UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE ENGENHARIA DE SOFTWARE

GUILHERME BUZATTO DONAT

ANÁLISE DOS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE NAS EMPRESAS DE TECNOLOGIA: PRÁTICAS, DESAFIOS E OPORTUNIDADES DE MELHORIA

RUSSAS

2025

GUILHERME BUZATTO DONAT

ANÁLISE DOS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE NAS EMPRESAS DE TECNOLOGIA: PRÁTICAS, DESAFIOS E OPORTUNIDADES DE MELHORIA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Prof. Dra. Jacilane de Holanda Rabelo

RUSSAS

2025

GUILHERME BUZATTO DONAT

ANÁLISE DOS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE NAS EMPRESAS DE TECNOLOGIA: PRÁTICAS, DESAFIOS E OPORTUNIDADES DE MELHORIA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em 07/03/2025.

BANCA EXAMINADORA

Prof. Dra. Jacilane de Holanda Rabelo (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dra. Patrícia freitas campos de vasconcelos
Universidade Federal do Ceará (UFC)

Prof. Dr. Cenez araújo de rezende
Universidade Federal do Ceará (UFC)

À minha família por todo o apoio nessa jornada

“Termine cada dia e esteja contente com ele. Você fez o que pôde. Alguns enganos e tolices se infiltraram indubitavelmente; esqueça-os tão logo você consiga. Amanhã é um novo dia; comece-o bem e serenamente com um espírito elevado demais para ser incomodado pelas tolices do passado.”

(Ralph Waldo Emerson)

RESUMO

O desenvolvimento de software é um fator essencial para a inovação e competitividade das empresas de tecnologia, sendo diretamente influenciado pelas metodologias adotadas. No entanto, a escolha e aplicação dessas metodologias enfrentam desafios relacionados à complexidade dos projetos, adaptação das equipes e necessidade de entrega contínua de valor. Diante desse cenário, este trabalho tem como objetivo analisar os processos de desenvolvimento de software em empresas de tecnologia, investigando as práticas mais utilizadas, os desafios enfrentados e as oportunidades de melhoria. Para isso, foram comparadas abordagens tradicionais e ágeis, como o modelo em cascata, Scrum e Kanban. A pesquisa baseou-se na aplicação de questionários a profissionais da área de tecnologia, permitindo identificar que a adoção de metodologias ágeis tem sido predominante, embora muitas empresas ainda combinem abordagens tradicionais para atender a demandas específicas. Os principais desafios encontrados foram a resistência à mudança, dificuldades na comunicação entre equipes e a falta de capacitação adequada. Com base nos dados coletados, o estudo sugere soluções para aprimorar os processos de desenvolvimento de software, incluindo a adoção de metodologias híbridas, investimentos em treinamento e automação de processos.

Palavras-chave: desenvolvimento de software; metodologias ágeis; scrum; kanban; desafios; melhoria contínua.

ABSTRACT

Software development is a key factor for innovation and competitiveness in technology companies, being directly influenced by the methodologies adopted. However, selecting and implementing these methodologies pose challenges related to project complexity, team adaptation, and the need for continuous value delivery. Given this scenario, this study aims to analyze software development processes in technology companies, investigating the most commonly used practices, the challenges faced, and opportunities for improvement. To this end, traditional and agile approaches such as the waterfall model, Scrum, and Kanban were compared. The research was based on questionnaires applied to technology professionals, identifying that agile methodologies have been predominantly adopted, although many companies still combine traditional approaches to meet specific demands. The main challenges found include resistance to change, communication difficulties between teams, and a lack of adequate training. Based on the collected data, the study suggests solutions to enhance software development processes, including the adoption of hybrid methodologies, investments in training, and process automation.

Keywords: software development; agile methodologies; scrum; kanban; challenges; continuous improvement.

SUMÁRIO

1.INTRODUÇÃO

O desenvolvimento de software é um componente vital na era digital atual, impulsionando inovações e soluções em diversos setores (Sommerville, I. 2016). A tecnologia avança rapidamente, e as empresas de tecnologia estão constantemente adaptando seus processos de desenvolvimento para atender às demandas do mercado, melhorar a eficiência e garantir a qualidade dos produtos (SCHWARZ, J., LEGNE, 2020).

1.1. Contextualização

Diversas metodologias são adotadas para gerenciar projetos de software, cada uma com suas características e vantagens específicas. O Scrum é amplamente utilizado devido à sua estrutura iterativa e incremental, que facilita a adaptação às mudanças e promove entregas rápidas e frequentes (ALMEIDA; CARNEIRO, 2023). Kanban, por outro lado, é valorizado por sua abordagem visual e flexível, que permite um fluxo contínuo de trabalho e adaptação constante, sendo particularmente útil em ambientes onde as prioridades mudam com frequência (VERWIJS, C., & RUSSO, D., 2023).

Além dessas metodologias ágeis, há também o Scrumban, que combina elementos do Scrum e do Kanban, proporcionando uma abordagem híbrida que une as práticas estruturadas do Scrum com a flexibilidade do Kanban. Isso permite otimizar a eficiência e a capacidade de resposta das equipes de desenvolvimento de software, especialmente em projetos com requisitos em constante evolução (REIS, 2021).

Metodologias tradicionais, como o modelo em cascata, seguem uma abordagem linear e sequencial de desenvolvimento. Embora sejam menos flexíveis e adaptáveis a mudanças, são amplamente utilizadas em projetos com requisitos bem definidos desde o início, como em projetos governamentais, onde a previsibilidade e a documentação rigorosa são essenciais (GABOROV MAJA *et al*. 2021).

Cada uma dessas abordagens oferece benefícios distintos e pode ser mais adequada dependendo da natureza do projeto, do ambiente organizacional e das preferências da equipe. Por exemplo, enquanto metodologias ágeis como Scrum e Kanban permitem maior adaptabilidade e resposta rápida às mudanças, métodos como o cascata oferecem uma estrutura mais rígida e previsível, que pode ser ideal para projetos onde as mudanças são menos frequentes e os requisitos são claros desde o início.

As organizações estão continuamente ajustando suas práticas de desenvolvimento de software para manter a competitividade e responder rapidamente às mudanças nas necessidades dos clientes. Isso inclui a adoção de práticas que combinam aspectos de diferentes metodologias para criar um ambiente de trabalho equilibrado e eficaz (BANIJAMALI *et al.* 2016). No entanto, desafios significativos ainda permanecem, como a complexidade dos projetos, a necessidade de integração contínua, entrega rápida e a gestão de equipes multidisciplinares (SOŃTA-DRĄCZKOWSKA E KROGULEC, 2024).

1.2. Problematização

A implementação de processos de desenvolvimento de software nem sempre ocorre de forma suave e eficiente. Muitas empresas relatam dificuldades na aplicação de metodologias ágeis, problemas de comunicação entre equipes, atrasos nos cronogramas, e desafios na manutenção da qualidade do software. De acordo com (SOŃTA-DRĄCZKOWSKA E KROGULEC, 2024), a transição para metodologias ágeis pode ser problemática devido a resistências culturais e à falta de treinamento adequado. Além disso, (REGINALDO E SANTOS, 2020) destacam que a comunicação entre equipes é frequentemente citada como uma das principais dificuldades, impactando diretamente na coordenação e na integração de tarefas.

Essas dificuldades podem resultar em defasagens no desenvolvimento, comprometendo a entrega de produtos e serviços de alta qualidade. Por exemplo, uma pesquisa realizada por (DIKERT, PAASIVAARA E LASSENIUS, 2016) revelou que atrasos nos cronogramas são uma consequência comum da má implementação de práticas ágeis, afetando negativamente a satisfação do cliente e a qualidade do software entregue.

Diante desse cenário, surge a necessidade de investigar como as empresas de tecnologia estão aplicando seus processos de desenvolvimento de software, quais são as principais dificuldades enfrentadas, e quais pontos positivos têm sido percebidos. Através dessa análise, será possível identificar os principais desafios e sugerir melhorias que possam contribuir para a eficácia dos processos de desenvolvimento. (BEHUTIYE *et al*. 2022) sugerem que a identificação de pontos de melhoria e a adoção de práticas de gerenciamento de projetos mais robustos podem levar a um aumento significativo na eficiência e na qualidade dos produtos desenvolvidos.

1.3. Justificativa

A relevância deste estudo está na possibilidade de fornecer uma visão detalhada sobre os processos de desenvolvimento de software utilizados nas empresas de tecnologia, identificando práticas eficazes e áreas que necessitam de aprimoramento. De acordo com (BEHUTIYE *et al*. 2022), uma compreensão profunda dos processos de desenvolvimento de software pode levar a melhorias significativas na eficiência operacional e na qualidade do produto. Compreender melhor os desafios e os pontos positivos desses processos permitirá que as empresas adotem estratégias mais eficazes, melhorando a eficiência e a qualidade dos seus produtos.

Além disso, este trabalho contribuirá para a literatura acadêmica e prática na área de engenharia de software, oferecendo dados empíricos que poderão ser utilizados por outros pesquisadores e profissionais da área para desenvolver novas abordagens e soluções. Como apontado por (REGINALDO E SANTOS 2020), a investigação contínua sobre práticas e desafios em desenvolvimento de software é essencial para impulsionar a inovação e a competitividade no setor de tecnologia.

2. OBJETIVOS

O desenvolvimento de software envolve uma série de desafios e variáveis que impactam diretamente a eficiência dos processos adotados pelas empresas. Esta seção apresenta os objetivos deste estudo, detalhando a intenção de investigar as metodologias utilizadas, as dificuldades enfrentadas e as oportunidades de aprimoramento nos processos de desenvolvimento de software.

2.1. Objetivo Geral

Analisar os processos de desenvolvimento de software utilizados nas empresas de tecnologia, identificando práticas, desafios e oportunidades de melhoria, com o intuito de proporcionar insights e sugestões que contribuam para a eficácia desses processos.

2.2. Objetivos Específicos

- Identificar as metodologias de desenvolvimento de software mais utilizadas nas empresas de tecnologia investigadas.

- Analisar as principais dificuldades e defasagens encontradas na aplicação dessas metodologias.
- Avaliar os pontos positivos e benefícios percebidos pelos profissionais em relação aos processos utilizados.

3. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados os principais conceitos que fundamentam este estudo, incluindo as metodologias de desenvolvimento de software mais utilizadas, seus desafios e oportunidades de melhoria. Primeiramente, discute-se a evolução das metodologias, destacando abordagens tradicionais e ágeis. Em seguida, são exploradas as dificuldades enfrentadas pelas empresas na adoção dessas metodologias e as estratégias utilizadas para aprimorar os processos. Essa revisão teórica é essencial para embasar a análise dos dados coletados, fornecendo um referencial para a interpretação dos resultados da pesquisa.

3.1. Introdução à Engenharia de Software

3.1.1. Definição e Importância da Engenharia de Software

A Engenharia de Software é uma disciplina da engenharia que se ocupa do desenvolvimento, operação e manutenção de software de forma sistemática e quantificável. Ela envolve a aplicação de teorias, métodos e ferramentas para a criação de software de alta qualidade que atenda às necessidades dos usuários e esteja dentro do prazo e do orçamento previstos (SOMMERVILLE, 2016). A importância da Engenharia de Software reside na sua capacidade de abordar a complexidade crescente dos sistemas de software, garantir a qualidade e a segurança do software, e melhorar a eficiência e a produtividade dos processos de desenvolvimento (PRESSMAN & MAXIM, 2020).

A definição de Engenharia de Software pode ser expandida para incluir diversos subcampos, como o desenvolvimento de software, a garantia de qualidade, a gestão de projetos e a engenharia de requisitos. A aplicação adequada de práticas de Engenharia de Software é crucial para evitar problemas comuns, como atrasos no cronograma, estouro de orçamento e falhas no sistema. Segundo (ABRAHAMSSON, OZA E SIPONEN, 2019), a engenharia de software bem executada pode reduzir significativamente os riscos associados

ao desenvolvimento de software, garantindo que o produto final seja confiável, seguro e eficiente.

3.1.2. Histórico e Evolução dos Processos de Desenvolvimento de Software

O desenvolvimento de software tem uma história rica que se estende por várias décadas, evoluindo através de diversas metodologias e práticas. Nos anos 1950 e 1960, o desenvolvimento de software era caracterizado por abordagens ad hoc, onde a codificação e a depuração eram as principais atividades. Com o aumento da complexidade dos sistemas, emergiu a necessidade de uma abordagem mais estruturada, levando ao desenvolvimento do Modelo Cascata (Waterfall) na década de 1970. Este modelo, introduzido por (ROYCE, 1970), propunha um processo sequencial de fases distintas, incluindo requisitos, design, implementação, verificação e manutenção.

Na década de 1980, surgiram críticas ao Modelo Cascata devido à sua rigidez e à incapacidade de lidar com mudanças nos requisitos durante o ciclo de desenvolvimento. Isso levou ao desenvolvimento de modelos mais iterativos e incrementais, como o Modelo Espiral proposto por (BOEHM, 1988). O Modelo Espiral enfatizava a análise de riscos e a iteração contínua, permitindo uma maior flexibilidade e adaptação às mudanças.

Nos anos 1990 e 2000, a indústria de software testemunhou o advento das metodologias ágeis, como o Scrum e o Extreme Programming (XP). Estas metodologias focam em entregas rápidas, colaboração constante com o cliente e a capacidade de responder rapidamente às mudanças. De acordo com (HOHL *et al.* 2018), as metodologias ágeis promovem a comunicação contínua, feedback frequente e entregas incrementais, resultando em uma maior satisfação do cliente e uma melhor qualidade do produto.

A evolução dos processos de desenvolvimento de software reflete uma busca contínua por métodos que possam lidar eficientemente com a complexidade e a dinâmica do desenvolvimento de software. Atualmente, a integração de práticas DevOps e a adoção de metodologias híbridas que combinam elementos ágeis e tradicionais estão se tornando cada vez mais comuns. Estas abordagens híbridas visam equilibrar a flexibilidade e a estrutura, permitindo que as equipes de desenvolvimento entreguem softwares de alta qualidade de forma mais eficiente e eficaz (FITZGERALD *et al.* 2017).

3.2. Metodologias de Desenvolvimento de Software

3.2.1. Metodologias Tradicionais

As metodologias tradicionais de desenvolvimento de software, também conhecidas como metodologias preditivas ou sequenciais, são caracterizadas por uma abordagem estruturada e linear, onde cada fase do desenvolvimento é rigorosamente definida e seguida de maneira sequencial. Estas metodologias são amplamente utilizadas em projetos onde os requisitos são bem compreendidos e menos sujeitos a mudanças ao longo do tempo.

3.2.1.1. Modelo Cascata

O Modelo Cascata, ou Waterfall, é uma das metodologias mais antigas e conhecidas no desenvolvimento de software. Como já descrito em tópicos anteriores, este modelo segue uma abordagem linear e sequencial, onde cada fase deve ser concluída antes que a próxima comece. As fases típicas do Modelo Cascata incluem requisitos, design, implementação, verificação e manutenção (ROYCE, 1970).

A principal vantagem do Modelo Cascata é sua simplicidade e clareza. Cada fase tem objetivos específicos e bem definidos, o que facilita o planejamento e a gestão do projeto. No entanto, sua principal desvantagem é a falta de flexibilidade. Mudanças nos requisitos podem ser difíceis e custosas de implementar após o início do desenvolvimento, tornando este modelo menos adequado para projetos onde os requisitos são incertos ou sujeitos a mudanças frequentes (PRESSMAN & MAXIM, 2020).

3.2.1.2. Modelo em V

O Modelo em V, também conhecido como Modelo de Verificação e Validação, é uma extensão do Modelo Cascata que enfatiza a verificação e validação em cada fase do desenvolvimento. Cada fase de desenvolvimento tem uma fase de teste correspondente, criando uma forma de "V" quando representado graficamente. As fases incluem planejamento de requisitos, design de sistema, design detalhado, implementação, testes de unidade, testes de integração, testes de sistema e testes de aceitação (BALAJI & MURUGAIYAN, 2012).

O Modelo em V oferece a vantagem de identificar e corrigir defeitos em etapas iniciais do desenvolvimento, o que pode reduzir custos e melhorar a qualidade do produto final. No

entanto, assim como o Modelo Cascata, ele é menos flexível e pode ser inadequado para projetos com requisitos voláteis (SOMMERVILLE, 2022).

3.2.1.3. Outros modelos

Além dos modelos Cascata e em V, existem outras metodologias tradicionais que têm sido utilizadas no desenvolvimento de software. O Modelo Espiral, também já comentado em tópicos anteriores, combina elementos do Modelo Cascata com uma abordagem iterativa, permitindo a revisão e refinamento contínuo dos requisitos e soluções. Este modelo é particularmente útil para projetos complexos e de grande escala, onde os riscos precisam ser gerenciados cuidadosamente.

Outro modelo é o Modelo Incremental, que divide o desenvolvimento em pequenas partes incrementais, permitindo que partes funcionais do software sejam entregues e avaliadas ao longo do tempo. Cada incremento adiciona funcionalidades ao sistema, o que facilita a incorporação de feedback e mudanças nos requisitos (LARMAN & BASILI, 2003).

3.2.1.4. Comparação das Metodologias Tradicionais

A comparação das metodologias tradicionais de desenvolvimento de software revela que cada uma tem suas próprias vantagens e desvantagens, dependendo do contexto do projeto. O Modelo Cascata é ideal para projetos com requisitos bem definidos e estáveis, proporcionando uma abordagem clara e linear ao desenvolvimento. No entanto, sua rigidez torna difícil a adaptação a mudanças, o que pode ser problemático em projetos mais dinâmicos (PRESSMAN & MAXIM, 2020).

O Modelo em V adiciona uma camada de verificação e validação, melhorando a qualidade do produto final ao identificar defeitos em estágios iniciais. No entanto, como o Modelo Cascata, ele é menos adequado para projetos com requisitos que podem evoluir ao longo do tempo (SOMMERVILLE, 2022).

O Modelo Espiral, com sua ênfase na gestão de riscos e iteração contínua, oferece maior flexibilidade e adaptabilidade, tornando-o apropriado para projetos complexos e incertos. O Modelo Incremental, por sua vez, permite entregas frequentes e a incorporação de feedback contínuo, mas pode exigir uma gestão mais intensiva para coordenar as diferentes partes do sistema (BOEHM, 1988; LARMAN & BASILI, 2003).

