trabalho de Hilton et al. (HILTON et al., 2017) discute os trade-offs em integração contínua, incluindo a segurança e a flexibilidade, evidenciando como os testes de regressão são essenciais para garantir que essas práticas não comprometam a qualidade do software.

Na **Indústria**, com **7.7%**, o estudo de Haghighatkhah et al. (HAGHIGHATKHAH et al., 2018) analisa a priorização de testes em ambientes de integração contínua, ressaltando a importância dos testes de regressão para manter a qualidade em ambientes dinâmicos. Sneed e Verhoef (SNEED; VERHOEF, 2019) também destacam a importância desses testes ao reimplementar sistemas legados, garantindo que novas implementações não introduzam regressões.

Na área de **Tecnologia**, que ocupa **38.5%**, a relevância dos testes de regressão é evidenciada em diversos estudos, como o de Virmani (VIRMANI, 2015), que explora a transição da integração contínua para a entrega contínua. Além disso, o trabalho de Elbaum et al. (ELBAUM; ROTHERMEL; PENIX, 2014) foca em técnicas para melhorar os testes de regressão em ambientes de desenvolvimento ágil, reafirmando que a qualidade do software é uma prioridade a ser mantida. O artigo de Chen (CHEN, 2015) também destaca os desafios e benefícios da entrega contínua, sublinhando a importância de testes de regressão eficazes.

No contexto de **Telecomunicações**, que representa **30.8%**, a automação de

testes, conforme descrito por Collins e De Lucena (COLLINS; LUCENA, 2012), é essencial para assegurar que o software atenda às necessidades dos clientes. Os testes de regressão, em particular, permitem que as equipes detectem rapidamente problemas que possam surgir após atualizações.

Por fim, na área de **Processamento de Dados**, representando **7.7%**, os testes de regressão são fundamentais para garantir que as alterações no sistema não afetem negativamente a integridade dos dados e a funcionalidade do software.

Em conclusão, a adoção de testes de regressão é vital para o sucesso no desenvolvimento de software em diversas indústrias. Esses testes não apenas garantem a qualidade e segurança das aplicações, mas também promovem uma cultura de melhoria contínua e agilidade nas organizações. À medida que a tecnologia avança, a implementação dessas práticas se torna cada vez mais imprescindível.

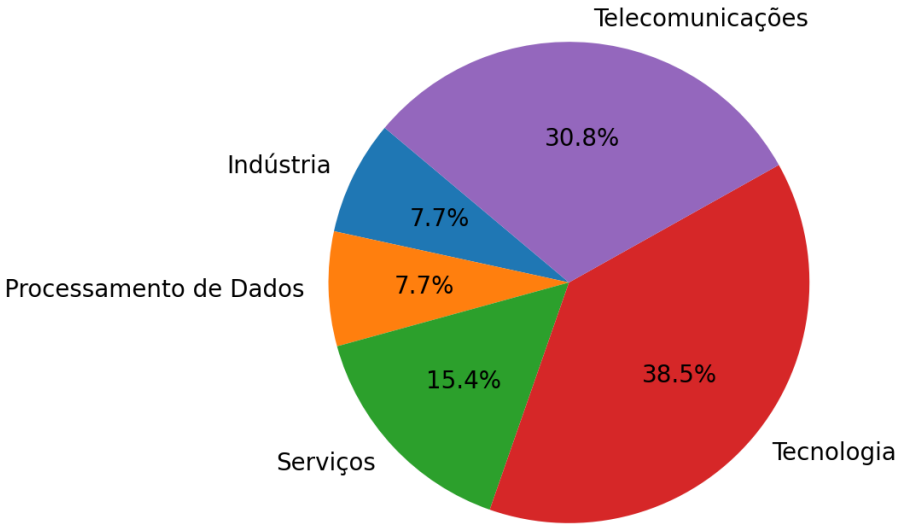


Figura 5.6: Imagem da autora: Distribuição do Domínio de Aplicação para Testes de Regressão

5.2.6 Teste de Aceitação

A crescente complexidade dos sistemas de software atuais exige práticas de teste robustas e eficazes. Os testes de aceitação são fundamentais para garantir que as aplicações atendam às expectativas dos usuários e funcionem corretamente em todos os cenários de uso. Diversos artigos em diferentes domínios refletem a adoção dessas práticas, destacando

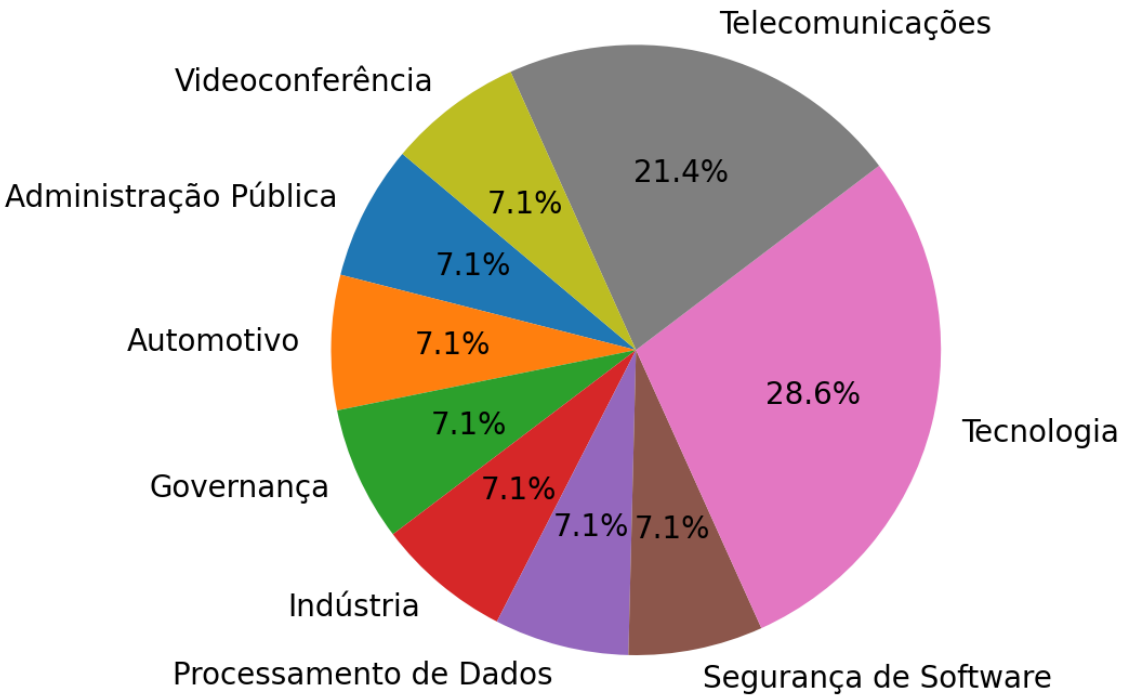


Figura 5.7: Imagem da autora: Distribuição do Domínio de Aplicação para Testes de Aceitação

sua importância para a qualidade do software. Abaixo será apresentado uma análise sobre a Figura 5.7.

No setor **Administração Pública**, representando **7.1%**, o artigo de Siqueira et al. (SIQUEIRA et al., 2018) discute a entrega contínua e a construção de confiança em grandes organizações governamentais. A implementação de testes de aceitação é crucial nesse contexto para assegurar que os sistemas atendam às necessidades da população e funcionem de maneira confiável.

Na área **Automotiva**, que também ocupa **7.1%**, o trabalho de Vöst e Wagner (V¨OST; WAGNER, 2016) explora a transição para práticas de integração contínua e entrega contínua. Os testes de aceitação garantem que as novas funcionalidades estejam em conformidade com as especificações e expectativas do usuário, essencial para a competitividade neste setor.

No contexto da **Indústria**, que representa **7.1%**, o artigo de Tanzil et al. (TANZIL et al., 2023) apresenta um estudo sobre os desafios do DevOps, enfatizando que os testes de aceitação são uma prática essencial para garantir a qualidade em ambientes dinâmicos.

Na área de **Telecomunicações**, que representa **21.4%**, Lwakatare et al. (LWAKATARE et al., 2019) abordam a importância dos testes de aceitação como parte das práticas de DevOps, assegurando que os sistemas desenvolvidos atendam aos requisitos de desempenho e qualidade esperados pelos clientes.

Na área de **Tecnologia**, que ocupa **28.6%**, diversos estudos, como o de Mäkinen et al. (M¨AKINEN et al., 2016) e Rubert e Farias (RUBERT; FARIAS, 2022), mostram como a implementação de testes de aceitação é fundamental para garantir a qualidade em ambientes de desenvolvimento ágil. Esses testes ajudam a validar a funcionalidade e a qualidade do software antes de sua entrega final.

Por fim, na área de **Segurança de Software**, representando **7.1%**, o trabalho de Batra e Jatain (BATRA; JATAIN, 2020) destaca a importância dos testes de aceitação na avaliação de desempenho e segurança de pipelines de entrega contínua.

Em conclusão, a adoção de testes de aceitação é vital para o sucesso no desenvolvimento de software em diversas indústrias. Esses testes não apenas garantem a qualidade e segurança das aplicações, mas também promovem uma cultura de melhoria contínua e agilidade nas organizações. À medida que a tecnologia avança, a implementação dessas práticas se torna cada vez mais imprescindível.

5.2.7 Teste de Cobertura de Código

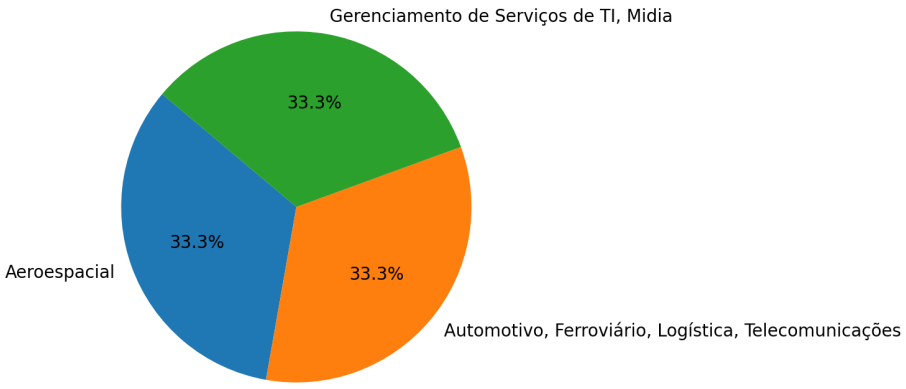


Figura 5.8: Imagem da autora: Distribuição do Domínio de Aplicação para Teste de Cobertura de Código

A crescente complexidade dos sistemas de software atuais exige práticas de teste robustas e eficazes. Os testes de cobertura de código são fundamentais para garantir

que as aplicações sejam testadas de maneira abrangente, assegurando que todas as partes do código sejam verificadas para falhas e inconsistências. Diversos artigos em diferentes domínios refletem a adoção dessas práticas, destacando sua importância para a qualidade do software. Abaixo será apresentando com maiores detalhes as informações da Figura 5.8.

No setor **Aeroespacial**, que representa **33.3%**, os trabalhos de Baron e Louis (BARON; LOUIS, 2023) abordam a necessidade de certificação ágil de software embarcado em sistemas críticos. A implementação de testes de cobertura de código é essencial para garantir que todos os aspectos do software sejam testados, assegurando sua segurança e funcionalidade em condições reais de operação.

