UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MARINILSA SANSIGOLO

UMA PROPOSTA DE MODELO DE AVALIAÇÃO DE MATURIDADE DEVOPS PARA EMPRESAS DE DESENVOLVIMENTO DE SOFTWARE

DOIS VIZINHOS

2022

MARINILSA SANSIGOLO

UMA PROPOSTA DE MODELO DE AVALIAÇÃO DE MATURIDADE DEVOPS PARA EMPRESAS DE DESENVOLVIMENTO DE SOFTWARE

A PROPOSAL FOR A DEVOPS MATURITY ASSESSMENT MODEL FOR SOFTWARE DEVELOPMENT COMPANIES

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Software do Curso de Bacharelado em Engenharia de Software da Universidade Tecnológica Federal do Paraná.

Orientador: Profa. Dra. Marisângela P. Brittes

DOIS VIZINHOS

2022

MARINILSA SANSIGOLO

UMA PROPOSTA DE MODELO DE AVALIAÇÃO DE MATURIDADE DEVOPS PARA EMPRESAS DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia de Software do Curso de Bacharelado em Engenharia de Software da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 01/dezembro/2022

Marisângela Pacheco Brittes
Doutorado
Universidade Tecnológica Federal do Paraná

Franciele Beal
Doutorado
Universidade Tecnológica Federal do Paraná

Simone de Sousa Borges
Doutorado
Universidade Tecnológica Federal do Paraná

DOIS VIZINHOS
2022

Dedico este trabalho a todos que de alguma
forma me ajudaram a concretiza-lo e
principalmente aos meus pais por todo o apoio
em todo o processo de formação.

“Se você esperar até que se sinta pronto, vai esperar pelo resto da sua vida”. Renato Maia

RESUMO

Atualmente um dos grandes desafios das empresas de tecnologia é encontrar formas mais eficientes de desenvolver software, dessa forma muitas empresas estão aderindo a novos modelos de processos, como o movimento DevOps. Nos últimos anos o termo DevOps – colaboração entre desenvolvimento e operação - e suas práticas foram amplamente usados e discutidos sob diferentes aspectos tendo como objetivo aproximar os setores de desenvolvimento e operações, os quais tradicionalmente não trabalhavam com muita proximidade. Por se tratar de uma abordagem emergente, as empresas ainda estão lapidando os seus processos, no que tange a aplicação dos conceitos de DevOps necessitando de um modelo de avaliação que permita avaliar o nível de maturidade das suas práticas. Nesse contexto, esse trabalho visa elaborar um modelo de avaliação de práticas DevOps abrangendo diversas áreas, com níveis iniciais até a melhoria contínua, sendo posteriormente aplicado em uma empresa de desenvolvimento de software para varejo do sudoeste do Paraná, oferecendo uma visão geral do estado das práticas de DevOps.

Palavras-chave: devops; desenvolvimento e operações; engenharia de software; melhoria de processos; modelo de maturidade.

ABSTRACT

Currently, one of the great challenges for technology companies is finding more efficient ways to develop software, so many companies are adhering to new process models, such as the DevOps movement. In recent years, the term DevOps - collaboration between development and operations - and its practices have been widely used and discussed under different aspects with the aim of bringing the development and operations sectors together, which traditionally did not work very closely. Because it is an emerging approach, companies are still polishing their processes, regarding the application of DevOps concepts, requiring an evaluation model that allows assessing the level of maturity of their practices. In this context, this work aims to develop a model for assessing DevOps practices covering several areas, with initial levels up to continuous improvement, and subsequently applied in a software development company for retail in the southwest of Paraná, offering an overview of the state of DevOps practices.
Keywords: devops; development and operations; software engineering; process improvement; maturity model.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A produção de software no Brasil tem apresentado crescimento constante, segundo o estudo "Mercado Brasileiro de Software – Panorama e Tendências 2021", realizado pela Associação Brasileiras das Empresas de Software (ABES, 2021), com dados do IDC, apresenta que a indústria de tecnologia no Brasil cresceu 22,9%, passando da 10ª posição em 2019 para 9ª em 2020 no ranking mundial de TI.

Com esse crescimento, ocorre também o aumento da competitividade, demandando entregas mais rápidas, levando as empresas a adaptar seus processos e seus comportamentos. Uma recente pesquisa feita pela universidade McKinsey revelou que existe uma relação direta entre a velocidade de desenvolvimento de software pelas empresas e seu desempenho nos negócios. Entre as maiores influências ligadas ao desempenho do negócio estão: ferramentas, cultura, gerenciamento de produtos e gerenciamento de talentos (SRIVASTAVA *et al.*, 2020).

No processo de desenvolvimento, ocorrem rotineiramente alterações devidas às solicitações feitas pelo cliente para ajuste ou adição de algum requisito. Nesse contexto para conseguir acompanhar as novas mudanças, as empresas estão cada vez mais aderindo às práticas ágeis.

Os métodos ágeis de desenvolvimento surgiram no final dos anos 90 reformulando os processos de levantamento de requisitos, arquitetura, implantação e teste. Diferentemente dos modelos tradicionais onde os desenvolvedores trabalham de forma isolada do restante da equipe, na metodologia ágil tanto desenvolvedores, testadores, analistas de negócios e gerentes de projetos trabalham juntos. Como reflexo do movimento ágil às entregas passaram a ser mais rápidas, gerando assim para a equipe de operações, demandas de novas *builds* de forma muito mais acelerada, o que acabou acarretando uma taxa muito elevada de erros e atrasos na implantação (HUMBLE; FARLEY, 2010).

Surgiu assim um conflito entre a equipe de desenvolvedores e operação, como entregar um software de qualidade em pouco tempo e com constantes mudanças de requisitos. A partir dessa necessidade de aproximar as equipes de desenvolvimento e operações criou-se uma nova cultura chamada DevOps.

A palavra DevOps deriva dos termos *Development* e *Operations* em inglês que referenciam os membros envolvidos no processo de desenvolvimento e implantação de softwares:

- *Development* (Desenvolvimento): os membros da equipe de desenvolvimento são responsáveis pela identificação e análise dos requisitos com os clientes, criação do projeto, teste e implementação.

- *Operations* (Operação): os membros da equipe de operação são responsáveis pela implantação em produção, monitoramento e por dar suporte em incidentes e problemas.

DevOps é uma cultura em crescimento que busca a colaboração entre os desenvolvedores de sistemas e a equipe de operações buscando a automação completa e eficiente dos processos de implantação e gerenciamento de software (WETTINGER; ANDRIKOPOULOS; LEYMANN, 2015), dessa forma um dos preceitos da cultura DevOps é que os membros da equipe se sintam genuinamente proprietários do projeto final (ANTONIO MUNIZ; ANALIA IRIGOYEN, 2020).

Com base no contexto apresentado, o presente trabalho realizou o desenvolvimento de um estudo de caso dentro de uma empresa de desenvolvimento de software de porte médio, a partir do levantamento das principais práticas, ferramentas e às influências da cultura DevOps dentro das equipes e setores.

1.2 OBJETIVO GERAL

Propor um modelo de avaliação de maturidade de práticas DevOps que possa ser aplicado em empresas de desenvolvimento de software, abrangendo processos, metodologias, técnicas e ferramentas, visando fornecer uma visão geral de suas práticas DevOps, por meio da aplicação de um *guideline.*

1.3 OBJETIVOS ESPECÍFICOS

Para que o desenvolvimento do objetivo geral fosse alcançado, foram propostos os seguintes objetivos específicos:

- Estruturar um modelo de avaliação de maturidade DevOps a partir do estado da arte da literatura e de práticas do mercado;
- Elaborar um *guideline* para avaliação de maturidade DevOps;
- Validar o modelo proposto por meio da aplicação de um estudo de caso; e em empresa de desenvolvimento de software.

1.4 JUSTIFICATIVA

O termo DevOps foi mencionado a primeira vez em meados 2008 durante uma conferência ágil no Canadá(Toronto) por Jez Humble e Patrick Debois, desde então surgiram diversas definições, alguns autores a identificam como sendo uma metodologia ou abordagem de desenvolvimento de software, método para a gestão de projetos de desenvolvimento de software (HUMBLE *et al.*, 2021), outros a trazem como sendo uma cultura organizacional e até mesmo um movimento dos profissionais de TI (SOUSA, 2019).

Apesar dos grandes benefícios, por ser uma cultura relativamente nova, um dos grandes desafios está em inclui-la dentro das empresas. Entre os principais motivos para a resistência está a falta de orientação estratégica da alta administração, falta de educação em processos DevOps, risco de desintermediação de papéis, resistência à mudança além da baixa exposição dos seus benefícios (HUSSAINI, 2014).

Outro ponto elencado é a infinidade de informações, práticas e ferramentas relacionadas, o que acaba gerando dúvidas de seu uso dentro das organizações e sua baixa aderência a cultura.

Tendo em vista essa situação, a justificativa para o presente estudo é a necessidade de se investigar as práticas de DevOps e modelos que buscam identificar os níveis de maturidade das práticas nas empresas, a partir de buscas na literatura e de um estudo de caso em empresa de desenvolvimento de software, visando propor um modelo para avaliação de maturidade DevOps para melhoria dos processos nas empresas.

1.5 ESTRUTURA DO TRABALHO

Este trabalho está dividido em cinco capítulos principais, cujo objetivo é detalhar o estudo a ser realizado. O Capítulo 2 apresenta a fundamentação teórica do estudo, considerando os principais conceitos necessários para o entendimento do estudo proposto. No Capítulo 3, apresenta trabalhos relacionados com os tópicos de pesquisa da presente monografia. No Capítulo 4, é apresentada a metodologia utilizada para a condução do trabalho e os passos para sua execução. Os resultados são apresentados no Capítulo 5 e, por último, são apresentadas as conclusões, no Capítulo 6.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ENGENHARIA DE SOFTWARE

Nas décadas de 1950 e 1960 os sistemas de softwares não possuíam processos e padrões bem definidos, sendo assim desenvolvidos de forma simples e artesanal. Esse período ficou conhecido como “crise do software”, devido aos programas desenvolvidos não atenderem as necessidades do usuário, sendo de baixa qualidade, não confiáveis e de difícil manutenção.

Atrelado a isso no final dos anos 60 houve um rápido crescimento do poder computacional e com a baixa dos custos foi possível utilizar os computadores para tarefas cada vez mais complicadas. Dessa forma no final dos anos 60 devido ao crescimento de demandas por softwares mais elaborados notou-se a necessidade por um padrão de desenvolvimento, sendo conhecido como “engenharia de software” (VICTORINO; BRÄSCHER, 2009).

Para contornar a chamada crise do software começou-se a instaurar fortemente os princípios da engenharia de software. O termo foi criado na década de 1960 e utilizado oficialmente em 1968 *na NATO Science Committee*, com a pretensão de dar um tratamento mais sistemático, controlado e de melhor qualidade ao desenvolvimento de programas (RANDELL; NAUR, 1969).

Para SOMMERVILLE (2011), “A engenharia de software é uma disciplina de engenharia que se preocupa com os aspectos da produção de software, desde sua concepção inicial até sua operação e manutenção”. Além disso o mesmo elenca que um bom programa deve possuir adaptabilidade, dependabilidade e segurança das informações.

Portanto, a engenharia de software (ES) engloba não apenas o desenvolvimento de programas, mas todo ciclo de vida, abrangendo também toda a documentação necessária para o desenvolvimento, instalação, uso e manutenção dos programas de modo que, ao final de cada uma destas etapas, um ou mais documentos são produzidos.

2.2 PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

A utilização de processos tem sido apontada como um fator primordial para o sucesso de empresas. Durante o desenvolvimento os processos auxiliam na estabilidade, controle e organização do software. Processo ainda pode ser definido como “uma metodologia para as atividades, ações e tarefas necessárias para desenvolver um software de alta qualidade” (PRESSMAN; MAXIM, 2021).

Os processos utilizados durante o desenvolvimento depende do tipo e das necessidades do programa. Para um processo ser de qualidade o mesmo deve possuir bem definido as atividades a serem executadas, os recursos e artefatos necessários, além dos procedimentos a serem

adotados. SOMMERVILLE (2011) aponta que independentemente do processo utilizado durante o desenvolvimento, possui-se quatro atividades fundamentais (SOMMERVILLE, 2011):

- **Especificação do software:** são definidos os requisitos e as restrições do software;
- **Desenvolvimento do software:** são efetuadas as implementações dos requisitos;
- **Validação de software:** é efetuada a certificação de que o sistema atende as necessidades e expectativas do cliente;
- **Evolução de software:** o software deve evoluir para atender os novos requisitos elencados pelos clientes.

Por não existir um processo de desenvolvimento padrão as empresas adaptam os modelos existentes conforme as suas necessidades, auxiliando também na avaliação dos objetivos, forças e fraquezas (SOMMERVILLE, 2011).

2.2.1 METODOLOGIAS DE DESENVOLVIMENTO TRADICIONAIS

No final da década de 1960, a demanda por sistemas mais completos mostrou que uma abordagem informal no processo de produção de programas não estava suprindo a necessidade das empresas. Com as novas exigências criou-se um novo modelo de desenvolvimento, considerado tradicional, cercado de normas e padrões de desenvolvimento (VICTORINO; BRÄSCHER, 2009).

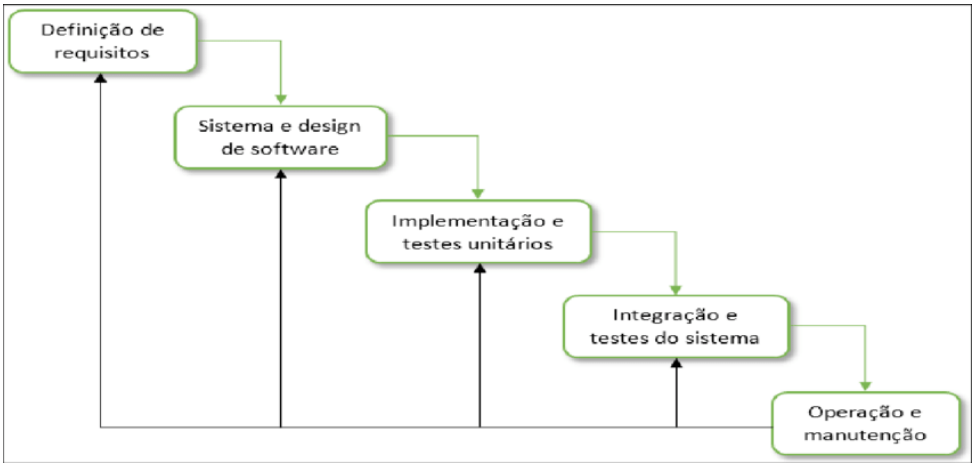
As metodologias tradicionais são conhecidas também como pesadas ou orientadas a documentação, por seguir processos sequencias. No contexto atual utiliza-se o desenvolvimento tradicional comumente em sistemas que possuem requisitos estáveis (SOARES, 2004).

O modelo cascata foi o primeiro modelo de desenvolvimento proposto no início da década de 1970. Também chamado de clássico ou linear, possui como principal característica a divisão das tarefas em etapas predeterminadas, dessa forma a próximo passo do processo somente será executado quando seu antecessor estiver finalizado (VICTORINO; BRÄSCHER, 2009).