3.3. Metodologias Ágeis

3.3.1. Manifesto Ágil e seus Princípios

O Manifesto Ágil, criado em 2001 por um grupo de desenvolvedores de software, estabelece uma nova abordagem para o desenvolvimento de software, centrada em valores e princípios que promovem a agilidade, a colaboração e a adaptação às mudanças. O Manifesto Ágil propõe quatro valores fundamentais:

- Indivíduos e interações mais que processos e ferramentas.
- Software funcionando mais que documentação abrangente.
- Colaboração com o cliente mais que negociação de contratos.
- Responder a mudanças mais que seguir um plano.

Além desses valores, o Manifesto Ágil estabelece doze princípios que orientam as práticas ágeis, incluindo a entrega contínua de software funcional, a aceitação de mudanças nos requisitos, o envolvimento diário de negócios e desenvolvedores, e a importância de equipes motivadas e auto-organizadas (BECK *et al.* 2001).

3.3.2. Principais Metodologias Ágeis

As metodologias ágeis são diversas e variam em suas abordagens e práticas específicas, mas todas compartilham os valores e princípios do Manifesto Ágil, entre elas, é importante falar das metodologias que serão apresentadas nos seguintes tópicos.

3.3.2.1. Scrum

Scrum é uma das metodologias ágeis mais populares, caracterizada por ciclos de desenvolvimento iterativos chamados sprints, que geralmente duram de duas a quatro semanas. O Scrum enfatiza a colaboração, a responsabilidade e o progresso contínuo através de reuniões diárias (daily stand-ups), revisões de sprint e retrospectivas. Os papéis chave no Scrum incluem o Product Owner, o Scrum Master e o Time de Desenvolvimento. O Product Owner é responsável por maximizar o valor do produto, o Scrum Master facilita o processo e remove impedimentos, e o Time de Desenvolvimento é auto-organizado e multifuncional (SCHWABER & SUTHERLAND, 2020).

3.3.2.2. Kanban

Kanban é uma metodologia ágil que se concentra na visualização do fluxo de trabalho, na limitação do trabalho em progresso (WIP) e na melhoria contínua. Através do uso de um quadro Kanban, as equipes podem visualizar as tarefas em diferentes estágios de conclusão,

identificar gargalos e otimizar o fluxo de trabalho. Ao contrário do Scrum, Kanban não prescreve papéis específicos ou sprints, proporcionando uma abordagem mais flexível e adaptável às necessidades da equipe (ANDERSON, 2010).

3.3.2.3. Extreme Programming (XP)

Extreme Programming (XP) é uma metodologia ágil que enfatiza a melhoria contínua, a comunicação e a qualidade do software através de práticas como programação em par, desenvolvimento orientado a testes (TDD), integrações contínuas e revisões de código. XP valoriza a simplicidade no design e incentiva a resposta rápida às mudanças nos requisitos. A abordagem iterativa e incremental de XP permite que as equipes entreguem software funcional de forma frequente e com alta qualidade (BECK, 2004).

3.3.2.4. Scrumban

Scrumban é uma metodologia híbrida que combina elementos do Scrum e do Kanban, aproveitando a estrutura iterativa do Scrum e a flexibilidade e visualização do fluxo de trabalho do Kanban. Esta abordagem é útil para equipes que desejam a organização do Scrum mas com a flexibilidade adicional para adaptar o fluxo de trabalho conforme necessário. Scrumban permite que as equipes se beneficiem das reuniões e papéis do Scrum enquanto usam práticas de Kanban para gerenciar o fluxo de trabalho e limitar o WIP (LADAS, 2009).

3.3.3. Comparação das Metodologias Ágeis

A comparação das metodologias ágeis revela que cada uma oferece vantagens distintas e pode ser mais adequada para diferentes tipos de projetos e equipes.

- **Scrum** é ideal para equipes que preferem uma estrutura bem definida e ciclos de trabalho regulares, proporcionando clareza e ritmo através de sprints e reuniões regulares (SCHWABER & SUTHERLAND, 2020).
- **Kanban** é melhor para equipes que necessitam de maior flexibilidade e um foco contínuo na melhoria do fluxo de trabalho, sendo menos prescritivo e mais adaptável a mudanças imediatas (ANDERSON, 2010).
- **Extreme Programming (XP)** é excelente para equipes que valorizam a alta qualidade do código e a comunicação constante, com práticas rigorosas de desenvolvimento e teste (BECK, 2004).
- **Scrumban** combina os benefícios de Scrum e Kanban, sendo uma boa escolha para equipes que desejam a estrutura do Scrum com a flexibilidade do Kanban,

especialmente em ambientes dinâmicos onde os requisitos mudam frequentemente (LADAS, 2009).

Cada metodologia ágil tem suas próprias práticas e enfoques, e a escolha da metodologia mais adequada depende das necessidades específicas do projeto e da equipe.

3.4. Comparação entre Metodologias Tradicionais e Ágeis

A escolha entre metodologias tradicionais e ágeis no desenvolvimento de software depende de diversos fatores, incluindo a natureza do projeto, os requisitos dos clientes e as características da equipe de desenvolvimento. É importante entender como cada metodologia pode ser aplicada em diferentes contextos para maximizar a eficiência e a qualidade do desenvolvimento.

3.4.1. Vantagens e Desvantagens

As metodologias tradicionais, como o Modelo Cascata e o Modelo em V, são conhecidas por sua estruturação rígida e sequencial. Essas metodologias são frequentemente escolhidas por sua previsibilidade e facilidade de gestão, especialmente em projetos onde os requisitos são bem definidos desde o início. Entre as principais vantagens dessas metodologias estão:

- **Clareza e Planejamento**: As fases bem definidas permitem um planejamento detalhado e a alocação de recursos de maneira eficiente (PRESSMAN & MAXIM, 2020).
- **Documentação Completa**: A ênfase na documentação garante que todos os aspectos do projeto sejam bem documentados, facilitando a manutenção e a transferência de conhecimento (SOMMERVILLE, 2022).

Entretanto, as desvantagens dessas metodologias incluem:

- **Rigidez**: A falta de flexibilidade dificulta a adaptação a mudanças nos requisitos ao longo do projeto (Balaji & Murugaiyan, 2012).
- **Entrega Tardia**: A entrega do produto funcional ocorre apenas no final do ciclo de desenvolvimento, o que pode atrasar o feedback dos usuários e a correção de problemas emergentes (Royce, 1970).

Por outro lado, as metodologias ágeis, como Scrum, Kanban e XP, oferecem uma abordagem mais flexível e iterativa. As vantagens dessas metodologias são:

- **Adaptabilidade**: A capacidade de responder rapidamente a mudanças nos requisitos e prioridades do cliente (BECK *et al.* 2001).
- **Entrega Contínua**: A entrega frequente de incrementos de software funcional permite feedback constante e ajustes rápidos (SCHWABER & SUTHERLAND, 2020).

As desvantagens das metodologias ágeis incluem:

- **Documentação Menor:** A menor ênfase na documentação pode dificultar a manutenção e a escalabilidade do projeto a longo prazo (FOWLER & HIGHSMITH, 2001).
- **Dependência da Equipe:** O sucesso das metodologias ágeis depende fortemente da capacidade e da colaboração da equipe de desenvolvimento, o que pode ser um desafio em equipes menos experientes ou menos coesas (DINGSØYR *et al.* 2012).

3.4.2. Estudos de Caso e Análises Comparativas

Para entender melhor as vantagens e desvantagens das metodologias tradicionais e ágeis, é útil examinar estudos de caso específicos e análises comparativas que destacam a aplicação prática dessas metodologias em diferentes cenários.

3.4.3. Contextos de Aplicação

A escolha entre metodologias tradicionais e ágeis deve considerar o contexto específico do projeto. Ambientes de desenvolvimento variam significativamente, e a eficácia de uma metodologia pode depender de fatores como a estabilidade dos requisitos, o tamanho da equipe e a cultura organizacional. Metodologias tradicionais são mais apropriadas para:

- **Projetos de Grande Escala**: Onde o planejamento detalhado e a documentação completa são essenciais (SOMMERVILLE, 2022).
- **Ambientes Regulamentados**: Onde a conformidade com normas e regulamentos exige documentação rigorosa e processos bem definidos (FITZGERALD *et al.* 2017).

Por outro lado, metodologias ágeis são mais indicadas para:

- **Projetos de Desenvolvimento Rápido**: Onde a rapidez de entrega e a capacidade de adaptação são críticas (BECK *et al.* 2001).
- **Equipes Colaborativas e Auto-organizadas**: Que podem aproveitar a flexibilidade e a comunicação contínua proporcionadas pelas metodologias ágeis (SCHWABER & SUTHERLAND, 2020).

Em suma, a decisão entre metodologias tradicionais e ágeis deve ser baseada em uma avaliação cuidadosa das necessidades do projeto, das capacidades da equipe e do ambiente de desenvolvimento. Cada abordagem oferece vantagens distintas e pode ser mais ou menos adequada dependendo do contexto específico.

3.5. Práticas de Desenvolvimento de Software nas Empresas de Tecnologia

3.5.1. Adoção de Metodologias

A adoção de metodologias de desenvolvimento de software nas empresas de tecnologia é um processo complexo e multifacetado. Envolve não apenas a escolha de uma metodologia específica, mas também a adaptação dessa metodologia ao contexto e às necessidades da organização. A eficácia da metodologia escolhida pode impactar diretamente a qualidade do produto, a satisfação do cliente e a eficiência dos processos internos.

3.5.1.1. Fatores que Influenciam a Escolha da Metodologia

Vários fatores influenciam a escolha da metodologia de desenvolvimento de software em uma organização. Entre esses fatores, destacam-se:

- **Natureza do Projeto**: Projetos com requisitos bem definidos e estáveis tendem a se beneficiar de metodologias tradicionais, como o Modelo Cascata, enquanto projetos com requisitos dinâmicos e voláteis são mais adequados para metodologias ágeis PRESSMAN & MAXIM, 2020).
- **Tamanho da Equipe**: Equipes menores e altamente colaborativas podem se beneficiar mais de metodologias ágeis, como Scrum ou Kanban, que enfatizam a comunicação constante e a flexibilidade. Equipes maiores podem achar útil a estrutura e a clareza das metodologias tradicionais (SOMMERVILLE, 2022).
- **Cultura Organizacional**: A cultura da empresa desempenha um papel crucial na adoção de metodologias. Empresas com uma cultura aberta à mudança e à experimentação tendem a adotar metodologias ágeis, enquanto aquelas com uma

cultura mais conservadora podem preferir metodologias tradicionais (MISRA *et al.* 2021).

- **Pressões de Mercado**: A necessidade de respostas rápidas às mudanças do mercado pode levar as empresas a adotar metodologias ágeis, que são mais adaptáveis e permitem entregas rápidas e frequentes (BECK *et al.* 2001).

3.5.1.2. Adaptação das Metodologias ao Contexto Organizacional

A implementação eficaz de uma metodologia de desenvolvimento de software muitas vezes requer a adaptação dessa metodologia ao contexto específico da organização. Isso pode envolver a combinação de práticas de diferentes metodologias ou a modificação de práticas existentes para melhor atender às necessidades da equipe e dos projetos.

- **Híbridos Metodológicos**: Muitas empresas adotam uma abordagem híbrida, combinando elementos de metodologias tradicionais e ágeis. Por exemplo, uma empresa pode usar Scrum para gerenciar o desenvolvimento iterativo, mas incorporar práticas de Kanban para gerenciar o fluxo de trabalho contínuo e a limitação do trabalho em progresso (DINGSØYR *et al.* 2012).
- **Customização de Práticas**: A customização de práticas específicas pode ser necessária para alinhar a metodologia escolhida com as práticas e processos internos da organização. Isso pode incluir a adaptação das cerimônias do Scrum, a personalização do quadro Kanban ou a modificação das práticas de Extreme Programming (XP) para melhor atender às necessidades específicas do projeto (SCHWABER & SUTHERLAND, 2020).
- **Integração de Ferramentas**: A adoção de ferramentas de suporte adequadas, como softwares de gestão de projetos, ferramentas de integração contínua e plataformas de comunicação, pode facilitar a implementação e o uso eficaz da metodologia escolhida. Essas ferramentas ajudam a automatizar processos, aumentar a transparência e melhorar a colaboração entre as equipes (FITZGERALD *et al*. 2017).

A adaptação eficaz das metodologias de desenvolvimento de software ao contexto organizacional requer uma compreensão profunda das necessidades e características específicas da organização. A flexibilidade e a disposição para experimentar e ajustar práticas são fundamentais para alcançar os melhores resultados possíveis.

3.6. Desafios no Desenvolvimento de Software

O desenvolvimento de software envolve uma série de desafios que podem impactar negativamente a eficiência, a qualidade e o sucesso dos projetos. A seguir, são discutidos alguns dos principais desafios enfrentados pelas equipes de desenvolvimento de software nas empresas de tecnologia.

3.6.1. Resistências Culturais

A resistência cultural é um desafio significativo no desenvolvimento de software, especialmente quando se introduzem novas metodologias ou práticas. A mudança de uma abordagem tradicional para uma metodologia ágil, por exemplo, pode encontrar resistência devido à inércia organizacional e à relutância em abandonar processos familiares (Schein, 2010). A resistência cultural pode manifestar-se através de:

- **Resistência à Mudança**: Equipes acostumadas a metodologias tradicionais podem resistir a novas abordagens que exigem maior colaboração e flexibilidade (WANG *et al.* 2012).
- **Falta de Adesão**: A falta de compromisso dos membros da equipe e dos gerentes pode minar a implementação eficaz de novas práticas (DIKERT, PAASIVAARA, & LASSENIUS, 2016).

3.6.2. Problemas de Comunicação entre Equipes

A comunicação eficaz é fundamental para o sucesso dos projetos de software. No entanto, problemas de comunicação entre equipes podem levar a mal-entendidos, retrabalho e atrasos. Esses problemas podem ser exacerbados por:

- **Distribuição Geográfica**: Equipes distribuídas geograficamente podem enfrentar desafios de comunicação devido a fusos horários diferentes e barreiras linguísticas (OLSON & OLSON, 2014).
- **Falta de Ferramentas Adequadas**: A ausência de ferramentas de comunicação eficazes pode dificultar a colaboração e a coordenação entre equipes (MISHRA *et al.* 2020).

3.6.3. Atrasos e Defasagens nos Cronogramas

Atrasos e defasagens nos cronogramas são desafios comuns no desenvolvimento de software, resultando frequentemente de:

- **Estimativas Inadequadas**: Estimativas imprecisas de tempo e recursos podem levar a cronogramas irrealistas (JØRGENSEN & MOLØKKEN-ØSTVOLD, 2004).
- **Mudanças nos Requisitos**: Mudanças frequentes nos requisitos podem desestabilizar o planejamento e causar atrasos ().

3.6.4. Manutenção da Qualidade do Software

Manter a qualidade do software ao longo do ciclo de vida do desenvolvimento é um desafio constante, envolvendo:

- **Garantia de Qualidade**: Implementar práticas de garantia de qualidade eficazes, como testes contínuos e revisões de código, é essencial para detectar e corrigir defeitos precocemente ().
- **Gestão da Dívida Técnica**: A dívida técnica acumulada pode comprometer a qualidade e a manutenção do software a longo prazo (KRUCHTEN, NORD, & OZKAYA, 2012).

3.6.5. Integração Contínua e Gestão de Dependências

A integração contínua (CI) e a gestão de dependências são práticas cruciais no desenvolvimento moderno de software, mas apresentam seus próprios desafios:

- **Integração de Código**: Integrar mudanças de código frequentemente pode resultar em conflitos e problemas de integração se não for gerido adequadamente (DUVALL, MATYAS, & GLOVER, 2007).
- **Gestão de Dependências**: Projetos complexos frequentemente dependem de múltiplos componentes e bibliotecas, tornando a gestão de dependências crítica para evitar incompatibilidades e falhas (LI *et al.* 2017).

3.7. Oportunidades de Melhoria nos Processos de Desenvolvimento de Software

Melhorias nos processos de desenvolvimento de software podem ser alcançadas através da adoção de práticas de melhoria contínua, capacitação e treinamento das equipes, melhoria da comunicação e colaboração, automação de processos e análise de estudos de casos de sucesso.

3.7.1. Adoção de Práticas de Melhoria Contínua

A adoção de práticas de melhoria contínua é fundamental para a evolução dos processos de desenvolvimento de software. A melhoria contínua envolve a análise regular dos processos existentes e a implementação de mudanças incrementais para otimizar a eficiência e a qualidade. Práticas como retrospectivas de sprint em metodologias ágeis, onde a equipe reflete sobre o que funcionou bem e o que pode ser melhorado, são exemplos de iniciativas de melhoria contínua (SCHWABER & SUTHERLAND, 2020). Segundo (BASS *et al.* 2018), a implementação de ciclos de feedback curtos e frequentes pode ajudar as equipes a identificar rapidamente áreas de melhoria e a ajustar suas práticas de forma ágil.

3.7.2. Capacitação e Treinamento de Equipes

A capacitação e o treinamento das equipes são cruciais para garantir que todos os membros estejam atualizados com as melhores práticas, ferramentas e tecnologias. Investir em programas de treinamento contínuo pode melhorar significativamente a competência técnica e a moral da equipe. De acordo com (DIAS *et al.* 2021), programas de capacitação bem estruturados aumentam a eficiência e reduzem erros, resultando em uma qualidade superior do produto. Além disso, a formação cruzada (cross-training) pode aumentar a flexibilidade da equipe, permitindo que os membros assumam diferentes papéis conforme necessário (HODA & NOBLE, 2017).

3.7.3. Melhoria da Comunicação e Colaboração

A melhoria da comunicação e colaboração entre as equipes é essencial para o sucesso dos projetos de software. Ferramentas de comunicação eficazes, como Slack ou Microsoft Teams, e práticas colaborativas, como reuniões diárias e pair programming, podem aumentar a transparência e a cooperação dentro das equipes (OLSON & OLSON, 2014). Segundo (PETERSEN *et al.* 2010), a implementação de ferramentas e práticas que facilitam a comunicação constante e a resolução rápida de problemas pode reduzir significativamente os riscos de mal-entendidos e retrabalho.

3.7.4. Automação de Processos e Ferramentas

A automação de processos de desenvolvimento de software é uma oportunidade significativa para melhorar a eficiência e a qualidade. Ferramentas de integração contínua

(CI) e entrega contínua (CD), como Jenkins e GitLab CI, permitem a automação de testes, build e deploy, reduzindo o tempo e os erros associados a essas atividades (DUVALL, MATYAS, & GLOVER, 2007). Segundo (KUMAR *et al.* 2019), a automação não só acelera o processo de desenvolvimento, mas também libera os desenvolvedores para focar em tarefas mais complexas e criativas.