Na área **Automotiva, Ferroviário, Logística e Telecomunicações**, também representando **33.3%**, o estudo de Ponsard e Ramon (PONSARD; RAMON, 2022) analisa práticas de automação em desenvolvimento orientado a modelos. Os testes de cobertura de código são cruciais nesse contexto, pois ajudam a identificar partes não testadas do software, garantindo que ele funcione corretamente em todos os cenários.

Na área de **Gerenciamento de Serviços de TI e Mídia**, que representa **33.3%**, o artigo de Luz et al. (LUZ; PINTO; BONIFÁCIO, 2019) discute a adoção do DevOps e a importância dos testes de cobertura de código para assegurar que os serviços atendam aos padrões de qualidade exigidos.

Em conclusão, a adoção de testes de cobertura de código é vital para o sucesso no desenvolvimento de software em diversas indústrias. Esses testes não apenas garantem a qualidade e segurança das aplicações, mas também promovem uma cultura de melhoria contínua e agilidade nas organizações. À medida que a tecnologia avança, a implementação dessas práticas se torna cada vez mais imprescindível.

5.2.8 Teste de Performance

crescente complexidade dos sistemas de software atuais exige práticas de teste robustas e eficazes. Os testes de performance são fundamentais para garantir que as aplicações suportem a carga esperada e operem eficientemente sob diferentes condições. Diversos artigos em diferentes domínios refletem a adoção dessas práticas, destacando sua

importância para a qualidade do software.

Na Figura 5.9, apresentamos a distribuição percentual dos artigos por área que aplicam testes de performance. Essa distribuição evidencia como a prática é amplamente adotada em diferentes setores.

No setor **Indústria**, representando **20.0%**, o trabalho de Tanzil et al. (TANZIL et al., 2023) apresenta um estudo sobre os desafios do DevOps, enfatizando que os testes de performance são essenciais para manter a qualidade em ambientes dinâmicos, onde a eficiência e a rapidez são cruciais.

Na área de **Processamento de Dados**, que também ocupa **20.0%**, o artigo de Chen (CHEN, 2015) discute os benefícios e desafios da entrega contínua, sublinhando a importância dos testes de performance para garantir que as soluções implementadas funcionem de maneira eficaz.

Na área de **Telecomunicações**, que representa **40.0%**, Stradowski e Madeyski (STRADOWSKI; MADEYSKI, 2023) exploram os desafios nos testes de software do sistema 5G, destacando como os testes de performance são cruciais para validar a funcionalidade e a eficiência do sistema em um cenário em constante evolução. Além disso, o trabalho de Collins e De Lucena (COLLINS; LUCENA, 2012) relata práticas de automação de testes em ambientes ágeis, ressaltando a necessidade de testes de performance para atender às exigências do mercado.

Por fim, na área de **Videoconferência**, que representa **20.0%**, Marijan et al. (MARIJAN; LIAAEN; SEN, 2018) discutem como as melhorias no DevOps, com foco na integração contínua, são essenciais para otimizar os testes de performance e reduzir os ciclos de desenvolvimento.

Em conclusão, a adoção de testes de performance é vital para o sucesso no desenvolvimento de software em diversas indústrias. Esses testes não apenas garantem a qualidade e segurança das aplicações, mas também promovem uma cultura de melhoria contínua e agilidade nas organizações. À medida que a tecnologia avança, a implementação dessas práticas se torna cada vez mais imprescindível.

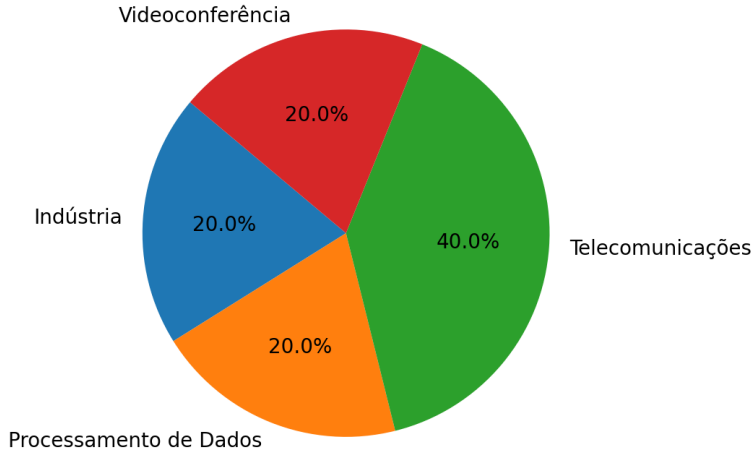


Figura 5.9: Imagem da autora: Distribuição do Domínio de Aplicação para Teste de Performance

5.2.9 Demais Testes

A crescente complexidade dos sistemas de software atuais exige práticas de teste robustas e eficazes. Os "demais testes", que englobam uma variedade de abordagens e metodologias, são fundamentais para garantir a qualidade e a confiabilidade das aplicações em diferentes contextos. Diversos artigos em diferentes domínios refletem a adoção dessas práticas, destacando sua importância para a qualidade do software.

Na Figura 5.10, apresentamos a distribuição percentual dos artigos por área que aplicam "demais testes". Essa distribuição evidencia como a prática é amplamente adotada em diferentes setores.

No setor **Aeroespacial**, que representa **9.1%**, o artigo de Baron e Louis (BARON; LOUIS, 2021) discute a certificação contínua de software crítico para a aviação, sublinhando a importância de testes baseados em riscos para garantir a segurança dos sistemas.

Na área **Indústria**, também com **9.1%**, o trabalho de Bruneliere et al. (BRUNELIERE et al., 2022) apresenta um framework para o desenvolvimento contínuo, enfatizando a necessidade de testes baseados em modelos para assegurar a qualidade das soluções desenvolvidas.

Além disso, na mesma área, Tanzil et al. (TANZIL et al., 2023) discutem a aplicação de testes não funcionais, que são cruciais para validar a usabilidade e a segurança das aplicações em ambientes de alta complexidade.

Na área de **Tecnologia**, que representa **40.0%**, vários artigos, como o de Rubert e Farias (RUBERT; FARIAS, 2022), exploram práticas de testes exploratórios e de revisão de código, que são essenciais para garantir a qualidade do software em ciclos de desenvolvimento ágeis.

Na área de **Rede de Comunicação**, que ocupa **9.1%**, o estudo de Leite et al. (LEITE et al., 2021) discute a aplicação do TDD (Desenvolvimento Orientado a Testes) e sua relevância para a entrega contínua e eficiente de software.

Finalmente, na área de **Telecomunicações**, que representa **9.1%**, o trabalho de Stradowski e Madeyski (STRADOWSKI; MADEYSKI, 2023) analisa os desafios nos testes de competência em sistemas 5G, ressaltando a importância de métodos de teste de segurança e usabilidade.

Em conclusão, a adoção de "demais testes"é vital para o sucesso no desenvolvimento de software em diversas indústrias. Esses testes não apenas garantem a qualidade e segurança das aplicações, mas também promovem uma cultura de melhoria contínua e agilidade nas organizações. À medida que a tecnologia avança, a implementação dessas práticas se torna cada vez mais imprescindível.

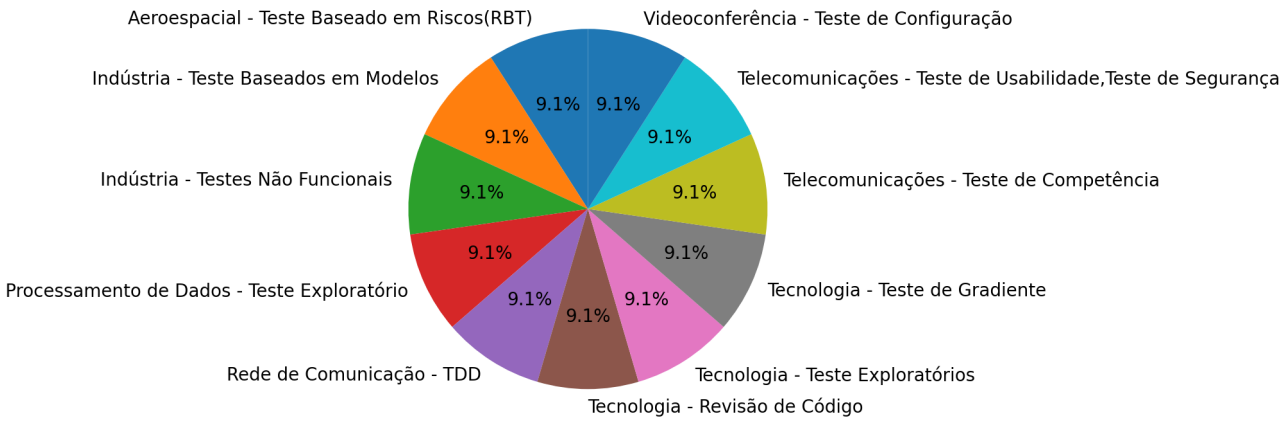


Figura 5.10: Imagem da autora: Distribuição do Domínio de Aplicação para Demais Testes identificados

A identificação dos domínios em que os testes automatizados são aplicados na integração contínua ajuda as organizações a entender onde focar esforços de desenvolvimento e garantia de qualidade. Isso também indica áreas com maior adoção de práticas avançadas de CI/CD (Integração Contínua e Entrega Contínua).

Essas percepções fornecem uma visão clara sobre como diferentes indústrias apli-

cam testes automatizados na integração contínua, o que é essencial para empresas que buscam otimizar suas práticas de desenvolvimento de software e garantir produtos de alta qualidade.

5.3 Ferramentas de Apoio ao Teste Automatizado

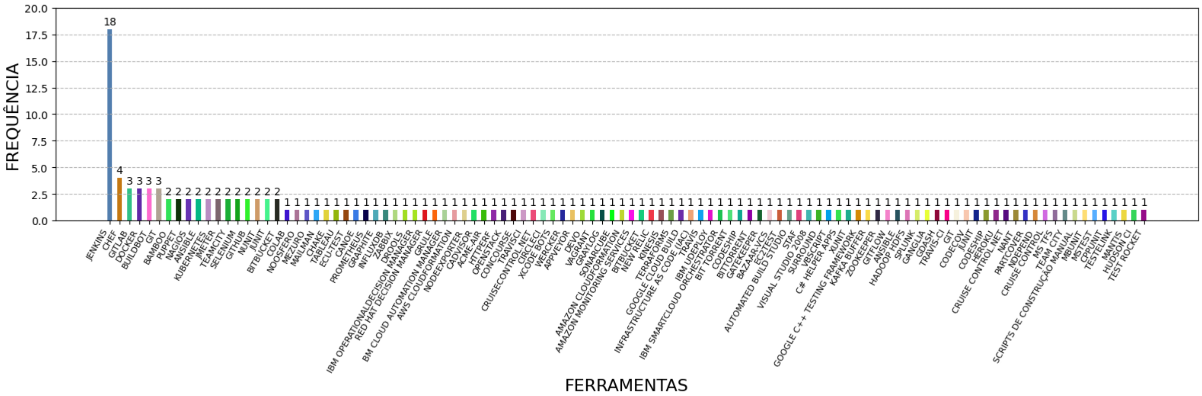


Figura 5.11: Imagem da autora: Ferramentas utilizadas nos estudos

Com base na análise dos estudos primários, foram identificadas as ferramentas de apoio ao teste automatizado, com ênfase na integração contínua, utilizadas em contexto industrial, relacionados à QP3, conforme apresentado no gráfico da Figura 5.11.