Como citado anteriormente os processos de software são as atividades envolvidas na produção de um programa, assim sendo as metodologias podem ser definidas como representações abstratas desses processos. A partir dos modelos de processos podem ser definido as atividades para o desenvolvimento do software, especificar os produtos de cada atividade e a função dos envolvidos nos processos (SOMMERVILLE, 2011).

As metodologias tradicionais de desenvolvimento evoluíram, mas seu processo continua inflexível, com etapas predefinidas que devem ser cumpridas de maneira sequencial, isso acaba sendo um problema devido às constantes mudanças que acontecem no meio do processo de

Figura 1 – Representação do Modelo Cascata



Fonte: SOMMERVILLE (2011).

desenvolvimento. Dessa forma as empresas estão utilizando de metologias com processos mais flexíveis, principalmente as metodologias ágeis (MUNIZ *et al.*, 2021).

2.2.2 METOLOGIAS ÁGEIS

As metologias ágeis surgiram constituindo uma nova classe de metodologias com processos mais ágeis, leves e ciclos de desenvolvimento cada vez mais curtos visando entregas mais frequentes, além de permitir mudanças durante o desenvolvimento (ABRAHAMSSON *et al.*, 2003). A filosofia do pensamento ágil se resume em "maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo" (BECK *et al.*, 2001).

O termo "Metodologias Ágeis" tornou-se popular em 2001 quando dezessete especialistas em processos de desenvolvimento de software representando diversos métodos ágeis se reuniram com o intuito de discutir maneiras de melhorar o desempenho de seus projetos. Nessa reunião concluíram que havia um conjunto de princípios que quando aplicados aumentava-se a taxa de sucesso dos projetos, sendo esses princípios elencados abaixo (PONTES; ARTHAUD, 2018):

- Indivíduos e interação entre eles mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração do cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

Na metodologia ágil, processos, ferramentas, documentação e planejamento ainda estão presentes no desenvolvimento, mas o que se difere das metodologias tradicionais são o enfoque e os valores. O enfoque do modelo ágil esta nas pessoas e não em processos ou algoritmos.

Dessa forma o esforço maior está em gastar menos tempo com documentação e mais com a implementação (SOARES, 2004).

Para ser considerada ágil, a metodologia deve ser adaptativas ao invés de preditivas. Dessa forma elas se adaptam a novas mudanças como alteração dos requisitos decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer durante o desenvolvimento (SOARES, 2004).

2.2.3 DEVOPS - DESENVOLVIMENTO E OPERAÇÕES

Embora as metodologias ágeis tenham revolucionado o desenvolvimento de software, um problema ainda persiste, o conflito que existe entre as equipes de desenvolvimento e operações, com isso surge uma nova cultura chamada DevOps .

Atualmente utilizando-se de práticas, pilares e processos, o DevOps está permitindo a entrega de um software com mais rapidez e qualidade e consequentemente aumentando a satisfação do cliente, gerando mais lucro as empresas (MUNIZ *et al.*, 2021).

Ao analisar a estrutura organizacional de uma empresa nota-se uma divisão entre desenvolvimento e operações. Enquanto o time de desenvolvimento se ocupa da parte da produção do software bem como a sua manutenção, a parte de operações é responsável por gerenciar essas mudanças em produção. Ocasionalmente essa divisão acaba gerando conflitos de modo que uma equipe é responsável por novas implementações a outra pela preza pela estabilidade do sistema.

Como forma de diminuir os atritos entre as equipes, o objetivo do DevOps é possibilitar a aproximação dessas equipes, transformando os silos existentes num conjunto de times que trabalham em prol da organização e não em torno da sua atividade dentro dela (MUNIZ *et al.*, 2021).

A cultura DevOps propõe um conjunto de práticas e pilares principais para alcançar a melhoria contínua dentro das empresas:

2.2.3.1 PILARES DO DEVOPS

Colaboração(*Collaboration*), afinidade(*Affinity*), ferramentas(*Tools*) e dimensionamento (*Scaling*), são elencados como os quatro pilares para a implementação efetiva do Devops, que juntos promovem mudanças e soluções de problemas culturais e técnicos que podem impactar no desenvolvimento, qualidade e rapidez do software, produzido:

- **Colaboração:** é o processo de atingir objetivos em comum com a colaboração e interação de pessoas com diferentes experiências. Esse sentimento de colaboração deve te inicio entre os membros da mesma equipe e posteriormente expandir para outras equipes. Profissionais que não estejam trabalhando bem de forma individual ou in-

ternamente terão mais dificuldades na colaboração com indivíduos de outros setores (GARCIA, 2020);

- **Afinidade:** é o processo de incentivo da colaboração entre as equipes, considerando os objetivos e métricas pessoais de cada integrante, mantendo em mente os objetivos organizacionais (GARCIA, 2020);
- **Ferramentas:** ferramentas dentro do DevOps são utilizadas para melhorar os procedimentos de desenvolvimento de software, além de funcionarem como impulsionadores da mudança cultural, quando escolhidas corretamente. É importante entender o valor dessas ferramentas e o impacto que elas terão sobre o fluxo de trabalho existente para evitar que tragam problemas aos times (GARCIA, 2020); e
- **Dimensionamento:** é referente a adaptação dos pilares e processos conforme a expansão ou diminuição do crescimento das empresas, isso por que existem particularidades culturais e técnicas para organizações de tamanho diferentes (GARCIA, 2020).

2.2.3.2 CULTURA, AUTOMAÇÃO, *LEAN*, MEDIÇÃO E COMPARTILHAMENTO

A sigla CAMS surge das 4 principais praticas (*Culture, Automation, Measurement* e *Sharing*), mais tarde esse termo foi aperfeiçoado por Jez Humble com a inclusão do L para destacar a importância do Lean para a melhoria contínua. A estrutura CALMS compreende todas partes as partes interessadas no DevOps, contemplando a parte de negócios, operações, qualidade, segurança, desenvolvimento, implantação e integração. Sendo utilizada para avaliar o sucesso e maturidade do DevOps dentro das empresas (MUNIZ *et al.*, 2021).

- **Cultura:** para que os objetivos sejam alcançados é necessária uma cultura de compartilhamento de conhecimento e respeito entre os Dev e Ops, juntando pessoas com diferentes conhecimentos, mas um objetivo em comum (MUNIZ *et al.*, 2021);
- **Automação:** além da mudança cultural o DevOps foca bastante, na prática, da automação em processos rotineiros e repetitivos, de modo a garantir uma melhor qualidade do produto final e redução com retrabalho (MUNIZ *et al.*, 2021);
- *Lean*: dentro do DevOps a prática Lean é citada como uma filosofia colaborativa para redução de desperdícios e entregas de valor auxiliando as empresas a atingirem maior nível de maturidade (MUNIZ *et al.*, 2021);
- **Medição:** a medição dos dados é muito importante para o DevOps, para a estimativa de melhorias e verificação de processos com gargalos (MUNIZ *et al.*, 2021). Segundo (MOLESKY; HUMBLE, 2011) “a medição inclui o monitoramento de métricas de negócios de alto nível, como receita ou transações de ponta a ponta por unidade de tempo”;

- **Compartilhamento:** compartilhar conhecimento, ferramentas, *feedbacks*, é um dos pilares do DevOps, com essa troca toda a equipe tem noção sobre o processo e quais as novas funcionalidades que estão sendo implementadas. O compartilhamento de ferramentas e técnicas de desenvolvimento para gerenciar ambientes e infraestrutura também é uma parte fundamental do DevOps (MUNIZ *et al.*, 2021).

A Figura 2 reúne alguns pontos fundamentais e características de cada elemento do acrônimo CALMS (ALVES, 2020).

Figura 2 – Representação das práticas CALMS

Fonte: ALVES (2020).

2.2.3.3 PRATICAS DEVOPS

O sucesso do modelo DevOps depende da adoção de boas práticas por parte de toda a equipe. São ações que ajudam a acelerar, automatizar e otimizar cada uma das fases do ciclo de vida do software. Consideram-se as principais práticas a integração contínua (*continuous*

integration), entrega continua (*continuous delivery*), implantação contínua (*continuous deployment*) e monitoramento Contínua (*Continuous Monitoring*) (MUNIZ *et al.*, 2021).

- **Integração Contínua:** na cultura DevOps se utiliza muito a prática da integração continua, em que cada participante do time integra seu trabalho ao menos uma vez no dia, dessa forma cada integração é verificada por um *build* automatizado de modo a detectar erros de imediato, permitindo que as tarefas de desenvolvimento de software tenham mais qualidade e agilidade (MUNIZ *et al.*, 2021);
- **Entrega Continua:** essa prática normalmente envolve atividades como análise de código, teste de aceitação, verificação de requisitos, sendo considerada a evolução da integração continua garante a entrega de software da equipe de desenvolvedores para o ambiente de produção em um processo confiável, previsível, visível, automatizado com riscos quantificáveis e compreendidos (MUNIZ *et al.*, 2021);
- **Implantação Contínua:** essa prática compreende o processo de implantar as mudanças, advindas da entrega contínua, no ambiente de produção regularmente, após a execução com sucesso dos testes automatizados. A ideia é disponibilizar pequenas mudanças em produção com maior frequência. Deste forma, é possível aumentar a produtividade e diminuir os ciclos de *release* (MUNIZ *et al.*, 2021); e
- **Monitoramento Contínuo:** monitoramento é uma prática importante na entrega contínua onde permite a identificação quando um serviço está indisponível, fornecendo um sinal para que o problema seja investigado e resolvido (FERNANDES *et al.*, 2018).

2.3 MODELOS DE MATURIDADE DE SOFTWARE

Ao longo dos anos foram sendo criados inúmeros modelos visando produzir um software de qualidade. Com isso foram propostos modelos de maturidade de software, os primeiros modelos datam da década de 60, mas estes acabaram não fazendo muito sucesso por não atenderem as necessidades das organizações produtoras de software de forma satisfatória (MEZZENA; ZWICKER, 2007). Mais tarde, em 1991 surgiu um novo modelo intitulado como *Capability Maturity Model* – CMM (MACHADO; AMÊNDOLA, 2004).

O modelo CMM foi criado através de uma iniciativa do instituto SEI (*Software Engineering Institute*), da Carnegie Melon *University*, a partir de uma solicitação do governo dos Estados Unidos, cuja função era de avaliar a capacitação dos fornecedores de software contratados pela Força Aérea Norte Americana.

Atualmente este modelo é utilizado para diagnosticar, avaliar e melhorar a maturidade dos processos empresariais além de identificar as questões mais críticas, focando em um conjunto limitado de atividades, os quais possibilitam ganhos contínuos e duradouros na capacidade dos processos de software (SILVEIRA, 2009).

Para esse modelo o processo de melhoria é gradativo com várias pequenas etapas, dividido em 5 níveis de maturidade que permitem estratificar a posição ocupada pela companhia desenvolvedora de softwares com relação à maturidade de seus processos de gerenciamento de projetos, tais níveis são definidos como (MEZZENA; ZWICKER, 2007):

- **Inicial:** neste nível os processos são caracterizados como caóticos, não há nenhuma metodologia implementada e o sucesso depende de esforços pessoais dos colaboradores;
- **Repetível:** esse nível é considerado disciplinado, práticas de gestão de projetos estão bem estabelecidas, embora em toda a organização os processos possam não existir;
- **Definido:** nesse nível estão presentes padrões de projetos integrados assim toda a organização pode aprender com diferentes trabalhos, aprimorando cada vez mais o seu trabalho. Além disso, todos os projetos utilizam uma versão aprovada e adaptada do processo padrão de software da organização para o desenvolvimento e manutenção;
- **Quantitativamente Gerenciado:** nesse nível são realizadas medições de forma detalhada do processo e da qualidade do produto produzido. A organização passa a ter uma gestão feita com bases quantitativas; e
- **Otimizado:** processo melhorado continuamente, possibilitada por meio do *feedback* quantitativo dos processos, de modo a reduzir drasticamente o retrabalho e desperdício.

2.3.1 CMMI

Nas pequenas e grandes organizações produtoras de software, são encontrados diversos fatores que influenciam na geração de um produto de qualidade. Os autores VASCONCELOS e MORAIS (2012) destacam alguns problemas corriqueiros encontrados nesse ciclo, tais como "prazos e orçamentos não cumpridos, insatisfações dos clientes, produtos com erros, entre outros". Para de amenizar essas situações muitas empresas estão adotando o CMMI, ajudando as organizações a desenvolver software de alta qualidade de forma mais eficiente.

Assim como o CMM, o modelo CMMI (*Capability Maturity Model Integration)* foi criado através de uma iniciativa do instituto SEI, sendo considerado seu sucessor, mas com a vantagem de sua aplicação ser empregada nas mais diversas áreas e processos (VASCONCELOS; MORAIS, 2012). Como base para a sua criação foi utilizado a combinação de três modelos, *Capability Maturity Model for Software* (SW-CMM). Mais informações sobre os modelos Capability Maturity Model for Software (SW-CMM) e Integrated Product Development Capability Matutity Model (IPD_CMM) podem ser visto nos *links*[1] (SOUSA, 2009).

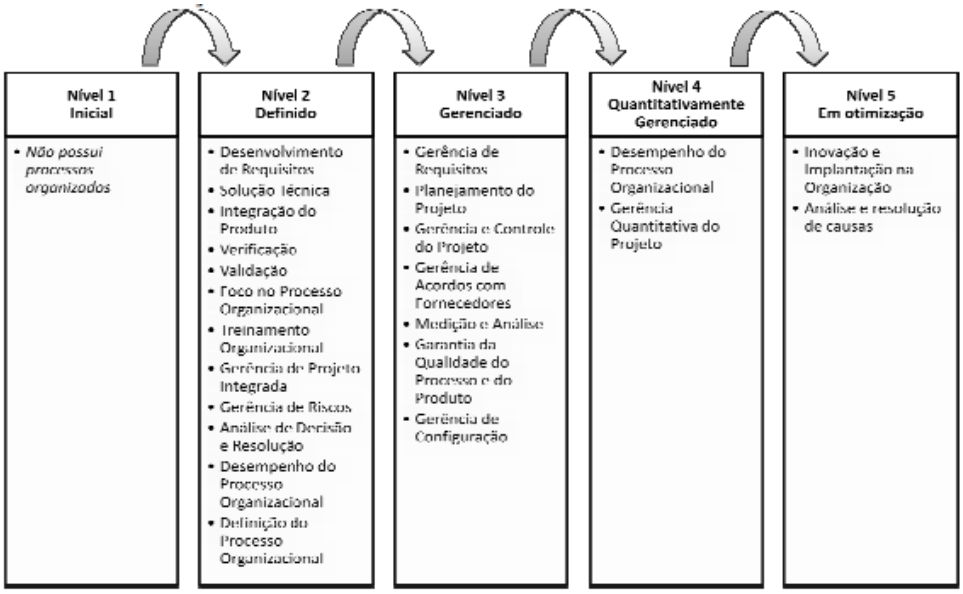
[1] http://archive.adaic.com/ase/ase02_01/bookcase/se_sh/cmms/systems_engineering/eia_731_secm/SECMovrvw_lkd.pdf e https://www.researchgate.net/publication/285405401_The_Integrated_Product_Development_CMM_SM

Existem três tipos de análise sobre os processos de TI os quais apresentam o quanto uma organização é madura para desenvolver e manter os produtos/serviços (ATAÍDES; GUEDES, 2006):

- **Capacidade:** essa analise apresenta o potencial de resultados que podem ser conquistados com o uso de processos CMMI;
- **Desempenho:** realiza a análise da situação dos processos atuais e os resultados adquiridos pelo seu uso; e
- **Maturidade:** realiza a análise dos níveis que um processo específico está definido, gerenciado, mensurado, controlado e efetivo. Também está ligado ao potencial de crescimento consistente utilizando processos de software em projetos.

