DESENVOLVIMENTO DE SOFTWARE ÁGIL COM DEVOPS: BENEFÍCIOS E DESAFIOS

JOÃO GABRIEL OLIVEIRA SOARES[1], NÍDIA MARA MELCHIADES CASTELLI[2]

[1] Discente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas / joao.soares33@fatec.sp.gov.br
[2] Docente do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas / nidia.fernandes@fatec.sp.gov.br

RESUMO

Este artigo vem analisar o impacto da cultura DevOps e a implantação de um framework ágil em uma empresa de tecnologia da informação que atua no desenvolvimento e distribuição de sistemas para supermercados, atacados e varejos no território nacional. O estudo explora os quatro pilares do DevOps juntamente com a aplicação de um framework ágil para melhorar a eficiência e qualidade no desenvolvimento de software. Após a realização dos questionários junto aos funcionários das empresas foi constatado que, após a aplicação de metodologias ágeis junto a modernização dos processos de entrega de software utilizando práticas adotadas pela cultura DevOps, houve um aumento na qualidade do software disponibilizado, juntamente com uma diminuição do tempo de entrega.

Palavras-chave: DevOps; Desenvolvimento de Software; Cultura Ágil; Entrega Rápida;

1 INTRODUÇÃO

A indústria de desenvolvimento de software está constantemente evoluindo para atender às demandas crescentes do mercado. Quando se trata de ramos específicos, como empresas do setor atacadista e varejista, é crucial que as melhorias e correções sejam entregues o mais rápido possível, abordando questões como legislação, atualizações fiscais e financeiras, ao mesmo tempo em que se garante a qualidade dos novos recursos. Nesse contexto, surgiu o termo DevOps em meados de 2009, adicionando uma nova dimensão ao universo ágil (KIM et al., 2015).

O termo DevOps foi introduzido oficialmente nesse universo em 2009, durante a convenção Variety organizada pela O'Reilly Media. Com o principal objetivo de unir as etapas de desenvolvimento e implantação, o conceito de DevOps propõe a introdução de quatro grandes pilares nos modelos de desenvolvimento para que a entrega do software e implantação dele aconteça na menor janela de tempo possível.

Para que esse objetivo fosse atingido, os profissionais que já utilizavam as metodologias ágeis junto a um modelo de desenvolvimento, desenvolveram quatro grandes pilares para que alcançassem o sucesso: a criação de uma cultura dentro das equipes de desenvolvimento para que todos trabalhem juntos incentivando a

multidisciplinaridade dos profissionais; a automação das tarefas manuais executadas pelos times (como geração de executáveis, publicação de sites, instalação e configuração de ambientes produtivos, etc.); a criação de métricas para avaliação se a implantação da metodologia foi efetiva; e, para finalizar, o compartilhamento de informações e conhecimento entre os times.

Diante deste cenário, este artigo vem propor como a aplicação da cultura DevOps juntamente com a adesão de um framework ágil para o desenvolvimento de software, podem aumentar o nível de qualidade e diminuir o tempo da entrega de produtos.

Por fim, o objetivo deste artigo é analisar o cenário de antes e depois da implantação da cultura DevOps em uma empresa de tecnologia da informação, que atua no desenvolvimento e distribuição a nível nacional de softwares para empresas do ramo varejista (supermercados, atacados e varejos).

2 REFERENCIAL TEÓRICO

A seção de embasamento teórico em um artigo científico desempenha um papel fundamental ao fornecer a fundamentação conceitual e teórica necessária para sustentar as ideias, argumentos e hipóteses apresentados no trabalho. Por meio de uma revisão e análise da literatura existente sobre o tema de estudo, busca-se identificar referências, teorias, modelos e estudos prévios relevantes que contribuam para embasar e contextualizar a pesquisa em questão. Neste sentido, o embasamento teórico proporciona uma base sólida para o desenvolvimento do estudo e a compreensão do contexto em que ele se insere.

2.1. CICLO DE VIDA

O ciclo de vida do software é um modelo que descreve os diferentes estágios pelos quais o software passa, desde a concepção até a desativação. É um processo que inclui especificação de requisitos, projeto, implementação, teste, implantação e manutenção de software. O principal objetivo do ciclo de vida do software é garantir que o software atenda às necessidades do usuário, seja de alta qualidade, entregue no prazo e dentro do orçamento (PRESSMAN, 2021).

Existem vários modelos de ciclo de vida de software, os mais comuns são: modelo cascata, modelo iterativo e incremental, modelo espiral, modelo ágil, etc. Cada padrão tem suas próprias características e é adequado para diferentes tipos de projetos de desenvolvimento de software.

Em geral, o ciclo de vida do software é uma abordagem estruturada e disciplinada para o desenvolvimento de software que permite um melhor gerenciamento de projetos e maior eficiência na entrega de software de alta qualidade.

2.2. MODELOS DE DESENVOLVIMENTO

Modelos de desenvolvimento de software são estruturas conceituais que descrevem os procedimentos e estágios necessários para o desenvolvimento de um sistema. Eles oferecem uma estrutura para o trabalho da equipe de desenvolvimento, servindo como um manual para o planejamento, gerenciamento e execução de projetos de software. Esta estrutura é chamada de framework, pois disponibiliza um padrão para o desenvolvimento da atividade em questão (PRESSMAN, 2021).