A Figura 5.11 evidência popularidade de **Jenkins**, que se destaca como a ferramenta mais popular para integração contínua, indicando sua robustez, flexibilidade e ampla adoção em diferentes setores industriais. Isso ressalta sua capacidade de se integrar a uma variedade de tecnologias e sua eficácia em automatizar o ciclo de vida de desenvolvimento de software.

A evolução das práticas de desenvolvimento de software trouxe à tona a necessidade de integrar testes automatizados nos processos de entrega contínua. Ferramentas como Jenkins, GitLab e Travis CI se destacam como pilares nesse contexto, permitindo uma abordagem mais ágil e confiável na entrega de produtos de software.

Diversos estudos têm explorado a aplicação dessas ferramentas em diferentes setores. Por exemplo, Siqueira et al. (2018) em seu artigo "*Continuous delivery: Building trust in a large-scale, complex government organization*" destacam o uso de ferramentas

como Colab, Gitlab e Chef para melhorar a confiança em ambientes governamentais complexos.

Por outro lado, Baron e Louis (2023) em "*Framework and tooling proposals for Agile certification of safety-critical embedded software in avionic systems*"enfatizam a necessidade de Jenkins e Bamboo para garantir a segurança e eficácia em sistemas embarcados.

No âmbito da automação de testes, Ponsard e Ramon (2022) em "*Survey of automation practices in model-driven development and operations*"utilizaram GitLab e Jenkins como principais ferramentas para automatizar práticas em desenvolvimento orientado a modelos. Esses exemplos ilustram a versatilidade e aplicabilidade das ferramentas de testes em diferentes contextos.

A tabela 5.3 resume as ferramentas de testes mencionadas nos artigos analisados:

Tabela 5.3: Artigos e suas respectivas ferramentas de testes

Artigo	**Conjunto de Ferramentas**
(SIQUEIRA et al., 2018)	Colab, Noosfero, Gitlab, Mezuro, Mailman, Chef, Chake
(BARON; LOUIS, 2023)	Jenkins, Bamboo
(SHAHIN et al., 2017)	Jenkins, Chef
(SPIEKER et al., 2017)	Tableau
(V¨OST; WAGNER, 2016)	ECU-Test, Docker, Puppet, CanOE
(PONSARD; RAMON, 2022)	GitLab, Jenkins
(TANZIL et al., 2023)	Jenkins
(ROUF et al., 2021)	Prometheus, Graphite, InfluxDB, Nagios, Zabbix, Drools, IBM Operational Decision Manager, Red Hat Decision Manager, Grule, BM Cloud Automation Manager, Ansible, AWS CloudFormation, Docker, Kubernetes, NodeExporter, cAdvisor, Acme-Air, JMeter, httperf, OpenStack

Artigo	**Conjunto de Ferramentas**
(HILTON et al., 2017)	Concourse, Jenkins, TravisCI, CruiseControl.NET, CircleCI, TeamCity, XCode Bots, Buildbot, Wercker, AppVeyor
(LWAKATARE et al., 2019)	Devo, Jenkins, Git, Vagrant, Selenium, Ansible, GrayLog, SonarCube, Docker, Amazon CloudFormation, Amazon Monitoring Services, Bitbucket, Chef, New Relic, Puppet, Kinesis
(PIANINI; NERI, 2021)	GitLab, Terraform, Kubernetes
(M¨AKINEN et al., 2016)	Jenkins, TeamCity, Buildbot, Bamboo
(STURM; POLLARD; CRAIG, 2017)	Jenkins
(WANG et al., 2022)	Google Cloud Build, Infrastructure as Code (IaC)
(RUBERT; FARIAS, 2022)	Travis, Jenkins
(BATRA; JATAIN, 2020)	GitHub, Jenkins
(PAULE; D¨ULLMANN; HOORN, 2019)	Jenkins
(VIRMANI, 2015)	IBM uDeploy, IBM SmartCloud Orchestrator
(RAHMAN et al., 2015)	Bit Torrent, Codeship
(FEITELSON; FRACHTENBERG; BECK, 2013)	BitTorrent, Gatekeeper
(LACOSTE, 2009)	Buildbot, Bazaar-vcs, ec2test
(STOLBERG, 2009)	Automated Build Studio, STAF, Visual Studio 2008, Surround, NUnit, VBScript, C helper apps
(MEMON et al., 2017)	JUnit
(ELBAUM; ROTHERMEL; PENIX, 2014)	xUnit, Google C++ Testing Framework
(SHAHIN; BABAR, 2020)	Kafka Buffer, Zookeeper, Jenkins, Gitflow, Ansible, Bitbucket, Hadoop HDFS

Artigo	**Conjunto de Ferramentas**
(NEELY; STOLT, 2013)	Git, Splunk, Ganglia, Nagios, GDash, Jenkins
(RAUSCH et al., 2017)	GitHub, Travis-CI
(D¨ULLMANN; PAULE; HOORN, 2018)	Jenkins
(RODRIGUES et al., 2017)	Git, Selenium, Codecov, JUnit, Bitbucket, Codeship, Heroku
(MELLADO et al., 2010)	Cruise Control.Net, Nant, NUnit, PartCover, NDepend
(LUZ; PINTO; BONIFÁCIO, 2019)	Jenkins
(LEPP¨ANEN; KILAMO; MIKKONEN, 2015)	Jenkins, Git, Chef
(T¨UZ¨UN et al., 2019)	Cruise Control, Jenkins, MS TFS, Team City, scripts de construção manual
(RAMLER; PUTSCH¨OGL; WINKLER, 2014)	JUnit, MBUnit, MSTest, CppUnit
(LANGE et al., 2023)	Jenkins, GitLab
(WANG; PYH¨AJ¨ARVI; M¨ANTYL¨A, 2020)	Tools Root, Scripts, TestLab infra, WinOS image, Environment virtualization service, Jenkins, Radiator, and TA telemetry
(SONI, 2015)	Jenkins
(HEMON-HILDGEN; ROWE; MONNIER-SENICOURT, 2020)	GitLab, Jenkins
(COLLINS; LUCENA, 2012)	TestLink, Mantis, Jmeter, Hudson CI
(MARIJAN; GOTLIEB; SEN, 2013)	Test Rocket

5.4 Papéis no Processo de Integração Contínua

Com base na análise dos estudos primários, foram identificados os seguintes papéis envolvidos no processo de integração contínua: desenvolvedores, devOps, engenheiros de software e equipe de operação, como ilustrado na Figura 5.12 e na tabela 5.4, como resposta à QP4. Esses papéis podem variar dependendo da estrutura da empresa.

O papel de *DevOps* é central na integração contínua, como pode ser observado na Figura 5.12, destacando-se como o mais influente entre os papéis analisados. Isso reflete a evolução das práticas de desenvolvimento de software, em que a integração entre desenvolvimento e operações é crucial para a agilidade e eficiência. Profissionais de *DevOps* são essenciais na criação de *pipelines* de CI/CD (*Continuous Integration/Continuous Delivery*), facilitando a colaboração entre equipes, automatizando testes e *deployment*, e garantindo a qualidade e segurança do software.

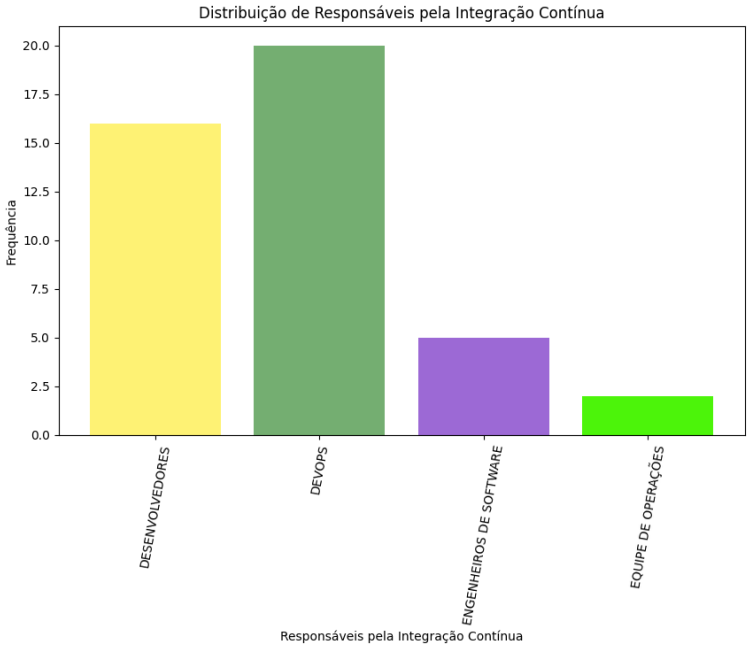


Figura 5.12: Imagem da autora: Distribuição de responsáveis pela integração contínua

A integração contínua envolve uma variedade de profissionais além de *DevOps*, incluindo desenvolvedores, engenheiros de software e equipes de operações, conforme ilustrado na Figura 5.12. Isso denota um ambiente interdisciplinar, onde a colaboração entre diferentes especialistas é fundamental para abordar as complexidades do desenvolvimento moderno de software.

Enquanto desenvolvedores e engenheiros de software tradicionalmente se concentravam mais nos aspectos internos do desenvolvimento, o envolvimento deles em tarefas de CI indica uma expansão de suas responsabilidades para incluir operações de *deployment* e manutenção, ressaltando a tendência de *"shift-left"* em testes e qualidade.

Na Figura 5.12, a menor frequência da equipe de operações sugere que muitas das suas funções tradicionais podem estar integradas sob o papel de *DevOps*. Isso pode indicar uma fusão de responsabilidades ou uma maior automação das tarefas operacionais dentro do processo de integração contínua.

Para organizações que buscam aprimorar ou implementar práticas de integração contínua, essas conclusões sublinham a importância de investir em capacitação e ferramentas para *DevOps* e de promover uma cultura de colaboração entre desenvolvimento, operações e engenharia. Além disso, sugere-se que as empresas reavaliem as funções tradicionais para melhor alinhá-las com as demandas da entrega contínua de software.

Essas conclusões podem guiar decisões estratégicas, desde a formação de equipes até o desenvolvimento de políticas internas e a escolha de tecnologias, para maximizar a eficácia dos processos de desenvolvimento de software na era digital.