Para a avaliação são utilizados cinco níveis de maturidade, os quais definem uma escala ordinal que medirá a maturidade dos processos de softwares e avaliará sua capacidade, além de auxiliar na priorização de esforços e melhorias (ATAÍDES; GUEDES, 2006). Na Figura 3 estão representadas as áreas-chave de cada nível do modelo CMMI.

Figura 3 – Áreas-chave de Processos do CMMI - Representação por estágios



Fonte: Ataídes e Guedes (2006).

A versão 1.3 do CMMI®... (2010) definem esses níveis como:

- **NÍVEL 1 - INICIAL:** Caracterizado como um processo executado, ou seja, ele cumpre as metas propostas das áreas de processos. Apesar de existirem melhorias importantes neste nível, estas podem ser perdidas ao decorrer do tempo se não forem institucionalizadas.

- **NÍVEL 2 - DEFINIDO:** Recebe essa classificação, pois o mesmo é executado de acordo com políticas propostas, dessa forma ajuda a garantir que práticas existentes sejam mantidas durante períodos de estresse. Além disso, estes processos passam a ser monitorados, controlados, revisados e medidos quanto sua aderência.

- **NÍVEL 3 - GERENCIADO:** Esse nível é considerado como definido, com processos gerenciados de forma proativa e adaptados a partir de um conjunto de processos padronizados, portanto, são mais consistentes. Além disso, estes são descritos com maior rigor que no nível 2, especificando clareamento o seu propósito, os critérios de entrada quais as atividades, papéis e medidas, também são verificados os critérios de saídas.

- **NÍVEL 4 - QUANTITATIVAMENTE GERENCIADO:**

  Caracterizado como um processo gerenciado quantitativamente, baseados nas necessidades dos clientes, usuários finais, organizações bem como os implementadores de processos. O gerenciamento é realizado através da análise dos níveis de qualidade e desempenho estatisticamente, estes são aplicados em projetos com maior valor geral de negócio. Posteriormente os dados das coletas dos níveis são usados em técnicas e previsões.

- **NÍVEL 5 - EM OTIMIZAÇÃO:**

  Neste nível os processos são melhorados continuamente de forma incremental e inovadora, com base na compreensão quantitativa dos objetivos dos negócios e necessidades de desempenho. A resultância da aplicabilidade dos processos implantados são medidos estatística e quantitativamente, seguido da comparação com os objetivos de qualidade e desempenho propostos. A análise dos dados identifica deficiências ou lacunas no desempenho. Essas lacunas são utilizadas para impulsionar a melhoria do processo organizacional. São plicados grandes esforços em entender e controlar o desempenho no nível do subprocesso para posteriormente utiliza-las para gerenciar projetos.

2.3.2 MÉTRICAS

Atualmente as empresas vivem períodos de grandes mudanças e adaptações na forma de desenvolver software, devido as grandes dificuldades na comunicação entre as equipes de desenvolvimento e operações e como consequência impactando na produtividade. Obter métricas e informações sobre um projeto desenvolvido é algo comum ao desenvolvimento de software, sendo utilizadas para avaliar situações, acompanhar o resultado de um processo, ação ou estratégia específica (NORMAN; JAMES, 2020).

Além disso, a análise e medição é fundamental para tomadas de decisões, dando apoio as diferentes visões, tanto no projeto, como no produto ou para o processo de produção do sistema

(SANTOS, 2019). Quanto mais cedo uma organização iniciar a medição, mais cedo uma linha de base será estabelecida e poderá ser usada para avaliar a melhoria relativa.

Com o intuito de auxiliar no processo de medição a empresa Google criou um grupo de pesquisa denominado DORA *Metrics* o qual propõem o acompanhamento de cinco métricas que apresentam o desempenho das equipes DevOps dos softwares desenvolvidos pela organização (GOOGLE CLOUD, 2014).

2.3.3 DORA *METRICS*

*DevOps Research and Assessment (*DORA*)*, ou em tradução livre "*Pesquisa e Avaliação de DevOps"*, foi criada em 2014 com o intuito de avaliar o desempenho das equipes DevOps para identificação de possíveis pontos de melhorias ou de mudanças mais profundas (PEREIRA, 2022). Nesta pesquisa são examinados como as equipes desenvolvem, entregam e operam sistemas de software. Através de algumas perguntas são definidos os níveis de desempenho podendo ser: elite, alto, médio e baixo desempenho (SMITH *et al.*, 2021).

Dentro do DevOps as chamadas métricas DORA são utilizadas para é avaliar o grau de implementação da cultura com base em um conjunto de questões que permitem conhecer o percentual de conformidade com as práticas, dimensões e valores do DevOps. Através de dados objetivos a DORA *Metrics* propôs quatro métricas que demonstram o desempenho das equipes DevOps e dos softwares envolvidos na empresa que são *Deployment Frequency, Lead Time for Changes, Mean Time to Recover e Time to Restore Service,* e uma métrica que mede o desempenho operacional denominada *Reliability* (SMITH *et al.*, 2021):

- *Deployment Frequency:* utilizada para medir a frequência com que o novo código é implementado na produção pela equipe. Esta métrica está ligada ao rendimento da equipe;
- *Lead Time for Changes:* utilizada para medir o tempo demandado para que a equipe DevOps desenvolva melhorias ou correções nas suas aplicações.
- *Mean Time to Recover:* utilizada para mediar as taxas de falhas nas adaptações e mudanças propostas pela equipe DevOps;
- *Time to Restore Service:* utilizada para medir o tempo necessário para que um serviço se recupere de uma falha. Está métrica está ligada a estabilidade do sistema; e
- *Reliability:* utilizada para medir o grau em que uma equipe pode cumprir promessas e afirmações sobre o software que opera.

Na Figura 4 estão disponíveis as perguntas relativas ao desempenho nas entregas e os níveis que cada equipe pode ter dependendo das respostas:

Figura 4 – Métricas de desempenho de entregas de software

Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Fonte: SMITH *et al.* (2021).

2.4 REVISÃO DA LITERATURA

Com o intuito de investigar o estado da arte de práticas DevOps nas empresas, foi desenvolvida uma pesquisa referente a publicações sobre o tema, conforme apresentado a seguir na Tabela 1.

Tabela 1 – Trabalhos relacionados identificados.

Trabalhos	Objetivo
PERERA, BANDARA e PERERA (2016)	Este trabalho buscou identificar os principais desafios na adoção do DevOps. Como resultado foi observado a resistência à mudança do processo existente, alto custo para a implantação e a dificuldade da mudança cultural principalmente por parte dos desenvolvedores.
RONG, ZHANG e SHAO (2016)	Neste trabalho procurou-se avaliar a viabilidade de aplicar modelos CMMI para orientar a melhoria de processos para DevOps e identificar possíveis ajustes do modelo CMMI. Para a análise foi utilizada somente as áreas de processo que residem nos níveis Gerenciado e Definido dos modelos CMMI-DEV e CMMI-SVC.
BATRA e JATAIN (2020)	Este trabalho procurou apresentar os ganhos de produtividade ao utilizar DevOps em relação ao modelo cascata, juntamente com uma medição de qualidade através da comparação de duas equipes, cada um utilizando um modelo.
TOMAS e JINGYUE (2019)	Este trabalho evidencia os principais desafios das práticas de segurança na cultura DevSecOps realizada com seis desenvolvedores. Como resultado foram elencados a falta de comprometimento das empresas e responsáveis, insatisfação e inexperiências por parte dos desenvolvedores e a escassez de padrões de desenvolvimento seguro.
GASPARAITE e RAGAIŠIS (2019)	Neste trabalho é apresentado uma análise das semelhanças entre os modelos de maturidade.
RADSTAAK (2019)	Neste trabalho é proposto uma nova forma de avaliar a maturidade das empresas em relação ao DevOps e orientar as organizações para o nível de maturidade necessário. Esta nova proposta de avaliação combina o modelo CMMI comum, juntamente com os recursos citados na literatura através da revisão.
ZAROUR *et al.* (2019)	Complementando os artigos citados anteriormente a pesquisa denominada identifica e compara os modelos de maturidade de DevOps disponíveis na literatura. Entre os modelos comparados no estudo estão, modelo de Maturidade Bahrs da IBM DevOps, modelo de Maturidade Mohamed, modelo de maturidade Capgemini DevOps, modelo de maturidade Bucena DevOps, modelo de maturidade Eficode e o modelo de maturidade Feijter.
NEUBRAND e HAENDLER (2020)	No estudo é apresentado uma abordagem baseada em GQM (Goal Question Metric) para avaliação o grau de maturidade DevOps nas empresas. O modelo proposto é composto por um questionário com 98 perguntas relativas à cultura DevOps, estruturadas em objetivos e subjetivos sendo avaliadas através de perguntas a serem respondidas pelos profissionais.
ANISETT e ARDAGNA (2019)	No artigo é proposto uma forma de melhorar a qualidade do produto, através das chamadas avaliações continuas para DevOps. O processo de garantia é orientado por um modelo de certificação que aciona um conjunto de ações de garantia em cada etapa do processo de desenvolvimento de DevOps .

Fonte: Autoria própria.

3 METODOLOGIA

Este capítulo possui como objetivo apresentar os elementos pertencentes a este trabalho, divididos em 4 seções. Primeiramente, é apresentada a caracterização da pesquisa, na Seção 3.1. Então, é explicado o conceito de estudo de caso na Seção 3.2 e, em seguida, o *survey* desenvolvido é descrito, na Seção 3.3 e a metodologia de criação do *guideline* de maturidade proposta na Seção 3.4.

3.1 CARACTERIZAÇÃO DA PESQUISA

A pesquisa consiste em um estudo de caso em conjunto a um *survey* realizado em uma empresa de desenvolvimento de software localizada no sudoeste do Paraná. O estudo abrangeu dois setores, o primeiro trabalha com a criação de produtos novos, denominado setor A e o segundo com produtos legados, denominado setor B, após a identificação dos setores e equipes que utilizam DevOps foi aplicado um *guideline* para avaliação de maturidade.

3.2 ESTUDO DE CASO

Um estudo de caso tem como foco o entendimento e a produção de conhecimento dos fenômenos grupais, organizacionais, sociais, políticos e relacionados, através da coleta e análise de dados, sendo geralmente organizado em torno de um pequeno número de questões que se referem ao como e ao porquê da investigação (YIN, 2015).

Desse modo para o trabalho proposto foi executado um estudo de caso qualitativo em uma empresa real de desenvolvimento de software, utilizando-se de múltiplas fontes de coletas, tais como, entrevistas presenciais e remotas e aplicações de formulários. Uma explicação mais abrangente do estudo de caso conduzido é apresentado nas subsequentes Seções.

3.3 *SURVEY*

Possui-se três abordagens de pesquisa sendo elas qualitativas, quantitativas e métodos mistos. A metodologia qualitativa exige um estudo amplo do objeto de pesquisa, considerando o contexto em que ele está inserido e as características da sociedade a que pertence sendo estruturada pelo uso de palavras. Enquanto na quantitativa utiliza-se de números para sua estruturação, buscando resultados objetivos e palpáveis, baseados em experimentos, e possíveis de serem quantificados. A abordagem mista acaba incorporando às duas formas pressupondo que a integração das duas gera uma compreensão que vai além dos resultados fornecidos pelas abordagens quantitativas e qualitativas (CRESWELL; CRESWELL, 2021).

O *survey* utilizado no trabalho atual apresenta caráter descritivo, visando compreender como o DevOps está presente na empresa, utilizando-se de entrevistas e questionários para o levantamento dos dados. Neste cenário, foram analisadas as práticas, métodos e como a cultura está presente nas equipes, não buscando explicar o porquê do uso de determinadas técnicas, mas sim qual a distribuição de uso das mesmas.

O questionário de investigação baseados no *guideline* modelo de maturidade proposto foi enviado por e-mail para ser respondido pelas equipes, sem identificação dos respondentes, via formulários online.

Os resultados foram coletados e analisados sendo dispostos em gráficos apresentados no capítulo 5, seção de resultados do trabalho.

3.4 DESENVOLVIMENTO DA PROPOSTA DO MODELO DE MATURIDADE DEVOPS

Os níveis maturidade dos processos de uma empresa podem variar conforme as práticas DevOps utilizadas, além disso, dentro da mesma empresa podem ser encontrados processos com alto nível de maturidade, enquanto outros com um baixo nível. Com base nessa discrepância de maturidade foi elaborado um modelo que permite verificar os níveis de implementação de diferentes processos com base em algumas métricas.

Para definir um modelo de métricas homogêneo e unificado para avaliar o DevOps, optou-se por realizar um processo de harmonização de três modelos, buscando relacionar as principais práticas DevOps em um só, propondo a criação de um modelo simplificado e de fácil aplicação nas empresas.

Na Figura 5 possível visualizar o fluxo para criação e aplicação do modelo de maturidade.

Com base teórica foram estudados a fundo os 3 artigos citados na Tabela 1, abrangendo ainda, de forma complementar, conceitos presentes nos trabalhos descritos na revisão narrativa desenvolvida no Capítulo 2. OS Trabalhos utilizados na elaboração da proposta são apresentados na Tabela 2.

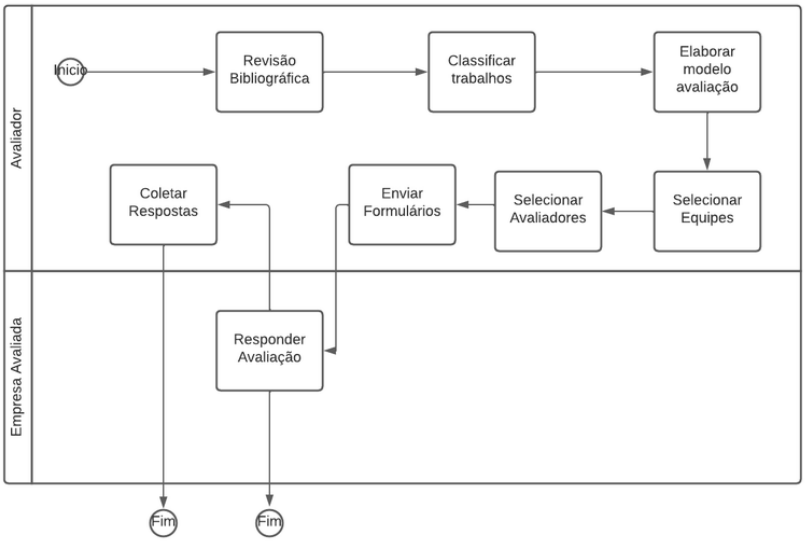
Tabela 2 – Trabalhos baseados para elaboração do modelo de maturidade.

Referência	Trabalho
ARAUJO (2018)	Grafos de Maturidade.
LEVITA (2018)	Proposta de Modelo para Avaliação da Maturidade DevOps.
OROZCO e CALVACHE (2022)	Modelo de métricas para apoyar la evaluación de DevOps en empresas de desarrollo de software.

Fonte: Autoria própria.