Existem dois tipos diferentes de modelos de desenvolvimento de software, de acordo com Sommerville (2011): modelos tradicionais e modelos ágeis. Não há como ir e voltar entre as etapas de análise, projeto, implementação e teste, que acontecem em uma ordem pré-determinada em modelos tradicionais como o modelo em cascata. Métodos ágeis, como Scrum, por outro lado, colocam mais ênfase na colaboração, feedback e entrega frequente de software, tornando-os mais adaptáveis e flexíveis.

Os métodos convencionais incluem benefícios como extensa documentação e previsibilidade do projeto. No entanto, eles também têm desvantagens, como a incapacidade de se adaptar aos requisitos de mudança e a ausência de feedback contínuo do usuário. As abordagens ágeis, em comparação, colocam mais ênfase na colaboração e na entrega frequente de software, ao mesmo tempo em que são mais flexíveis e permitem a adaptação aos requisitos em constante mudança. Eles podem, no entanto, ser menos previsíveis em termos de orçamento e tempo e exigir muita disciplina da equipe de desenvolvimento (SOMMERVILLE, 2011).

Em conclusão, os frameworks que dão uma estrutura ao desenvolvimento são conhecidos como modelos de desenvolvimento de software. O melhor modelo a ser escolhido para um determinado projeto dependerá de seus requisitos, bem como da experiência da equipe de desenvolvimento.

2.2.1. Modelo de Desenvolvimento em Cascata

O modelo em cascata é um método sequencial e linear para o desenvolvimento de software que segue um processo de design sequencial. O progresso é feito através dos estágios de concepção, iniciação, análise, projeto, construção, teste, implantação e manutenção em uma única direção, como uma

cachoeira. Cada uma deve ser concluída antes que a próxima possa começar e a entrada para a fase seguinte é a saída da fase anterior. Com foco em fornecer um produto acabado na conclusão do ciclo de desenvolvimento, este modelo se distingue por sua forte ênfase no planejamento e documentação (PRESSMAN, 2021).

O modelo cascata é frequentemente empregado em projetos onde os requisitos são claramente definidos e compreendidos, onde é improvável que qualquer mudança seja feita durante o desenvolvimento do projeto. Pressman também defende que, por tornar a conclusão do projeto mais previsível, também é o modelo de desenvolvimento mais apropriado para projetos com limites de gastos e datas de conclusão definidas.

Apesar de também defender o paradigma cascata e explicar que ele oferece alguns benefícios em termos de documentação e previsibilidade, Sommerville (2011) também explicita que o modelo tem desvantagens significativas. Pode ser um desafio incorporar modificações ou comentários de usuários ou partes interessadas devido à natureza linear do modelo. Quando falhas ou problemas podem não ser descobertos até o final do ciclo de desenvolvimento, eles podem ser mais caros para corrigir, aumentando a probabilidade de falha do projeto. Sommerville (2011) complementa dizendo que o paradigma também pode ser menos apropriado para projetos desafiadores ou extensos, onde os requisitos podem mudar ou se desenvolver ao longo do tempo.

No geral, o modelo em cascata ainda é um método popular de desenvolvimento de software, especialmente em campos mais tradicionais ou regulamentados, como governo, finanças ou saúde. No entanto, abordagens mais flexíveis e interativas, como metodologias Ágeis ou *DevOps*, que permitem maior colaboração e adaptabilidade ao longo do processo de desenvolvimento, estão constantemente substituindo-o (KIM, et al, 2015).

2.2.2. Modelo de Desenvolvimento em Espiral

Outro modelo de desenvolvimento muito utilizado é o *Waterfall* (ou modelo de desenvolvimento em cascata). A prototipação é um dos aspectos do modelo espiral, este modelo é considerado interativo e incremental quando falamos em desenvolvimento de software (SOMMERVILLE, 2011). Ele é dividido em quatro ciclos, cada um representando uma interação diferente do processo de desenvolvimento. As etapas do padrão espiral são as seguintes:

- Planejamento: Nesta etapa são definidos os objetivos do projeto, as opções de desenvolvimento e os fatores de risco.
- Análise de Risco: Nesta fase são avaliados os perigos identificados na fase anterior e desenvolvidas estratégias de mitigação para lidar com eles.
- Implementação: Um protótipo de software é criado e testado durante esta etapa usando as especificações descritas nas etapas anteriores.
- Avaliação: Nesta etapa, o protótipo é avaliado pelo cliente e pelo desenvolvedor. Com base nas conclusões desta avaliação, são estabelecidos os requisitos para a próxima iteração.

A abordagem em espiral é recomendada para projetos de software complicados e de grande escala, onde a identificação e redução de riscos são essenciais para o sucesso do projeto. Sommerville (2011) explica que o modelo é mais adequado para projetos dinâmicos devido à sua abordagem adaptável e flexível, o que permite a inserção de modificações nos requisitos já levantados. O modelo espiral é considerado mais completo e potente do que outros modelos de desenvolvimento de software.