3.7.5. Estudo de Casos de Sucesso

Estudos de casos de sucesso podem fornecer insights valiosos sobre como outras organizações implementaram melhorias em seus processos de desenvolvimento de software. Analisar exemplos de empresas que adotaram práticas ágeis com sucesso, como Spotify e Atlassian, pode oferecer modelos e estratégias que podem ser adaptados a diferentes contextos organizacionais (KNIBERG & IVARSSON, 2012). Segundo (PATEL *et al*. 2020), estudar casos de sucesso ajuda a identificar práticas eficazes e evita a repetição de erros comuns.

3.8. Tendências e Inovações no Desenvolvimento de Software

O desenvolvimento de software está em constante evolução, impulsionado por novas tecnologias, metodologias e práticas que buscam aumentar a eficiência, a qualidade e a inovação. A seguir, são discutidas algumas das tendências e inovações mais recentes na área.

3.8.1. DevOps e sua Integração com Metodologias Ágeis

DevOps é uma prática emergente que combina o desenvolvimento de software (Dev) e as operações de TI (Ops) para melhorar a colaboração, a integração contínua e a entrega contínua de software. A integração de DevOps com metodologias ágeis é cada vez mais comum, proporcionando uma abordagem mais holística ao ciclo de vida do desenvolvimento de software (BASS, WEBER, & ZHU, 2015). DevOps enfatiza a automação de processos, a integração contínua (CI) e a entrega contínua (CD), facilitando a rápida implementação de mudanças e a redução do tempo de entrega (HUMBLE & FARLEY, 2010).

A integração de DevOps com metodologias ágeis resulta em benefícios significativos, como maior velocidade de entrega, melhor qualidade do software e maior capacidade de

resposta às mudanças nos requisitos dos clientes. Segundo (KIM, HUMBLE E DEBOIS, 2016), a combinação dessas práticas permite que as equipes de desenvolvimento e operações colaborem de maneira mais eficaz, reduzindo silos e melhorando a comunicação.

3.8.2. Inteligência Artificial e Machine Learning no Desenvolvimento de Software

A inteligência artificial (IA) e o machine learning (ML) estão revolucionando o desenvolvimento de software, introduzindo novas maneiras de automatizar tarefas, melhorar a qualidade do código e prever problemas antes que eles ocorram. Ferramentas de IA e ML são usadas para diversas aplicações, incluindo:

- **Detecção de Bugs e Anomalias**: Algoritmos de machine learning podem ser treinados para detectar padrões em código e identificar bugs ou anomalias que podem não ser facilmente detectados por métodos tradicionais de testes (LI *et al.* 2019).
- **Automação de Testes**: Ferramentas baseadas em IA podem automatizar a criação e execução de casos de teste, melhorando a cobertura dos testes e reduzindo o tempo necessário para testar novos lançamentos (GREILER *et al.* 2015).
- **Assistentes de Codificação**: Assistentes baseados em IA, como o GitHub Copilot, podem sugerir linhas de código e ajudar os desenvolvedores a escrever código mais eficiente e livre de erros (ZHANG *et al.* 2021).

A aplicação de IA e ML no desenvolvimento de software está crescendo rapidamente, proporcionando novas oportunidades para aumentar a eficiência e a qualidade dos projetos de software.

3.8.3. Metodologias Ágeis Escaladas (Scaled Agile Frameworks)

À medida que as metodologias ágeis se tornaram populares, surgiu a necessidade de escalá-las para grandes organizações e projetos complexos. Metodologias ágeis escaladas, como o Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS) e Disciplined Agile Delivery (DAD), foram desenvolvidas para atender a essa necessidade. Essas frameworks proporcionam diretrizes para a implementação de práticas ágeis em escala, mantendo os princípios e benefícios das metodologias ágeis tradicionais (LEFFINGWELL, 2021).

- **SAFe**: O Scaled Agile Framework é uma abordagem estruturada que combina práticas ágeis e lean para coordenar grandes equipes e portfólios de projetos. SAFe oferece uma série de práticas, papéis e eventos para gerenciar o desenvolvimento ágil em grande escala (LEFFINGWELL, 2021).
- **LeSS**: Large-Scale Scrum é uma extensão do Scrum que se concentra em aplicar os princípios do Scrum em grandes organizações, mantendo a simplicidade e a flexibilidade. LeSS enfatiza a coordenação de múltiplas equipes Scrum que trabalham juntas em um único produto (LARMAN & VODDE, 2016).
- **DAD**: Disciplined Agile Delivery é uma abordagem que integra práticas ágeis, lean e tradicionais, proporcionando um framework híbrido que pode ser adaptado a diferentes contextos organizacionais (AMBLER & LINES, 2020).

Essas metodologias ágeis escaladas oferecem uma maneira estruturada de aplicar práticas ágeis em ambientes complexos e em larga escala, permitindo que as organizações mantenham a agilidade e a capacidade de resposta às mudanças, mesmo em projetos grandes e multifuncionais.

4.TRABALHOS RELACIONADOS

Para esse estudo, foram realizadas pesquisas nas bases de dados *IEEE Xplore*, *ACM Digital Library, Google Scholar* e Periódicos da CAPES. Nas pesquisas as palavras-chave eram relacionadas à micro e pequenas empresas, metodologia ágil, Scrum e gerenciamento de projetos.

4.1. Base de estudos

A análise dos processos de desenvolvimento de software é um campo de estudo amplamente investigado, com inúmeras pesquisas abordando práticas, desafios e oportunidades de melhoria. Este capítulo apresenta uma revisão dos principais trabalhos relacionados, destacando as contribuições e limitações de cada estudo, bem como suas implicações para a pesquisa em questão.

Um estudo significativo foi realizado por (GABOROV MAJA *et al.* 2021), que conduziram uma análise comparativa entre metodologias ágeis e tradicionais de gestão de projetos de TI. Os autores argumentam que as metodologias ágeis, como Scrum e Kanban, oferecem maior flexibilidade e adaptabilidade às mudanças do que as metodologias

tradicionais, como o modelo em cascata. No entanto, eles também destacam que a adoção de metodologias ágeis pode enfrentar resistência devido à necessidade de mudança cultural e à falta de experiência das equipes.

Outro estudo relevante é o de (SERRADOR E PINTO, 2015), que investigaram a relação entre o uso de metodologias ágeis e o sucesso dos projetos. Os autores concluíram que projetos gerenciados com metodologias ágeis têm maior probabilidade de serem bem-sucedidos em termos de satisfação do cliente, prazo e custo. Eles atribuem esse sucesso à natureza iterativa e incremental das metodologias ágeis, que permite ajustes contínuos e uma melhor resposta às mudanças.

No contexto das startups de tecnologia, um estudo de (PATERNOSTER *et al.* 2014) destaca as práticas ágeis mais utilizadas e os desafios enfrentados por essas empresas. Os autores descobriram que as startups tendem a adotar metodologias ágeis devido à necessidade de rápida adaptação ao mercado e à escassez de recursos. No entanto, a falta de experiência e a pressão para entregar rapidamente podem comprometer a qualidade do software.

Além disso, (DINGSØYR, NERUR, BALIJEPALLY E MOE, 2012) realizaram uma revisão sistemática da literatura sobre metodologias ágeis, identificando as principais áreas de pesquisa e lacunas no conhecimento. Eles concluíram que, apesar do aumento significativo no número de estudos sobre metodologias ágeis, ainda há uma necessidade de mais pesquisas empíricas que investiguem os impactos dessas metodologias em diferentes contextos organizacionais.

Outro estudo notável é o de (WANG *et al.* 2012), que investigaram a implementação de metodologias ágeis em ambientes de desenvolvimento distribuído. Eles descobriram que, embora as metodologias ágeis possam melhorar a comunicação e a colaboração em equipes distribuídas, a diferença de fusos horários e a barreira linguística ainda representam desafios significativos. Eles sugerem que a utilização de ferramentas de colaboração e a adaptação das práticas ágeis às necessidades específicas das equipes distribuídas podem mitigar esses desafios.

CONBOY, (2009), propõe uma definição abrangente de agilidade no desenvolvimento de software. Ele argumenta que a agilidade não deve ser vista apenas como a capacidade de responder rapidamente às mudanças, mas também como a capacidade de prever e antecipar

mudanças. Sua pesquisa destaca a necessidade de um entendimento mais profundo e holístico da agilidade para aprimorar os processos de desenvolvimento de software.

O trabalho de (CHOW E CAO, 2008) também é de grande importância, pois examina os fatores críticos de sucesso na adoção de metodologias ágeis. Eles identificaram fatores como o apoio da alta administração, a competência das equipes e a clareza dos requisitos como determinantes cruciais para o sucesso dos projetos ágeis. Além disso, eles sugerem que a falta de uma cultura organizacional compatível pode ser uma barreira significativa para a adoção dessas metodologias.

Por fim, é importante mencionar o estudo de (Boehm e Turner 2003) o qual contribuiu significativamente para o entendimento das diferenças entre metodologias ágeis e tradicionais. Em seu livro "Balancing Agility and Discipline: A Guide for the Perplexed", eles exploram como equilibrar a necessidade de agilidade com a disciplina necessária para garantir a qualidade e a previsibilidade. Eles argumentam que não existe uma abordagem única que seja adequada para todos os projetos e que a escolha da metodologia deve ser baseada nas características específicas do projeto e da equipe.

Esses estudos fornecem uma base sólida para a compreensão dos processos de desenvolvimento de software e suas implicações práticas. Eles evidenciam a importância de escolher a metodologia adequada para cada contexto específico e de considerar os fatores organizacionais e culturais que podem influenciar o sucesso dos projetos. A presente pesquisa busca contribuir para esse corpo de conhecimento, investigando práticas, desafios e oportunidades de melhoria nos processos de desenvolvimento de software em empresas de tecnologia.

Tabela 1 - Relação entre trabalhos relacionados

	Metodologias Ágeis	**Scrum**	**Micro e Pequenas Empresas**	**Gerenciamento de Projetos**
Presente trabalho	x	x	x	x
Gaborov Maja *et al.* 2021	x			x

Dingsøyr *et al.* 2012	x	x		
Chow e Cao, 2008				x
Serrador e Pinto, 2015				x
Boehm e Turner, 2003	x			x
Wang *et al.* 2012		x		
Paternoster *et al.* 2014	x		x	
Conboy, 2009	x			x

Fonte: *Elaborado pelo autor, 2025.*

Após a análise dos estudos relacionados apresentados na Tabela 1, destacam-se algumas diferenças significativas em relação ao presente trabalho, as quais serão detalhadas nos subtópicos a seguir.

4.2. Enfoque na análise dos processos de desenvolvimento de software nas empresas de tecnologia

O presente estudo se diferencia por focar diretamente nos processos de desenvolvimento de software dentro de empresas de tecnologia, buscando compreender de maneira aprofundada as práticas atuais, os desafios enfrentados e as oportunidades de melhoria. Enquanto estudos como os de Gaborov, Maja *et al.* (2021) e Serrador & Pinto (2015) abordam metodologias específicas ou contextos mais amplos de gestão de projetos, este trabalho examina o desenvolvimento de software de forma mais holística, considerando tanto as metodologias ágeis quanto as tradicionais, e suas implicações práticas no ambiente empresarial.

4.3. Metodologia de pesquisa com coleta de dados primários

Diferentemente dos trabalhos que utilizam predominantemente revisões de literatura ou estudos de caso como principais métodos de investigação, este estudo adota uma abordagem empírica baseada na aplicação de questionários estruturados a alunos da Universidade

Federal do Ceará (UFC) e profissionais da área de tecnologia. Essa metodologia permite a obtenção de dados primários atualizados e específicos para o contexto brasileiro, proporcionando uma análise quantitativa que complementa e contrasta com as abordagens qualitativas encontradas em estudos como os de Boehm e Turner (2003) e Conboy (2009).

4.4. Contextualização no cenário brasileiro

Outro aspecto diferenciador do presente estudo é o seu foco no cenário brasileiro. Ao contrário de muitos dos trabalhos relacionados, que são realizados em contextos internacionais, este trabalho investiga as práticas de desenvolvimento de software em empresas de tecnologia no Brasil, fornecendo insights diretamente aplicáveis ao mercado nacional. A contextualização local permite uma análise mais precisa dos desafios enfrentados pelas empresas brasileiras, oferecendo recomendações mais relevantes e aplicáveis ao contexto específico do país.

4.5. Contribuições práticas para pequenas e microempresas

Além disso, este trabalho oferece uma análise detalhada das práticas de desenvolvimento de software em pequenas e microempresas, um segmento frequentemente menos explorado nos estudos acadêmicos. Ao abordar os desafios específicos enfrentados por essas empresas e propor estratégias de melhoria contínua, este estudo complementa e expande as discussões iniciadas por Serrador & Pinto (2015) e Paternoster *et al.* (2014), adicionando uma perspectiva prática e voltada para a realidade desses negócios menores.

4.6. Recomendações para melhoria contínua

Por fim, uma contribuição importante deste estudo é a formulação de recomendações práticas para a melhoria contínua dos processos de desenvolvimento de software nas empresas de tecnologia. Enquanto os trabalhos relacionados frequentemente se limitam a descrever metodologias ou a analisar casos específicos, o presente estudo se propõe a fornecer diretrizes práticas baseadas nos dados coletados, oferecendo um conjunto de ações que podem ser implementadas pelas empresas para aprimorar seus processos e alcançar melhores resultados.

5. PROCEDIMENTOS METODOLÓGICOS

A escolha da metodologia é essencial para garantir a validade e a confiabilidade dos resultados obtidos. Nesta seção, são detalhados os procedimentos adotados para a coleta e análise dos dados, incluindo a definição dos participantes, a estrutura do questionário aplicado e os critérios utilizados para interpretar as informações coletadas.

5.1. Tipo de Pesquisa

Esta pesquisa caracteriza-se como um estudo descritivo com abordagem mista, combinando elementos quantitativos e qualitativos. O objetivo é descrever e analisar os processos de desenvolvimento de software em empresas de tecnologia, identificando práticas comuns, desafios enfrentados, abordagens adotadas para a captura e validação de requisitos, e as metodologias utilizadas. A pesquisa quantitativa será complementada por insights qualitativos obtidos através de perguntas abertas, onde os participantes poderão compartilhar suas percepções detalhadas sobre os processos de desenvolvimento de software.

5.2. Métodos de Coleta de Dados

Para a coleta de dados, serão utilizados os seguintes métodos:

5.2.1. Questionário

O questionário foi composto por uma combinação de perguntas fechadas e abertas, com o objetivo de capturar as percepções dos participantes sobre os processos de desenvolvimento de software. As questões abordaram tópicos como as metodologias utilizadas, os desafios enfrentados pelas empresas de tecnologia, a gestão de mudanças nos requisitos, a comunicação entre equipes e stakeholders, e as estratégias de melhoria adotadas pelas empresas.

O questionário completo está incluído como **Apêndice A**, onde são detalhadas todas as perguntas aplicadas.

5.3. Seleção dos Participantes

Os participantes da pesquisa serão selecionados com base em dois grupos principais:

- **Profissionais da Área de Tecnologia:** Desenvolvedores, gerentes de projeto, analistas de requisitos, e líderes de equipe que atuam em empresas de tecnologia de diversos portes e setores. Esses profissionais serão selecionados para capturar uma visão abrangente das práticas e desafios enfrentados na indústria.

5.4. Análise dos Dados

Os dados coletados por meio dos questionários serão analisados utilizando uma abordagem mista, quantitativa e qualitativa, e a análise será conduzida em duas etapas principais:

5.4.1. Análise Descritiva

Serão utilizadas técnicas de estatística descritiva para sumarizar as respostas dos participantes. Isso incluirá a análise de frequências, médias, medianas, modas e desvios padrão das respostas às perguntas fechadas.

5.4.2. Análise de Conteúdo

As respostas às perguntas abertas serão analisadas qualitativamente para identificar padrões, temas recorrentes, e insights significativos que complementem os dados quantitativos. Isso permitirá uma compreensão mais profunda das percepções e experiências dos participantes em relação aos processos de desenvolvimento de software.

5.5. Considerações Éticas

O trabalho será conduzido com base nos princípios éticos que regem a pesquisa científica. Todos os participantes serão informados sobre os objetivos do estudo e deverão consentir formalmente com sua participação. Será garantida a confidencialidade e o anonimato dos participantes e das informações fornecidas. Os dados coletados serão utilizados exclusivamente para os fins desta pesquisa e armazenados de forma segura.

6. ANÁLISE E DISCUSSÃO DOS RESULTADOS

O capítulo de Análise e Discussão dos Resultados tem como objetivo apresentar a interpretação dos dados coletados durante a pesquisa, contextualizando-os com a literatura existente e destacando os principais achados. O foco principal será a comparação entre as metodologias de desenvolvimento de software, com ênfase nas metodologias ágeis e tradicionais, bem como os desafios enfrentados pelas empresas de tecnologia e as oportunidades de melhoria identificadas na pesquisa. Este estudo foi baseado em um questionário aplicado a profissionais da área de desenvolvimento de software, com o intuito de entender como as empresas de tecnologia estão lidando com as metodologias de desenvolvimento, os principais problemas que enfrentam e as estratégias que utilizam para melhorar seus processos.

Este capítulo se estrutura em duas seções principais: a análise dos dados quantitativos, com base nas respostas objetivas do questionário, e a análise dos dados qualitativos, que envolvem a interpretação das respostas abertas e das discussões sobre os desafios e estratégias adotadas pelas empresas.

6.1. Características demográficas

Os participantes da pesquisa foram compostos por profissionais da área de desenvolvimento de software. A amostra foi intencionalmente composta por profissionais de diferentes áreas de atuação, com o objetivo de capturar uma variedade de perspectivas sobre os processos de desenvolvimento de software.

O questionário foi enviado para mais de **300** potenciais respondentes, e um total de **116 participantes** responderam à pesquisa, resultando em uma taxa de resposta de aproximadamente **38,7%**. A seguir, apresentamos as principais características demográficas dos participantes:

- **Cargo/Função Principal:** Os profissionais que participaram da pesquisa foram, predominantemente, desenvolvedores (backend e frontend), seguidos por analistas de requisitos, designers UI/UX, entre outros. A distribuição entre as funções pode ser observada no gráfico abaixo.