Considerando sua importância para o trabalho proposto, os artigos serão descritos detalhadamente nas próximas Seções 3.4.1, (**??**) e (**??**).

Figura 5 – Fluxograma de criação e aplicação do modelo de maturidade



Fonte: Autoria própria.

3.4.1 GRAFOS DE MATURIDADE (ARAUJO, 2018)

Nível de maturidade é um estágio evolutivo bem definido em busca de um processo de software maduro. Cada nível de maturidade fornece um apanhado elementos para a melhoria contínua do processo (JUCÁ JUNIOR; CONFORTO; AMARAL, 2010). Dessa forma a utilização de modelo de maturidade em projetos tem por objetivo, portanto, medir o estágio de organização das empresas na gestão de seus projetos e apontar formas para a melhoria desse nível de organização.

Como forma de avaliar o nível de implementação DevOps nas empresas, o autor ARAUJO (2018) elaborou um modelo de avaliação de maturidade baseado em algumas métricas DevOps retiradas do livro “Manual de DevOps: como obter agilidade, confiabilidade e segurança em organizações tecnológicas” de (KIM, 2018).

O modelo propõe uma avaliação com cinco níveis de maturidade:

- **Inicial:** neste nível as entregas são feitas de forma semestral ou anual, retrabalho acima de 30% dos projetos e há grande discordância entre os setores de desenvolvimento, qualidade e operações;
- **Consciente:** neste nível a demanda de esforços para retrabalhos são menores, maior usabilidade de ferramentas de automação em projetos pilotos e com menor divergências entre as equipes de desenvolvimento, qualidade e operações;
- **Gerenciado:** demandas de esforços em retrabalhos em cerca de 15 a 30%, entregas em homologação são realizadas de forma semanal e em produção de forma mensal;

- **Avançado:** as práticas DevOps, integração, entrega, implantação e monitoramento contínuos estão presentes na maioria dos projetos existentes, e os novos já são desenvolvidos com o uso das práticas e na maioria de forma automatizada através de ferramentas; e
- **Melhoria Contínua:** possui dominância no uso de práticas DevOps, o tempo demandado em retrabalhos está abaixo de 15%, entregas em homologação são realizadas diariamente e em produção de forma semanal e os times de operação, qualidade e desenvolvimento trabalham em união.

Um ponto a destacar é que o nível de melhoria contínua não é o último, nele o time atingiu uma maturidade tão alta que melhorar é uma tarefa presente no dia a dia. Além dos níveis de maturidade, o modelo é composto por oito categorias diferentes:

- Qualidade de Código;
- Configuração como Código;
- Gestão de *Builds;*
- Gestão de Testes;
- Testes de Carga;
- Gestão de Configuração;
- Gestão de *Releases; e*
- Monitoração de Aplicações.

3.4.2 PROPOSTA DE MODELO PARA AVALIAÇÃO DA MATURIDADE DEVOPS (LEVITA, 2018)

Assim como a pesquisa avaliativa “Grafos de Maturidade” criada por ARAUJO (2018), o autor LEVITA (2018), propõe um modelo que busca analisar a maturidade DevOps nas empresas por meio de algumas perguntas.

Através da revisão da literatura em formato guarda-chuva foi estruturada uma teoria que buscou relacionar as principais práticas DevOps e posteriormente para cada prática foram definidos alguns constructos da pesquisa, as quais são justamente os objetos a serem mensurados por este modelo. A avaliação é formada por perguntas e respostas graduadas de 1 a 5, com base nas respostas é possível avaliar o quanto a empresa esta aderente as abordagens DevOps no desenvolvimento e implementação de aplicações.

Como resultado da construção do referencial teórico, foram estabelecidos os constructos do modelo, elaborados com base nas práticas DevOps que foram identificadas (LEVITA, 2018):

- Pessoas e Cultura: colaboração e estratégia;
- Planejamento Contínuo: requisitos, planejamento e painel de controle;
- Integração Contínua: gerenciamento, colaboração e painel de controle;
- Testes Contínuos: gerenciamento, automação e painel de controle;
- Entrega Contínua: gerenciamento, automação e painel de controle;
- Infraestrutura como código: gerenciamento, automação e painel de controle;
- Monitoração Contínua: gerenciamento e capacidade;
- Feedback Contínuo: feedback e otimização.

3.4.3 MODELO DE MÉTRICAS PARA APOYAR LA EVALUACIÓN DE DEVOPS EN EMPRESAS DE DESARROLLO DE SOFTWARE DE (OROZCO; CALVACHE, 2022)

Os autores OROZCO e CALVACHE (2022) propõe um novo modelo de avaliação do grau de implementação de DevOps baseado na abordagem GQM, estabelecendo métricas necessárias para avaliar o grau de implementação das práticas fundamentais de DevOps propostas na literatura, de forma a apoiar as organizações a identificar oportunidades de melhoria em seus processos DevOps.

O modelo foi elaborado a partir da revisão da literatura utilizado o mapeamento sistemático filtrando por trabalhos relacionados á avaliação do grau de implementação de DevOps em organizações produtoras de software Para a criação do modelo foram unificadas as práticas capturadas na revisão sistemática, que posteriormente passou pela avaliação de especialistas membros da comunidade acadêmica. Este modelo conta com um total de 12 práticas fundamentais, 6 práticas complementares, 4 dimensões e 4 valores, seguindo uma estrutura hierárquica em que valores compõem o mais alto nível de abstração, de forma a garantir que a cultura DevOps seja realizada de forma adequada. Através da estrutura GQM ficou estabelecido que:

- **Nível Conceitual:** foram identificadas as dimensões práticas e valores propostas pelo DevOps;
- **Nível Operacional:** foram estabelecidas as questões relacionadas as práticas DevOps;
- **Nível Quantitativo:** foram estabelecidas um conjunto de métricas, em que com base nas respostas as questões descritas no nível operacional, permite conhecer o grau de implementação das práticas, dimensões e valores DevOps.

Como resultado da aplicação do GQM estabeleceu-se um conjunto de objetivos e questões associadas às práticas centrais e complementares, na Tabela 3 é possível visualizar o número de metas e questões resultantes e na Tabela 4 o resumo com o número de métricas por práticas, dimensões e valores:

Tabela 3 – Objetivos e questões obtidas após a aplicação do GQM.

Elementos do Processo	Objetivos	Perguntas
Práticas Fundamentais	42	72
Práticas Complementares	19	23

Fonte: Adaptado de OROZCO e CALVACHE (2022).

Tabela 4 – Número de métricas obtidas após a aplicação do GQM.

Não	Aspecto a Avaliar	Métricas
1	Práticas	7
2	Dimensões	2
3	Valores	2
	Total	**11**

Fonte: Adaptado de OROZCO e CALVACHE (2022).

As questões foram elaboradas utilizando escalas nominais com as opções de "Sim" que condiz a 100% e "Não" que condiz a 0%, seguindo os critérios expostos na Tabela 5.

Tabela 5 – Resposta preenchida no instrumento de avaliação

Resposta	Descrição
Sim	São apresentadas evidências suficientes e necessárias para garantir que o conjunto de atividades associadas à questão seja realizado corretamente. As evidências podem ser extraídas de diferentes mecanismos, como: (i) evidências diretas obtidas por meio da observação de uma prática, (ii) coleta de opiniões por cada uma das funções envolvidas na prática ou (iii) registros históricos consistentes que permitem evidências de conformidade com a prática.
Não	Essa resposta é dada em dois cenários: (i) a empresa não possui algum tipo de comprovação ou (ii) observase cumprimento parcial da prática, ou seja, não são atendidos todos os aspectos necessários para garantir que a questão seja respondida completamente.

Fonte: Adaptado de OROZCO e CALVACHE (2022).

Para a avaliação foram selecionadas diferentes áreas separadas como Práticas Fundamentais e Práticas Complementares, apresentadas no Quadro 1.

O processo de aplicação do modelo é realizado seguindo o seguinte conjunto de atividades: Planejamento da avaliação, Execução, Geração de Resultados, e por fim a Geração do relatório de resultados:

- Planejamento da avaliação: para o planejamento da avaliação o avaliador passa as informações da empresa, a data para a avaliação juntamente com as atividades a serem realizadas e o esforço e tempo necessários para realizar a avaliação;

Quadro 1 – Práticas Fundamentais e Complementares

Práticas Fundamentais	Integração Contínua Entrega Contínua Testes Contínuos Gerenciamento de Requisitos Gerenciamento de Dados Supervisão de Segurança Direção Estratégica Gerenciamento de Configuração Monitoramento e Observabilidade Contínuos Educação de DevOps FeedBack e Inovação Contínuos
Práticas Complementares	Medição de Cultura Implantação Contínua Infraestrutura como Código Gerenciamento de Acessos de Privilégios Aprendizado Contínuo Experimentação Contínua Satisfação no Trabalho

Fonte: Adaptado de OROZCO e CALVACHE (2022).

- Execução da avaliação: o avaliador leva os dados coletados com as informações necessárias para a aplicação das métricas;
- Geração de resultados: os dados são analisados pelo avaliador após a aplicação das métricas; e
- Relatório de resultados: o avaliador expõe um relatório detalhado dos resultados obtidos ao responsável pelo processo.

A partir dos dados coletados no estado da arte, realizou-se a análise das principais ferramentas existentes para a avaliação DevOps, identificando o nível de detalhamento da avaliação realizada por cada ferramenta, para isso foram seguidos alguns critérios.

- C1: a ferramenta realiza a avaliação do grau de adoção das práticas, dimensões e valores;
- C2: a ferramenta é disponibilizada de forma gratuita;
- C3: a ferramenta é suportada por um conjunto de métricas bem definidas, suportada por um consultor externo e por um modelo de referência;
- C4: a ferramenta fornece serviços de acompanhamento realizados por consultores especializados que fornecem informações adicionais sobre os resultados produzidos por cada ferramenta; e
- C5: nenhuma ferramenta é suportada por meio de modelos de referência.

Como resultado foram encontradas 15 ferramentas, apresentadas na Tabela 6.

Tabela 6 – Comparação de ferramentas para avaliar DevOps

Nombre	C1			C2		C3		C4		C5	
	Prac	Dim	Val	SI	NO	SI	NO	SI	NO	SI	NO
IVI's DevOps Assessment	X				X		X		X		X
DevOps Maturity Assessment	X			X			X		X		X
Microsoft DevOps Self-Assessment	X			X			X	X			X
Eficode	X				X		X		X		X
Infostretch	X			X			X		X		X
InCycle Evaluacion de devops	X			X			X		X		X
Veritis	X				X		X		X		X
Boxboat	X				X		X		X		X
IBM DevOps Self Assesment	X			X		X			X		X
DevOps Maturity Survey Report	X			X		X		X			X
Humanitec DevOps Assessment	X				X		X		X		X
Attlasian DevOps Assessment	X				X		X		X		X
DevOps Maturity model	X			X			X		X		X
DevOps Heath Radar Assessment	X	X		X		X					X
DORA DevOps Quick Check	X			X			X		X		X

Fonte: Autoria própria.

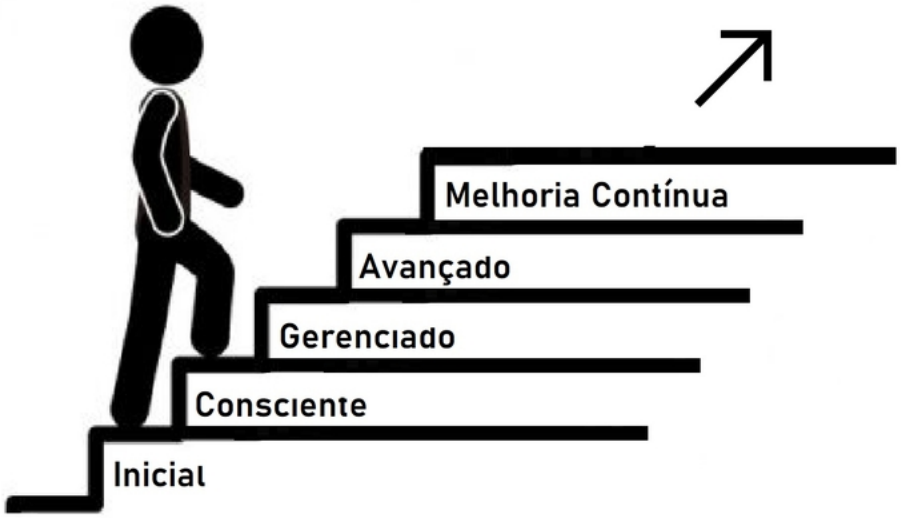
Estes 3 trabalhos serviram como base referencial para a proposta do modelo de avaliação de maturidade, objeto deste trabalho, a qual será apresentada na próxima Seção.

4 DESENVOLVIMENTO DA PROPOSTA

O enfoque desta proposta é apresentar um modelo que possa ser utilizado por pequenas e médias empresas, por meio de um *guideline* de fácil utilização, pois apesar de existirem muitos modelos de avaliação encontrados tanto na literatura acadêmica quanto ofertados pelo mercado, estes são mais voltados para grandes empresas e apresentam maior complexidade para compreensão.

O modelo desenvolvido conta com 5 níveis de maturidade abrangendo 10 categorias de processos. O primeiro estágio é Inicial, seguido do Consciente, Gerenciado, Avançado e por fim Melhoria Contínua conforme pode ser observado na Figura 7.

Tabela 7 – Escala do Modelo de Maturidade Proposto em Práticas DevOps



Fonte: Autoria própria.

Para a elaboração dos níveis de maturidade, foi utilizado o critério de ser de fácil aplicação e compreensão além de possuir uma abertura para a melhoria contínua. Para a definição das áreas DevOps avaliadas, foram analisadas o grau de cobertura dos outros modelos de forma a identificar possíveis lacunas, dessa forma o novo modelo abrange todas as áreas não cobertas entre os trabalhos. Em relação às práticas presentes em cada nível estas foram projetadas identificando os principais métodos de trabalho utilizados em cada grau de maturidade de forma que pude-sem ser aplicadas em pequenas e médias empresas de forma clara e objetiva.

Na Tabela 8 estão elencados os processos de cada nível que demonstram o grau de evolução em que uma organização ou equipe se encontra num determinado momento.

Além dos níveis de maturidade, o modelo propõe a separação de habilidades em dez diferentes categorias: (i) Pessoas e Cultura; (ii) Qualidade de Código; (iii) Configuração como Código; (iv) Gestão de *Builds*; (v) Gestão de Testes; (vi) Testes de Carga; (vii) Gestão de Con-

Tabela 8 – Níveis de Maturidade DevOps

Melhoria Contínua
- Melhoria contínua de processo e maestria no uso das práticas DevOps. - Entregas em homologação diariamente e em produção semanalmente, cooperação entre os times de desenvolvimento, qualidade e operações. - Alinhamento na qualidade da organização e o desempenho do processo, objetivos são estabelecidos e continuamente revisados, tomados como base para na gestão e melhoria de processos. - Foco no gerenciamento e aprimoramento do desempenho organizacional através da coleta de dados em vários projetos para identificar falhas nos processos e deficiências no desempenho.
Avançado
- Práticas DevOps estão em curso na maioria dos projetos e existe forte automação das práticas DevOps usando ferramentas de automação (ALM). - Novos projetos e produtos iniciam fazendo uso das práticas DevOps, com estimativas de desempenho do processo. - Criação de metas para a qualidade e desempenho dos processos as quais são utilizadas como base para gerenciamento de projetos.
Gerenciado
- Processos são bem caracterizados e compreendidos sendo descritos de forma padronizada, com o uso de procedimentos, ferramentas e métodos. - Práticas DevOps são utilizadas em projetos importantes, apresentando resultados de negócios reconhecidos pela empresa. - Entregas em homologação desenvolvidas de forma semanal e produção de forma mensal, com processos sendo gerenciados de forma mais proativa.
Consciente
- Busca pela redução do tempo de ciclo e retrabalhos, com ações iniciais em projetos pilotos, com ferramentas de automação. - Aproximação entre desenvolvimento, qualidade e operações, sem integração de equipes. - Processos são monitorados, controlados e revisados e os projetos são executados e gerenciados de acordo com seus planos documentados.
Inicial
- Entregas em bases semestrais ou anuais são comuns. - Conflitos entre desenvolvimento, qualidade e operações são comuns. - Não possui um padrão de processo definido para toda a empresa e não há comprometimento com padronização das entregas.