2.3. METODOLOGIAS ÁGEIS

Os métodos ágeis de desenvolvimento de software promovem a entrega contínua de valor aos clientes, em vez de aderir a um plano linear e prescritivo. São abordagens incrementais e iterativas para o desenvolvimento de software. Eles são centrados na adaptabilidade, cooperação e comunicação efetiva entre equipes de desenvolvimento, clientes e usuários e são fundamentados nos valores e princípios do Manifesto Ágil (RIGBY, D.; ELK, S.; BEREZ, 2020). As chamadas metodologias ágeis, que incorporam técnicas como Scrum, Kanban e XP *(Extreme Programming)*, entre outras, colocam forte a comunicação rápida, aprendizado baseado em experiência e adaptação de ambiente e requisitos como o foco do processo de desenvolvimento. Metodologias ágeis são amplamente empregadas em projetos de desenvolvimento de software, principalmente quando os requisitos são ambíguos e sujeitos a mudanças frequentes (PRESSMAN, 2021).

2.3.1. Kanban

A abordagem do Kanban é uma estratégia de gerenciamento fácil de usar que auxilia no controle e monitoramento do fluxo de trabalho em um projeto de desenvolvimento de software. O Kanban é baseado em um quadro, que pode ser físico ou digital, e que mostra o status das tarefas em várias fases do processo de

desenvolvimento (SOUZA, 2021). Cada tarefa é representada por um cartão que passa pelas colunas do quadro Kanban, que é separado em linhas que correspondem às diversas etapas do fluxo de trabalho.

O método Kanban enfatiza fortemente o trabalho em equipe, a visibilidade do trabalho, a melhoria contínua do processo e a entrega contínua de valor ao cliente. Ele permite que a equipe resolva problemas imediatamente e aumente a produtividade, auxiliando na identificação de desperdícios e gargalos no processo de desenvolvimento de software. A abordagem Kanban também é muito adaptável e pode ser personalizada de acordo com os requisitos exclusivos do projeto e da equipe (SOUZA, 2021).

2.3.2. Scrum

A técnica Scrum é uma abordagem interativa e incremental baseada em ciclos de trabalho curtos, as chamadas sprints, para o desenvolvimento de software. Ele traz como princípio o fornecimento contínuo de valor aos clientes, trabalhando em equipe e ajustando-se às mudanças nas necessidades do projeto (SOARES, 2004).

Uma equipe multifuncional colabora usando a metodologia Ágil para atingir um objetivo comum. A equipe é liderada por um Scrum Master, que é responsável por garantir que a equipe siga os princípios do Scrum e cumpra os objetivos da equipe. O Scrum Master organiza a equipe e os ajuda a se livrar de quaisquer impedimentos ao desenvolvimento do projeto (RIGBY, D.; ELK, S.; BEREZ, 2020). Uma lista priorizada de tarefas que devem ser executadas para concluir uma meta do projeto compõe o chamado backlog do produto. A partir do backlog do produto, a equipe escolhe um conjunto de tarefas para trabalhar durante um determinado período, a sprint, e assume o compromisso de concluir esses itens até a data limite acordada pela equipe (SOARES, 2004).

A equipe realiza revisões e retrospectivas na conclusão de cada sprint para avaliar o progresso do projeto e identificar áreas que podem ser melhoradas para o próximo sprint. A fim garantir que o projeto atenda aos requisitos do cliente e às expectativas da equipe, a metodologia Scrum foca na transparência, no teste e na adaptação.

2.3.3. XP (Extreme Programming)

O desenvolvimento de software usando a técnica XP, ou Extreme Programming, concentra-se em fornecer aos consumidores produtos funcionais

consistentemente de alta qualidade (SOARES, 2004). Ele é construído com base em princípios de engenharia, como testes automatizados, programação em pares, integração contínua e usabilidade do usuário, além de valores como comunicação, simplicidade e feedback.

A metodologia XP traz o foco para o trabalho em equipe entre os desenvolvedores e o cliente, a fim de atender melhor às necessidades do consumidor. Como o código bem projetado é mais simples de manter e evoluir ao longo do tempo, a qualidade do código é crucial para o XP (SOARES, 2004).

2.3.4. O Desuso de Modelos de Desenvolvimento Tradicionais

A forma como o desenvolvimento de software é abordada mudou significativamente, com o resultado da adoção da metodologia ágil. Metodologias de desenvolvimento de software mais antigas, incluindo o modelo Waterfall, perderam a preferência e agora são frequentemente vistas como antiquadas (KIM, et al, 2015).

Por exemplo, o modelo Waterfall exige um método linear e sequencial de desenvolvimento de software. Antes de iniciar a próxima, cada fase do procedimento deve ser finalizada. Esse modelo foi bem-sucedido no passado, mas tem várias falhas que o tornam menos desejável no ambiente acelerado e em constante evolução de hoje.

A suposição do modelo Waterfall de que todos os requisitos são conhecidos antecipadamente e podem ser corrigidos antes do início do desenvolvimento é uma de suas principais falhas. Na prática, as necessidades podem mudar durante o desenvolvimento, e especificações rígidas podem resultar em atrasos e excessos de preços.

A Atlassian, uma empresa australiana que desenvolve sistemas para outros desenvolvedores, publicou um artigo abordando que a metodologia ágil, em contraste, adota uma abordagem mais adaptável e iterativa. O procedimento é dividido em partes menores e mais fáceis de gerenciar, e cada parte é projetada e testada separadamente. Como resultado, uma entrada mais regular é possível, o que pode ajudar na identificação e resolução de problemas no início.