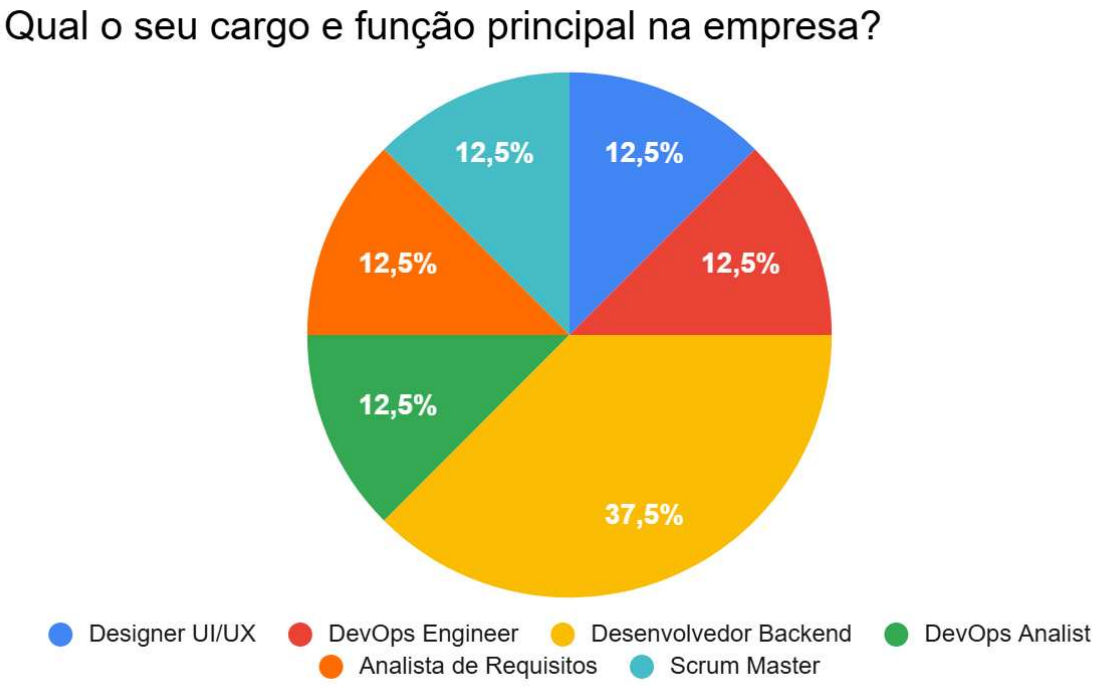

Gráfico 1 *- Distribuição dos Cargos/Funções dos Participantes*

Fonte: *Elaborado pelo autor, 2025.*

- **Setor de Atuação das Empresas:** A maioria dos participantes trabalha no setor de Tecnologia da Informação (TI), com uma representação menor nos setores de Educação e Telecomunicação. Esta distribuição é crucial para a análise dos resultados, pois pode refletir como as necessidades do setor influenciam a adoção de metodologias de desenvolvimento.

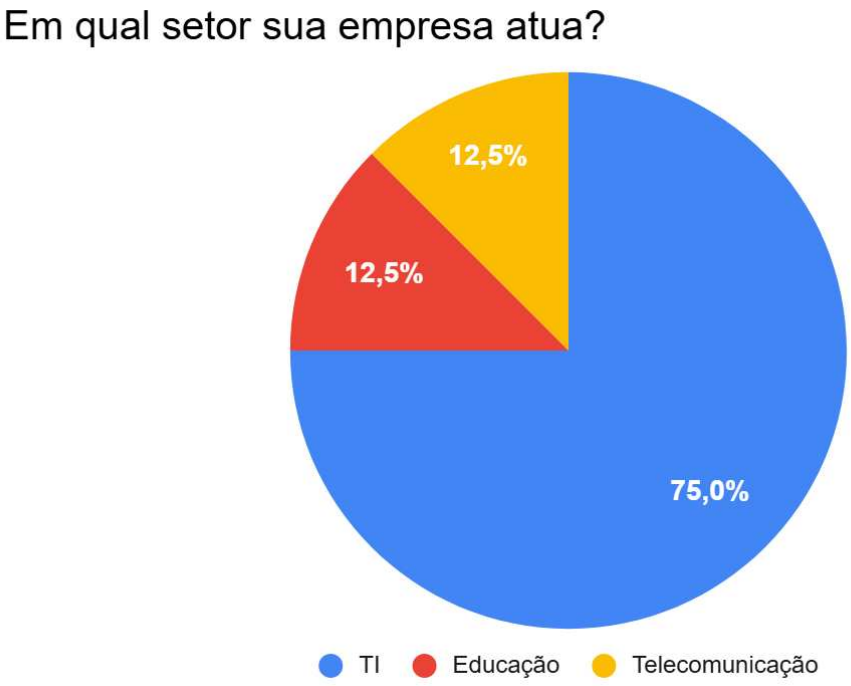

Gráfico 2 *- Setor de Atuação das Empresas*

Fonte: *Elaborado pelo autor, 2025.*

- **Porte da Empresa:** A pesquisa envolveu profissionais de empresas de diferentes portes, sendo a maioria proveniente de empresas de médio e grande porte, o que reflete a maior implementação de processos formais de desenvolvimento de software em tais empresas.

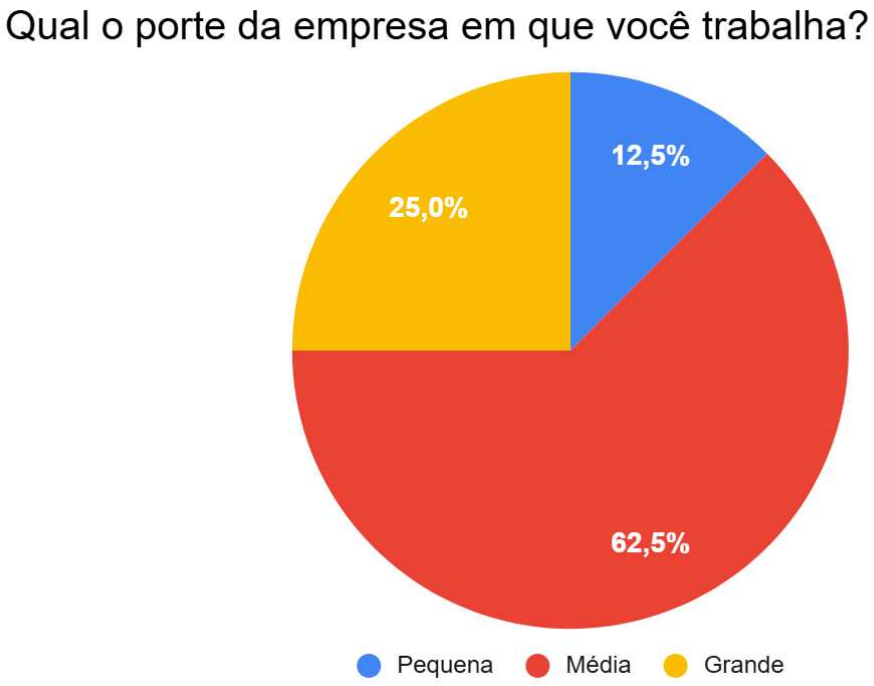

Gráfico 3 *- Porte das Empresas dos Participantes*

Fonte: *Elaborado pelo autor, 2025.*

6.2. Análise dos dados quantitativos

6.2.1. Metodologias Utilizadas pelas Empresas

A análise dos dados revela que as metodologias ágeis dominam o cenário de desenvolvimento de software nas empresas participantes da pesquisa. O Scrum se destaca como a metodologia mais utilizada, com 87,5% das empresas adotando essa prática. O Scrum é uma metodologia ágil baseada em ciclos iterativos e incrementais chamados de sprints, o que permite a adaptação contínua às mudanças durante o ciclo de vida do projeto. Este resultado corrobora a tendência observada em estudos de Boehm e Turner (2003), que afirmam que o Scrum tem se consolidado como uma das metodologias preferidas em ambientes dinâmicos e rápidos, como as empresas de tecnologia.

Logo após o Scrum, a metodologia Kanban foi adotada por 62,5% das empresas. O Kanban, com seu foco na visualização do fluxo de trabalho e na limitação do trabalho em progresso (WIP), é particularmente eficaz em ambientes onde a flexibilidade e a continuidade no fluxo de trabalho são essenciais. A flexibilidade do Kanban, ao contrário do Scrum, que

exige ciclos de trabalho fixos, permite que equipes em constante adaptação escolham a quantidade de trabalho que podem lidar em determinado momento, o que é uma vantagem em contextos mais fluidos. A combinação de visibilidade e controle do fluxo de trabalho é uma das principais razões de sua popularidade.

O uso de Scrumban, uma metodologia híbrida que mescla o Scrum e o Kanban, foi observada em 50% das empresas, refletindo uma tendência crescente de integração de práticas para balancear a estruturação do Scrum com a flexibilidade do Kanban. Este modelo híbrido tem sido bem-sucedido em equipes que necessitam de uma gestão de tarefas mais fluida, sem comprometer a organização das entregas.

Por outro lado, o Modelo Cascata, uma abordagem tradicional e sequencial, foi utilizado por apenas 12,5% das empresas, indicando uma preferencia pela flexibilidade oferecida pelas metodologias ágeis. Embora o modelo cascata ainda seja útil em projetos com requisitos bem definidos e onde a previsibilidade e a documentação sejam essenciais, sua falta de adaptabilidade e flexibilidade faz com que seja menos atraente para a maioria das empresas de tecnologia atuais.

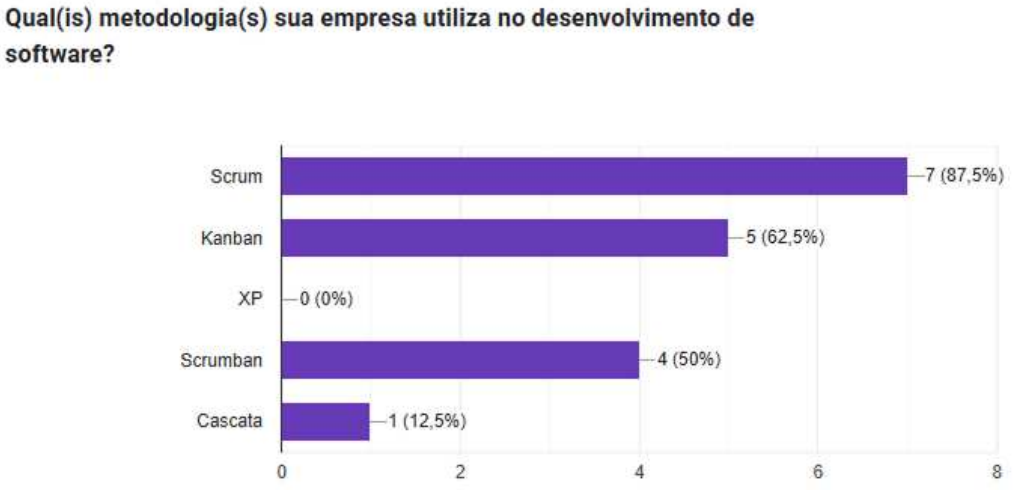

Gráfico 4 *- Metodologias Utilizadas pelas Empresas*

Fonte: *Elaborado pelo autor, 2025.*

Esses resultados indicam que as empresas têm se afastado das metodologias tradicionais, optando por modelos mais dinâmicos e colaborativos, como o Scrum e o Kanban, para lidar com as exigências de inovação constante e com a necessidade de adaptar-se rapidamente às mudanças do mercado.

6.2.2. Fatores Determinantes na Escolha da Metodologia

Ao analisarmos os fatores que influenciam a escolha da metodologia de desenvolvimento, os dados revelam que a cultura organizacional é o fator mais relevante para as empresas, com 50% dos participantes apontando isso como a razão principal para a escolha de uma metodologia. Em muitos casos, as empresas adotam uma metodologia que esteja alinhada à sua cultura interna, priorizando valores como colaboração, flexibilidade e adaptabilidade. A cultura organizacional pode, portanto, facilitar a adoção de metodologias ágeis, como o Scrum e o Kanban, que são mais colaborativas e dinâmicas.

Além disso, as exigências de clientes foram apontadas por 37,5% dos participantes como um fator determinante na escolha da metodologia. Em um ambiente altamente competitivo, onde as necessidades dos clientes mudam rapidamente, as empresas optam por metodologias que favoreçam entregas rápidas e incrementais, permitindo uma maior interação com os stakeholders e garantindo que o produto final esteja alinhado com as expectativas dos clientes.

A facilidade de adaptação foi identificada como fator determinante por 12,5% dos participantes. Embora importante, a capacidade de adaptar processos e equipes de forma rápida e eficaz não é a principal razão pela qual as empresas escolhem suas metodologias. No entanto, em contextos de constante mudança, a flexibilidade de metodologias ágeis como o Scrum e o Kanban ainda é vista como uma vantagem significativa para equipes que precisam se ajustar rapidamente a mudanças nos requisitos e nas demandas do mercado.

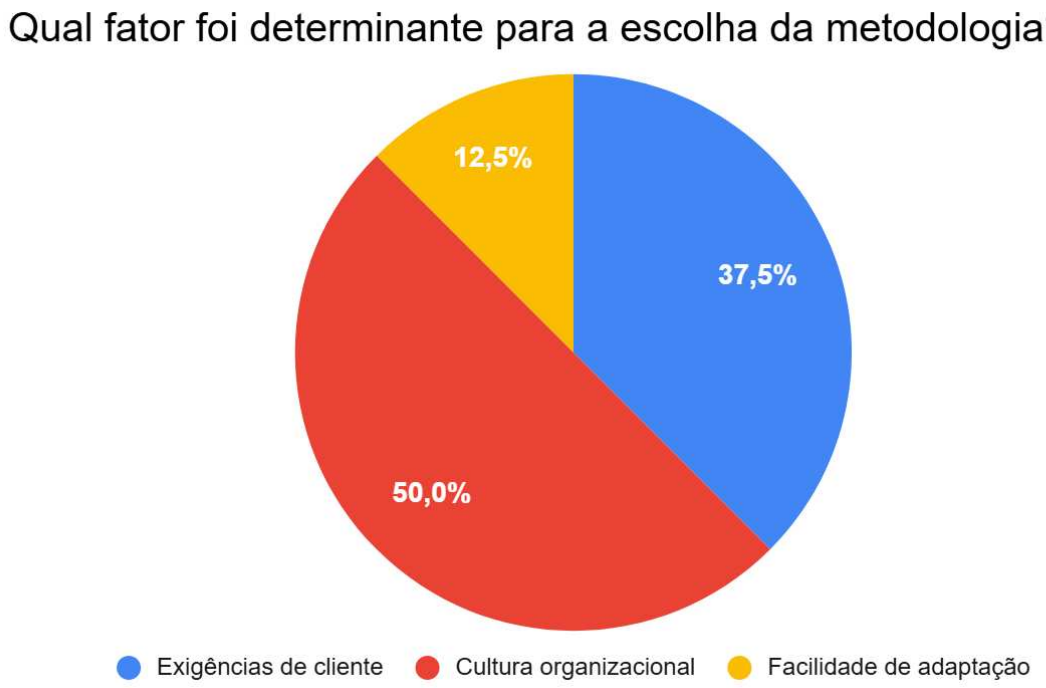

Gráfico 5 *- Fatores Determinantes na Escolha da Metodologia*

Fonte: *Elaborado pelo autor, 2025.*

6.2.3. Frequência de Mudanças na Metodologia

A flexibilidade das metodologias ao longo do ciclo de vida do projeto é uma variável crucial para a análise dos processos de desenvolvimento. Segundo os dados da pesquisa, 87,5% das empresas afirmaram que as mudanças na metodologia ocorrem às vezes durante o ciclo de vida de um projeto, enquanto 12,5% indicaram que as modificações acontecem sempre. Esses resultados sugerem que, embora as empresas escolham uma metodologia inicial, há uma abertura significativa para ajustes conforme a evolução do projeto e as demandas do mercado.

Essa adaptabilidade das metodologias é considerada uma vantagem importante, pois permite que as empresas ajustem suas abordagens de acordo com novos desafios ou mudanças nas condições do projeto. A metodologia ágil, como o Scrum, que promove ajustes regulares durante as retrospectivas de sprint, está alinhada com essa necessidade de flexibilidade. Nesse contexto, a disposição para modificar a metodologia, seja de forma ocasional ou constante, facilita a adaptação às mudanças dinâmicas que surgem ao longo do desenvolvimento.

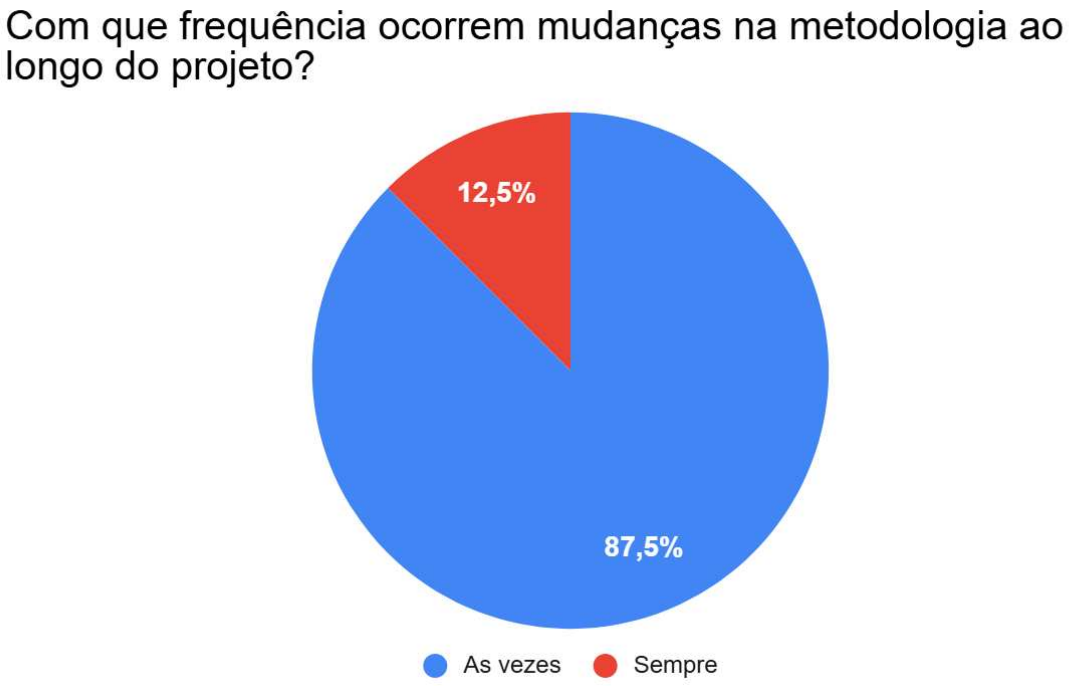

Gráfico 6 *- Frequência de Mudanças nas Metodologias*

Fonte: *Elaborado pelo autor, 2025.*

6.2.4. Treinamentos Internos e Adaptação de Metodologias

Outro ponto relevante foi a presença de treinamentos internos para a adoção das metodologias. 50% das empresas afirmaram não realizar treinamentos estruturados para a adoção das metodologias, o que pode indicar que, em algumas organizações, a implementação de metodologias ágeis não é totalmente acompanhada por uma estratégia formal de capacitação. Por outro lado, 50% das empresas afirmaram que realizam treinamentos esporadicamente, o que demonstra que, embora o treinamento não seja contínuo, ele ainda é reconhecido como uma prática importante para apoiar a adoção das metodologias.