Fonte: Autoria própria.

figuração; (viii) Gestão de *Releases;* (ix) Monitoração de Aplicações; e (x) Requisitos de Software. A seguir, são descritas as práticas e suas relações com os níveis de maturidade propostos.

4.0.1 PRÁTICAS RELACIONADAS À PESSOAS E CULTURA

Na Tabela 9, estão representadas as práticas para avaliação dos níveis de maturidade da cultura DevOps, entre as equipes. Esta métrica busca avaliar o quanto a cultura está presente dentro da empresa, validando processos como a troca de conhecimento entre as equipes de desenvolvimento e operações, institucionalização das práticas DevOps em toda a organização e unificação de processos entre Dev e Ops.

Tabela 9 – Avaliação das práticas de Pessoas e Cultura

Melhoria Contínua
- Todos os processos são unificados entre Dev e Ops e as informações compartilhadas por meio de ferramentas integradas. - Os princípios e as práticas DevOps são institucionalizados em toda a empresa, com qualificação das equipes e alinhamento sobre a cultura DevOps. - A metodologia de entrega está alinhada fim-a-fim em toda a infraestrutura, com práticas de gestão do conhecimento compartilhadas.
Avançado
- Dev e Ops usam o mesmo conjunto de processos, mas alguns pontos ainda não estão unificados e as ferramentas são compartilhadas entre Dev, QA e Ops, mas não há um único fluxo entre elas. - Acontecem trocas de conhecimento regulares entre Dev e Ops e um consenso dentro da empresa acerca dos princípios básicos do Dev Ops. - Vários times têm total experiência e habilidades em DevOps e a maioria das equipes estão motivadas e alinhadas sobre a cultura DevOps, com a estrutura organizacional parcialmente adaptada para DevOps e o processo ágil é executado fim-a-fim, até a produção, por quase todas as equipes.
Gerenciado
- Alguns elementos do processo são compartilhados entre Dev e Ops e as ferramentas são parcialmente compartilhadas, com compartilhamento moderado de conhecimento. - Existe um consenso acerca de criações de atividades a respeito dos princípios básicos de DevOps e como aplicá-los e vários times têm alguma experiência e habilidades em práticas DevOps. - A empresa possui alguns entusiastas das práticas DevOps, com um portfólio de aplicações DevOps funcionando separadamente, com aplicação de metodologias ágeis nos processos de QA, mas os processos de *releases* não são ágeis.
Consciente
- Dev e Ops usam processos distintos, mas as interfaces são alinhadas e compartilhadas, com pouca integração de ferramentas e algum compartilhamento de status, via painéis de controle. - O compartilhamento de conhecimento entre Dev e Ops é limitado e as equipes estão entendendo melhor o conceito, mas a empresa está desalinhada em relação às práticas DevOps, com apenas algumas pessoas na empresa que possuem experiência e conhecimento relevantes. - Nenhum incentivo especial para alinhar Dev e Ops e o sentimento é neutro, Dev segue um portfólio de aplicações, enquanto Ops permanece guiado por infraestrutura e tecnologia. Há alguma metodologia ágil, mas os processos de QA e *release* são separados e não ágeis.
Inicial
- Dev e Ops usam processos separados e redundantes, com compartilhamento limitado e ferramentas distintas e redundantes, não havendo compartilhamento de conhecimento entre Dev e Ops. - Não há cultura interna sobre DevOps e não existe experiência nem conhecimento sobre DevOps na empresa, com nenhum incentivo especial para alinhar Dev e Ops e, apresentando uma postura defensiva quanto à mudança. - A estrutura da empresa é guiada por tecnologia, com abordagem separada entre Dev e Ops para os *releases*, adotando metodologia tradicional de desenvolvimento.

Fonte: Autoria própria.

4.0.2 PRÁTICAS RELACIONADAS À QUALIDADE DO CÓDIGO

Na Tabela 10 estão representadas as práticas para avaliação dos níveis de maturidade para garantia da Qualidade do Código. Esta métrica busca avaliar como os desenvolvedores estão assegurando a qualidade dos seus códigos, se existe o apoio de ferramentas nesse processo.

Se as empresas consideram em suas análises o nível de desempenho, segurança e usabilidade, bem como revisão e verificação da qualidade do que foi desenvolvido e realização de testes.

Tabela 10 – Avaliação das práticas de Pessoas e Cultura

Melhoria Contínua
- A qualidade do código é verificada no momento em que são gerados os *commits*, com os defeitos e não-conformidades sendo resolvidos de forma diária. - Informações, métricas e indicadores de qualidade de código são compartilhados publicamente e as revisões de código são efetuadas de forma automatizada e integrada com o gerenciamento de *build*. - São efetuados testes unitários de forma automatizada em torno de 80 a 90%.
Avançado
- Análises e revisões de código são realizadas diariamente de forma automatizada, onde são analisados o nível de desempenho, segurança e usabilidade. - São efetuados testes unitários de forma automatizada em torno de 40 a 50%.
Gerenciado
- A revisão de código é realizada de forma manual e em pares, mas sua monitorização é realizada através de ferramentas, sendo gerados alertas ao identificar defeitos na análise da qualidade do código. - A verificação da qualidade do código ocorre em ambientes direcionados à integração. - Os testes unitários são realizados de forma manual conforme acordado com o cliente.
Consciente
- A revisão de código é realizada de forma manual e em pares. - Início de automações de qualidade em ambientes de desenvolvimento. - Testes unitários são realizados de forma manual, sendo levado em consideração a performance de cada desenvolvedor ao realizá-lo.
Inicial
- Não ocorrem revisões de código e uso de padrões de codificação. - Não há aplicabilidade de testes unitários, com a qualidade dependendo de cada desenvolvedor.

Fonte: Autoria própria.

4.0.3 PRÁTICAS RELACIONADAS Á CONFIGURAÇÃO COMO CÓDIGO

Na Tabela 11 estão representadas as práticas para avaliação dos níveis de maturidade dos processos de Configuração como Código dentro das organizações. Esta métrica visa medir os processos relacionados ao gerenciamento e a providência de infraestruturas por meio de códigos de forma automatizada como criação de ambientes, hospedagem de aplicações e utilização de serviços em nuvem.

4.0.4 PRATICAS RELACIONADAS À GESTÃO DE *BUILDS*

Na Tabela 12 estão representadas as práticas para avaliação dos níveis de maturidade dos processos de Gestão de *Builds*. Esta métrica visa avaliar os processos de compilação e gerenciamento de código-fonte, validando se são utilizados ferramentas de automação, proces-

Tabela 11 – Avaliação das práticas Configuração como Código

Melhoria Contínua
- Utilização de infra-estrutura como código, automatizando o processo de criação dos ambientes de trabalho necessários e estimulando a cooperação dos times de desenvolvimento e operações. - As aplicações são hospedadas respeitando as políticas de definições de redes, assim como os serviços em nuvem. - É encorajado a utilização de serviços SaaS, desde que o fornecedor garanta segurança e qualidade do serviço. - Criação de ambientes de testes e desenvolvimento em máquinas virtuais de forma flexível e controlada, com as versões de software iguais às de produção.
Avançado
- *Scripts* para criação de ambientes de máquinas virtuais começam a ser utilizados e desenvolvidos. - Aproximação dos times de desenvolvimento e operações ao utilizar infraestrutura como código. - Utilização de serviços em nuvem PaaS para hospedagem e implementação de hardware e software. - Criação de ambientes de teste e desenvolvimento em máquinas virtuais flexíveis, mas nem sempre igual ao de produção.
Gerenciado
- Utilização de *scripts* para a maioria dos trabalhos repetitivos de forma compartilhada entre os times e organizados em repositórios. - A verificação da qualidade do código ocorre em ambientes direcionados à integração. - Os testes unitários são realizados de forma manual conforme acordado com o cliente.
Consciente
- *Scripts* de automação começam a ser criados e compartilhados entre times de desenvolvimento, qualidade e operações. - Não são utilizados serviços SaaS (Software as a Service). - Ambientes de testes e desenvolvimento são geralmente diferentes dos ambientes de produção.
Inicial
- Configurações de ambientes são efetuadas de forma manual. - Não são utilizados serviços em nuvem, os projetos são hospedados em Data Centers. - Não é possível a criação dos ambientes de testes e desenvolvimento igual ao de produção.

Fonte: Autoria própria.

sos de testes automatizados, se possuem controle de mudanças no ciclo de desenvolvimento, gerenciamento de códigos e problemas e medição de débito técnico.

4.0.5 PRATICAS RELACIONADAS À GESTÃO DE *RELEASE*

Na Tabela 13 estão representadas as práticas para avaliação dos níveis de maturidade dos processos de Gestão de *Release*. Esta métrica visa avaliar os processos relacionados a gestão do desenvolvimento de um software, como processos de implantação, utilização de automação, monitoramento de *releases* e itens de configuração.

Tabela 12 – Avaliação das práticas de Qualidade do Código

Melhoria Contínua
- Gerenciamento de códigos e controle de versão é realizado por meio de sistemas compartilhados de código fonte. - Débito técnico é medido de forma contínua a cada *check-in*. - Processo de integração de código, gerenciamento de *build*, gerenciamento de defeitos, configuração e testes unitários efetuados de forma automatizada e contínua. Além disso, o sistema de gerenciamento de configuração é integrado com desenvolvimento, teste e implantação.
Avançado
- *Builds* abrangem obrigatoriamente automação de testes, configuração de ambiente, qualidade código, geração de dados, geração de tags, documentação técnica, entre outros. - O sistema de gerenciamento de configuração de testes não é integrado com o sistema SCM (*source code management*). - Débito técnico é medido a cada *build* e o seu monitoramento é realizado através de ferramentas com logs. - As mudanças no ciclo de desenvolvimento são controladas por meio de ferramentas de gerenciamento.
Gerenciado
- *Builds* incorporam alguns processos de qualidade tais como automação de testes, configuração de ambientes, qualidade de código, geração de tags e documentação técnica. - As fases de produção e teste não são integradas com o sistema de SCM. - Mudanças durante o ciclo de desenvolvimento são gerenciadas e monitoradas através de um processo manual de gerenciamento de requisitos
Consciente
- Geração da *build* é realizada em ambientes propícios com gestão centralizada de dependências e compilação do código fonte. - O gerenciamento de códigos e o controle de versões é efetuado por meio de um controlador instalado em um servidor. - Produção e testes não são integrados com o sistema SCM. - É somente notificado o sucesso da *build*, não abrangendo coleta de logs.
Inicial
- Uso inicial de ferramentas de linhas de comando de *builds*, não há planejamento de *release* e monitoramento de *logs*. - Não há integração de nenhum sistema com o sistema de SCM. - As mudanças são gerenciadas por e-mail ou verbalmente. - O monitoramento das *builds* são realizados através de ferramentas SCM e compilado na própria IDEs com a revisão de código feita de forma manual.

Fonte: Autoria própria.

4.0.6 PRÁTICAS RELACIONADAS À GESTÃO DE TESTES

Na Tabela 14 estão representadas as práticas para avaliação dos níveis de maturidade dos processos de Gestão de Testes. Esta métrica visa avaliar os processos preparação e automatização de ambientes, utilização de infraestrutura como código e coletas de requisitos.

Tabela 13 – Avaliação das práticas de Gestão de Releases

Melhoria Contínua
- Automação da implantação incorpora infraestrutura como código (IAC) sendo realizado de forma automática sendo possível implantar uma ou mais vezes por dia usando ferramentas DevOps. - Os processos de implantação são totalmente replicáveis, medidos e melhorados continuamente, sendo implantados em larga escala. - O processo de *release* é totalmente automatizado por ferramentas DevOps. - Os itens de configuração são monitorados através de relatórios automáticos usando integração de ferramentas DevOps assim como o provisionamento completamente virtualizado.
Avançado
- Processo de definição da infraestrutura começa a ocorrer com micro-contêineres (ex.Docker). - Implantação automatizada incorpora o provisionamento de ambientes com uso de conteinerização. - O processo de gerenciamento de *release* é realizado por meio de ferramentas como (JIRA, RTC), mas sem rastreabilidade até a produção, e com novas funcionalidades implantadas pelo menos uma vez por semana. Controle de *release* é em grande parte automatizado, mas ainda é necessário algum envolvimento manual.
Gerenciado
- Automação dos processos de aprovação entre ambientes de desenvolvimento, homologação e produção. - Implantação baseada em *scripts*, sem utilização de ferramentas de implantação de *builds*. - Gerenciamento de *release* realizado utilizando planilhas Excel, com rastreabilidade. - Novas implantações em produção são realizadas pelo menos uma vez a cada duas semanas, para as implantações em homologação ocorre sempre que um build é gerado.
Consciente
- Automação das cópias dos *builds* entre ambientes. - Implantação semi-automatizada, usando procedimentos manuais e *scripts*, com novas funcionalidades implantadas em produção ao menos uma vez por mês. - PImplantação é replicável, mas o processo de *release* é realizado manualmente. - O processo de gerenciamento de *release* é manual, através de planilhas Excel, assim como a monitoração de configurações.
Inicial
- *Builds* são copiados de forma manual entre ambientes de desenvolvimento, homologação e produção. Parâmetros nos ambientes e implantações também são realizados de forma manual. - NNão há um processo formal para o gerenciamento de *release*, funcionalidades são somente implantadas em produção após um mês.

Fonte: Autoria própria.

4.0.7 PRATICAS RELACIONADAS A GESTÃO DE CONFIGURAÇÕES

Na Tabela 15 estão representadas as práticas para avaliação dos níveis de maturidade dos processos de Gestão de Configurações. Esta métrica visa avaliar os processos relacionados ao controle de mudanças inerentes ao desenvolvimento de software como armazenamento do código, utilização de ferramentas e padrões, automação de tarefas, controle de código-fonte entre outras práticas.

4.0.8 PRATICAS RELACIONADAS Á TESTES DE CARGA

Na Tabela 16 estão representadas as práticas para avaliação dos níveis de maturidade dos processos de Teste de Carga. Esta métrica visa avaliar as práticas relacionadas a execução de testes que simulam cenários do mundo real, como disponibilidade do sistema, criação das bases de testes, variações de testes, padrões de qualidade, comunicação entre outros.

4.0.9 PRÁTICAS RELACIONADAS A MONITORAÇÃO DE APLICAÇÕES

Na Tabela 17 estão representadas as práticas para avaliação dos níveis de maturidade dos processos de Monitoração de Aplicações. Esta métrica visa avaliar as práticas de controle de desempenho de sistemas e redes, como gerenciamento de incidentes, ferramentas de monitoração, informações de disponibilidade, carga e desempenho.