A técnica também incentiva os membros da equipe a trabalharem juntos e, como consequência, a comunicação do time também melhora. As equipes podem criar um software que satisfaça com mais eficácia os desejos e expectativas dos

clientes, cooperando e focando nas necessidades do usuário final. No geral, a metodologia ágil substituiu os modelos tradicionais de desenvolvimento de software como o método preferido para muitos projetos de desenvolvimento de software.

2.4. *DEVOPS*

A chamada cultura *DevOps* é uma prática da engenharia de software que busca promover a cooperação e a comunicação entre as equipes de desenvolvimento e operações, melhorando o processo de desenvolvimento de um software de forma mais rápida e consistente. Patrick Debois, um profissional de TI e autor do livro *"The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations"*, apresentou o termo "*DevOps*" pela primeira vez em meados de 2009.

DevOps, de acordo com Debois (2015), é uma cultura organizacional que enfatiza a cooperação entre as equipes de desenvolvimento e operações com o objetivo de aumentar a produtividade, qualidade e velocidade na entrega de software. A metodologia *DevOps* defende que a automação de processos, o uso de ferramentas de CI/CD e a adoção de uma cultura de experimentação são a chave para a melhoria contínua dos processos do ciclo de desenvolvimento.

O uso de métricas e feedback para melhorar a qualidade do software e o desempenho de entrega também fazem parte do conceito defendido por Debois (2015), com destaque para a responsabilidade compartilhada das equipes de desenvolvimento e operações com referência ao software.

2.4.1. Integração Contínua (CI)

A chamada *Continuous Integration*, ou "integração contínua" é o pilar de *DevOps* responsável por integrar, testar e conferir alterações no código-fonte de um projeto regularmente. A integração contínua procura identificar problemas de integração o mais cedo possível no ciclo de vida do software, para que possam ser corrigidos de forma rápida e eficaz (BRAGA, 2015).

Na realidade, as ferramentas que validam e verificam automaticamente as alterações de código assim que são confirmadas em um repositório central são usadas para fornecer integração contínua (BRAGA, 2015). Essas ferramentas realizam várias operações, incluindo compilação de código, testes automatizados, análise de código estático e confirmação da adesão aos padrões de código.

O desenvolvimento ágil de software requer o uso de integração contínua porque permite que a equipe de desenvolvimento libere software com mais frequência e com maior qualidade, reduzindo o tempo e as despesas associadas à descoberta e à solução de problemas. A integração contínua também facilita a colaboração entre as equipes de desenvolvimento e operações, acelerando e simplificando a entrega de software (BRAGA, 2015).

2.4.2. Entrega Contínua (CD)

A técnica de desenvolvimento de software conhecida como *Continuous Delivery*, ou "entrega contínua", automatiza todo o processo de criação, teste e implantação de software para fornecer novos recursos de maneira eficiente e consistente. O principal objetivo da entrega contínua é automatizar todo o ciclo de entrega de software, do desenvolvimento à implantação, para permitir que as equipes de desenvolvimento forneçam novos recursos de maneira rápida e eficaz, sem sacrificar a qualidade (SOUSA, 2019).

Na realidade, a entrega contínua envolve a criação de *pipelines* [1] de integração e entrega contínuas (CI/CD) que automatizam todas as etapas da entrega de software, desde a compilação de código até o teste automatizado, a criação de artefatos de implantação e a implantação automatizada em ambientes ativos (BRAGA, 2015). Isso permite que as equipes de desenvolvimento implantem novos recursos com frequência, eficiência e confiabilidade sem diminuir o nível de qualidade no desenvolvimento.

O desenvolvimento ágil de software precisa do conceito de entrega contínua porque permite que as equipes de desenvolvimento produzam software com qualidade e no menor espaço de tempo possível (SOUSA, 2019). Além disso, a entrega contínua promove a comunicação entre as equipes de desenvolvimento e operações, agilizando o processo de entrega de software.

2.4.3. A Relação Entre *DevOps* e Agilidade

A relação entre agilidade e *DevOps* tem como objetivo a entrega de software da forma mais rápida e confiável, *DevOps* atua como um conjunto de técnicas projetadas para incentivar a colaboração e a comunicação entre as equipes de

[1] Pipelines são sequências automatizadas de processos que conectam e executam tarefas de forma contínua, permitindo a integração e fluxo eficiente do trabalho. Utilizados em áreas como desenvolvimento de software e ciência de dados, os pipelines facilitam a passagem de informações entre etapas, assegurando uma execução consistente e fluida do processo.

desenvolvimento e operações (KIM, et al, 2015). Ele destaca o valor da experimentação, aprendizado, melhoria contínua e integração e entrega contínuas (CI/CD).

Ao contrário, a ideologia da agilidade coloca uma forte ênfase na flexibilidade, adaptabilidade e resposta à mudança. É uma maneira de pensar que enfatiza a resposta rápida, o desenvolvimento iterativo e os ciclos de feedback contínuos. Em vez de lançar software de uma só vez, as técnicas ágeis enfatizam a produção de software de alta qualidade em partes gerenciáveis (SOUZA, 2021).