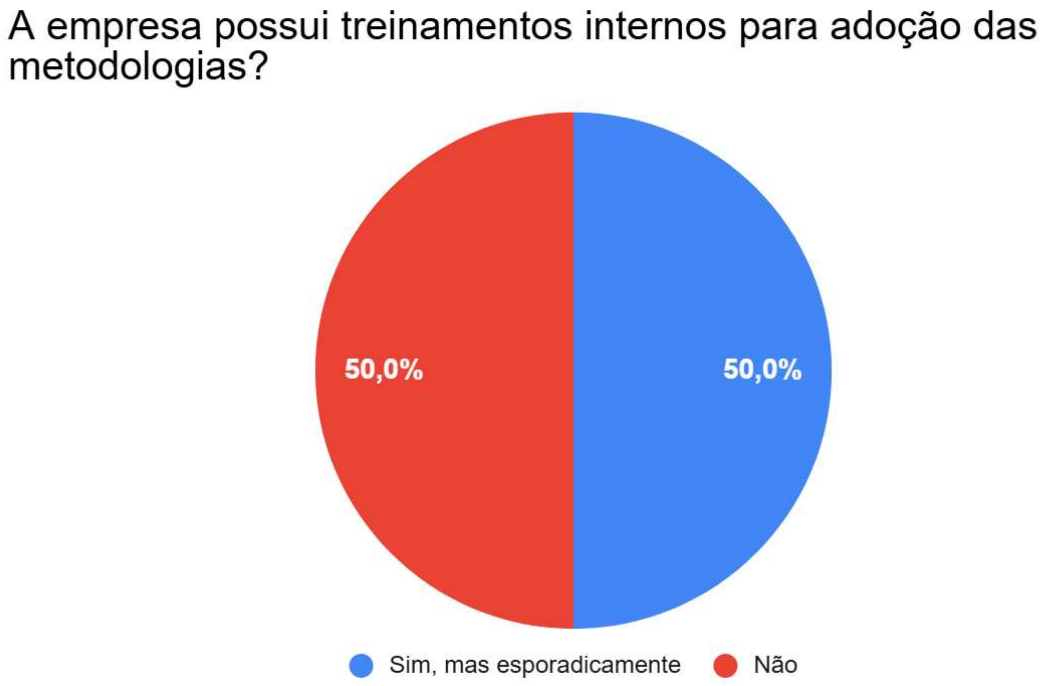

Gráfico 7 *- Treinamentos Internos*

Fonte: *Elaborado pelo autor, 2025.*

A adaptação das metodologias ao contexto do projeto foi apontada como essencial por 100% dos participantes, refletindo a flexibilidade necessária para personalizar as metodologias conforme as características específicas de cada empresa e projeto. A adoção de metodologias híbridas, como o Scrumban, também reflete essa tendência de adaptação e personalização das práticas de desenvolvimento, permitindo que as metodologias se ajustem melhor às necessidades e à dinâmica de cada equipe e contexto de trabalho.

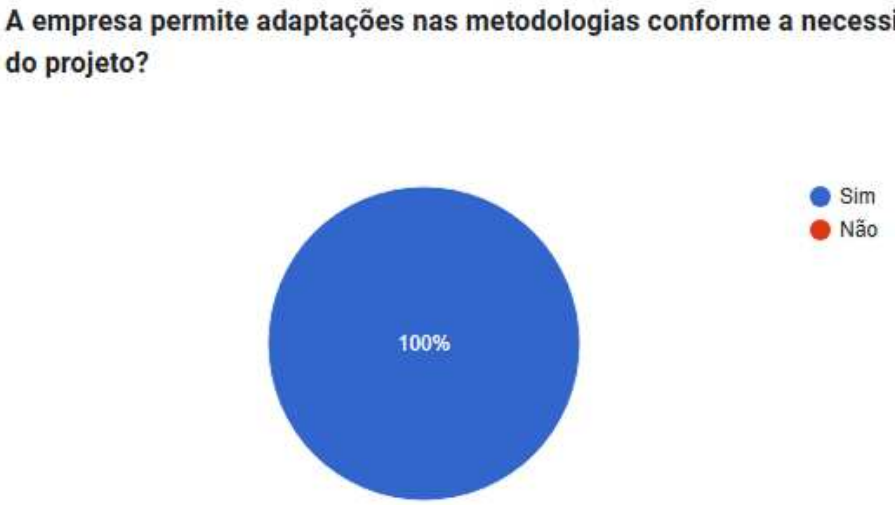

Gráfico 8 *- Adaptação de Metodologias*

Fonte: *Elaborado pelo autor, 2025.*

6.2.5. Impacto da Metodologia no Sucesso dos Projetos

Quando questionados sobre o impacto da metodologia no sucesso dos projetos, 87,5% dos participantes avaliaram a relação entre a escolha da metodologia e o sucesso dos projetos como muito positiva ou positiva. Esse dado demonstra que as metodologias ágeis, especialmente Scrum e Kanban, têm um impacto direto e positivo na eficiência e no sucesso das entregas de software. A entrega contínua e a flexibilidade nas práticas de desenvolvimento são vistas como fatores cruciais para o sucesso dos projetos, permitindo que as equipes se adaptem rapidamente às mudanças e entreguem produtos de alta qualidade dentro dos prazos estipulados.

Esses resultados também corroboram a visão de autores como Schwaber e Sutherland (2020), que defendem que metodologias ágeis como o Scrum são eficazes para a gestão de projetos dinâmicos, em constante evolução, onde a interação com os stakeholders e a entrega incremental são fundamentais para o sucesso.

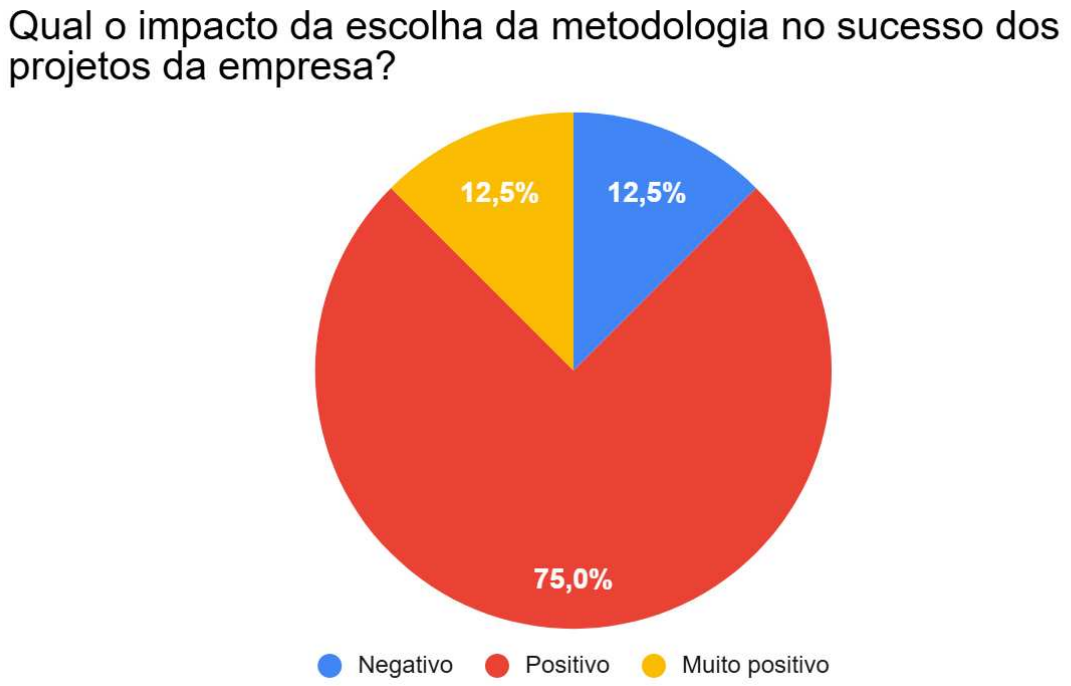

Gráfico 9 *- Impacto da Metodologia no Sucesso dos Projetos*

Fonte: *Elaborado pelo autor, 2025.*

6.2.6. Principais Desafios Enfrentados pelas Empresas

A análise qualitativa das respostas dos participantes revelou uma série de desafios que as empresas de tecnologia enfrentam durante o desenvolvimento de software. Entre os principais desafios, destacam-se:

- **Mudanças nos Requisitos (50%)**: A gestão de mudanças nos requisitos foi identificada como o maior desafio pelas empresas. Embora as metodologias ágeis, como o Scrum e o Kanban, incentivem a adaptação contínua, mudanças frequentes e mal gerenciadas podem prejudicar a produtividade das equipes e a qualidade do software. A falta de clareza nas definições iniciais do projeto, além da pressão dos clientes para implementar novas funcionalidades constantemente, contribui para esse problema. A adoção de um processo mais robusto de controle de mudanças, como o uso de ferramentas de gestão de requisitos, pode ajudar a mitigar esse desafio.
- **Falta de Documentação (25%)**: A falta de documentação adequada foi outro problema destacado pelos participantes. Embora as metodologias ágeis, como o Scrum, defendam a entrega de software funcional em detrimento de documentação extensiva, muitos participantes indicaram que a ausência de documentação pode

resultar em dificuldades futuras, como a manutenção do sistema e a integração de novos membros à equipe. Além disso, a falta de documentação também pode dificultar a comunicação entre as partes interessadas, comprometendo a transparência no andamento do projeto.

- **Comunicação entre Equipes (12,5%)**: A comunicação eficaz entre as equipes de desenvolvimento e outras áreas, como marketing e operações, é fundamental para o sucesso de qualquer projeto de software. No entanto, muitos participantes destacaram dificuldades em manter uma comunicação fluida, especialmente em empresas onde as equipes são distribuídas geograficamente ou operam em diferentes fusos horários. O problema é ainda mais exacerbado em metodologias ágeis, como o Scrum e o Kanban, que exigem comunicação constante e colaboração contínua. A falta de ferramentas de comunicação adequadas e a resistência cultural em adotar práticas mais colaborativas são fatores que agravam esse desafio.
- **Atrasos nos Cronogramas (12,5%)**: Outro desafio mencionado foi a dificuldade em cumprir os cronogramas estabelecidos. O atraso na entrega de funcionalidades é uma consequência direta de uma má implementação das metodologias de desenvolvimento, especialmente quando as equipes não têm uma definição clara de escopo e prioridades. A pressão por entregas rápidas pode resultar em decisões apressadas e na falta de planejamento adequado. A gestão ineficaz dos requisitos também contribui para esses atrasos, especialmente quando mudanças frequentes são feitas sem o devido controle.
- **Baixa Qualidade do Código (0%)**: A baixa qualidade do código foi um desafio menor destacado pelos participantes, o que indica que, ao menos nas empresas consultadas, a pressão para entregas rápidas não comprometeu significativamente a qualidade do código. A maioria das empresas parece adotar boas práticas de controle de qualidade, como testes automatizados e revisões de código, minimizando os impactos negativos.

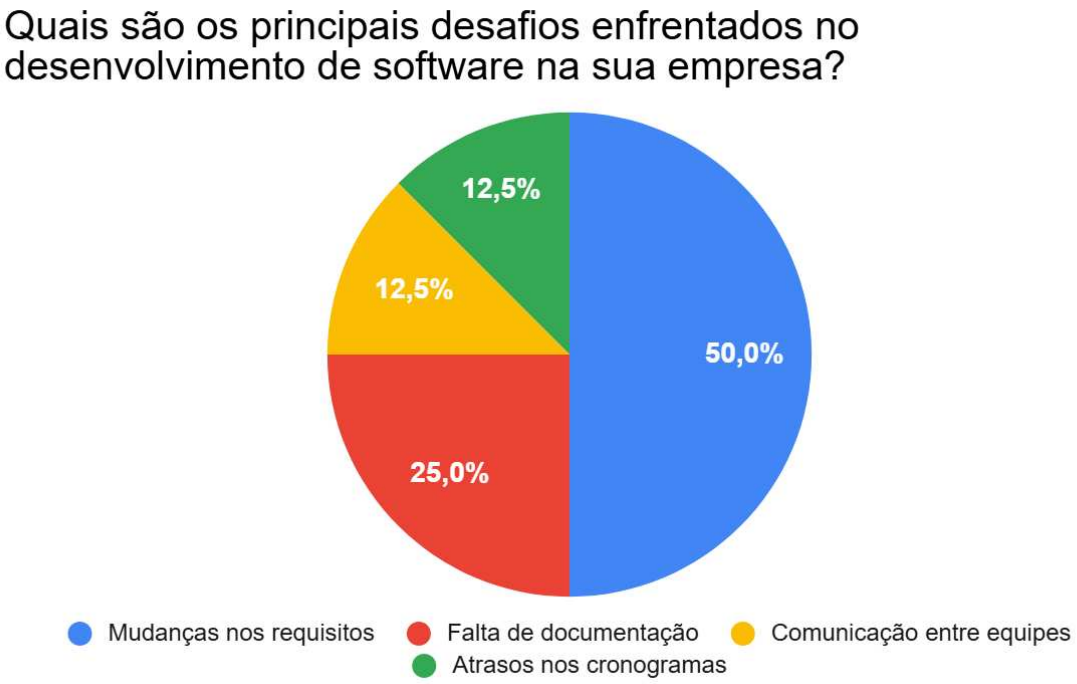

Gráfico 10 *- Estratégias Adotadas para Melhorar os Processos de Desenvolvimento*

Fonte: *Elaborado pelo autor, 2025.*

6.2.7. Estratégias Adotadas para Melhorar os Processos

Em resposta aos desafios mencionados, as empresas estão adotando diversas estratégias para melhorar seus processos de desenvolvimento de software. Algumas dessas estratégias incluem:

- **Treinamento Contínuo (25%)**: A capacitação das equipes é uma das estratégias mais adotadas para melhorar os processos. Empresas que oferecem programas de treinamento contínuo conseguem melhorar a competência técnica de suas equipes, reduzir erros e aprimorar a qualidade das entregas. Além disso, o treinamento ajuda a integrar as melhores práticas do mercado, como o uso de ferramentas de automação de testes e integração contínua.
- **Nenhuma Estratégia Específica (37,5%)**: Embora muitas empresas adotem estratégias de melhoria, uma parte significativa (37,5%) dos participantes indicou que não adotam uma estratégia específica para melhorar seus processos de desenvolvimento. Esse dado sugere que, embora o reconhecimento de desafios seja

comum, muitas empresas ainda não implementaram soluções estruturadas para resolver essas questões de forma sistemática.

- **Adoção de Metodologias Híbridas (25%)**: A combinação de metodologias ágeis e tradicionais, como o Scrumban, está sendo cada vez mais adotada. Esta abordagem híbrida combina a estrutura do Scrum com a flexibilidade do Kanban, permitindo que as equipes se beneficiem da organização dos ciclos de trabalho do Scrum, enquanto mantêm a adaptabilidade e o fluxo contínuo do Kanban. Empresas que adotam Scrumban demonstraram maior capacidade de adaptação aos requisitos em constante mudança, ao mesmo tempo que mantêm um controle estruturado das entregas.
- **Melhor Comunicação entre Equipes (12,5%)**: Para resolver os problemas de comunicação, muitas empresas estão adotando ferramentas de colaboração como Slack e Microsoft Teams, que facilitam a interação constante entre as equipes. Além disso, práticas como reuniões diárias e reuniões de planejamento com clientes e stakeholders estão sendo implementadas para melhorar a comunicação e garantir que todos os envolvidos no projeto estejam alinhados.

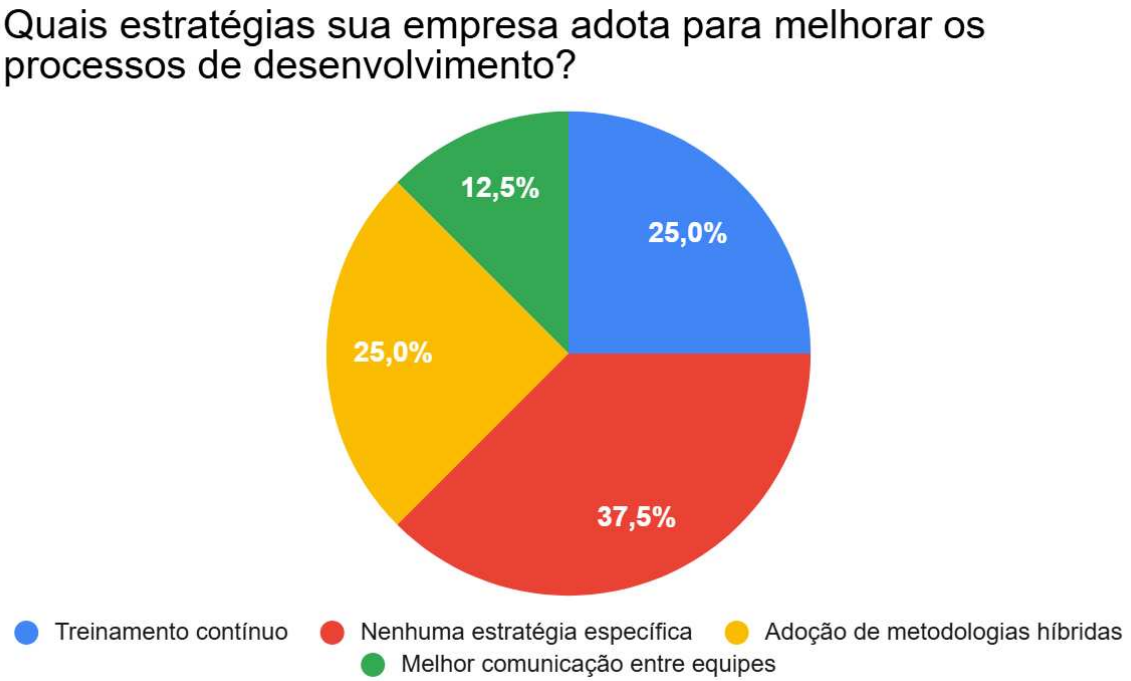

Gráfico 11 *- Estratégias Adotadas para Melhorar os Processos de Desenvolvimento*
Fonte: *Elaborado pelo autor, 2025.*

6.2.8. Adoção de Ferramentas de Gestão de Mudanças

A gestão de mudanças foi identificada como uma área que exige mais atenção nas empresas. A adoção de ferramentas de rastreabilidade e controle de mudanças, como Jira, Git e Confluence, foi vista como essencial para garantir que todas as alterações no projeto sejam documentadas e monitoradas adequadamente. No entanto, o controle adequado das mudanças também é gerido por diferentes abordagens, com 75% das empresas utilizando reuniões frequentes para alinhar as mudanças de forma contínua e garantir que as alterações sejam discutidas com todos os envolvidos. Já 25% das empresas preferem utilizar processos formais, nos quais as mudanças são registradas e acompanhadas de maneira mais estruturada, com foco em um controle rigoroso e documentação das alterações.