4.0.10 PRÁTICAS RELACIONADAS AOS REQUISITOS DE SOFTWARE

Na Tabela 18 estão representadas as práticas para avaliação dos níveis de maturidade dos processos de Requisitos de Software. Esta métrica visa avaliar as práticas relacionadas a definição dos objetivos e funções que um software necessita, como coleta e especificação de requisitos, formas de controle, priorização e medição do trabalho.

4.1 APLICAÇÃO DO *GUIDELINE* DE MATURIDADE PROPOSTO

A partir do modelo de avaliação elaborado, realizou-se entrevistas com as principais lideranças das equipes dos setores A e B, buscando identificar as equipes que utilizam DevOps. Posteriormente foi criado um formulário contendo as 10 áreas descritas no capítulo anterior.

Os grupos-alvo do questionário consistiram em funcionários ocupando cargos de liderança das equipes que utilizam de práticas DevOps e possuíam conhecimento em todas as áreas do processo de produção.

O envio do questionário foi realizado no período entre 14/11/2022 e 18/11/2022, por envio de e-mails contando com a participação de 5 lideres de equipes, sem a identificação dos mesmos ao longo da pesquisa.

Tabela 14 – Avaliação das práticas de Gestão de Testes

Melhoria Contínua
- Práticas de injeção de falhas são inicializadas. - Testes canários(testes A/B) são incorporados nas *releases*. - Os ambientes de teste são idênticos aos ambientes de produção, incluindo a configuração de hardware e de software, sendo realizados continuamente de forma ágil, ocorrendo testes de sanidade em ambientes de produção.Casos de teste unitários são automatizados de ponta a ponta e um painel de controle da cobertura do código o qual é disponibilizado para todos os releases. - O provisionamento do ambiente é totalmente automatizado por meio de ferramentas DevOps. - Utilização de infraestrutura como código para definição dos ambientes para desenvolvimento, teste, homologação e produção. - Requisitos de teste não-funcional são obtidos na fase inicial e modificados durante o ciclo de desenvolvimento.
Avançado
- Processos de *build* possuem suítes de automação de testes robustas e com excelente cobertura de código. - Os ambientes de teste são parcialmente semelhantes ao ambiente de produção, definições de software são idênticas, mas a de hardware não. - Os casos de teste unitário são executados manualmente pelos desenvolvedores de forma individual posteriormente um relatório consolidado é compartilhado antes de cada *release*. - Planos, casos e *scripts* de teste são criados antes do desenvolvimento, posteriormente os testes são feitos por *releases* iterativas. - Testes funcionais são parcialmente automatizados, pois nem tudo é coberto, para os não funcionais, são automatizados processos de segurança ou usabilidade.
Gerenciado
- Automação de testes ocorre para testes funcionais e testes de unidade, testes manuais possuem apoio de ferramentas de gerenciamento de teste, em relação aos testes contínuos existem muitos desafios devido à disponibilidade de recursos. - A cobertura automatizada do código é monitorada, automação de são realizados ao nível de UI para aplicações, com cobertura média. - Alguns ambientes de teste são idênticos aos de produção e outros similares. - Planos, casos e *scripts* de teste são criados em paralelo com o desenvolvimento.
Consciente
- Testes funcionais bem estabelecidos, teste de unidade são introduzidos para regras de negócio complexas e/ou conforme instruções da equipe de desenvolvimento. Resultados de testes manuais são registrados no sistema de monitoração de defeitos e / ou em planilhas. - Automação de testes começa a ser experimentada em projetos pilotos. - Os ambientes de testes possuem configurações semelhantes ao ambiente de produção.
Inicial
- Não possuem a cultura de automação. Testes são executados de forma exploratória e manual sem a criação de documentações, testes de unidade são realizados conforme a necessidade. - Defeitos se acumulam nas fases finais dos projetos, registrados em planilhas. - Os ambientes de teste não se assemelham com o ambiente de produção. - Todos os requisitos são testados no final do *release* (*waterfall*).

Fonte: Autoria própria.

Tabela 15 – Avaliação das práticas de Gestão de Configurações

Melhoria Contínua
- Práticas de injeção de falhas são inicializadas. - Todos os binários de componentes e bibliotecas fazem parte da gestão de configuração. - Gestão da configuração incorpora ambientes virtualizados e trata a infraestrutura como código (IAC), com processos sendo mensurados e controlados. - Preocupação em tratar causas comuns de variação de processos de modo a melhorar o desempenho e satisfazer os objetivos quantitativos.
Avançado
- Gerência de configuração generalizada com ferramentas e metodologias padronizadas. - Grande parte dos binários de componentes e bibliotecas fazem parte da gestão de configuração.. - Políticas de controle de código-fonte incorporam barreiras sólidas de qualidade, sendo estas automatizadas. - Processos de produção são mensurados e controlados, além de medições iniciais de desempenho e qualidade, com os processos sendo baseados em termos estatísticos e gerenciados ao longo da vida.
Gerenciado
- Processos gerenciais e técnicos básicos bem definidos. - Gestão da configuração começa a abranger binários de automação de tarefas. - Automações de políticas de controle de código-fonte começam a ser experimentadas. - Padrões, descrições de processo e procedimentos para um projeto são adaptados a partir de um conjunto de processos padronizados da organização que se ajustam a um projeto específico.
Consciente
- Processo gerenciado, infraestrutura básica e versionamento de código com controle automatizado. - Utilização de ferramentas de gestão de configuração de código (SCM), além de políticas de controle.
Inicial
- Ausência de configuração de código, ambientes com processos caóticos e reativos. - Código-fonte e outros artefatos são armazenados em sistemas de arquivos. - Escassez de ferramentas para gerenciamento de configuração.

Fonte: Autoria própria.

Tabela 16 – Avaliação das práticas de Testes de Carga

Melhoria Contínua
- Testes de carga fornecem políticas automatizadas de infraestrutura de código para responder a picos de utilização em ambientes de produção. - Disponibilidade do sistema a maioria do tempo em produção. - O início da execução dos testes é comunicado de forma automatizada, possuindo painéis de controle e relatório de monitoração, com captura de defeitos feita de forma automatizada. - Equipe de desenvolvimento cria e fornece as bases de dados de teste, que se assemelha ao de produção.
Avançado
- Testes de carga, estresse e maturidade incorporados aos processos de *builds* e *releases*. As bases de teste são criadas pela equipe de teste, alinhado com os dados de produção. - O planejamento de capacidade fornece diretrizes de provisionamento de ambiente em produção. - A comunicação do início da execução dos testes é realizada de forma automatizada com uma lista de distribuição. - Os defeitos são capturados utilizando ferramentas, mas sem métricas automatizadas, pois são geradas manualmente, usando planilhas. - Requisitos identificados através de planilhas.
Gerenciado
- Variações dos testes de carga como testes de performance, estresse e testes de maturidade começam a ser adotadas, com o time de testes criando sua própria base de dados, mas com baixo nível de semelhança com o de produção. - Problemas de indisponibilidade são menores na produção. - A comunicação do início da execução dos testes é realizada a partir de e-mails automáticos. - Os defeitos são identificados em planilhas, mas não são geradas métricas.
Consciente
- Padrões de qualidade são inseridos no desenvolvimento juntamente com a revisão por pares. -Início da automação da qualidade nos ambientes de desenvolvimento(IDEs). - A comunicação do início da execução dos testes é realizada de forma manual através de e-mails. - Sem utilização de ferramentas para gerenciamento de defeitos, os mesmos são identificados individualmente pelos testadores em planilhas. O time de testes cria a sua própria base de dados, de forma desalinhada com o de produção.
Inicial
- A comunicação do início da execução dos testes é realizada de forma verbal em reuniões. - Os defeitos identificados são comunicados via e-mail. - Não existe a cultura de gerenciamento de requisitos e gerenciamento de testes. - EAs bases de testes são feitas caso a caso pelo testador, durante os testes.

Fonte: Autoria própria.

Tabela 17 – Avaliação das práticas de Monitoração de Aplicações

Melhoria Contínua
- Utilização de ferramentas DevOps para a monitoração, de forma integrada com os processos e ferramentas de desenvolvimento. - Gerenciamento de incidentes e eventos são realizados através de ferramentas DevOps, de forma integrada com processos de *release* e desenvolvimento. - Limites de desempenho são monitorados de forma automatizada, com rápido tempo de resposta para a escalada de ambiente. - Monitoração de erros de desempenho, alertas e erros através de ferramentas DevOps, com feedback automatizado para o time de desenvolvimento.
Avançado
- Monitoramento de aplicativos incorporando práticas de telemetria os quais coletam dados de uso para gerar aprendizados de negócio para as equipes de desenvolvimento. - Os problemas na produção são informados às equipes por meio de ferramentas que geram alertas de forma automatizada. - Monitoração de desempenho automatizada. - Monitoração de erros, desempenho e alertas através de ferramentas DevOps, mas sem integração com as ferramentas de desenvolvimento.
Gerenciado
- Monitoramento de aplicativos em produção em formato contínuo. - Informações de disponibilidade, carga e desempenho são regularmente repassados para as equipes de desenvolvimento. - Bancos de dados são sincronizados usando *scripts* para a criação de ambientes iguais. - O gerenciamento de incidentes e eventos são realizados através de ferramentas como (Remedy ou CQ), mas de forma não automatizada com o processo de *release*.
Consciente
- Aplicações monitoradas não geram feedback acionáveis para times de desenvolvimento. - Problemas na produção são informados às equipes por meio de solicitações/reclamações dos usuários finais por telefone, *e-mail*, *helpdesk* ou gerentes de conta. - Incidentes são comunicados por e-mail e registrados em planilha. - Monitoração de erros, desempenho e alertas realizados de forma manual.
Inicial
- Aplicações rodam sem supervisão no ambiente de produção. - Ambientes de teste e produção diferentes. - Não há um processo de gerenciamento de incidentes assim como formas de reportar aos usuários. - Não há um processo de monitoração de aplicações quanto a erros, alertas e desempenho.

Fonte: Autoria própria.

Tabela 18 – Avaliação das práticas de Requisitos de Software

Melhoria Contínua
- Requisitos funcionais e de negócio são coletados junto ao cliente através de técnicas e ferramentas DevOps de forma integrada no processo de desenvolvimento de Dev e Ops. - São utilizados *backlogs* compartilhados para controle de trabalhos a serem desenvolvidos e quadros Kanban abrangendo diferentes equipes. - A priorização dos requisitos é feita de forma ágil, através de ferramentas de gerenciamento de projetos com recursos compartilhados e integração com outras ferramentas, com métricas bem definidas com a coleta de dados e análise feitas através de ferramentas DevOps. - Medição e modelagem de processos.
Avançado
- Requisitos funcionais e de negócio são coletados junto ao cliente através de ferramentas DevOps e as tarefas a serem realizadas com os times são coletadas através de *backlogs*. - Requisitos são priorizados de forma ágil utilizando uma ferramenta de gerenciamento de requisitos, mas sem integração com outras ferramentas e as métricas para gerenciamento são coletadas de forma parcial através de ferramentas DevOps. - Os requisitos são identificados e rastreados para a *release* do sistema de produção através de planilhas excel, de forma muito bem planejada e rastreada, mas as atualizações são efetuadas de forma manual.
Gerenciado
- Requisitos funcionais e de negócio são coletados junto ao cliente através de ferramentas de documentação de requisitos e as tarefas serem feitas junto ao time são documentadas através de *backlog* do time, com uso de quadro Kanban físico ou virtual ou através de Ticket/ Incidente. - Requisitos são priorizados de forma ágil sem o uso de ferramentas para gerenciamento de requisitos e o planejamento dos projetos são realizados de forma ágil através de planilhas Excel. - Métricas são coletadas e analisadas utilizando métodos manuais através de planilhas e os relatórios são gerados por meio da automação no Excel e os requisitos são rastreados utilizando planilhas Excel ou ferramentas próprias, sem planos de dados disponíveis ou não há indicadores de dados suficientes disponíveis.
Consciente
- Requisitos funcionais e de negócio são coletados junto ao cliente através de ferramentas DevOps e as tarefas a serem realizadas juntamente com os times são coletadas através de *backlogs*. - Requisitos são priorizados de forma ágil utilizando uma ferramenta de gerenciamento de requisitos, mas sem integração com outras ferramentas e as métricas para gerenciamento são coletadas de forma parcial através de ferramentas DevOps. - Os requisitos são identificados e rastreados para a *release* do sistema de produção através de planilhas excel, de forma muito bem planejada e rastreada, mas as atualizações são efetuadas de forma manual.
Inicial
- Requisitos funcionais e de negócio são coletados junto ao cliente através de técnicas e ferramentas DevOps de forma integrada no processo de desenvolvimento de Dev e Ops. - São utilizados *backlogs* compartilhados para controle de trabalhos a serem desenvolvidos e quadros Kanban abrangendo diferentes equipes. - A priorização dos requisitos é feita de forma ágil, através de ferramentas de gerenciamento de projetos com recursos compartilhados e integração com outras ferramentas, com métricas bem definidas com a coleta de dados e análise feitas através de ferramentas DevOps. - Utilização de medições e modelagens de processos.

Fonte: Autoria própria.

5 RESULTADOS

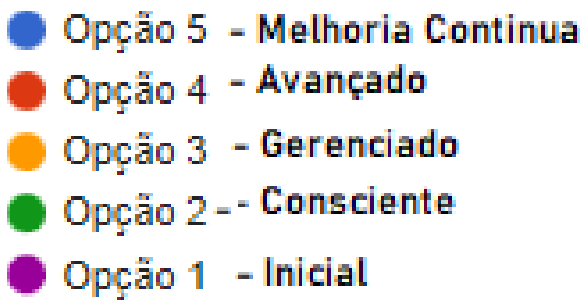
Este capítulo apresenta os resultados obtidos pelo processo de execução da pesquisa com a apresentação dos questionários aplicados.

5.0.1 RESULTADOS QUESTIONÁRIOS

Para realizar a avaliação e obter uma visão geral da Cultura DevOps dos níveis de maturidade de cada equipe foi disponibilizado para as lideranças de equipe um questionário sem teor de obrigatoriedade obtendo um total de 5 respostas, dois participantes do setor A de novos produtos e três do setor B de software legado.

Visando evitar induzir as respostas para as melhores classificações propostas no modelo de maturidade, no questionário não foram disponibilizados os nives que correspondiam a cada área deixando cada alternativa declarada apenas como opções de 1 a 5. Na Figura 6 é possível visualizar os níveis correspondentes a cada opção.

Figura 6 – Opções de níveis de maturidade



Fonte: Autoria Própria.