O desenvolvimento ágil e o *DevOps* estão relacionados, pois ambos visam produzir software de forma rápida e eficaz. Enquanto a agilidade fornece a mentalidade e os princípios orientadores para dar suporte ao *DevOps*, o *DevOps* fornece a base técnica e processual que torna a agilidade possível (KIM, et al, 2015). Os dois métodos são complementares e se fortalecem mutuamente, tornando o processo de entrega de software mais bem-sucedido e eficiente.

3 METODOLOGIA

Este estudo tem como objetivo analisar a eficácia do uso das metodologias ágeis em conjunto com as práticas de DevOps em empresas de software, utilizando uma empresa de tecnologia da informação que desenvolve e distribui sistemas para grandes empresas de atacado, mercados e varejos em todo o território nacional.

A amostra deste estudo consistirá em cerca de 50 funcionários que atuam diretamente no processo de desenvolvimento, desde o refinamento dos requisitos até a disponibilização e implantação dos sistemas nos clientes. Para coletar os dados necessários, uma pesquisa será realizada com nove questões, sendo três sobre o perfil do funcionário, três sobre o conhecimento do funcionário sobre a cultura DevOps e três sobre metodologias ágeis e como elas são aplicadas na rotina de desenvolvimento.

A análise dos dados será realizada por meio de respostas dos questionários da pesquisa, contabilizando os resultados quantitativos e classificando os qualitativos. Todas as normas éticas serão seguidas, incluindo a obtenção de consentimento informado dos participantes, garantindo a privacidade e a confidencialidade dos dados coletados. Isso garantirá que o estudo seja realizado de forma ética e com rigor científico.

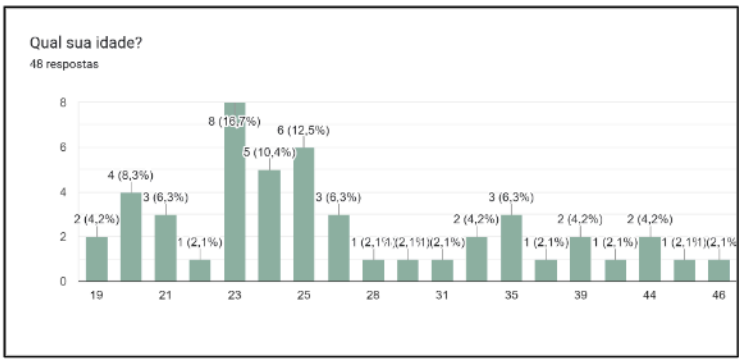
4 RESULTADOS E DISCUSSÃO

Dentro deste capítulo abordaremos a discussão e apresentação dos resultados obtidos através da pesquisa explicitada na metodologia.

4.1. EM RELAÇÃO AO PERFIL PROFISSIONAL DOS QUESTIONADOS

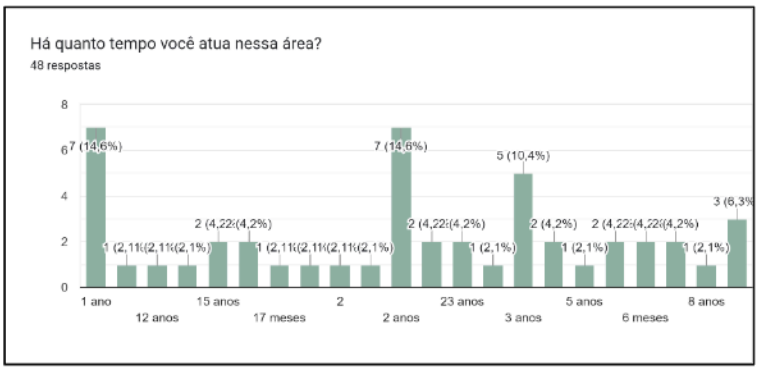
Foram realizadas três questões a respeito do perfil dos respondentes, sendo os resultados apresentados nos gráficos abaixo:

Gráfico 1 - Idade



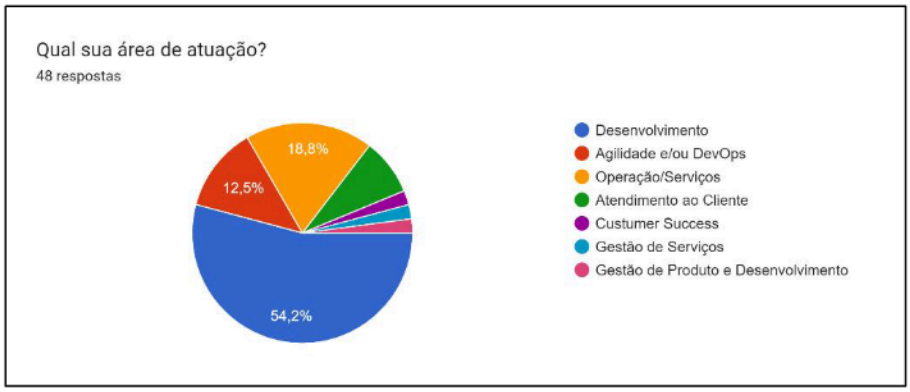
Fonte: Autor, 2023.