Além disso, os participantes do questionário indicaram que o processo de gestão de mudanças contribui para o alinhamento das expectativas dos stakeholders, garantindo que as modificações sejam feitas de acordo com os requisitos e as prioridades do projeto.

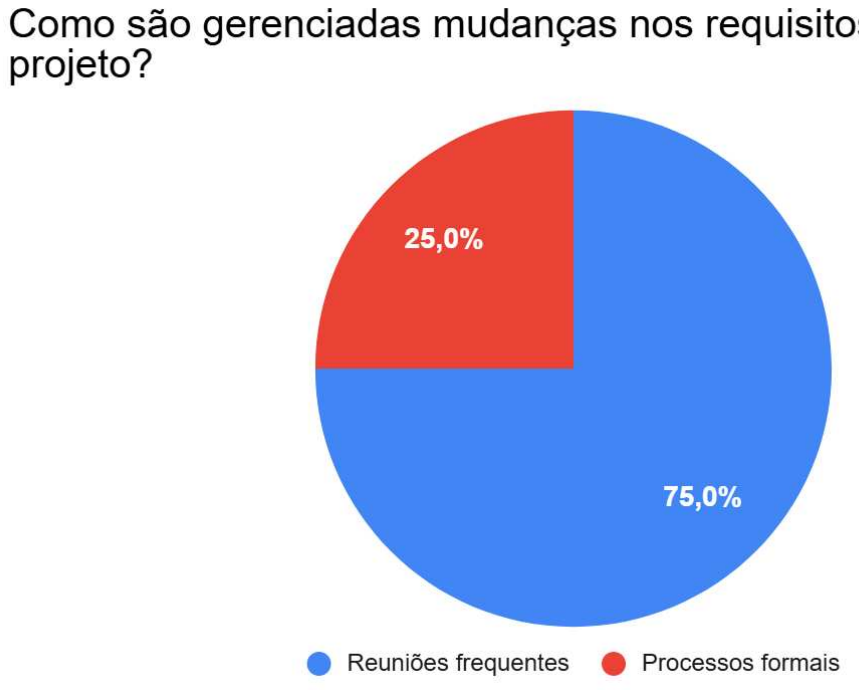

***Gráfico 12** - Adoção de Ferramentas de Gestão de Mudanças*

Fonte: *Elaborado pelo autor, 2025.*

6.3. Análise dos dados qualitativos

Além das perguntas objetivas, os participantes tiveram a oportunidade de compartilhar sugestões para a melhoria dos processos de desenvolvimento de software em suas empresas. As respostas obtidas foram analisadas qualitativamente, identificando-se padrões e temas recorrentes.

A análise revelou três principais eixos de melhoria mencionados pelos respondentes: capacitação contínua da equipe, gestão estruturada de mudanças e melhoria na documentação e comunicação.

6.3.1. Capacitação contínua da equipe

Vários participantes destacaram a importância do aprimoramento das habilidades da equipe por meio de treinamentos técnicos e metodológicos. Segundo um dos respondentes:

> *"A empresa deveria investir mais em treinamentos práticos voltados para tecnologias emergentes e boas práticas de desenvolvimento, como testes automatizados e arquitetura escalável."*

Outro participante reforçou a necessidade de capacitação em metodologias de trabalho:

> *"Os times ainda enfrentam dificuldades com Scrum e Kanban. A adoção dessas metodologias sem um treinamento adequado gera confusão e retrabalho."*

Essa percepção sugere que, apesar do uso disseminado de metodologias ágeis, há desafios na sua implementação devido à falta de capacitação formal dos profissionais.

6.3.2. Gestão estruturada de mudanças

Outro ponto recorrente nas respostas foi a dificuldade na gestão de mudanças nos requisitos e no escopo dos projetos. Um participante mencionou:

> *"Muitas vezes as mudanças ocorrem sem um processo bem definido. Não há uma priorização clara, o que faz com que o backlog se torne desorganizado e a equipe perca tempo tentando entender o que deve ser feito."*

Outro participante sugeriu soluções para minimizar esse problema:

"A empresa deveria adotar um fluxo estruturado de mudanças nos requisitos, estabelecendo momentos específicos para revisão e validação, reduzindo decisões impulsivas que afetam prazos e qualidade."

Isso indica que um dos desafios enfrentados pelas equipes está na falta de processos claros para lidar com solicitações de alteração, o que pode comprometer a produtividade e a entrega dos projetos.

6.3.3. Melhoria na documentação e comunicação

A documentação insuficiente e a comunicação ineficaz entre as equipes também foram citadas como obstáculos para a eficiência do desenvolvimento. Um dos participantes apontou:

"Sem uma documentação mínima, fica difícil para novos membros entenderem o projeto e para os desenvolvedores manterem o código no longo prazo."

Outro profissional enfatizou a necessidade de melhorar a comunicação entre áreas:

"A falta de alinhamento entre produto, desenvolvimento e stakeholders leva a expectativas desalinhadas e mudanças frequentes que poderiam ser evitadas com uma comunicação mais eficiente."

Essas respostas indicam que, embora a flexibilidade seja uma característica valorizada em metodologias ágeis, a ausência de documentação e comunicação eficaz pode gerar dificuldades no acompanhamento dos projetos.

6.3.4. Percepções da análise

A análise qualitativa das respostas reforça desafios que também foram identificados nos dados quantitativos, como dificuldades na gestão de mudanças e a necessidade de aprimoramento de processos internos. A capacitação das equipes, a estruturação de processos para lidar com mudanças e a melhoria na comunicação e documentação aparecem como sugestões recorrentes para tornar o desenvolvimento de software mais eficiente. Essas percepções destacam a necessidade de um equilíbrio entre a flexibilidade das metodologias ágeis e a organização dos processos para garantir entregas eficazes e sustentáveis.

6.4. Comparação com a literatura

A comparação entre os resultados obtidos nesta pesquisa e a literatura existente sobre o desenvolvimento de software é fundamental para verificar se as tendências observadas no estudo estão alinhadas com as práticas comuns na área, além de contribuir para a discussão sobre a eficácia das metodologias e estratégias adotadas pelas empresas de tecnologia.

6.4.1. Metodologias Ágeis no Desenvolvimento de Software

Os resultados da pesquisa indicam que a maioria das empresas adotou metodologias ágeis, especialmente o Scrum (87,5%), seguido por Kanban (62,5%) e Scrumban (50%). Esse achado está em consonância com diversos estudos, como o de Sommerville (2016), que destaca a popularização das metodologias ágeis devido à sua capacidade de fornecer entregas incrementais e flexibilidade para responder às mudanças nos requisitos ao longo do desenvolvimento. Segundo Sommerville (2016), o Scrum se tornou a metodologia ágil mais adotada, devido ao seu ciclo iterativo e à estrutura organizada que ajuda as equipes a manterem o foco e a disciplina durante o processo de desenvolvimento.

A flexibilidade das metodologias ágeis também é ressaltada por Boog, V. e Oliveira, F. (2017), que apontam que o Kanban, embora menos estruturado do que o Scrum, tem se mostrado eficaz em ambientes de desenvolvimento que exigem uma abordagem mais fluida. Este estudo é corroborado pelos dados da pesquisa, onde a adoção do Kanban foi significativa (62,5%), especialmente em empresas que buscam uma gestão de fluxo de trabalho mais visual e com menos restrições em termos de ciclos fixos.

Por outro lado, a utilização do Modelo Cascata, embora menor (12,5%), ainda reflete a escolha de algumas empresas para projetos onde os requisitos são mais definidos e as mudanças durante o desenvolvimento são mínimas. A literatura de Pressman (2014) explica que, embora o Modelo Cascata tenha sido amplamente superado pelas metodologias ágeis, ele ainda é aplicado em projetos de grande porte onde a previsibilidade e a documentação detalhada são essenciais. Este padrão observado na pesquisa reflete a continuidade da utilização de modelos tradicionais em algumas empresas, apesar da predominância das metodologias ágeis.

6.4.2. Impacto da Escolha da Metodologia no Sucesso dos Projetos

De acordo com os dados da pesquisa, 75% dos participantes acreditam que a escolha da metodologia tem um impacto positivo ou muito positivo no sucesso dos projetos. Este resultado alinha-se com os estudos de Schwaber e Sutherland (2020), que enfatizam que a adoção de Scrum e outras metodologias ágeis melhora significativamente o desempenho de equipes, pois estas abordagens proporcionam visibilidade contínua sobre o progresso e facilitam a adaptação rápida a mudanças. Além disso, essas metodologias incentivam uma comunicação constante e uma colaboração estreita entre os membros das equipes, elementos fundamentais para a entrega de projetos bem-sucedidos.

Já Boehm e Turner (2003) argumentam que, embora as metodologias ágeis tenham se mostrado eficazes em muitos contextos, seu impacto no sucesso do projeto pode ser mitigado pela falta de controle e estrutura em ambientes com requisitos altamente regulados ou em organizações com pouca experiência na implementação dessas metodologias. Esse ponto é relevante para entender a perspectiva de algumas empresas que ainda enfrentam dificuldades com a implementação ágil, como demonstrado nas respostas que mencionam a falta de adaptação às mudanças e a pressão por cronogramas apertados.

6.4.3. Desafios Enfrentados no Desenvolvimento de Software

A pesquisa identificou que o principal desafio enfrentado pelas empresas é a gestão de mudanças nos requisitos, com 50% dos participantes apontando esse fator como o mais significativo. As mudanças frequentes nos requisitos são comuns em ambientes de desenvolvimento de software, especialmente em metodologias ágeis, onde a adaptabilidade é fundamental. Contudo, mudanças mal gerenciadas podem prejudicar o andamento do projeto e a qualidade do software. Sommerville (2016) discute que mudanças mal gerenciadas em projetos ágeis podem prejudicar a produtividade das equipes e a qualidade final do software, o que é corroborado pelos resultados da pesquisa, que indicam que a falta de clareza nas definições iniciais do projeto e a pressão dos clientes para implementar novas funcionalidades constantemente agravam a situação.

A questão das mudanças nos requisitos também é discutida por Pressman (2014), que observa que a gestão de mudanças precisa ser realizada de forma estruturada para evitar impactos negativos no projeto. Da mesma forma, Boehm e Turner (2003) afirmam que a

pressão para entregar rapidamente pode resultar em alterações apressadas nos requisitos, o que, frequentemente, causa atrasos e aumenta a complexidade do projeto.

Além disso, a falta de documentação, mencionada por 25% dos participantes da pesquisa, é outro desafio relevante. Sommerville (2016) destaca que, em metodologias ágeis, o foco está na entrega de software funcional, o que pode reduzir a ênfase na documentação extensiva. No entanto, a falta de documentação pode levar a dificuldades futuras, como problemas na manutenção do sistema e integração de novos membros à equipe. A literatura reconhece a necessidade de balancear a agilidade com a documentação adequada para garantir que o conhecimento seja preservado e compartilhado entre as partes interessadas.

A comunicação entre equipes, embora apontada por 12,5% dos participantes, também continua sendo um desafio relevante. Schwaber e Sutherland (2020) destacam que as equipes Scrum bem-sucedidas dependem de comunicação constante e clara. Em ambientes de desenvolvimento distribuídos, como os relatados na pesquisa, as dificuldades de comunicação são um obstáculo significativo, o que é corroborado por Sommerville (2016), que aponta a comunicação entre equipes de diferentes especialidades como uma das maiores dificuldades em projetos de software.

6.4.4. Estratégias de Melhoria: Treinamento e Automação

As estratégias de melhoria adotadas pelas empresas para otimizar seus processos de desenvolvimento incluem uma variedade de abordagens, com destaque para treinamento contínuo (25%), adoção de metodologias híbridas (25%), nenhuma estratégia específica (37,5%) e melhor comunicação entre equipes (12,5%). Esses dados indicam que, embora o treinamento e a adaptação de metodologias híbridas sejam abordagens comuns, uma parte significativa das empresas ainda não implementa estratégias específicas para melhoria contínua.

A automação de processos foi amplamente mencionada na literatura, com Sommerville (2016) destacando a importância da automação no desenvolvimento de software, especialmente nas práticas de integração contínua (CI) e entrega contínua (CD). Essas práticas são vistas como fundamentais para aumentar a eficiência do processo de desenvolvimento e garantir a qualidade do código. Embora a automação tenha sido uma estratégia comum, a pesquisa revelou que 37,5% das empresas não adotam uma estratégia

específica, o que pode indicar que a automação ainda não é amplamente implementada ou priorizada em muitas organizações.

O treinamento contínuo (25%) também se destaca como uma estratégia importante, especialmente para empresas que buscam capacitar suas equipes para lidar com novas tecnologias e metodologias. Boog, V. e Oliveira, F. (2017) enfatizam a importância de preparar as equipes, promovendo a aprendizagem contínua para melhorar a eficiência e reduzir erros. Empresas que investem em capacitação contínua conseguem não só melhorar o desempenho das equipes, mas também garantir que as melhores práticas e ferramentas sejam utilizadas de forma eficiente.

A adoção de metodologias híbridas (25%), como o Scrumban, foi apontada como uma estratégia importante para algumas empresas. Ao combinar elementos de metodologias ágeis e tradicionais, como o Scrum e o Kanban, as empresas podem adaptar suas práticas de desenvolvimento às necessidades específicas de cada projeto, mantendo a flexibilidade sem perder a estrutura necessária para um controle eficaz.

Por fim, 12,5% das empresas apontaram a melhor comunicação entre equipes como uma estratégia essencial para melhorar os processos de desenvolvimento. O uso de ferramentas de colaboração, como Slack e Microsoft Teams, é frequentemente mencionado para facilitar a interação constante entre as equipes, promovendo alinhamento e transparência nas tarefas e decisões do projeto.

6.5. Discussão dos resultados

6.5.1. Adoção de Metodologias Ágeis: Tendência e Implicações

Os resultados desta pesquisa confirmam a crescente adoção de metodologias ágeis pelas empresas de desenvolvimento de software, com destaque para o Scrum e o Kanban. A escolha dessas metodologias reflete uma tendência global, conforme observada por Sommerville (2016), que aponta a necessidade de flexibilidade, comunicação constante e entrega incremental, características essenciais em ambientes dinâmicos e inovadores.

A pesquisa revelou que, embora uma grande parte das empresas utilize metodologias ágeis, 37,5% das empresas não adotam uma estratégia específica para lidar com os desafios de desenvolvimento, enquanto 25% das empresas adotam metodologias híbridas, como o Scrumban. Isso sugere que, apesar da popularidade do Scrum e do Kanban, muitas empresas

ainda buscam uma abordagem personalizada que combine as práticas ágeis com a estrutura necessária para projetos de maior porte.

O Scrum, que é amplamente utilizado, ainda apresenta desafios, especialmente em comunicação entre equipes. Como destacado por Schwaber e Sutherland (2020), a comunicação clara e constante é um pilar essencial para o sucesso do Scrum. No entanto, a pesquisa revela que dificuldades de comunicação, especialmente em equipes distribuídas ou com alta rotatividade, continuam sendo um obstáculo significativo. A comunicação eficiente é fundamental para o bom andamento da metodologia, mas quando não é bem gerida, pode levar a falhas na entrega e desorganização.

A adoção de metodologias híbridas (25%), como o Scrumban, tem se mostrado eficaz, principalmente em ambientes onde a flexibilidade do Kanban e a estrutura do Scrum são necessárias para lidar com a natureza dinâmica dos projetos. Essa combinação permite que as empresas se beneficiem da organização dos ciclos de trabalho do Scrum, enquanto mantêm a adaptabilidade e o fluxo contínuo do Kanban, ajudando a atender a requisitos em constante mudança.

Porém, como Boehm e Turner (2003) apontam, empresas com requisitos altamente definidos e pouca flexibilidade, como as que ainda utilizam o Modelo Cascata (12,5%), podem enfrentar dificuldades ao tentar aplicar metodologias ágeis. A resistência à mudança e a falta de adaptação dos processos podem gerar desconforto e insatisfação nas equipes, o que reforça a necessidade de uma análise detalhada das características do projeto antes de escolher a metodologia a ser aplicada.

6.5.2. Desafios no Desenvolvimento de Software: Comunicação e Mudanças nos Requisitos

Os principais desafios enfrentados pelas empresas de software, conforme os dados da pesquisa, são comunicação entre equipes (12,5%), atrasos nos cronogramas (25%) e mudanças nos requisitos (50%). Esses desafios são clássicos no campo do desenvolvimento de software e são amplamente discutidos na literatura.

A comunicação entre equipes continua sendo um dos maiores obstáculos para o sucesso dos projetos. Schwaber e Sutherland (2020) destacam que, embora o Scrum proponha um modelo de trabalho colaborativo, a eficácia desse modelo depende fortemente de uma

comunicação clara e eficiente. O fato de 50% dos participantes indicarem a comunicação como um desafio reflete que, em muitas empresas, o modelo de comunicação ágil ainda não está plenamente implementado. Isso pode ser atribuído à resistência cultural ou à falta de treinamento adequado. A literatura sugere que a adoção de ferramentas de comunicação, como Slack, Microsoft Teams ou ferramentas de gestão de projetos como Jira, pode ser uma solução para superar esse obstáculo. No entanto, a simples adoção dessas ferramentas não é suficiente; é necessário um esforço contínuo de adaptação cultural e treinamento para garantir que a comunicação flua de forma eficiente.