Tabela 5.4: Artigos e Responsáveis pela Integração Contínua

Artigo	**Responsável pela Integração Contínua**
(SIQUEIRA et al., 2018)	DevOps
(BARON; LOUIS, 2023)	Desenvolvedores
(BARON; LOUIS, 2021)	DevOps
(SHAHIN et al., 2017)	Equipe de Operações
(BRUNELIERE et al., 2022)	DevOps
(PONSARD; RAMON, 2022)	DevOps
(TANZIL et al., 2023)	DevOps
(ROUF et al., 2021)	DevOps
(HILTON et al., 2017)	Desenvolvedores
(LWAKATARE et al., 2019)	DevOps

Artigo	**Responsável pela Integração Contínua**
(LEPP¨ANEN et al., 2015)	Desenvolvedores
(PIANINI; NERI, 2021)	Desenvolvedores
(M¨AKINEN et al., 2016)	Desenvolvedores
(CLAPS; SVENSSON; AURUM, 2015)	Desenvolvedores
(STURM; POLLARD; CRAIG, 2017)	DevOps
(SNEED; VERHOEF, 2019)	Desenvolvedores
(WANG et al., 2022)	DevOps
(RUBERT; FARIAS, 2022)	DevOps
(BATRA; JATAIN, 2020)	Desenvolvedores
(PAULE; D¨ULLMANN; HORN, 2019)	DevOps
(VIRMANI, 2015)	DevOps
(FEITELSON; FRACHTENBERG; BECK, 2013)	Engenheiros de Software
(LACOSTE, 2009)	Desenvolvedores
(STOLBERG, 2009)	Desenvolvedores
(MEMON et al., 2017)	Desenvolvedores
(ELBAUM; ROTHERMEL; PENIX, 2014)	Desenvolvedores
(ERICH; AMRIT; DANEVA, 2017)	DevOps
(SHAHIN; BABAR, 2020)	Engenheiros de Software
(NEELY; STOLT, 2013)	Engenheiros de Software
(D¨ULLMANN; PAULE; HORN, 2018)	Desenvolvedores
(RODRIGUES et al., 2017)	DevOps

Artigo	**Responsável pela Integração Contínua**
(MELLADO et al., 2010)	Desenvolvedores
(LUZ; PINTO; BONIFÁCIO, 2019)	DevOps
(LEPP¨ANEN; KILAMO; MIKKONEN, 2015)	Desenvolvedores
(RAMLER; PUTSCH¨OGL; WINKLER, 2014)	Engenheiros de Software
(CHEN, 2015)	Equipe de Operações
(LEITE et al., 2021)	DevOps
(STRADOWSKI; MADEYSKI, 2023)	DevOps
(WANG; PYH¨AJ¨ARVI; M¨ANTYL¨A, 2020)	DevOps
(SONI, 2015)	DevOps
(HEMON-HILDGEN; ROWE; MONNIER-SENICOURT, 2020)	DevOps
(STÅHL; BOSCH, 2013)	Desenvolvedores
(COLLINS; LUCENA, 2012)	Engenheiros de Software

5.5 Problemas Recorrentes na Integração Contínua

Buscando responder à QP5, a partir da análise dos estudos primários, foi identificado um conjunto de problemas e desafios relacionados à integração contínua, especialmente em relação à automação de testes, que foram agrupados nas seguintes categorias:

A integração contínua (IC) é uma prática vital para garantir a eficiência e a qualidade no desenvolvimento de software. No entanto, a sua implementação traz uma série de desafios que precisam ser enfrentados. A partir da análise de estudos primários, foi possível identificar um conjunto de problemas e desafios relacionados à IC, especialmente em relação à automação de testes, que foram agrupados nas seguintes categorias:

1. **Desafios Técnicos**:

   - **Infraestrutura inadequada**: A falta de uma infraestrutura robusta para suportar a IC pode levar a falhas na implementação. Siqueira et al. (SIQUEIRA et al., 2018) relatam que problemas de configuração e manutenção podem ser impeditivos para a entrega contínua em organizações governamentais complexas.

   - **Problemas de implantação do produto**: De acordo com Baron e Louis (BARON; LOUIS, 2021), as dificuldades na implantação contínua incluem não apenas atualizações, mas também mudanças em esquemas de banco de dados, o que pode criar riscos significativos em ambientes críticos.

   - **Falta de automação eficiente**: A automação de testes é crucial para o sucesso da IC. Wang et al. (WANG et al., 2022) destacam que a maturidade da automação de testes é um fator determinante para melhorar a qualidade do produto. No entanto, a falta de ferramentas integradas e a ocorrência de *builds* longos ou falhas de compilação podem comprometer essa automação.

2. **Desafios Organizacionais**:

   - **Coordenação e comunicação entre equipes**: Hilton et al. (HILTON et al., 2017) enfatizam que a falta de uma comunicação eficaz entre as equipes de desenvolvimento e operações pode resultar em problemas de coordenação, dificultando o fluxo de trabalho necessário para a IC.

   - **Resistência à mudança e adaptação**: Como mencionado por Claps et al. (CLAPS; SVENSSON; AURUM, 2015), a resistência à adoção de novas práticas e tecnologias por parte das equipes pode ser um obstáculo significativo para a implementação bem-sucedida da IC.

   - **Pressão e expectativas**: Leppänen et al. (LEPP¨ANEN et al., 2015) observam que a pressão por entregas rápidas pode criar tensões entre a necessidade de velocidade e a manutenção da qualidade e estabilidade do software.

3. **Desafios de Qualidade e Testes**:

- **Balanceamento entre velocidade e qualidade**: A busca por testes rápidos pode se tornar um gargalo, conforme discutido por Sturm et al. (STURM; POLLARD; CRAIG, 2017), levando a uma redução na qualidade do produto devido à pressa.
- **Cobertura de testes e risco**: A necessidade de equilibrar a cobertura de testes com os riscos associados e os recursos disponíveis é um desafio reconhecido por Rubert e Farias (RUBERT; FARIAS, 2022), que investigaram os impactos da entrega contínua na qualidade do código.

4. **Problemas Específicos de DevOps**:

- **Atrito entre silos de desenvolvimento e operações**: Virmani (VIRMANI, 2015) destaca os conflitos de objetivos entre as equipes de desenvolvimento, que buscam inovação rápida, e as operações, que priorizam a estabilidade, como um grande desafio.
- **Gestão de ambiente de produção**: A gestão diária dos ambientes de produção por equipes de desenvolvimento é uma preocupação levantada por Düllmann et al. (D¨ULLMANN; PAULE; HOORN, 2018), que discutem como isso pode impactar a eficácia da IC.
- **Segurança e controles de acesso**: A segurança na entrega contínua é uma questão crítica, especialmente em contextos sensíveis. Neely e Stolt (NEELY; STOLT, 2013) enfatizam que as práticas de segurança precisam ser integradas aos processos de IC para garantir a proteção dos sistemas.

Identificar esses problemas é crucial para otimizar os processos de IC, melhorar a colaboração entre as equipes, aumentar a eficiência do desenvolvimento e reduzir o tempo de chegada ao mercado sem comprometer a qualidade do produto. Além disso, a análise da frequência de menção dos problemas nos estudos pode fornecer uma indicação da prevalência e do impacto desses desafios, conforme discutido por Erich et al. (ERICH; AMRIT; DANEVA, 2017).

A integração contínua, embora fundamental para o desenvolvimento ágil e para a entrega de software, vem com um conjunto de desafios que necessitam de uma abor-

dagem estratégica para serem superados. Reconhecendo e endereçando esses desafios de forma proativa, as organizações podem maximizar os benefícios da integração contínua, melhorando a eficiência, a qualidade e a satisfação do cliente.

5.6 Ameaça à Validação

As principais ameaças à validade identificadas para este mapeamento sistemático são a seleção do estudo, a validade dos dados e a validade da pesquisa. Nesta sessão será explicado como foram mitigadas as ameaças.

- **Validade da seleção de estudos**: a formulação da *string* de busca poderia representar riscos à pertinência dos estudos primários selecionados. Para atenuar esse risco, o processo de revisão foi conduzido com base nas diretrizes para revisões sistemáticas de literatura propostas por (KITCHENHAM et al., 2009). A busca por estudos primários foi conduzida nas bases de dados *ScienceDirect* e *IEEE Explore*, reconhecidas como fontes fundamentais na disciplina de engenharia de software. Implementaram-se filtros para a seleção dos estudos primários, abrangendo o período de 2015 a 2023. No decorrer da pesquisa, foram avaliados diversos tipos de publicações, incluindo artigos de conferências, artigos de periódicos, relatórios técnicos e capítulos de livro.

  Em relação à inclusão/exclusão de estudos, para assegurar um processo de seleção equânime, foram previamente definidas as questões de pesquisa e critérios de inclusão e exclusão durante a fase de planejamento. O protocolo deste mapeamento sistemático foi submetido à revisão por todos os autores do estudo, que possuem experiência na condução de estudos secundários. Com o intuito de aumentar a confiabilidade dos resultados, cada artigo foi examinado por, no mínimo, dois pesquisadores. Em casos de discrepâncias na aplicação dos critérios de inclusão e exclusão, o estudo em questão foi reavaliado por ambos os participantes para uma decisão consensual final.

- **Validação do Dados**: outro risco associado a este mapeamento sistemático de literatura diz respeito aos dados coletados dos estudos primários, dado que nem

todas as informações estavam claras para responder às questões de pesquisa, e alguns dados necessitaram de interpretação. Ademais, em situações de discordância entre os revisores, foram realizadas discussões para assegurar a obtenção de um consenso.

- **Validação da Pesquisa**: em relação às ameaças à validade da pesquisa, adotou-se a validação por pares para assegurar a confiabilidade dos resultados do estudo.

Quanto à generalização deste estudo, reconhece-se que limitar a pesquisa a estudos publicados apenas em repositórios acadêmicos ou obtidos através de *snowballing* pode representar um risco para a validade dos resultados. Além disso, a inclusão de dados do setor industrial baseou-se exclusivamente em evidências obtidas para generalização, sem considerar todas as arquiteturas potenciais desenvolvidas pela indústria ou as divulgadas na literatura cinzenta.

5.7 Avaliar a Qualidade da Produção Científica

O Qualis CAPES é um sistema de classificação de periódicos científicos brasileiros que avalia a qualidade e relevância de cada revista. Ele é utilizado para avaliar a produção científica de pesquisadores e instituições, orientar na escolha de revistas para publicação e estimular a melhoria da qualidade dos periódicos brasileiros. A classificação leva em consideração fatores como índice de citações, qualidade dos artigos, comitê editorial, processo de avaliação e visibilidade.

A Figura 5.13 mostra uma tendência geral de aumento no número de publicações ao longo dos anos referente a metodologias de apoio de testes na integração contínua aplicado na indústria, com picos significativos em 2017 e 2021, indicando anos de alta produtividade. No entanto, há oscilações notáveis, incluindo quedas em 2018 e 2022, sugerindo que eventos externos ou ciclos naturais de pesquisa podem ter influenciado essas variações. Em geral, o número de publicações se manteve relativamente estável, com flutuações que merecem investigação para entender melhor os fatores que impactam a produção acadêmica com relação ao tema pesquisado.