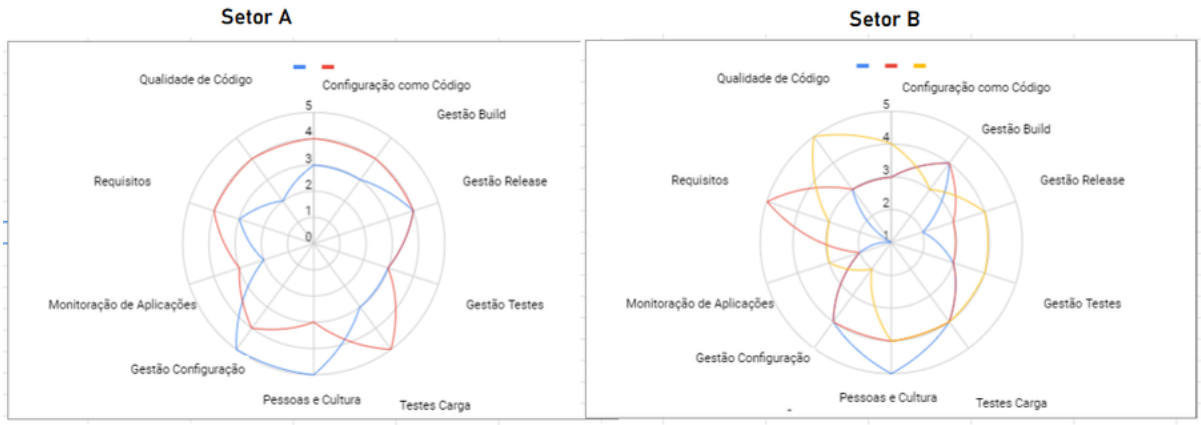
5.0.2 RESULTADO QUESTIONÁRIO UTILIZANDO GRAFOS DE MATURIDADE

Abaixo estão representados os grafos de maturidade em formato de teia de aranha gerados a partir das respostas dos setores A e B para uma melhor visualização da variação dos níveis de maturidade entres as equipes e entre os setores A e B.

Ao comparar os gráficos da Figura 7 nota-se uma grande variação de maturidade entre os setores, de forma geral, o setor A aparenta estar com seus processos semelhantes entre as equipes de forma geral como pode ser notado através das linhas azuis e vermelhas, variando dos níveis Avançado e Gerenciado com práticas DevOps em curso na maioria dos projetos e com processos seguidos com maior rigor, enquanto o setor B possui maior variação de maturidade entre as equipes, principalmente em relação à coleta de requisitos, onde a variação ocorre com

práticas de nível Inicial em uma equipe enquanto outras utilizam práticas de alto nível, sendo visualizados a partir das linhas amarelas, azuis e vermelhas.

Figura 7 – Grafo de Maturidade Setores A e B

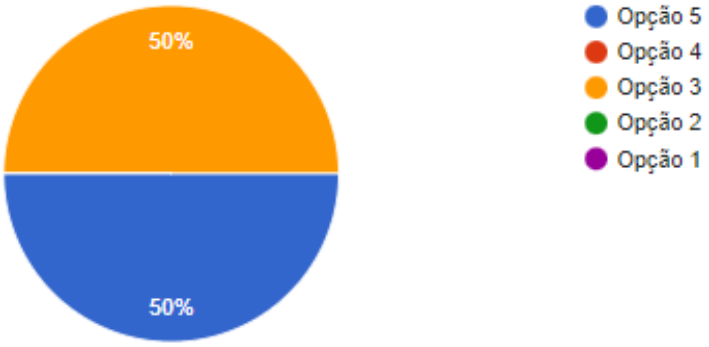


Fonte: Autoria Própria.

5.0.3 RESULTADO QUESTIONÁRIO PESSOAS E CULTURA

- **Setor A:** Na Figura 8, é apresentado o percentual das práticas DevOps relacionada a dispersão da cultura DevOps entre as pessoas do setor A. Como ilustra a figura, 50% dos líderes consideram que as práticas DevOps relacionadas a pessoas e cultura estão ao Nível de Melhoria Contínua e outros 50% ao Nível Gerenciado.

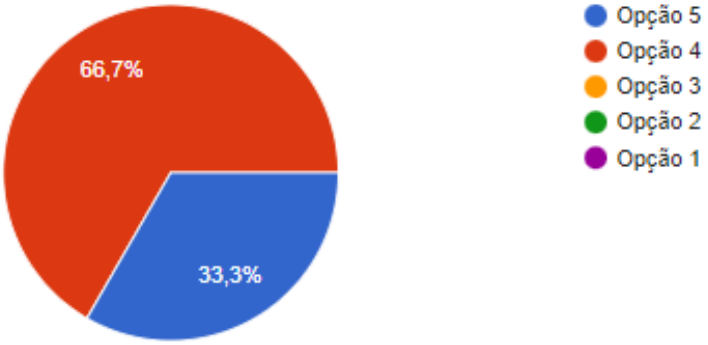
Figura 8 – Resultado questionário Pessoas e Cultura - Setor A



Fonte: Autoria Própria.

- **Setor B:** Na Figura 9, é apresentado o percentual das práticas DevOps relacionada a dispersão da cultura DevOps entre as pessoas do setor B. Como ilustra a figura, 66.7% dos líderes consideram que as práticas DevOps relacionadas a pessoas e cultura estão ao Nível de Melhoria Contínua e outros 33,3% ao Nível Avançado.

Figura 9 – Resultado questionário Pessoas e Cultura - Setor B

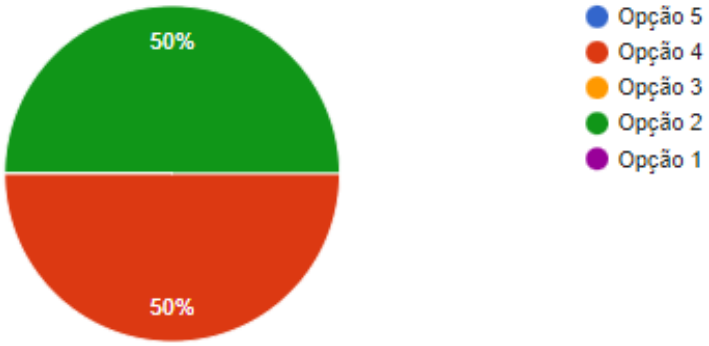


Fonte: Autoria Própria.

5.0.4 RESULTADO QUESTIONÁRIO QUALIDADE DE CÓDIGO

- **Setor A:** Na Figura 13, é apresentado o percentual das práticas DevOps relacionada a qualidade do código no setor A. Como ilustra a figura, 50% dos líderes consideram que as práticas DevOps relacionadas a qualidade do código estão ao Nível Avançado e outros 50% ao Nível Consciente.

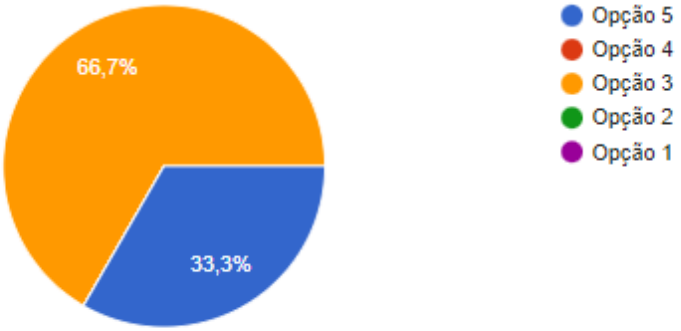
Figura 10 – Resultado questionário Qualidade de Código - Setor A



Fonte: Autoria Própria.

- **Setor B:** Na Figura 11, é apresentado o percentual das práticas DevOps relacionada a qualidade do código no setor B. Como ilustra a figura, 66.7% dos líderes consideram que as práticas DevOps relacionadas a qualidade do código estão ao Nível Melhoria Contínua e outros 33,3% ao Nível Gerenciado.

Figura 11 – Resultado questionário Qualidade de Código - Setor B

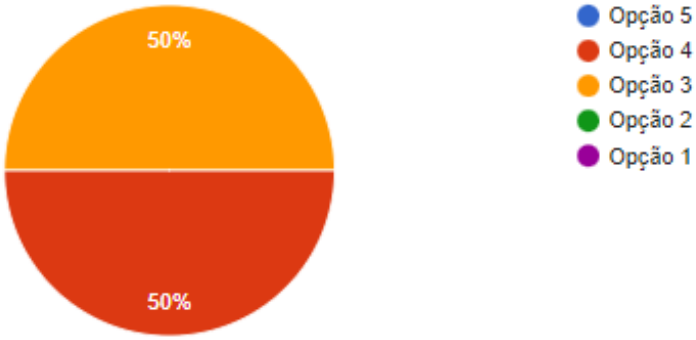


Fonte: Autoria Própria.

5.0.5 RESULTADO DO QUESTIONÁRIO CONFIGURAÇÃO COMO CÓDIGO

- **Setor A:** na Figura 12, é apresentado o resultado das respostas relacionados a configuração como código no setor A. Como resultado observa-se que 50% selecionaram as práticas de nível Avançado e outros 50% selecionaram o nível Gerenciado.

Figura 12 – Resultado questionário Configuração como Código - Setor A



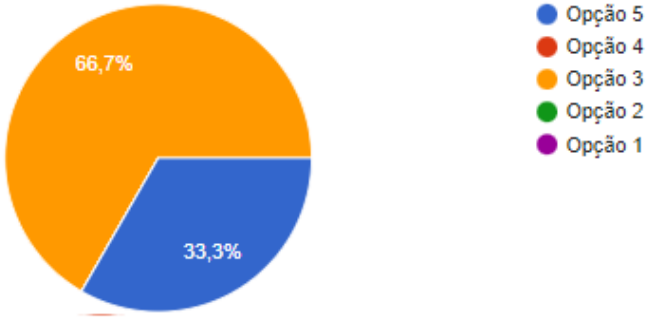
Fonte: Autoria Própria.

- **Setor B:** na Figura 13, é apresentado o resultado das respostas relacionados a configuração como código do setor B. Como resultado observa-se que 66,7% selecionaram as práticas de nível Melhoria Contínua e outros 33,3% selecionaram o nível Gerenciado.

5.0.6 RESULTADO DO QUESTIONÁRIO GESTÃO DE *BUILDS*

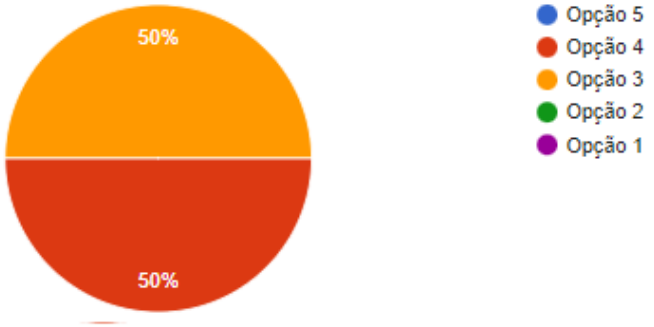
- **Setor A:** Na Figura 14, é apresentado o resultado das respostas relacionados a gestão de *Builds* no setor A. Como resultado observa-se que 50% selecionaram as práticas de nível Avançado e outros 50% de nível Gerenciado.

Figura 13 – Resultado questionário Configuração como Código - Setor B



Fonte: Autoria Própria.

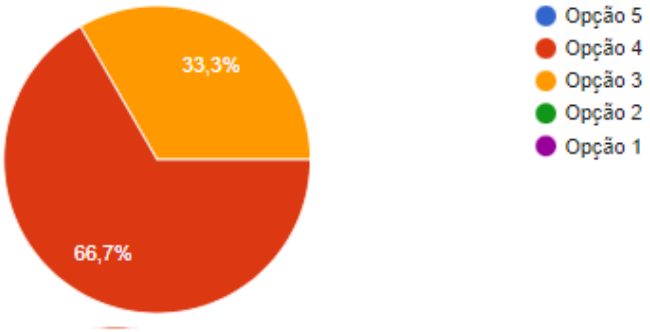
Figura 14 – Resultado questionário Gestão de *Builds* - Setor A



Fonte: Autoria Própria.

- **Setor B:** Na Figura 15, é apresentado o resultado das respostas relacionados a gestão de *Builds* no setor B. Como resultado observa-se que 66,7% selecionaram as práticas de nível Avançado e outros 33,3% de nível Gerenciado.

Figura 15 – Resultado questionário Gestão de *Builds* - Setor B

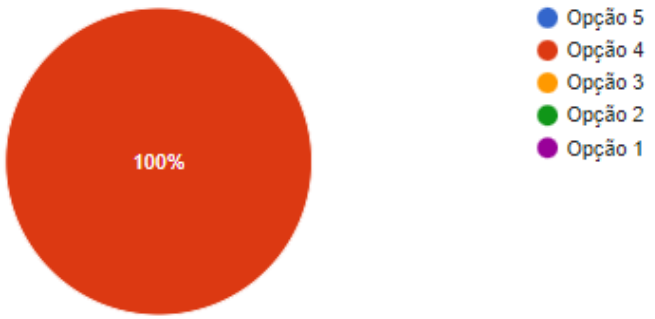


Fonte: Autoria Própria.

5.0.7 RESULTADO DO QUESTIONÁRIO GESTÃO DE *RELEASES*

- **Setor A:** Na Figura 16, é apresentado o resultado das respostas relacionados a gestão de *releases* do setor A. Como resultado observa-se que houve unanimidade na seleção das práticas de nível Avançado.

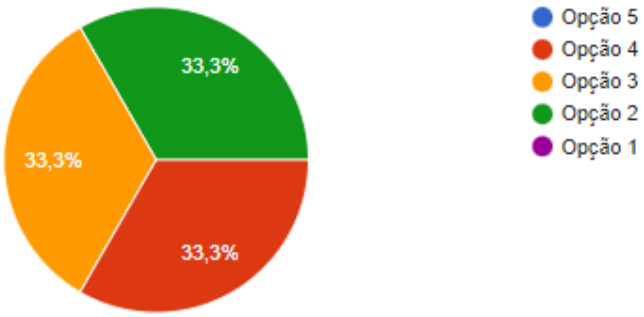
Figura 16 – Resultado questionário Gestão de Release -Setor A



Fonte: Autoria Própria.

- **Setor B:** Na Figura 17, é apresentado o resultado das respostas relacionados a gestão de *releases* do setor B. Como resultado observa-se que 33.3% selecionaram as práticas de nível Avançado, 33.3% de nível Gerenciado e outros 33.3% selecionaram as práticas de nível Consciente.

Figura 17 – Resultado questionário Gestão de *Release* - Setor B

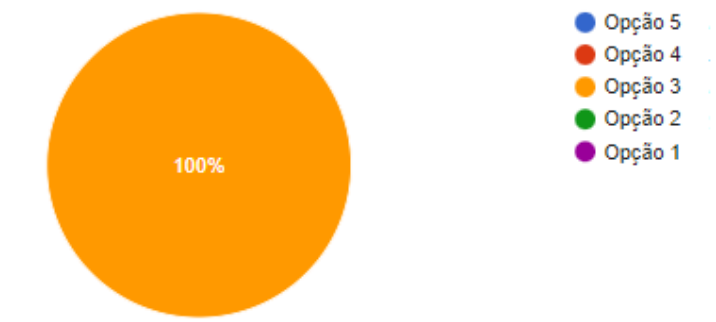


Fonte: Autoria Própria.

5.0.8 RESULTADO QUESTIONÁRIO GESTÃO DE TESTES

- **Setor A:** Na Figura 18, é apresentado o resultado das respostas relacionados a gestão de testes. Como resultado observa-se que de forma unanime são utilizadas práticas de nível Avançado.