Outro desafio significativo é o atraso nos cronogramas, que foi identificado por 25% dos participantes como um dos principais problemas no desenvolvimento de software. O atraso nas entregas é frequentemente associado à gestão inadequada de tempo e falta de planejamento, como argumentado por Pressman (2014). Este problema também pode ser exacerbado pela pressão por entregas rápidas e pela constante mudança dos requisitos. Em muitos casos, as equipes de desenvolvimento se veem sobrecarregadas pela necessidade de adaptar-se continuamente às mudanças, o que afeta diretamente a produtividade e a qualidade do código. Uma solução proposta pela literatura é a utilização de sprints curtos e entregas incrementais, como praticado no Scrum, o que permite ajustar o curso do projeto com maior agilidade. No entanto, para que isso seja eficaz, é necessário que as empresas implementem um processo de gestão de mudanças bem estruturado e com envolvimento contínuo dos stakeholders.

A questão das mudanças nos requisitos, que foi apontada como o principal desafio por 50% dos participantes, é uma das características fundamentais das metodologias ágeis, quando bem implementadas. Boehm e Turner (2003) argumentam que, enquanto as mudanças são inevitáveis em projetos dinâmicos, é necessário um controle rigoroso para que as modificações não sobrecarreguem as equipes e afetem a qualidade do software. A pesquisa sugere que as empresas devem investir em ferramentas de rastreabilidade e controle de mudanças, como o Jira, para garantir que todas as alterações sejam devidamente documentadas e comunicadas a todos os envolvidos, minimizando os impactos negativos no andamento do projeto.

6.5.3. Implicações Práticas para as Empresas

As implicações práticas dos resultados encontrados são claras: as empresas que buscam melhorar seus processos de desenvolvimento de software devem investir em treinamento contínuo de suas equipes, automação de processos, ferramentas de gestão de mudanças e melhoria da comunicação entre equipes. A adoção de metodologias híbridas, como Scrumban, pode ser uma solução eficaz para empresas que precisam equilibrar a estrutura do Scrum com a flexibilidade do Kanban, principalmente em ambientes dinâmicos e com requisitos em constante mudança.

Além disso, é essencial que as empresas implementem um processo de gestão de requisitos e mudanças bem definido, garantindo que as modificações sejam gerenciadas de forma controlada e eficiente, sem comprometer o andamento do projeto.

6.6. Oportunidades de melhoria

As oportunidades de melhoria identificadas nesta pesquisa estão baseadas nas dificuldades relatadas pelas empresas de desenvolvimento de software e nas práticas que podem ser implementadas para superar esses desafios. Abaixo, são detalhadas as propostas de melhoria, com uma explicação clara de como essas melhorias podem ser aplicadas nas empresas, utilizando ferramentas específicas e processos que podem ser implementados de forma prática.

6.6.1. Integração de Inteligência Artificial no Desenvolvimento Ágil

A incorporação de Inteligência Artificial (IA) no desenvolvimento ágil visa otimizar processos e aumentar a eficiência das equipes de software. Ferramentas baseadas em aprendizado de máquina (machine learning) possibilitam a previsão de gargalos em sprints, a automação de revisões de código e a sugestão de refatoramentos alinhados às boas práticas de desenvolvimento. Além disso, a análise de históricos de projetos por IA pode antecipar falhas recorrentes, permitindo uma abordagem proativa para solução de problemas.

A adoção de IA no desenvolvimento ágil pode ser realizada com ferramentas já estabelecidas, que oferecem soluções para análise de código e automação de testes, tais como DeepCode, Codacy, SonarQube e Tabnine para análise de código, e Testim e Applitools para automação de testes. Essas ferramentas utilizam IA para identificar padrões de erro, sugerir

melhorias e automatizar a revisão de código, promovendo a qualidade do software e reduzindo o tempo gasto em correções.

A integração da IA com sistemas de gestão de projetos permite um monitoramento dinâmico do progresso das tarefas e uma alocação eficiente de recursos. Ferramentas como Jira, Azure DevOps e Trello possibilitam essa integração por meio de APIs nativas, viabilizando a implementação de IA sem alterar significativamente o fluxo de trabalho da equipe.

Para o treinamento de modelos personalizados, podem ser utilizadas plataformas como Google AutoML, AWS SageMaker e TensorFlow. Essas ferramentas permitem a criação e implementação de modelos de aprendizado de máquina sem a necessidade de grandes investimentos em infraestrutura, tornando a adoção de IA acessível a empresas de diferentes portes.

A implementação da IA no desenvolvimento ágil deve ser feita de maneira estruturada para garantir sua eficácia e adesão pela equipe. O processo pode ser dividido em planejamento, prova de conceito (PoC), treinamento da equipe e acompanhamento de métricas. Inicialmente, identificam-se os processos que podem ser otimizados com IA e selecionam-se as ferramentas mais adequadas. Em seguida, realiza-se uma prova de conceito para avaliar a viabilidade da solução e seu impacto no fluxo de trabalho. Posteriormente, a equipe é treinada para a utilização eficaz das ferramentas, e KPIs (Key Performance Indicators) são monitorados para mensurar a eficiência da IA na melhoria dos processos.

A utilização de ferramentas de IA reduz custos operacionais e aumenta a acessibilidade das soluções para empresas de diferentes portes. A automação progressiva e escalável permite que as organizações adotem a tecnologia de forma gradual, minimizando riscos e garantindo uma transição eficiente para um modelo de desenvolvimento mais ágil e inteligente. A implementação bem-sucedida dessas soluções pode resultar em melhorias significativas na qualidade do software, na redução de falhas e no aumento da produtividade da equipe de desenvolvimento.

6.6.2. Criação de uma metodologia híbrida adaptável

A evolução dos modelos de desenvolvimento de software tem levado à necessidade de abordagens mais flexíveis e adaptáveis às especificidades de cada organização. O presente estudo propõe uma metodologia híbrida que **integra princípios ágeis, Lean, Design Thinking e gestão baseada em dados** para proporcionar maior personalização dos processos de desenvolvimento. A implementação dessa metodologia segue um modelo estruturado, garantindo um equilíbrio entre flexibilidade e previsibilidade.

A primeira etapa para a implementação da metodologia híbrida consiste na **análise dos processos existentes** e na **identificação de pontos de ineficiência**. Isso envolve o **mapeamento dos fluxos de trabalho atuais**, a **avaliação da maturidade da equipe** em relação a metodologias ágeis e tradicionais e a **definição de objetivos estratégicos** alinhados à nova abordagem. Com base nesses diagnósticos, é possível estruturar um modelo de desenvolvimento adaptado à realidade da empresa. A metodologia híbrida proposta é composta por quatro pilares fundamentais: **planejamento adaptável, execução baseada em fluxo de valor, medição e feedback contínuos, e flexibilidade na escolha de ferramentas**. A implementação de ciclos curtos de planejamento inspirados no Design Sprint permite maior clareza nos objetivos e validação rápida de soluções. Além disso, a adoção de princípios do Lean elimina desperdícios e garante foco nos itens de maior impacto para o negócio. Para assegurar ajustes iterativos e a melhoria contínua, a utilização de métricas quantitativas e qualitativas baseadas em Evidence-Based Management (EBM) é essencial, assim como a integração de frameworks como Scrum, Kanban e SAFe, conforme a complexidade dos projetos e a maturidade da equipe.

Antes da adoção em larga escala, a metodologia deve ser validada por meio de projetos piloto, nos quais a seleção de projetos de teste, a coleta de dados e o monitoramento de KPIs são essenciais para avaliar sua efetividade. Indicadores como tempo de ciclo, taxa de retrabalho e nível de satisfação da equipe são analisados para identificar melhorias necessárias. Com os insights obtidos na fase de provas de conceito, a metodologia híbrida pode ser refinada e gradativamente expandida para outras áreas da organização. O escalonamento inclui treinamentos, mecanismos de governança para garantir consistência e monitoramento contínuo através de painéis de controle.

Em uma empresa de tecnologia focada no desenvolvimento de produtos digitais, a implementação dessa metodologia pode ocorrer de maneira otimizada ao combinar diferentes abordagens conforme a necessidade dos times. Para produtos inovadores, a adoção de sprints curtos baseados no Design Sprint auxilia na validação de ideias antes da execução completa. Equipes de manutenção e suporte podem utilizar Kanban para garantir fluxo contínuo e resolução ágil de demandas, enquanto a gestão estratégica pode ser aprimorada através de Evidence-Based Management (EBM), priorizando projetos com base em indicadores de valor. A implementação de metodologias híbridas permite que as organizações equilibrem flexibilidade e previsibilidade, garantindo eficiência e alinhamento estratégico. Ao combinar princípios ágeis, Lean, Design Thinking e gestão baseada em dados, as empresas podem estruturar frameworks personalizados que maximizam o valor entregue. O monitoramento contínuo e a adaptação iterativa são essenciais para garantir a evolução constante dos processos.

6.6.3. Aplicação do Value Stream Mapping no Desenvolvimento Ágil

A otimização dos processos de desenvolvimento de software é um desafio constante para as empresas de tecnologia. Nesse contexto, a técnica de Value Stream Mapping (VSM) surge como uma abordagem visual e sistemática para a identificação e eliminação de desperdícios, promovendo maior eficiência no fluxo de valor dos projetos ágeis. Ao mapear todas as etapas do processo, desde a concepção até a entrega final, o VSM permite compreender onde ocorrem ineficiências e implementar melhorias estratégicas.

A primeira etapa da aplicação do VSM consiste na análise detalhada do processo atual, identificando possíveis gargalos e desperdícios. Para isso, é necessário realizar o mapeamento detalhado das etapas do fluxo de desenvolvimento de software, identificando tempos de espera, refações, filas e processos redundantes. O uso de entrevistas com desenvolvedores, gestores e stakeholders auxilia na compreensão dos desafios enfrentados e permite a representação gráfica do fluxo de desenvolvimento, desde a entrada de demandas até a entrega ao cliente final. Durante essa fase, também são analisadas as interações entre equipes, ferramentas utilizadas e dependências entre processos, além da utilização de métricas como lead time (tempo total de execução), cycle time (tempo efetivo de produção) e percentual de retrabalho para identificar pontos de melhoria.

Com base nessa análise, problemas e oportunidades de melhoria são diagnosticados. Essa etapa busca identificar e eliminar atividades que não agregam valor ao produto final, reduzir tempos de espera e aprimorar a comunicação entre equipes, além de automatizar processos manuais para aumentar a eficiência e reduzir erros operacionais. Para otimizar o fluxo, é desenvolvido um Mapa de Fluxo Futuro, onde são propostas mudanças para remover desperdícios e reduzir gargalos identificados. Além disso, são implementadas metodologias ágeis e DevOps para acelerar as entregas, bem como a automação de processos para minimizar atividades repetitivas e otimizar a alocação de recursos.

O monitoramento e os ajustes contínuos são essenciais para avaliar o impacto das mudanças. Indicadores de desempenho são utilizados para validar a eficiência das melhorias, enquanto a reavaliação contínua do processo possibilita ajustes conforme feedback das equipes e stakeholders. A aplicação do VSM ocorre de forma iterativa, garantindo um aperfeiçoamento contínuo do fluxo de desenvolvimento.

A aplicação do VSM permite uma análise objetiva e visual dos entraves no ciclo de desenvolvimento, viabilizando a implementação de melhorias que resultam em maior produtividade e melhor entrega de valor ao cliente. Essa abordagem também pode ser utilizada para aprimorar a gestão de backlog, o planejamento de releases e os processos de deploy, garantindo um fluxo de trabalho mais eficiente e alinhado às necessidades do mercado.

6.6.4. Gamificação para Engajamento da Equipe

A gamificação pode ser uma estratégia eficaz para aumentar a adesão dos desenvolvedores às boas práticas e incentivar a participação ativa nas atividades do time. A introdução de elementos de jogo no ambiente de trabalho tem como objetivo elevar o engajamento, melhorar a colaboração e impulsionar a produtividade por meio de sistemas de recompensas, progressão de níveis e desafios internos.

A implementação da gamificação no desenvolvimento de software deve iniciar com a definição de objetivos e métricas claras. Estabelecer Key Performance Indicators (KPIs) mensuráveis, como taxa de conclusão de tarefas, qualidade do código e tempo médio de resposta a incidentes, é essencial para garantir que os esforços gamificados resultem em melhorias tangíveis. Um sistema de pontuação pode ser adotado para avaliar tanto o

desempenho individual quanto o coletivo, promovendo uma cultura de colaboração e reconhecimento.

A estruturação de níveis e recompensas é um dos pilares centrais da gamificação. Criar um sistema de progressão no qual os desenvolvedores avançam de nível com base na participação em revisões de código, resolução de bugs e entrega de funcionalidades dentro do prazo torna o ambiente de trabalho mais dinâmico. Além disso, a oferta de recompensas tangíveis, como bônus, folgas e brindes, associada a recompensas intangíveis, como distintivos virtuais, rankings e reconhecimento público, aumenta a motivação dos colaboradores e incentiva a melhoria contínua.

Para potencializar o engajamento, é recomendada a criação de desafios e missões que estimulem a superação de metas e a inovação dentro do time. Desafios internos, como hackathons, competições de eficiência no código e eventos de inovação, podem ser estruturados para fomentar a resolução de problemas de maneira criativa. O estabelecimento de missões diárias ou semanais com metas específicas mantém a equipe motivada e focada em seu aprimoramento técnico e colaborativo.

A integração da gamificação com ferramentas de gestão de projetos facilita a sua aplicação prática no dia a dia do desenvolvimento. Sistemas como Jira, Trello, GitHub e Azure DevOps permitem a implementação de painéis de progresso, rankings de desempenho e notificações automatizadas sobre conquistas. Bots de automação podem ser utilizados para comunicar recompensas diretamente nos canais de comunicação da equipe, como Slack e Microsoft Teams, reforçando a cultura de engajamento de forma contínua.

Para garantir que a gamificação produza os efeitos desejados, o monitoramento e ajustes contínuos são fundamentais. A avaliação periódica do impacto da gamificação no ambiente de trabalho permite identificar possíveis ajustes em regras e recompensas conforme o feedback dos colaboradores. Além disso, é importante assegurar que a competitividade saudável seja mantida, evitando efeitos adversos como pressão excessiva ou desmotivação de membros que apresentem um desempenho abaixo do esperado.

A aplicação estruturada da gamificação pode transformar a cultura organizacional, tornando o trabalho mais dinâmico e motivador, enquanto aprimora a eficiência e a colaboração entre as equipes. O uso de mecânicas de jogo dentro das ferramentas de gestão

de projetos fortalece a adesão dos desenvolvedores às boas práticas, contribuindo para a evolução constante dos processos e da qualidade do software entregue ao cliente.

6.7. Limitações do estudo

Embora esta pesquisa tenha fornecido insights valiosos sobre os processos de desenvolvimento de software nas empresas de tecnologia, é importante reconhecer as limitações que podem ter influenciado os resultados e a generalização das conclusões. A seguir, são discutidas as principais limitações do estudo.

6.7.1. Amostra Limitada

Uma das principais limitações deste estudo foi o tamanho e a composição da amostra. A pesquisa foi realizada com um número limitado de participantes, compostos por profissionais da área de desenvolvimento de software, sendo que a amostra não foi randomizada nem estratificada. Como resultado, os dados podem não refletir a diversidade de experiências e práticas adotadas em um espectro mais amplo de empresas de tecnologia. Além disso, a amostra foi composta predominantemente por empresas de médio e grande porte, o que pode não representar com precisão as realidades enfrentadas por pequenas empresas ou startups, que podem ter desafios e práticas diferentes, especialmente em relação à adoção de metodologias ágeis e ferramentas de automação.

6.7.2. Respostas Subjetivas

Embora o questionário tenha sido estruturado para capturar dados objetivos e subjetivos, as respostas qualitativas podem ter sido influenciadas pela interpretação pessoal de cada participante. Isso significa que as percepções sobre os desafios, benefícios e práticas de melhoria podem variar de acordo com a experiência e a visão individual dos entrevistados, o que pode introduzir um viés nas respostas. Além disso, o questionário foi administrado de forma online, o que pode ter influenciado a clareza das respostas, uma vez que não houve a possibilidade de esclarecimento imediato de dúvidas por parte dos participantes.

6.7.3. Dependência de Dados Autorrelatados

A pesquisa se baseou fortemente em dados autorrelatados pelos participantes, o que pode ter afetado a precisão das informações fornecidas. Embora os participantes tenham sido

incentivados a fornecer respostas sinceras, fatores como a percepção de conformidade com as práticas recomendadas, o desejo de mostrar a empresa sob uma luz positiva ou a falta de autocrítica podem ter levado a respostas mais otimistas ou, em alguns casos, a uma subestimação de problemas reais. Isso pode afetar a precisão da análise, especialmente quando se trata de identificar áreas de melhoria que não foram relatadas pelos entrevistados.

6.7.4. Limitações na Aplicação das Melhoria e Metodologia Experimental

Embora as propostas de melhoria e as metodologias experimentais tenham sido baseadas nos resultados obtidos, a aplicação prática dessas propostas em larga escala não foi realizada no escopo deste estudo. As sugestões de melhorias, como a adoção de inteligência artificial, gamificação e Value Stream Mapping, são teorias que dependem de experimentações em ambientes reais para serem validadas. Não foi possível realizar uma intervenção direta nas empresas participantes para testar a eficácia das sugestões, o que limita a aplicação prática dos resultados obtidos.

6.7.5. Falta de Análise Longitudinal

Este estudo não incluiu uma análise longitudinal, ou seja, a avaliação dos efeitos das metodologias e práticas de desenvolvimento ao longo do tempo. Uma análise longitudinal seria importante para entender como as práticas e desafios evoluem dentro das empresas e qual o impacto das mudanças implementadas após o estudo. Sem esse acompanhamento ao longo do tempo, os resultados da pesquisa podem não refletir completamente as tendências de longo prazo ou os efeitos de mudanças na metodologia de desenvolvimento.

7. CONCLUSÃO E TRABALHOS FUTUROS

O desenvolvimento de software tem sido uma das áreas mais dinâmicas e inovadoras na atualidade, especialmente no contexto das empresas de tecnologia. Este trabalho teve como objetivo investigar os processos de desenvolvimento de software nas empresas de tecnologia, identificando as metodologias utilizadas, os principais desafios enfrentados e as estratégias adotadas para aprimorar esses processos. A pesquisa foi realizada por meio de um questionário aplicado a profissionais da área de desenvolvimento de software, permitindo uma análise detalhada das práticas e dificuldades no campo.

Os resultados obtidos demonstraram que as metodologias ágeis, como o Scrum e o Kanban, dominam o cenário das empresas de tecnologia, com 87,5% das empresas utilizando o Scrum como a metodologia principal. Essas metodologias, reconhecidas por sua flexibilidade e adaptabilidade, têm mostrado grande eficácia em ambientes dinâmicos, onde as mudanças nos requisitos são frequentes e os prazos apertados. No entanto, apesar de sua popularidade, os desafios relacionados à comunicação entre equipes e à gestão de mudanças nos requisitos foram amplamente identificados como obstáculos para o sucesso dos projetos.