A análise da Figura 5.14, o *top* de 10 dos "*Journal/Booktitle*", mostra as publicações agrupadas pelo título do jornal ou conferência. Aqui se encontram alguns dos

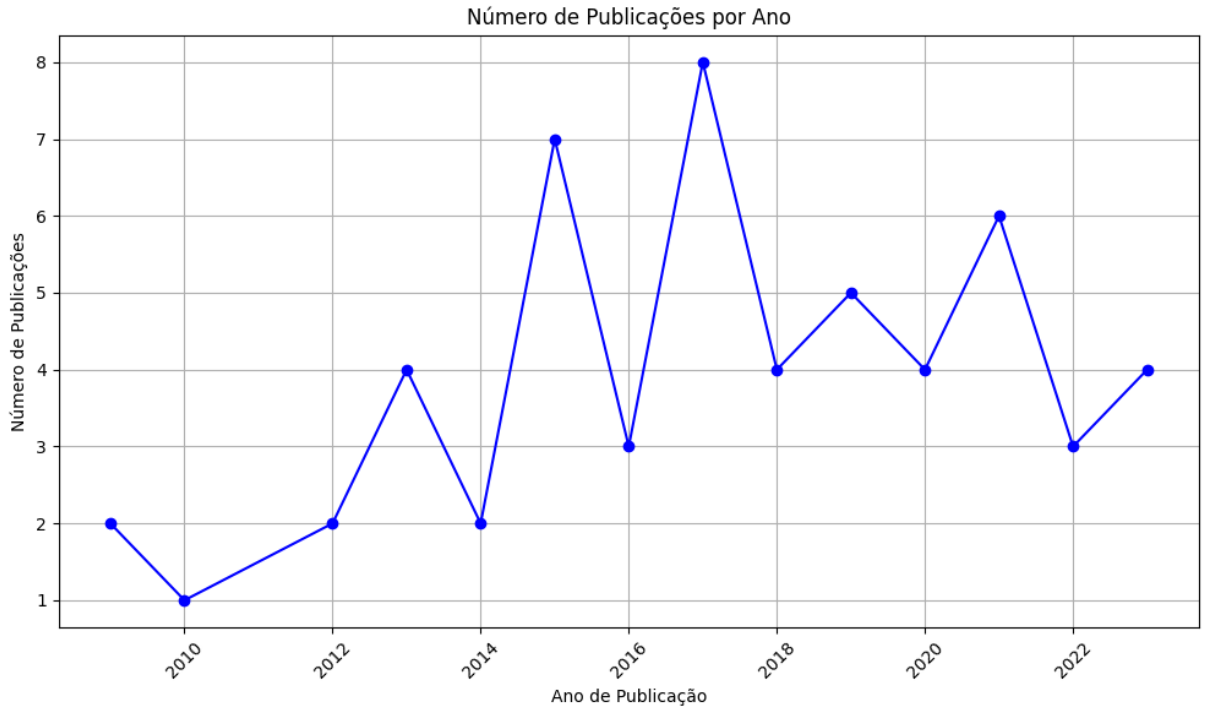


Figura 5.13: Imagem da autora: Número de Publicações por Ano

resultados: *Journal of Systems and Software*: 6 publicações, *Information and Software Technology*: 5 publicações, 2009 *Agile Conference*: 2 publicações, *Computers in Industry*: 2 publicações. Os demais títulos apresentaram uma publicação cada.

Esses resultados indicam que *"Journal of Systems and Software"* e *"Information and Software Technology"* são as fontes mais frequentes para as publicações, o que mostra que, para esse estudo, as revistas citadas são de maior relevância ou preferência na área de estudo.

No Apêndice B (Tabela 2), são expostos mais detalhes sobre fator de impacto, quartil e *H-Index* dos trabalhos analisados. A maioria dos trabalhos analisados que possuem fator de impacto estão classificados em estratos elevados, como A1 e A2, indicando publicações de alta qualidade. Com base na coluna E, observamos que esses periódicos e essas conferências apresentam fatores de impacto substanciais, reforçando ainda mais a relevância e o rigor acadêmico das publicações incluídas neste estudo. Essa alta qualificação dos trabalhos analisados contribui significativamente para a robustez e credibilidade dos resultados deste trabalho.

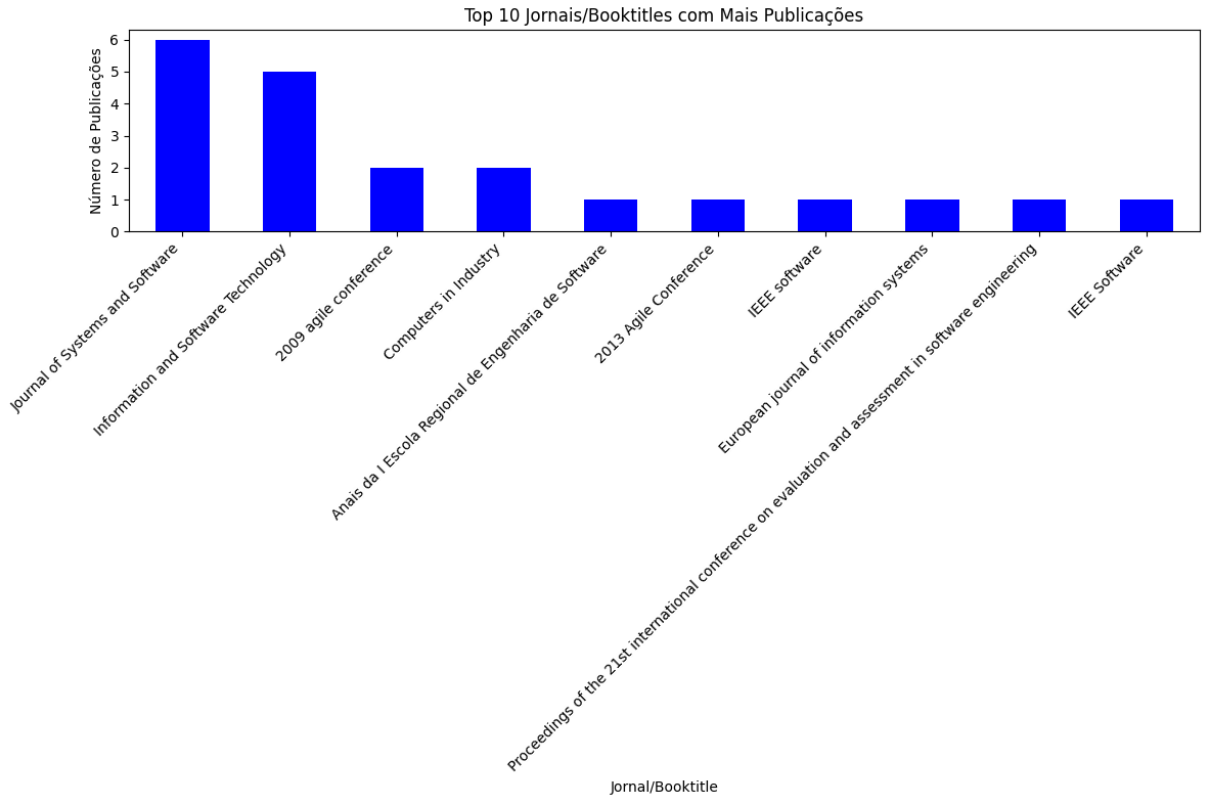


Figura 5.14: Imagem da autora: Top 10 *Jornal/Booktitles* com Mais Publicações

5.8 Conclusões do capítulo

Neste capítulo, foi realizada uma análise das técnicas de testes de software e integração contínua na indústria, com foco em estudos de caso em diversos domínios. As seções organizadas responderam às questões de pesquisa do mapeamento sistemático da literatura, oferecendo um panorama dos estudos empíricos, domínios de aplicação e ferramentas de teste automatizado. Embora a integração contínua traga benefícios, desafios significativos — técnicos, organizacionais e de qualidade — exigem soluções estratégicas. As conclusões destacam a necessidade de colaboração eficaz entre equipes e investimento em práticas de DevOps para otimizar o desenvolvimento de software. Além disso, a maioria das publicações analisadas está em periódicos de alto impacto, reforçando a relevância das contribuições na área. Este capítulo, assim, oferece uma base sólida para futuras investigações e práticas, direcionando esforços para aprimorar a qualidade e a eficiência no desenvolvimento de software.

6 Conclusão

Este trabalho buscou analisar a aplicação de testes automatizados na indústria dentro do contexto de integração contínua, destacando a importância crescente de testes automatizados para garantir a qualidade, segurança e manutenção do software. O estudo revelou que, apesar dos benefícios evidentes, como redução de falhas e eficiência operacional, muitas organizações ainda enfrentam desafios significativos para integrar plenamente essas práticas devido a limitações técnicas, custos e falta de habilidades adequadas entre os profissionais.

Identificar os papéis responsáveis pela integração contínua é crucial para entender como diferentes equipes e profissionais contribuem para o sucesso dessa prática. Isso permite às organizações otimizar suas equipes e processos para maximizar a eficiência e eficácia da entrega de software. Os papéis envolvidos na integração contínua e o número de estudos que identificam cada papel fornecem uma medida quantitativa de sua importância e envolvimento no processo. Essas métricas ajudam a destacar quais papéis são mais influentes ou críticos para a implementação bem-sucedida da integração contínua dentro das organizações.

O objetivo real deste trabalho é proporcionar uma compreensão abrangente da aplicação de testes automatizados no contexto da integração contínua, enfatizando sua relevância para a garantia da qualidade e segurança do software. A pesquisa demonstra como essas práticas podem ser utilizadas em diferentes setores, ajudando as organizações a otimizar processos e melhorar a entrega de produtos. Ao identificar os papéis responsáveis pela integração contínua, o estudo fornece insights valiosos sobre a dinâmica das equipes e sua contribuição para o sucesso dessa prática. Isso não apenas orienta as empresas na alocação de recursos, mas também destaca a importância de capacitar os profissionais nas habilidades necessárias para enfrentar os desafios da integração contínua.

A aplicação dos resultados deste trabalho pode se estender a diversos ambientes industriais, oferecendo diretrizes para a adoção efetiva de testes automatizados e integração contínua. Futuras pesquisas podem explorar a eficácia de novas ferramentas

emergentes e desenvolver estudos longitudinais para analisar o impacto dessas práticas na qualidade do software e na produtividade das equipes ao longo do tempo, contribuindo assim para um avanço contínuo no campo do desenvolvimento de software.

Este trabalho fornece uma visão geral para a compreensão dos testes automatizados e da integração contínua na indústria de software através de um aprofundamento teórico detalhado e um mapeamento sistemático empírico. Neste estudo, foi utilizado o método GQM, proposto por Kitchenham em 2009, que estrutura o mapeamento sistemático, possibilitando que seja replicado.

Uma limitação do estudo foi o fato de considerar apenas as bases de dados da *Science Direct* e IEEE. Entretanto, a base de dados da *Science Direct* indexa estudos de outras bases de dados como ACM e *Springer*. As ferramentas de teste avaliadas estão limitadas às disponíveis até a data de conclusão do estudo, não considerando avanços tecnológicos mais recentes que poderiam influenciar a eficácia dos testes.

No entanto, apesar dos esforços para abarcar todas as evidências pertinentes neste mapeamento sistemático de literatura, e embora existam métodos desenvolvidos com esse objetivo, é provável que algumas arquiteturas de referência tenham sido omitidas, especialmente aquelas originadas no meio acadêmico. Esse cenário inclui ainda aquelas propostas em contextos industriais e que não foram divulgadas em repositórios acadêmicos.

Como futuros trabalhos, pretende-se avaliar o impacto de novas ferramentas e tecnologias de automação de testes na integração contínua. Seria também relevante avaliar o impacto de novas ferramentas e tecnologias emergentes na automação de testes e integração contínua. Propõe-se o desenvolvimento de estudos longitudinais para avaliar os efeitos reais da implementação de testes automatizados sobre a qualidade do software e a produtividade das equipes ao longo do tempo. Adicionalmente, investigações mais profundas sobre as barreiras organizacionais, culturais e tecnológicas que impedem a adoção dessas práticas em algumas empresas poderiam prover *insights* valiosos para superar esses desafios.

Bibliografia

BARON, C.; LOUIS, V. Towards a continuous certification of safety-critical avionics software. *Computers in Industry*, Elsevier, v. 125, p. 103382, 2021.

BARON, C.; LOUIS, V. Framework and tooling proposals for agile certification of safety-critical embedded software in avionic systems. *Computers in Industry*, Elsevier, v. 148, p. 103887, 2023.