Gráfico 2 – Tempo de Atuação na Área



Fonte: Autor, 2023.

Gráfico 3 – Área de Atuação



Fonte: Autor, 2023.

O “Gráfico 1 – Idade” apresenta a relação das idades informadas, sendo a idade média de 27 anos, o “Gráfico 2 – Tempo de Atuação na Área” apresenta a quantidade de anos de atuação e o “Gráfico 3 – Área de Atuação” representa a área de atuação dos respondentes. Tratando-se de uma empresa de software o resultado da primeira parte da pesquisa foi satisfatório, tendo reunido informações de áreas diversas com faixas de idade e atuação diferentes.

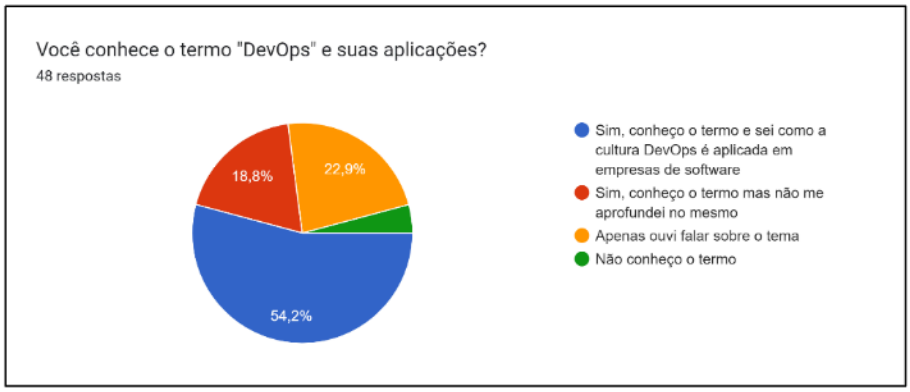
Assim, garantimos que a amostra analisada é diversa o suficiente para apresentar um viés de como profissionais com diferentes senioridades estão reagindo a aplicação de metodologias ágeis e como estão aderindo à cultura *DevOps*.

Em seu artigo Sousa (2019) apresenta a ideia de que a diversidade de profissionais é benéfica para a implantação da cultura *DevOps*, uma vez que os membros com uma senioridade maior podem atuar como mentores para os demais, replicando o conhecimento e melhorando a comunicação entre o time.

4.2. EM RELAÇÃO A DEVOPS

As próximas questões são relacionadas a adesão dos colaboradores à cultura *DevOps*. É visível que os respondentes, por estarem introduzidos em um ambiente que atua com tecnologia da informação, conhecem ou tiveram contato com o termo *DevOps*.

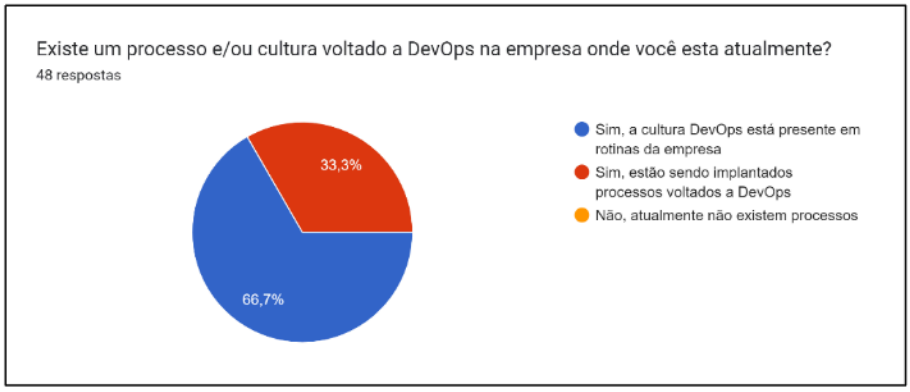
Gráfico 4 – Noções em DevOps



Fonte: Autor, 2023.

No entanto, ao analisarmos o Gráfico 4, é possível comparar as respostas de pessoas que atuam diretamente com a cultura e de pessoas que não tem contato direto com ela, uma vez que 12.5% dos respondentes da amostra atuam com agilidade e/ou *DevOps*. Nota-se que existe a automação de processos, verificação de qualidade do *software* oferecido e a aplicação de outros pilares voltados a DevOps, porém a difusão dos princípios da cultura ainda não atingiu todo o público necessário.

Gráfico 5 – Aplicação da Cultura DevOps



Fonte: Autor, 2023.

O Gráfico 5 em conjunto das respostas discursivas do questionário, comprovam a tese de que os colaboradores conhecem o básico de DevOps, porém não veem ou compreendem como o conceito está sendo aplicado dentro do ambiente no qual estão inseridos.