A pesquisa também revelou que, embora algumas empresas adotem estratégias para superar esses desafios, como automação de processos e treinamentos esporádicos, uma parcela significativa dos participantes indicou que não há uma abordagem estruturada ou contínua para esses esforços. A automação de processos, especialmente na área de testes e integração contínua, foi mencionada, mas sem uma ampla implementação. Já o treinamento contínuo, embora considerado essencial, não foi identificado como uma prática recorrente na maioria das empresas analisadas.

Um aspecto relevante identificado foi a utilização de metodologias híbridas, como o Scrumban, para combinar as vantagens das metodologias ágeis com a flexibilidade necessária para adaptação aos requisitos de cada projeto. A adoção de abordagens híbridas tem se mostrado eficaz em empresas que lidam com diferentes tipos de projetos, permitindo uma gestão mais eficiente e adaptável, sem perder a estrutura necessária para garantir o sucesso das entregas.

Embora a pesquisa tenha sido fundamental para entender os processos de desenvolvimento de software nas empresas de tecnologia, é importante ressaltar as limitações do estudo. O tamanho da amostra, embora adequado para o propósito da pesquisa, não

permite uma generalização dos resultados para todas as empresas de tecnologia, especialmente para aquelas de pequeno porte ou de setores diferentes. Além disso, a dependência de respostas autorrelatadas pode ter introduzido viés nas informações fornecidas pelos participantes, limitando a precisão em algumas áreas. A falta de uma análise longitudinal, por outro lado, impede a avaliação do impacto das mudanças sugeridas ao longo do tempo.

As sugestões de melhoria apresentadas, como a adoção de inteligência artificial, gamificação e Value Stream Mapping, visam otimizar os processos de desenvolvimento de software, contribuindo para a eficiência e a qualidade das entregas. Essas melhorias podem ser aplicadas de forma gradual, permitindo que as empresas ajustem suas práticas de acordo com suas necessidades específicas. A pesquisa também sugere a implementação de uma metodologia experimental adaptável, que combine o melhor das práticas ágeis juntamente com as demais melhorias propostas pela pesquisa.

Por fim, este estudo contribui para a compreensão dos processos de desenvolvimento de software nas empresas de tecnologia, fornecendo insights valiosos sobre as metodologias adotadas, os desafios enfrentados e as estratégias de melhoria. No entanto, novas pesquisas podem expandir este estudo, explorando a aplicação das melhorias sugeridas em diferentes contextos e empresas de diferentes portes. Além disso, estudos longitudinais são recomendados para avaliar os impactos das mudanças no longo prazo e a eficácia das metodologias híbridas implementadas. A implementação de uma metodologia experimental baseada nos resultados obtidos nesta pesquisa pode ser uma área promissora para futuras investigações.

Objetivo	**Resultados Alcançados**	**Seção Onde Estão**
Analisar os processos de desenvolvimento de software nas empresas de tecnologia.	Foram identificadas diferentes abordagens e metodologias utilizadas pelas empresas, destacando a predominância das metodologias ágeis e desafios na sua implementação.	Seção 6.2.1 - Metodologias Utilizadas pelas Empresas
Identificar os principais	Os desafios mais recorrentes	Seção 6.2.6 - Principais

desafios enfrentados pelas equipes de desenvolvimento.	foram a adaptação a novas tecnologias, dificuldades de comunicação entre equipes e problemas na definição de requisitos.	Desafios Enfrentados pelas Empresas
Avaliar oportunidades de melhoria nos processos de desenvolvimento.	Foram sugeridas estratégias para otimização dos processos e integração de ferramentas, incluindo boas práticas na adoção de metodologias ágeis e engajamento da equipe	Seção 6.6 – Oportunidades de Melhorias

7.1 Trabalhos futuros

Dentre as possibilidades de continuidade desta pesquisa, destaca-se a necessidade de testar na prática as sugestões apresentadas na seção de oportunidades de melhoria. A implementação de inteligência artificial para otimizar processos, a aplicação de gamificação para engajar equipes e o uso de Value Stream Mapping são aspectos que podem ser investigados em estudos futuros.

7.1.1 Criação de uma Metodologia Híbrida Experimental

Com base nos resultados obtidos, seria relevante desenvolver a metodologia híbrida personalizada proposta, que integrasse as melhores características das metodologias ágeis com práticas mais estruturadas e com uma maior ênfase na gestão de mudanças e na gestão de requisitos, utilizando todos os processos de melhorias propostos. Esta nova abordagem experimental poderia ser aplicada em diferentes tipos de projetos, permitindo que as empresas adotem a flexibilidade das metodologias ágeis, ao mesmo tempo que mantêm uma estrutura de planejamento detalhada, crucial para projetos de maior porte ou com requisitos mais rígidos.

O desenvolvimento dessa metodologia deve considerar as necessidades específicas das equipes, as características dos projetos e as dificuldades observadas, como a comunicação entre os membros das equipes e a constante mudança nos requisitos. Além disso, uma metodologia experimental deve ser testada e validada por meio de uma implementação prática em um caso real de desenvolvimento de software.

7.1.2 Aplicação Prática e Acompanhamento de Resultados

Após a definição e adaptação da metodologia híbrida, o passo seguinte seria a aplicação em campo. Para isso, seria necessário selecionar empresas dispostas a adotar a metodologia em seus projetos reais de desenvolvimento de software. O acompanhamento de resultados seria fundamental para avaliar a eficácia da nova metodologia e identificar quais aspectos precisam ser ajustados ao longo do tempo.

A aplicação prática permitiria que as empresas envolvidas no estudo aplicassem as melhorias sugeridas, como a adoção de inteligência artificial, gamificação e Value Stream Mapping, com o intuito de observar se essas mudanças geram um impacto positivo nos resultados de desenvolvimento. Esse processo seria monitorado por meio de métricas como tempo de entrega, qualidade do código e satisfação dos stakeholders, além da eficiência nas reuniões de planejamento e retrospectivas.

7.1.3 Coleta e Análise de Dados ao Longo do Tempo

Como parte importante de um estudo longitudinal, a coleta e análise de dados precisa ser feita ao longo de um período extenso, com múltiplos ciclos de feedback para garantir que os resultados sejam robustos e significativos. Este estudo exigiria um acompanhamento constante dos indicadores de desempenho da equipe e dos resultados do projeto para determinar se as mudanças propostas têm o efeito esperado na melhoria dos processos de desenvolvimento.

A metodologia híbrida experimental seria acompanhada com métricas de produtividade e qualidade do software, assim como com pesquisas de satisfação entre os membros da equipe e os clientes finais. O tempo médio de implementação de funcionalidades, a frequência de mudanças nos requisitos e a eficácia da comunicação seriam alguns dos parâmetros que ajudariam a avaliar o sucesso da nova abordagem.

7.1.4 Necessidade de Pesquisa Longitudinal

Este estudo experimental seria demorado, dado que a análise dos resultados em campo requereria tempo para coleta de dados precisos e para observar mudanças nas práticas de desenvolvimento ao longo de várias iterações de um projeto. Uma pesquisa longitudinal seria fundamental para garantir que as conclusões sejam baseadas em dados reais e ao longo de um

ciclo de vida de desenvolvimento completo, o que proporcionaria uma visão mais clara e fundamentada sobre a eficácia da metodologia híbrida proposta.

Além disso, o acompanhamento dos resultados ao longo do tempo permitiria a identificação de padrões e a avaliação contínua das melhorias implementadas, com ajustes sendo feitos conforme os dados indicarem. Uma análise mais demorada e detalhada seria crucial para que os benefícios das melhorias possam ser confirmados e para que a nova metodologia híbrida seja refinada e otimizada.

7.1.5 Contribuições para a Área

O desenvolvimento e a avaliação de uma metodologia híbrida experimental, com base nas práticas ágeis e melhorias propostas, pode representar uma contribuição significativa para a área de engenharia de software. O sucesso dessa abordagem poderá gerar insights valiosos sobre como as empresas podem adaptar suas práticas de desenvolvimento para lidar com projetos de diferentes características e complexidades. Além disso, as conclusões extraídas desta pesquisa podem servir como referência para futuras implementações em outras empresas, oferecendo um modelo flexível e eficaz para a gestão de processos de desenvolvimento de software.

REFERÊNCIAS

Kettunen, P., & Laanti, M. (2017). Future software organizations – agile goals and roles. *European Journal of Futures Research, 5*, 16. https://doi.org/10.1007/s40309-017-0123-7

Schwarz, J., & Legner, C. (2020). Business model tools at the boundary: Exploring communities of practice and knowledge boundaries in business model innovation. *Electronic Markets, 30*(3), 421–445.

Banijamali, A., Dawadi, R., Ahmad, M. O., Similä, J., Oivo, M., & Liukkunen, K. (2016, February). An empirical study on the impact of Scrumban on geographically distributed software development. In *2016 4th international conference on model-driven engineering and software development (MODELSWARD)* (pp. 567-577). IEEE.

Almeida, F., & Carneiro, P. (2023). Perceived importance of metrics for agile Scrum environments. *Information, 14*(6), 327. https://doi.org/10.3390/info14060327

Verwijs, C., & Russo, D. (2023). A theory of Scrum team effectiveness. *ACM Transactions on Software Engineering and Methodology, 32*(3), 1-51.

Reis, A. A. (2021). Scrumban-metodologia híbrida com Scrum e Kanban para desenvolvimento de software. *Revista de Tecnologia da Informação e Comunicação, 3*(1), 45-67.

Sońta-Drączkowska, E., & Krogulec, A. (2024). Challenges of scaling agile in large enterprises and implications for project management. *International Journal of Managing Projects in Business, 17*(2), 360-384.

Reginaldo, F., & Santos, G. (2020). Challenges in agile transformation journey: A qualitative study. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering* (pp. 11-20).

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software, 119*, 87-108. https://doi.org/10.1016/j.jss.2016.06.013

Behutiye, W., Rodríguez, P., Oivo, M., Aaramaa, S., Partanen, J., & Abhervé, A. (2022). Towards optimal quality requirement documentation in agile software development: A

multiple case study. *Journal of Systems and Software, 183*, 111112. https://doi.org/10.1016/j.jss.2021.111112

Gaborov, M., *et al*. (2021). Comparative analysis of agile and traditional methodologies in IT project management. *Journal of Applied Technical and Educational Sciences, 11*(4), 1-279. https://doi.org/10.24368/jates.v11i4.279

Arcos-Medina, G., & Mauricio, D. (2019). Aspects of software quality applied to the process of agile software development: A systematic literature review. *International Journal of System Assurance Engineering and Management, 10*, 867–897.

Abrahamsson, P., Oza, N. V., & Siponen, M. T. (2019). Agile software development methods: A comparative review. *ArXiv, abs/1903.10913*.

Ambler, S. W., & Lines, M. (2020). *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM Press.

Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A software architect's perspective*. Addison-Wesley Professional.

Beck, K., Beedle, M., Bennekum, A. V., *et al.* (2001). *Manifesto for agile software development*. Agile Alliance. Retrieved from https://agilemanifesto.org

Boehm, B. W. (1988). A spiral model of software development and enhancement. *ACM SIGSOFT Software Engineering Notes, 11*(4), 14-24. https://doi.org/10.1145/12944.12948

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software, 85*(6), 1213-1221. https://doi.org/10.1016/j.jss.2012.02.033

Dias, M. F., Alvaro, A., & Goldman, A. (2021). Improving software development through continuous training and learning. *Journal of Systems and Software, 174*, 110-122.

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software, 119*, 87-108. https://doi.org/10.1016/j.jss.2016.06.013

Duvall, P. M., Matyas, S., & Glover, A. (2007). *Continuous integration: Improving software quality and reducing risk*. Addison-Wesley Professional.

Fitzgerald, B., Stol, K. J., O'Sullivan, R., & O'Brien, D. (2017). Scaling agile methods to regulated environments: An industry case study. In *Proceedings of the 39th International Conference on Software Engineering (ICSE)*, 863-874.

Greiler, M., Van Deursen, A., & Storey, M. A. (2015). Automated detection of test fixture strategies and smells. *IEEE Transactions on Software Engineering, 41*(5), 429-451.

Hoda, R., & Noble, J. (2017). Becoming agile: A grounded theory of agile transitions in practice. In *Proceedings of the 39th International Conference on Software Engineering (ICSE)*, 141-151. https://doi.org/10.1109/ICSE.2017.23

Humble, J., & Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Addison-Wesley Professional.

Jørgensen, M., & Moløkken-Østvold, K. (2004). How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology, 48*(4), 297-301. https://doi.org/10.1016/j.infsof.2004.03.004

Kim, G., Humble, J., & Debois, P. (2016). *The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations*. IT Revolution Press.

Kniberg, H., & Ivarsson, A. (2012). *Scaling Agile @ Spotify*. Spotify AB.

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. *IEEE Software, 29*(6), 18-21. https://doi.org/10.1109/MS.2012.167

Kumar, A., Sharma, R., & Jain, V. (2019). Continuous integration and continuous deployment in agile development. *International Journal of Advanced Computer Science and Applications, 10*(4), 127-135. https://doi.org/10.14569/IJACSA.2019.0100416

Larman, C., & Vodde, B. (2016). *Large-scale Scrum: More with LeSS*. Addison-Wesley Professional.

Leffingwell, D. (2021). *SAFe 5.0: The scaled agile framework for lean enterprises*. Scaled Agile Inc.

Li, Z., Shang, W., Hassan, A. E., & Zou, Y. (2019). Towards just-in-time suggesting bug fixes. *Journal of Software: Evolution and Process, 31*(11), e2144. https://doi.org/10.1002/smr.2144

Mishra, D., Mishra, A., Ostrovska, S., & Weistroffer, H. R. (2020). Agile practices in global software engineering: A systematic map. *Journal of Software: Evolution and Process, 32*(12), e2298. https://doi.org/10.1002/smr.2298

Misra, S. C., Kumar, V., & Kumar, U. (2021). A comparative study of agile and traditional software development methodologies. *Journal of Software Engineering and Applications, 14*(1), 29-43. https://doi.org/10.4236/jsea.2021.141003

Olson, G. M., & Olson, J. S. (2014). *Working together apart: Collaboration over the Internet*. Morgan & Claypool Publishers.

Patel, N., Tamhane, M., & Sheth, S. (2020). Case study on agile transformation: Benefits and challenges. In *Proceedings of the 42nd International Conference on Software Engineering: Software Engineering in Practice*, 45-54. https://doi.org/10.1145/3377813.3381368

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2010). Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, 68-77.

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach*. McGraw-Hill Education.

Royce, W. W. (1970). Managing the development of large software systems. In *Proceedings of IEEE WESCON*, 1-9.

Schein, E. H. (2010). *Organizational culture and leadership*. John Wiley & Sons.

Schwaber, K., & Sutherland, J. (2020). *The Scrum guide: The definitive guide to Scrum: The rules of the game*. Scrum.org. Retrieved from https://scrumguides.org/scrum-guide.html

Sommerville, I. (2022). *Software engineering*. Pearson.

Wang, X., Conboy, K., & Pikkarainen, M. (2012). Assimilation of agile practices in use. *Information Systems Journal,* *22*(6), 435-455. https://doi.org/10.1111/j.1365-2575.2011.00392.x

Zhang, P., Wang, H., Liu, H., & Yu, Y. (2021). GitHub Copilot: A comprehensive study of its impacts on software development. In *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*, 195-206.

Apêndice A - QUESTIONÁRIO SOBRE OS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

1. **Qual o seu cargo e função principal na empresa?**
    - Desenvolvedor Backend
    - Desenvolvedor Frontend
    - Analista de Qualidade
    - Analista de Requisitos
    - Designer UI/UX
    - Outro: _____
2. **Em qual setor sua empresa atua?**
    - TI
    - Financeiro
    - Saúde
    - Educação
    - Outro: _____
3. **Qual o porte da empresa em que você trabalha?**
    - Pequena
    - Média
    - Grande
4. **Há quanto tempo você trabalha com desenvolvimento de software?**
    - <1 ano
    - 1-3 anos
    - 3-5 anos
    - +5 anos
5. **Qual(is) metodologia(s) sua empresa utiliza no desenvolvimento de software?**
    - Scrum
    - Kanban
    - XP
    - Scrumban
    - Cascata
    - Outro: _____
6. **Qual fator foi determinante para a escolha da metodologia?**
    - Facilidade de adaptação

- Cultura organizacional
- Pressão de mercado
- Exigências de cliente
- Outro: _____

7. **A empresa permite adaptações nas metodologias conforme a necessidade do projeto?**
   - Sim
   - Não
8. **Com que frequência ocorrem mudanças na metodologia ao longo do projeto?**
   - Nunca
   - Às vezes
   - Sempre
9. **A empresa possui treinamentos internos para adoção das metodologias?**
   - Sim, regularmente
   - Sim, mas esporadicamente
   - Não
10. **Quais são os principais desafios enfrentados no desenvolvimento de software na sua empresa?**
    - Comunicação entre equipes
    - Atrasos nos cronogramas
    - Mudanças nos requisitos
    - Falta de documentação
    - Baixa qualidade do código
    - Dificuldade na automação
    - Outro: _____
11. **Como são gerenciadas mudanças nos requisitos durante o projeto?**
    - (Resposta aberta)
12. **Como você avalia a comunicação entre as equipes de desenvolvimento e os stakeholders?**
    - Muito ruim
    - Ruim
    - Boa
    - Excelente

13. **Sua empresa adota alguma ferramenta para garantir rastreabilidade e controle de mudanças?**
    - Sim
    - Não
14. **A equipe enfrenta dificuldades com prazos apertados?**
    - Sim, frequentemente
    - Não, conseguimos atender aos prazos
15. **Quais estratégias sua empresa adota para melhorar os processos de desenvolvimento?**
    - Automação de processos
    - Treinamento contínuo
    - Melhor comunicação entre equipes
    - Adoção de metodologias híbridas
    - Nenhuma estratégia específica
    - Outro: _____
16. **Você percebe um esforço da empresa para melhorar a qualidade do software?**
    - Sim, de forma ativa
    - Sim, mas ainda há falhas
    - Não, pouco esforço percebido
17. **A empresa investe em automação de testes e integração contínua?**
    - Sim, amplamente
    - Sim, mas pouco estruturado
    - Não
18. **Qual o impacto da escolha da metodologia no sucesso dos projetos da empresa?**
    - Negativo
    - Positivo
    - Muito positivo
19. **O que poderia ser feito para melhorar a eficiência dos processos de desenvolvimento?**
    - (Resposta aberta)