BASILI, V. R.; ROMBACH, H. D. The tame project: Towards improvement-oriented software environments. *IEEE Transactions on software engineering*, IEEE, v. 14, n. 6, p. 758–773, 1988.

BATRA, P.; JATAIN, A. Measurement based performance evaluation of devops. In: IEEE. *2020 International Conference on Computational Performance Evaluation (ComPE)*. [S.l.], 2020. p. 757–760.

BOSCH, J. Exploratory testing of large-scale systems–testing in the continuous integration and delivery pipeline. 2017.

BOURQUE, P.; DUPUIS, R.; ABRAN, A.; MOORE, J.; TRIPP, L. The guide to the software engineering body of knowledge. *IEEE Software*, v. 16, n. 6, p. 35–44, 1999.

BRUNELIERE, H.; MUTTILLO, V.; ERAMO, R.; BERARDINELLI, L.; GOMEZ, A.; BAGNATO, A.; SADOVYKH, A.; CICCHETTI, A. Aidoart: Ai-augmented automation for devops, a model-based framework for continuous development in cyber–physical systems. *Microprocessors and Microsystems*, Elsevier, v. 94, p. 104672, 2022.

BSTQB. *Syllabus de Certificação CTFL 4.0*. 2024. Accessed: 15-09-2024. Disponível em: ⟨https://bstqb.online/files/syllabus_ctfl_4.0br.pdf⟩.

BURNSTEIN, I. *Practical software testing: a process-oriented approach*. [S.l.]: Springer Science & Business Media, 2006.

CALDIERA, V. R. B. G.; ROMBACH, H. D. The goal question metric approach. *Encyclopedia of software engineering*, p. 528–532, 1994.

CHEN, L. Continuous delivery: Huge benefits, but challenges too. *IEEE software*, IEEE, v. 32, n. 2, p. 50–54, 2015.

CLAPS, G. G.; SVENSSON, R. B.; AURUM, A. On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software technology*, Elsevier, v. 57, p. 21–31, 2015.

COLLINS, E. F.; LUCENA, V. F. D. Software test automation practices in agile development environment: An industry experience report. In: IEEE. *2012 7th International Workshop on Automation of Software Test (AST)*. [S.l.], 2012. p. 57–63.

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

DERMEVAL, D.; COELHO, J.; BITTENCOURT, I. I. Mapeamento sistemático e revisão sistemática da literatura em informática na educação. *JAQUES, Patrícia Augustin; SIQUEIRA; Sean; BITTENCOURT, Ig; PIMENTEL, Mariano.(Org.) Metodologia de Pesquisa Científica em Informática na Educação: Abordagem Quantitativa. Porto Alegre: SBC*, 2020.

D¨ULLMANN, T. F.; PAULE, C.; HOORN, A. van. Exploiting devops practices for dependable and secure continuous delivery pipelines. In: *Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering*. [S.l.: s.n.], 2018. p. 27–30.

DUVALL, P. M.; MATYAS, S.; GLOVER, A. *Continuous integration: improving software quality and reducing risk*. [S.l.]: Pearson Education, 2007.

ELBAUM, S.; ROTHERMEL, G.; PENIX, J. Techniques for improving regression testing in continuous integration development environments. In: *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. [S.l.: s.n.], 2014. p. 235–245.

ERICH, F. M.; AMRIT, C.; DANEVA, M. A qualitative study of devops usage in practice. *Journal of software: Evolution and Process*, Wiley Online Library, v. 29, n. 6, p. e1885, 2017.

FEITELSON, D. G.; FRACHTENBERG, E.; BECK, K. L. Development and deployment at facebook. *IEEE Internet Computing*, IEEE, v. 17, n. 4, p. 8–17, 2013.

HAGHIGHATKHAH, A.; M¨ANTYL¨A, M.; OIVO, M.; KUVAJA, P. Test prioritization in continuous integration environments. *Journal of Systems and Software*, Elsevier, v. 146, p. 80–98, 2018.

HAMEL, C.; MICHAUD, A.; THUKU, M.; SKIDMORE, B.; STEVENS, A.; NUSSBAUMER-STREIT, B.; GARRITTY, C. Defining rapid reviews: a systematic scoping review and thematic analysis of definitions and defining characteristics of rapid reviews. *Journal of Clinical Epidemiology*, v. 129, p. 74–85, 2021. ISSN 0895-4356. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0895435620311276⟩.

HEMON-HILDGEN, A.; ROWE, F.; MONNIER-SENICOURT, L. Orchestrating automation and sharing in devops teams: a revelatory case of job satisfaction factors, risk and work conditions. *European journal of information systems*, Taylor & Francis, v. 29, n. 5, p. 474–499, 2020.

HENRIQUE, J. *Primeiros Passos Com O Jenkins*. 2018. ⟨https://mundodevops.com/blog/primeiros-passos-com-o-jenkins/⟩. [Online; accessado em :27-outubro-2022].

HILTON, M.; NELSON, N.; TUNNELL, T.; MARINOV, D.; DIG, D. Trade-offs in continuous integration: assurance, security, and flexibility. In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. [S.l.: s.n.], 2017. p. 197–207.

HIRANO, M. *CI/CD: Continuous Integration and Continuous Delivery*. 2019. Accessed: 15-09-2024. Disponível em: ⟨https://medium.com/tecnologia-e-afins/ci-cd-continuous-integration-and-continuous-delivery-fb5d0aed4bf5⟩.

IBM. *Continuous Integration*. 2024. Accessed: 15-09-2024. Disponível em: ⟨https://www.ibm.com/br-pt/topics/continuous-integration⟩.

JABBARI, R.; ALI, N. bin; PETERSEN, K.; TANVEER, B. What is devops? a systematic mapping study on definitions and practices. In: *Proceedings of the Scientific Workshop Proceedings of XP2016*. [S.l.: s.n.], 2016. p. 1–11.

KITCHENHAM, B. Procedures for performing systematic reviews. *Keele, UK, Keele University*, v. 33, n. 2004, p. 1–26, 2004.

KITCHENHAM, B.; BRERETON, O. P.; BUDGEN, D.; TURNER, M.; BAILEY, J.; LINKMAN, S. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, Elsevier, v. 51, n. 1, p. 7–15, 2009.

KUMARESEN, P. P.; FRASHERI, M.; ENOIU, E. P. Agent-based software testing: A definition and systematic mapping study. In: IEEE. *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. [S.l.], 2020. p. 24–31.

LACOSTE, F. J. Killing the gatekeeper: Introducing a continuous integration system. In: IEEE. *2009 agile conference*. [S.l.], 2009. p. 387–392.

LANGE, M.; TRAN, K. C.; GRUNEWALD, A.; KOSCHEL, A.; PAKOSCH, A.; ASTROVA, I. Modern build automation for an insurance company tool selection. *Procedia Computer Science*, Elsevier, v. 219, p. 736–743, 2023.

LEITE, L.; PINTO, G.; KON, F.; MEIRELLES, P. The organization of software teams in the quest for continuous delivery: A grounded theory approach. *Information and Software Technology*, Elsevier, v. 139, p. 106672, 2021.

LEPP¨ANEN, M.; KILAMO, T.; MIKKONEN, T. Towards post-agile development practices through productized development infrastructure. In: IEEE. *2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering*. [S.l.], 2015. p. 34–40.

LEPP¨ANEN, M.; M¨AKINEN, S.; PAGELS, M.; ELORANTA, V.-P.; ITKONEN, J.; M¨ANTYL¨A, M. V.; M¨ANNIST¨O, T. The highways and country roads to continuous deployment. *Ieee software*, IEEE, v. 32, n. 2, p. 64–72, 2015.

LUZ, W. P.; PINTO, G.; BONIFÁCIO, R. Adopting devops in the real world: A theory, a model, and a case study. *Journal of Systems and Software*, Elsevier, v. 157, p. 110384, 2019.

LWAKATARE, L. E.; KILAMO, T.; KARVONEN, T.; SAUVOLA, T.; HEIKKIL¨A, V.; ITKONEN, J.; KUVAJA, P.; MIKKONEN, T.; OIVO, M.; LASSENIUS, C. Devops in practice: A multiple case study of five companies. *Information and Software Technology*, Elsevier, v. 114, p. 217–230, 2019.

M¨AKINEN, S.; LEPP¨ANEN, M.; KILAMO, T.; MATTILA, A.-L.; LAUKKANEN, E.; PAGELS, M.; M¨ANNIST¨O, T. Improving the delivery cycle: A multiple-case study of the toolchains in finnish software intensive enterprises. *Information and Software Technology*, Elsevier, v. 80, p. 175–194, 2016.

MALDONADO, J. C.; BARBOSA, E. F.; VINCENZI, A. M.; DELAMARO, M. E.; SOUZA, S. d. R. S. d.; JINO, M. *Introducao ao teste de software (versão 2004-01)*. [S.l.]: ICMC-USP, 2004.

MARIJAN, D.; GOTLIEB, A.; SEN, S. Test case prioritization for continuous regression testing: An industrial case study. In: IEEE. *2013 IEEE International Conference on Software Maintenance*. [S.l.], 2013. p. 540–543.

MARIJAN, D.; LIAAEN, M. Effect of time window on the performance of continuous regression testing. In: IEEE. *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. [S.l.], 2016. p. 568–571.

MARIJAN, D.; LIAAEN, M.; SEN, S. Devops improvements for reduced cycle times with integrated test optimizations for continuous integration. In: IEEE. *2018 IEEE 42nd annual computer software and applications conference (COMPSAC)*. [S.l.], 2018. v. 1, p. 22–27.

MÅRTENSSON, T.; STÅHL, D.; BOSCH, J. The emfis model—enable more frequent integration of software. In: IEEE. *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. [S.l.], 2017. p. 10–17.

MARTINS, G. *CI e CD: O que é?* 2024. Accessed: 15-09-2024. Disponível em: ⟨https://blog.hubdodesenvolvedor.com.br/ci-e-cd-o-que-e/⟩.

MELLADO, R. P.; MONTINI, D. Á.; DIAS, L. A. V.; CUNHA, A. M. da et al. Software product measurement and analysis in a continuous integration environment. In: IEEE. *2010 Seventh International Conference on Information Technology: New Generations*. [S.l.], 2010. p. 1177–1182.

MEMON, A.; GAO, Z.; NGUYEN, B.; DHANDA, S.; NICKELL, E.; SIEMBORSKI, R.; MICCO, J. Taming google-scale continuous testing. In: IEEE. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*. [S.l.], 2017. p. 233–242.

NEELY, S.; STOLT, S. Continuous delivery? easy! just change everything (well, maybe it is not that easy). In: IEEE. *2013 Agile Conference*. [S.l.], 2013. p. 121–128.

NETO, A.; CLAUDIO, D. Introdução a teste de software. *Engenharia de Software Magazine*, v. 1, p. 22, 2007.

NETO, J. G. d. R. *Usando técnicas de mineração de repositórios software para apoiar a automação de testes de software*. Dissertação (B.S. thesis) — Universidade Federal do Rio Grande do Norte, 2019.

OLIVEIRA, L. B. R. de; FELIZARDO, K. R.; FEITOSA, D.; NAKAGAWA, E. Y. Reference models and reference architectures based on service-oriented architecture: A systematic review. In: BABAR, M. A.; GORTON, I. (Ed.). *Software Architecture*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 360–367. ISBN 978-3-642-15114-9.