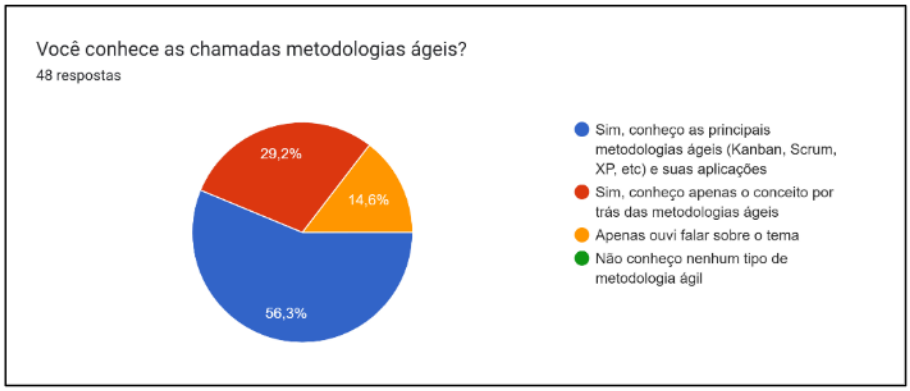
Em seu livro *"The DevOps Handbook How to Create World-class Agility, Reliability, and Security in Technology Organizations"* Patrick Debois (2015) explicita a necessidade de envolver os times em demandas multidisciplinares para que os

membros possam entender e aprender como melhorar o processo de desenvolvimento, aplicando os princípios de *DevOps* em sua rotina.

4.3. EM RELAÇÃO A AGILIDADE

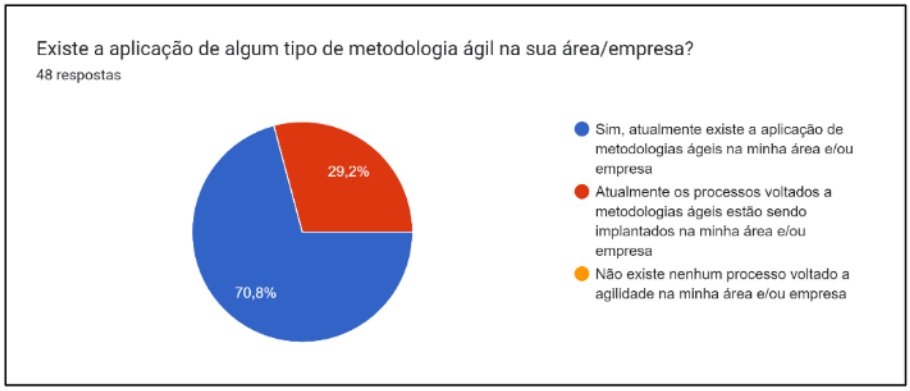
As três últimas questões são voltadas as metodologias ágeis e como são aplicadas dentro da instituição na qual a pesquisa foi realizada. Observando os gráficos “Gráfico 6 – Noções em Agilidade” e “Gráfico 7 – Aplicação das Metodologias Ágeis” é possível notar um resultado já esperado: todos os respondentes trabalham e/ou conhecem o mínimo sobre metodologias ágeis e como elas estão sendo aplicadas na organização.

Gráfico 6 – Noções em Agilidade



Fonte: Autor, 2023.

Gráfico 7 – Aplicação das Metodologias Ágeis



Fonte: Autor, 2023.

Em sua maioria, os colaboradores entendem o que são e como são aplicadas as metodologias ágeis na empresa. Ao analisar as questões discursivas é possível ver que o método utilizado é o Kanban, no entanto existe a adaptação de reuniões

para obter melhores resultados dentro das equipes que utilizam a metodologia, como pode ser observado na resposta de um dos gerentes de desenvolvimento:

> *"No modelo de desenvolvimento anterior não existia uma organização dos fluxos e da demanda. Tudo chegava através de solicitações do setor de atendimento ao cliente e não existia uma priorização além da cobrança dos próprios clientes. Com a aplicação dos fluxos essa demanda melhorou, hoje os times conseguem se organizar bem melhor usando o Kanban". (Gerente de Desenvolvimento, 35 anos).*

Esta adaptação é defendida por Souza (2021) em seu artigo, no mesmo é informado que as metodologias ágeis devem se adaptar a rotina do time que as utiliza, assim existe uma maior porcentagem de sucesso nos objetivos traçados. Tal ideia vai de encontro com o modo como a agilidade vem sendo utilizada na empresa.

4.4. HISTÓRICO DE EVOLUÇÃO

Ao realizar a análise das respostas discursivas, tanto sobre DevOps quanto sobre as metodologias ágeis, é possível ver como o processo evoluiu ao longo dos anos. Devido a diversidade da senioridade entre os profissionais entrevistados diversos pontos de vista foram coletados e puderam ser analisados, gerando um histórico do processo e apontando oportunidades de melhoria no mesmo.

Antes da aplicação de processos automatizados a empresa utilizava ferramentas internas para geração dos recursos que seriam liberados para os clientes, tais ferramentas não seguiam uma padronização e nem conversavam entre si, conforme a descrição disponibilizada através do questionário:

> *"Todo o processo dependia de um colaborador responsável por verificar a execução de cada etapa e, manualmente, iniciar a execução da próxima. Para cada etapa existia uma ferramenta e, cada uma delas, tinha um desenvolvimento próprio". (Analista de Dados e DevOps, 22 anos).*

Neste modelo havia diversos problemas, tais como a falta de integração do processo, que gerava um grande empenho caso fosse necessário realizar alguma manutenção nas ferramentas. Dentro deste cenário as manutenções e evoluções eram evitadas, sendo aplicadas soluções paralelas as ferramentas para evitar alterar o processo.