OLSSON, H. H.; ALAHYARI, H.; BOSCH, J. Climbing the stairway to heaven–a mulitiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In: IEEE. *2012 38th euromicro conference on software engineering and advanced applications*. [S.l.], 2012. p. 392–399.

PACHOULY, J.; AHIRRAO, S.; KOTECHA, K.; SELVACHANDRAN, G.; ABRAHAM, A. A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools. *Engineering Applications of Artificial Intelligence*, v. 111, p. 104773, 2022. ISSN 0952-1976. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0952197622000616⟩.

PAULE, C.; D¨ULLMANN, T. F.; HOORN, A. V. Vulnerabilities in continuous delivery pipelines? a case study. In: IEEE. *2019 IEEE international conference on software architecture companion (ICSA-C)*. [S.l.], 2019. p. 102–108.

PETERSEN, K.; VAKKALANKA, S.; KUZNIARZ, L. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and software technology*, Elsevier, v. 64, p. 1–18, 2015.

PETTICREW, M.; ROBERTS, H. *Systematic reviews in the social sciences: A practical guide*. [S.l.]: John Wiley & Sons, 2008.

PIANINI, D.; NERI, A. Breaking down monoliths with microservices and devops: an industrial experience report. In: IEEE. *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. [S.l.], 2021. p. 505–514.

PONSARD, C.; RAMON, V. Survey of automation practices in model-driven development and operations. In: *Proceedings of the Fourth International Workshop on Bots in Software Engineering*. [S.l.: s.n.], 2022. p. 14–17.

PRADHAN, S.; NANNIYUR, V. Large scale quality transformation in hybrid development organizations–a case study. *Journal of Systems and Software*, Elsevier, v. 171, p. 110836, 2021.

PRESSMAN, R. S.; MAXIM, B. Process models. *Software Engineering A Practitioner's Approach*, McGraw-Hill Education, p. 45–47, 2015.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software-9*. [S.l.]: McGraw Hill Brasil, 2021.

RAHMAN, A. A. U.; HELMS, E.; WILLIAMS, L.; PARNIN, C. Synthesizing continuous deployment practices used in software development. In: IEEE. *2015 Agile Conference*. [S.l.], 2015. p. 1–10.

RAMLER, R.; PUTSCH¨OGL, W.; WINKLER, D. Automated testing of industrial automation software: practical receipts and lessons learned. In: *Proceedings of the 1st International Workshop on Modern Software Engineering Methods for Industrial Automation*. [S.l.: s.n.], 2014. p. 7–16.

RAUSCH, T.; HUMMER, W.; LEITNER, P.; SCHULTE, S. An empirical analysis of build failures in the continuous integration workflows of java-based open-source software. In: IEEE. *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. [S.l.], 2017. p. 345–355.

RODRIGUES, P.; MACEDO, J.; FRANÇA, P.; FRANZ, L. P.; SILVA, J. P. da; CHEIRAN, J. Devops adoption in junior enterprise: an experience report of software development. In: SBC. *Anais da I Escola Regional de Engenharia de Software*. [S.l.], 2017. p. 89–96.

ROMAN, A. 2018 foundation syllabus overview. In: *A Study Guide to the ISTQB® Foundation Level 2018 Syllabus*. [S.l.]: Springer, 2018. p. 3–11.

ROUF, Y.; MUKHERJEE, J.; LITOIU, M.; WIGGLESWORTH, J.; MATEESCU, R. A framework for developing devops operation automation in clouds using components-off-the-shelf. In: *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. [S.l.: s.n.], 2021. p. 265–276.

RUBERT, M.; FARIAS, K. On the effects of continuous delivery on code quality: A case study in industry. *Computer Standards & Interfaces*, Elsevier, v. 81, p. 103588, 2022.

SHAHIN, M.; BABAR, M. A. On the role of software architecture in devops transformation: An industrial case study. In: *Proceedings of the International Conference on Software and System Processes*. [S.l.: s.n.], 2020. p. 175–184.

SHAHIN, M.; BABAR, M. A.; ZHU, L. Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, IEEE, v. 5, p. 3909–3943, 2017.

SHAHIN, M.; ZAHEDI, M.; BABAR, M. A.; ZHU, L. Adopting continuous delivery and deployment: Impacts on team structures, collaboration and responsibilities. In: *Proceedings of the 21st international conference on evaluation and assessment in software engineering*. [S.l.: s.n.], 2017. p. 384–393.

SIQUEIRA, R.; CAMARINHA, D.; WEN, M.; MEIRELLES, P.; KON, F. Continuous delivery: Building trust in a large-scale, complex government organization. *IEEE Software*, IEEE, v. 35, n. 2, p. 38–43, 2018.

SNEED, H.; VERHOEF, C. Re-implementing a legacy system. *Journal of Systems and Software*, Elsevier, v. 155, p. 162–184, 2019.

SOMMERVILLE, I. *Engenharia de Software. Capítulo 23: Teste de Software*. 8ed. ed. [S.l.]: Pearson, 2007.

SONI, M. End to end automation on cloud with build pipeline: the case for devops in insurance industry, continuous integration, continuous testing, and continuous delivery. In: IEEE. *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*. [S.l.], 2015. p. 85–89.

SPIEKER, H.; GOTLIEB, A.; MARIJAN, D.; MOSSIGE, M. Reinforcement learning for automatic test case prioritization and selection in continuous integration. In: *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*. [S.l.: s.n.], 2017. p. 12–22.

STÅHL, D.; BOSCH, J. Experienced benefits of continuous integration in industry software product development: A case study. In: *The 12th iasted international conference on software engineering,(innsbruck, austria, 2013)*. [S.l.: s.n.], 2013. p. 736–743.

STOLBERG, S. Enabling agile testing through continuous integration. In: IEEE. *2009 agile conference*. [S.l.], 2009. p. 369–374.

STRADOWSKI, S.; MADEYSKI, L. Exploring the challenges in software testing of the 5g system at nokia: A survey. *Information and Software Technology*, Elsevier, v. 153, p. 107067, 2023.

STURM, R.; POLLARD, C.; CRAIG, J. Chapter 10 - devops and continuous delivery. In: STURM, R.; POLLARD, C.; CRAIG, J. (Ed.). *Application Performance Management (APM) in the Digital Enterprise*. Boston: Morgan Kaufmann, 2017. p. 121–135. ISBN 978-0-12-804018-8. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/B9780128040188000103⟩.

TANZIL, M. H.; SARKER, M.; UDDIN, G.; IQBAL, A. A mixed method study of devops challenges. *Information and Software Technology*, Elsevier, v. 161, p. 107244, 2023.

T¨UZ¨UN, E.; TEKINERDOGAN, B.; MACIT, Y.; İNCE, K. Adopting integrated application lifecycle management within a large-scale software company: An action research approach. *Journal of Systems and Software*, Elsevier, v. 149, p. 63–82, 2019.

VINCENT, J.; KING, G.; LAY, P.; KINGHORN, J. Principles of built-in-test for run-time-testability in component-based software systems. *Software Quality Journal*, Springer, v. 10, p. 115–133, 2002.

VIRMANI, M. Understanding devops & bridging the gap from continuous integration to continuous delivery. In: IEEE. *Fifth international conference on the innovative computing technology (intech 2015)*. [S.l.], 2015. p. 78–82.

VITOR, M. $Revisao_Sistematica.[S.l.] : GitHub, 2024.¿¿$.

V¨OST, S.; WAGNER, S. Towards continuous integration and continuous delivery in the automotive industry. *arXiv preprint arXiv:1612.04139*, 2016.

WANG, Y.; M¨ANTYL¨A, M. V.; LIU, Z.; MARKKULA, J. Test automation maturity improves product quality—quantitative study of open source projects using continuous integration. *Journal of Systems and Software*, Elsevier, v. 188, p. 111259, 2022.

WANG, Y.; PYH¨AJ¨ARVI, M.; M¨ANTYL¨A, M. V. Test automation process improvement in a devops team: experience report. In: IEEE. *2020 ieee international conference on software testing, verification and validation workshops (icstw)*. [S.l.], 2020. p. 314–321.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. [S.l.: s.n.], 2014. p. 1–10.

WOLF, M.; SERPANOS, D. Safety and security in cyber-physical systems and internet-of-things systems. *Proceedings of the IEEE*, IEEE, v. 106, n. 1, p. 9–20, 2017.

XI, W. Integração contínua com software livre: um relato de implantação na fundação de amparo a pesquisa do estado de alagoas–fapeal. In: *XI Workshop de Software Livre*. [S.l.: s.n.], 2010. p. 16.

Apéndices

.1 Apêndice A: Tabela com Domínio da Aplicação e Técnicas e Critérios de Teste

Artigo	**Domínio de Aplicação**	**Técnicas e Critérios de Teste**
(SIQUEIRA et al., 2018)	Administração Pública	teste de unidade, testes de aceitação
(BARON; LOUIS, 2023)	Aeroespacial	teste de unidade , teste de integração, testes funcionais, teste de cobertura de código
(BARON; LOUIS, 2021)	Aeroespacial	Teste baseado em Riscos(RBT), teste de unidade e funcionais
(SHAHIN et al., 2017)	Agência de Pesquisa	teste de unidade, testes baseado em modelos
(PRADHAN; NANNIYUR, 2021)	Indústria	teste de aceitação, teste de integração
(BRUNELIERE et al., 2022)	Indústria	Testes unitarios, testes de cobertura de código, testes de integração, testes de sistemas
(SPIEKER et al., 2017)	Aprendizagem de Máquina	Teste de integração, teste de unidade, teste de desempenho, teste de aceitação , testes funcionais, testes não funcionais
(V¨OST; WAGNER, 2016)	Automotivo	teste de unidade, testes de regressão, teste de integração
(PONSARD; RAMON, 2022)	Automotivo, Ferroviário, Logística, Telecomunicações	teste de unidade, testes Ponta-a$_{p}onta, testedeaceitac\tilde{a}o$
(TANZIL et al., 2023)	Indústria	teste de unidade,Teste de integração, E2E
(HILTON et al., 2017)	Telecomunicações	teste de unidades, Teste de UI, teste de aceitação
(LWAKATARE et al., 2019)	Indústria	Teste de regressão
(LEPP¨ANEN et al., 2015)	Comunicação	teste de unidade, teste de componentes, teste de sistema, teste funcionais, testes de regressão
(PIANINI; NERI, 2021)	Editora	teste de unidade
(WANG; PYH¨AJ¨ARVI; M¨ANTYL¨A, 2020)	Engenharia de Software	teste de unidade, testes exploratórios , teste de aceitação
(ROUF et al., 2021)	Serviços	teste de unidade, testes de integração
(M¨AKINEN et al., 2016)	Indústria	teste de unidade, tesste de aceitação
(CLAPS; SVENSSON; AURUM, 2015)	Tecnologia	teste de unidade,testes funcionais, teste de regressão,