> *"Após passar por uma renovação do processo em 2019, o modo como a esteira de expedição era executada foi modernizado. As automações foram centralizadas no Jenkins, isso diminuiu a dificuldade de execução do processo como um todo. Com a simplificação dessa etapa, a velocidade e*

> *qualidade das entregas também melhorou, trazendo um resultado positivo para a organização" (Especialista II em Qualidade de Software, 39 anos).*

A resposta acima de um dos Especialistas em Qualidade de Software, aponta que atualmente o processo é executado através de um orquestrador, utilizando as automações desenvolvidas pela equipe. O sistema de apoio chamado Jenkins é uma ferramenta externa à organização que foi implantada para substituir as ferramentas internas, ele utiliza uma linguagem de programação em forma de *scripts,* sendo instruções de como o processo deve ocorrer e segue uma etapa cronológica para a execução das instruções em questão.

Ainda existem limitações e evoluções técnicas que precisam ser realizadas para garantir o processo de evolução da maturidade em *DevOps* da organização como um todo, mas já houve um grande avanço no processo que era executado anteriormente.

5 CONSIDERAÇÕES FINAIS

Em suma, este trabalho examinou o cenário de uma empresa de tecnologia da informação no ramo varejista antes e depois da implantação da cultura DevOps, com o objetivo de analisar o impacto dessa abordagem no nível de qualidade e no tempo de entrega de produtos. Os resultados obtidos demonstraram que através das respostas citadas anteriormente, a aplicação da cultura DevOps, em conjunto com um framework ágil, teve um efeito positivo na eficiência e na colaboração entre as equipes de desenvolvimento, resultando em melhorias significativas.

A introdução dos quatro pilares do DevOps, incluindo a criação de uma cultura colaborativa, automação de tarefas manuais, criação de métricas e compartilhamento de informações, mostrou-se efetiva na redução do tempo de entrega de software e no aumento da qualidade dos produtos desenvolvidos. Essas mudanças resultaram em benefícios tangíveis para a empresa, como a capacidade de lidar com atualizações legislativas e fiscais de forma mais ágil e a melhoria na satisfação do cliente.

No entanto, durante o estudo, também foram identificados pontos que requerem melhorias em relação à adesão da cultura *DevOps,* conforme relatado no questionário. Alguns desafios foram observados, como resistência à mudança por parte dos profissionais, falta de compreensão completa dos princípios e práticas do DevOps e dificuldades na integração de diferentes equipes e departamentos. Tais situações foram relatadas previamente em artigos como "Um panorama sobre o uso de práticas DevOps nas indústrias de software" escrito por Filipe A. M. Braga

publicado em 2015, onde o autor exemplifica possíveis dificuldades durante a implementação e disseminação da cultura DevOps.

Para superar esses desafios e melhorar a adesão à cultura DevOps, recomenda-se um esforço contínuo de treinamento e capacitação dos profissionais envolvidos, visando promover uma compreensão mais ampla dos conceitos e benefícios do DevOps (KIM, et al, 2015). Além disso, é importante fomentar uma mentalidade colaborativa e uma cultura de aprendizado constante dentro da organização.

Concluindo, este estudo demonstrou que a aplicação da cultura DevOps, em conjunto com um framework ágil, pode trazer benefícios significativos para empresas do ramo de desenvolvimento de software, em específico dentro de empresas voltadas a distribuição de software para mercados, varejos e atacados. Embora o objetivo de aumentar a qualidade e diminuir o tempo de entrega tenha sido alcançado, é fundamental abordar os pontos identificados que necessitam de melhorias na adesão da cultura DevOps. Ao superar esses desafios, as organizações estarão mais bem posicionadas para enfrentar as demandas em constante evolução do mercado e alcançar o sucesso sustentável em seus projetos de desenvolvimento de software.

REFERÊNCIAS

ATLASSIAN. **Gestão de projetos ágil versus cascata.** Disponível em: <https://www.atlassian.com/br/agile/project-management/project-management-intro>. Acesso em: 7 abr. 2023.

BRAGA, F. A. M. **Um panorama sobre o uso de práticas DevOps nas indústrias de software.** Disponível em: <https://repositorio.ufpe.br/handle/123456789/15989>. Acesso em: 7 abr. 2023.

KIM, G. et al. **The DevOps Handbook How to Create World-class Agility, Reliability, and Security in Technology Organizations.** [s.l.] It Revolution Pr, 2015.

Metodologia **Lean Agile: a relação entre Scrum e Lean.** Disponível em: <https://www.voitto.com.br/blog/artigo/relacao-entre-scrum-e-lean>. Acesso em: 7 abr. 2023.

RIGBY, D.; ELK, S.; BEREZ, S. **Ágil do Jeito Certo.** [s.l.] Saraiva Educação S.A., 2020.

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

SOARES, M. D. S. **Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. Revista Eletrônica de Sistemas de Informação**, v. 3, n. 1, 30 jun. 2004.

SOMMERVILLE, I. **Software engineering.** Boston; Munich: Pearson, 2011.

SOUSA, L. F. R. **DevOps: estudo de caso.** comum.rcaap.pt, 2019.

SOUZA, B. L. **Metodologias ágeis: análise e comparação do Scrum, Kanban e Lean aplicados ao desenvolvimento de software.** app.uff.br, 2021.