Continued on next page

Artigo	Domínio de Aplicação	Técnicas e Critérios de Teste
(HAGHIGHATKHAH et al., 2018)	Tecnologia	o teste de unidade, testes de integração, testes funcionais e testes A/B
(STURM; POLLARD; CRAIG, 2017)	Indústria	teste de unidade, Teste A/B, revisão de código, teste de regressão
(SNEED; VERHOEF, 2019)	Tecnologia	Testes de integração, Teste de sistema, end-to-end
(WANG et al., 2022)	Tecnologia	teste de unidades, teste de aceitação, teste de regressão
(RUBERT; FARIAS, 2022)	Educação	teste de unidade, Teste de gradiente
(BATRA; JATAIN, 2020)	Tecnologia	Testes de integração e Teste de regressão
(PAULE; D¨ULLMANN; HOORN, 2019)	Segurança de Software	teste de unidade
(VIRMANI, 2015)	Tecnologia	Testes A/B
(RAHMAN et al., 2015)	Tecnologia	Teste Mann-Whitney
(FEITELSON; FRACHTENBERG; BECK, 2013)	Tecnologia	Teste A/B, testes unitarios
(LACOSTE, 2009)	Tecnologia	teste de unidade, teste de aplicação
(STOLBERG, 2009)	Indústria	teste de unidades
(MEMON et al., 2017)	Tecnologia	Teste de cobertura de código
(ELBAUM; ROTHERMEL; PENIX, 2014)	Tecnologia	Testes de aceitação
(ERICH; AMRIT; DANEVA, 2017)	Tecnologia	teste de unidades, integração
(SHAHIN; BABAR, 2020)	Indústria	teste de unidade
(OLSSON; ALAHYARI; BOSCH, 2012)	Tecnologia	teste de unidade, Testes de integração, testes exploratórios, Teste de performance, Testes de aceitação
(NEELY; STOLT, 2013)	Tecnologia	TDD, teste de unidade
(RAUSCH et al., 2017)	Tecnologia	Teste de cenário do cliente, teste de desempenho, teste de competência
(D¨ULLMANN; PAULE; HOORN, 2018)	Segurança de Software	teste de unidade
(RODRIGUES et al., 2017)	Tecnologia	teste de unidade, teste de não regressão
(MELLADO et al., 2010)	Geoprocessamento	teste de unidade, testes de aceitação, teste de regressão

Continued on next page

Artigo	Domínio de Aplicação	Técnicas e Critérios de Teste
(LUZ; PINTO; BONIFÁCIO, 2019)	Financeira	Testes unitarios, testes de componentes, testes funcionais, testes exploratórios, teste A—B, teste de Usabilidade, teste de Perfomance, teste de segurança, tests de Aceitação, teste de refressão
(LEPP¨ANEN; KILAMO; MIKKONEN, 2015)	Gerenciamento de Serviços de TI, Midia	teste de sistema, teste de regressão
(T¨UZ¨UN et al., 2019)	Governança	Teste de regressão
(RAMLER; PUTSCH¨OGL; WINKLER, 2014)	Indústria	teste de unidade, testes de integração, teste de performance, testes de aceitação, teste de configuração
(LANGE et al., 2023)	Infraestrutura de Serviços Computacionais	-
(CHEN, 2015)	Jogos	-
(LEITE et al., 2021)	Processamento de Dados	-
(STRADOWSKI; MADEYSKI, 2023)	Rede de Comunicação	-
(SONI, 2015)	Seguros	-
(HEMON-HILDGEN; ROWE; MONNIER-SENICOURT, 2020)	Serviços	-
(STÅHL; BOSCH, 2013)	Telecomunicações	-
(COLLINS; LUCENA, 2012)	Telecomunicações	-
(MARIJAN; GOTLIEB; SEN, 2013)	Telecomunicações	-
(MARIJAN; LIAAEN, 2016)	Telecomunicações	-
(MARIJAN; LIAAEN; SEN, 2018)	Videoconferência	-

Tabela 1: Tabela com Domínio da Aplicação é Testes aplicados

.2 Apêndice B: Tabela com Análise do Qualis CAPES dos artigos

Artigo	**Ano**	**Jornal/booktitle**	**Publicação**	**Fator de impacto QUALIS-Jornal**	**Quartil**	**factor impacto jornal**	**H-Index**
M¨akinen et al. (2016)	2016	Information and Software Technology	Elsevier	A2	Q1	3.9	116
Stradowski e Madeyski (2023)	2023	Information and Software Technology	Elsevier	A1	Q1	3.9	116
Baron e Louis (2023)	2023	Computers in Industry	Elsevier	A1	Q1	10	117
Lange et al. (2023)	2023	Procedia Computer Science	Elsevier	C	Q1	2.56	109
Luz, Pinto e Bonifácio (2019)	2019	Journal of Systems and Software	Elsevier	A2	Q1	3.5	123
Claps, Svensson e Aurum (2015)	2015	Information and Software technology	Elsevier	A2	Q1	3.9	116
Haghighatkhah et al. (2018)	2018	Journal of Systems and Software	Elsevier	A2	Q1	3.9	123
Bruneliere et al. (2022)	2021	2021 24th Euromicro Conference on Digital System Design (DSD)	Elsevier	B1	-	0.9	-
Sturm, Pollard e Craig (2017)	2017	Application Performance Management (APM) in the Digital Enterprise	Morgan Kaufmann	-	-	-	-
Sneed e Verhoef (2019)	2019	Journal of Systems and Software	Elsevier	A2	Q1	4.5	123
Tanzil et al. (2023)	2023	Information and Software Technology	Elsevier	A1	Q1	5.34	116
Leite et al. (2021)	2021	Information and Software Technology	Elsevier	A2	Q1	5.03	116

Continua na próxima página

Artigo	Ano	Jornal/booktitle	Publicação	Fator de impacto QUALIS-Jornal	Quartil	factor impacto jornal	H-Index
Wang et al. (2022)	2022	Journal of Systems and Software	Elsevier	A1	Q1	4.85	123
Baron e Louis (2021)	2021	Computers in Industry	Elsevier	-	Q1	13.21	117
Pradhan e Nanniyur (2021)	2021	Journal of Systems and Software	Elsevier	A2	Q1	4.57	123
T¨uz¨un et al. (2019)	2019	Journal of Systems and Software	Elsevier	A2	Q1	4.5	123
Rubert e Farias (2022)	2022	Computer Standards & Interfaces	Elsevier	A1	Q1	6.28	71
Soni (2015)	2015	2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)	IEEE Xplore	B2	-	-	7
Marijan, Liaaen e Sen (2018)	2018	2018 IEEE 42nd annual computer software and applications conference (COMPSAC)	IEEE Xplore	A2	-	-	54
Batra e Jatain (2020)	2020	2020 International Conference on Computational Performance Evaluation (ComPE)	IEEE Xplore	B1	-	8.423	-
Ponsard e Ramon (2022)	2022	Proceedings of the Fourth International Workshop on Bots in Software Engineering	IEEE Xplore	-	-	-	-

Continua na próxima página

Artigo	Ano	Jornal/booktitle	Publicação	Fator de impacto QUALIS-Jornal	Quartil	factor impacto jornal	H-Index
Pianini e Neri (2021)	2021	2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)	IEEE Xplore	A2	-	4.9	-
Paule, D¨ullmann e Hoorn (2019)	2019	2019 IEEE international conference on software architecture companion (ICSA-C)	IEEE Xplore	B1	-	-	10
Ståhl e Bosch (2013)	2013	The 12th iasted international conference on software engineering,(innsbruck, austria, 2013)	-	-	-	-	-
Virmani (2015)	2015	Fifth international conference on the innovative computing technology (intech 2015)	IEEE	A4	-	-	6
Rahman et al. (2015)	2015	2015 Agile Conference	IEEE	-	-	-	-
Lepp¨anen, Kilamo e Mikkonen (2015)	2015	2015 IEEE/ACM 2nd International Workshop on Rapid Continuous Software Engineering	IEEE	-	-	-	-
Mellado et al. (2010)	2010	2010 Seventh International Conference on Information Technology: New Generations	IEEE	B1	Q2	6.35	46

Continua na próxima página

Artigo	Ano	Jornal/booktitle	Publicação	Fator de impacto QUALIS-Jornal	Quartil	factor impacto jornal	H-Index
Feitelson, Frachtenberg e Beck (2013)	2013	IEEE Internet Computing	IEEE	A1	Q1	3.18	120
Lacoste (2009)	2009	2009 agile conference	IEEE	A3	-	-	14
Stolberg (2009)	2009	2009 agile conference	IEEE	A3	-	-	14
Memon et al. (2017)	2017	2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)	IEEE	A1	-	-	14
Elbaum, Rothermel e Penix (2014)	2014	Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering	-	-	-	-	60
Spieker et al. (2017)	2017	Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis	-	-	-	-	18
Erich, Amrit e Daneva (2017)	2017	Journal of software: Evolution and Process	Wiley Online Library	-	Q3	1.6	35
Lwakatare et al. (2019)	2019	Information and Software Technology	Elsevier	A2	Q1	4.68	116
Chen (2015)	2015	IEEE software	IEEE	A2	Q2	1.82	121

Continua na próxima página

Artigo	Ano	Jornal/booktitle	Publicação	Fator de impacto QUALIS-Jornal	Quartil	factor impacto jornal	H-Index
Shahin et al. (2017)	2017	Proceedings of the 21st international conference on evaluation and assessment in software engineering	-	A1	-	3.57	166
Siqueira et al. (2018)	2018	IEEE Software	IEEE	A2	Q2	3.23	121
Shahin e Babar (2020)	2020	Proceedings of the International Conference on Software and System Processes	-	-	-	-	7
Lepp¨anen et al. (2015)	2015	Ieee software	IEEE	A2	Q2	1.82	121
Olsson, Alahyari e Bosch (2012)	2012	2012 38th euromicro conference on software engineering and advanced applications	IEEE	A4	-	-	13
Neely e Stolt (2013)	2013	2013 Agile Conference	IEEE	A3	-	-	-
Collins e Lucena (2012)	2012	2012 7th International Workshop on Automation of Software Test (AST)	IEEE	-	-	-	10
Hilton et al. (2017)	2017	Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering	-	-	-	-	18

Continua na próxima página

Artigo	Ano	Jornal/booktitle	Publicação	Fator de impacto QUALIS-Jornal	Quartil	factor impacto jornal	H-Index
Ramler, Putsch¨ogl e Winkler (2014)	2014	Proceedings of the 1st International Workshop on Modern Software Engineering Methods for Industrial Automation	-	-	-	-	5
Rausch et al. (2017)	2017	2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)	IEEE	-	-	-	-
Wang, Pyh¨aj¨arvi e M¨antyl¨a (2020)	2020	2020 ieee international conference on software testing, verification and validation workshops (icstw)	IEEE	-	-	-	-
D¨ullmann, Paule e Hoorn (2018)	2018	Proceedings of the 4th International Workshop on Rapid Continuous Software Engineering	-	-	-	-	-
V¨ost e Wagner (2016)	2016	arXiv preprint arXiv:1612.04139	-	-	-	-	-
Marijan, Gotlieb e Sen (2013)	2013	2013 IEEE International Conference on Software Maintenance	IEEE	-	-	-	68
Marijan e Liaaen (2016)	2016	2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)	IEEE	A2	-	-	20

Continua na próxima página

Artigo	Ano	Jornal/booktitle	Publicação	Fator de impacto QUALIS-Jornal	Quartil	factor impacto jornal	H-Index
Rodrigues et al. (2017)	2017	Anais da I Escola Regional de Engenharia de Software	SBC	-	-	-	-
Hemon-Hildgen, Rowe e Monnier-Senicourt (2020)	2020	European journal of information systems	Taylor & Francis	-	Q1	4.47	119
Rouf et al. (2021)	2021	Proceedings of the ACM/SPEC International Conference on Performance Engineering	-	A2	-	-	-

Tabela 2: Tabela de classificação Qualis - CAPES