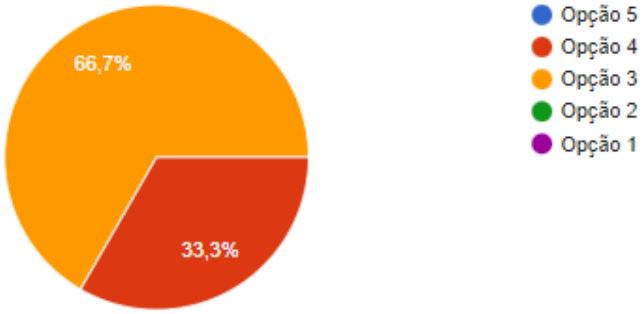
Figura 18 – Resultado questionário Gestão de Testes -Setor A



Fonte: Autoria Própria.

- **Setor B:** Na Figura 19, é apresentado o resultado das respostas relacionados a gestão de testes do setor B. Como resultado observa-se que 33,3% selecionaram as práticas de nível Avançado e outros 66,7% de nível Gerenciado.

Figura 19 – Resultado questionário Gestão de Testes -Setor B



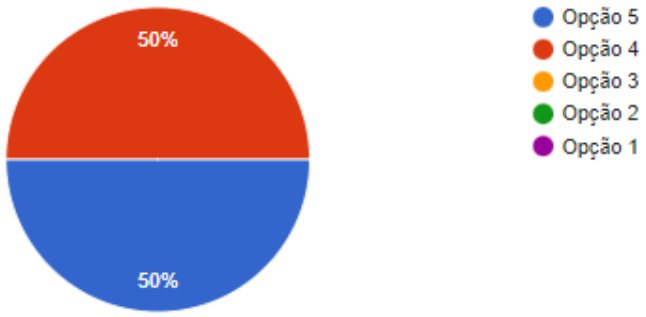
Fonte: Autoria Própria.

5.0.9 RESULTADO QUESTIONÁRIO GESTÃO DE CONFIGURAÇÃO

- **Setor A:** Na Figura 20, é apresentado o resultado das respostas relacionados a gestão de configuração no setor *A*. Como resultado observa-se que 250% selecionaram as práticas de Melhoria Contínua e outros 50% dos líderes selecionaram as práticas de nível Avançado.

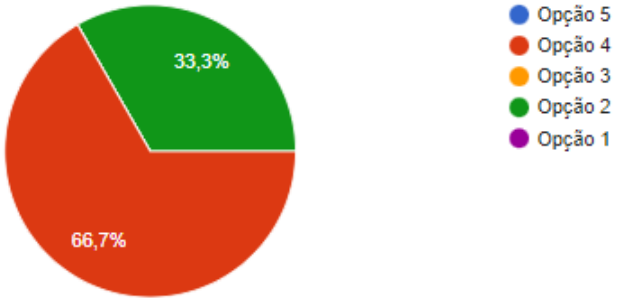
- **Setor B:** Na Figura 21, é apresentado o resultado das respostas relacionados a gestão de configuração no setor *B*. Como resultado observa-se que 66,7% selecionaram as práticas de Nível Avançado e outros 33,3% selecionaram as práticas de nível Consciente.

Figura 20 – Resultado questionário Gestão de Configuração - Setor A



Fonte: Autoria Própria.

Figura 21 – Resultado questionário Gestão de Configuração - Setor B



Fonte: Autoria Própria.

5.0.10 RESULTADO QUESTIONÁRIO TESTE DE CARGA

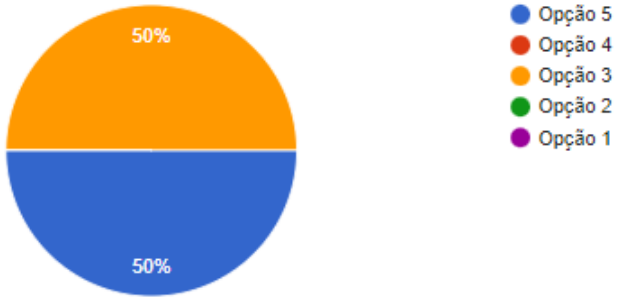
- **Setor A:** Na Figura 22, é apresentado o resultado das respostas relacionados as práticas de teste de carga do setor A. Como resultado observa-se que 50% selecionaram as práticas de Melhoria Contínua e outros 50% dos líderes selecionaram as práticas de nível Gerenciado.

- **Setor B:** Na Figura 23, é apresentado o resultado das respostas relacionados as práticas de teste de carga do setor B. Como resultado observa-se que 100% dos líderes selecionaram as práticas de Nível Avançado.

5.0.11 RESULTADO QUESTIONÁRIO MONITORAÇÃO DE APLICAÇÃO

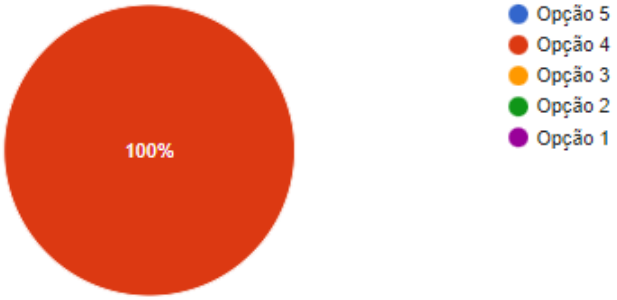
- **Setor A:** Na Figura 24, é apresentado o resultado das respostas relacionados as práticas de monitoração de aplicação do setor A. Como resultado observa-se que 50% dos líderes selecionam as práticas de nível Gerenciado e outros 50% de nível Consciente.

Figura 22 – Resultado questionário Teste de Carga - Setor A



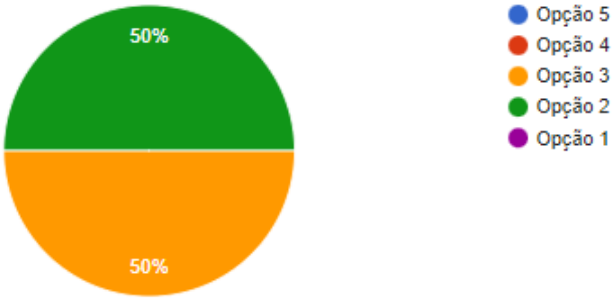
Fonte: Autoria Própria.

Figura 23 – Resultado questionário Teste de Carga - Setor B



Fonte: Autoria Própria.

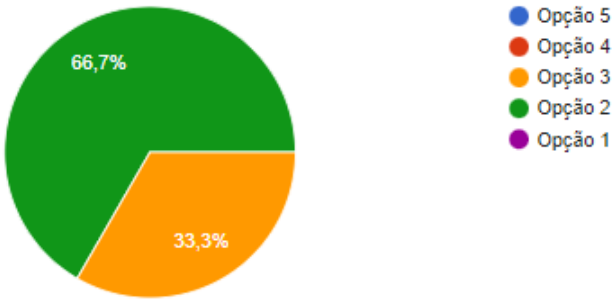
Figura 24 – Resultado questionário Monitoração de Aplicação - Setor A



Fonte: Autoria Própria.

- **Setor B:** Na Figura 25, é apresentado o resultado das respostas relacionados as práticas de monitoração de aplicação do setor B. Como resultado observa-se que 33.3% dos líderes selecionam as práticas de nível Gerenciado e outros 66.7% de nível Consciente.

Figura 25 – Resultado questionário Monitoração de Aplicação - Setor B

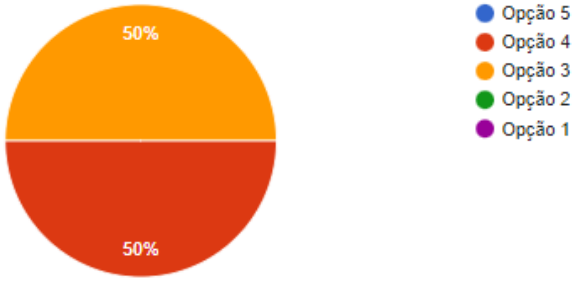


Fonte: Autoria Própria.

5.0.12 RESULTADO QUESTIONÁRIO REQUISITOS DE SOFTWARE

- **Setor A:** Na Figura 26, é apresentado o resultado das respostas relacionados a requisitos de software. Como resultado observa-se que 50% dos lideres selecionaram práticas de nível Avançado e outros 50% selecionaram as práticas de nível Gerenciado.

Figura 26 – Resultado questionário Requisitos de Software - Setor A



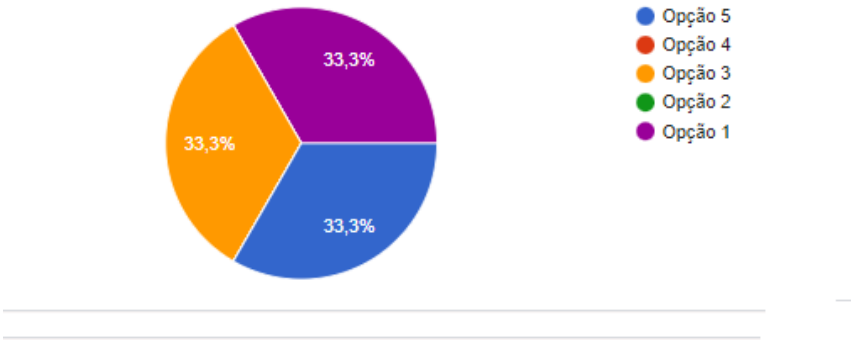
Fonte: Autoria Própria.

- **Setor B:** Na Figura 27, é apresentado o resultado das respostas relacionados a requisitos de software do setor B. Como resultado observa-se que 33.3% dos lideres selecionaram práticas de Melhoria Contínua, outros 33.3% selecionaram as práticas de nível Gerenciado e outros 33.3% dos líderes selecionaram as práticas de nível Inicial.

5.0.13 RESULTADO MAPEAMENTO DO CICLO DE VIDA

Através de um levantamento feito a partir de informações preliminares, constatou-se que a empresa conta com duas fábricas de desenvolvimento as quais trabalham com diferentes produtos. A primeira identificada como setor A, é responsável pela criação de novas tecnolo-

Figura 27 – Resultado questionário Requisitos de Software - Setor B

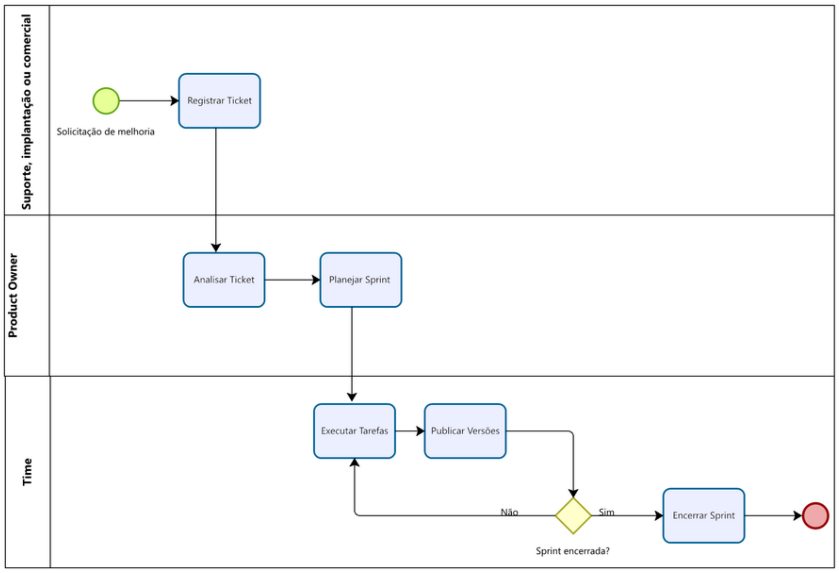


Fonte: Autoria Própria.

gias enquanto a segunda trabalha com a manutenção e melhoria de um produto legado, que denominaremos de setor B.

Abaixo são apresentados os ciclos de desenvolvimento utilizados em ambas as fábricas. No setor A, Figura 28, o *ticket* pode ser recebido através do suporte, implantação ou comercial, posteriormente analisado pelo PO (*Product Owner*) e alocado na *sprint* para o time de desenvolvimento e teste, podendo ser entregue várias versões antes do fim da *sprint*. Enquanto no setor B, Figura 29, todas as solicitações de mudança de software, originadas através processo de operação do suporte e/ou implantação ou comercial são registradas via ferramenta e enviadas ao processo de Gestão de Mudanças para revisão, análise e priorização da solicitação *Backlog* de Produto.

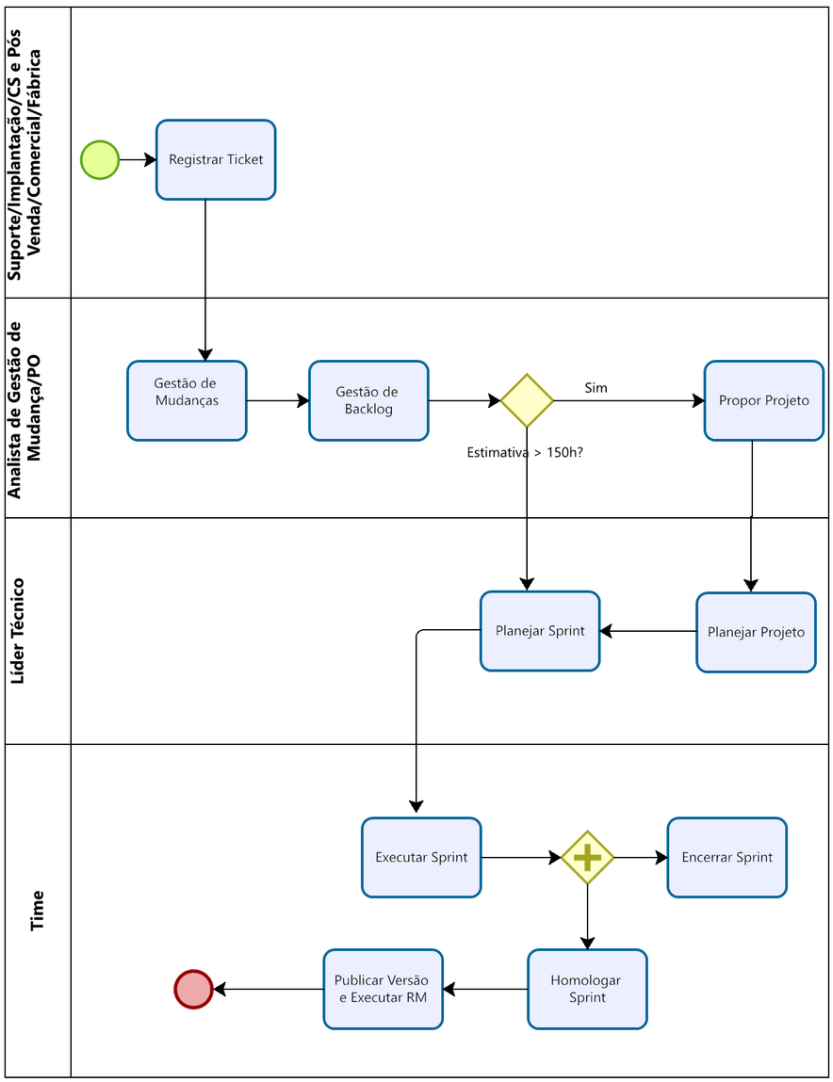
Figura 28 – Ciclo de vida - Setor A



Fonte: Autoria Própria.

Backlog de Produto, é uma lista que contém breves descrições de todas as funcionalidades desejadas para um produto específico. Essa lista traz os requisitos para um projeto, priorizados conforme o valor entregue para o cliente. O *backlog* deve ser gerenciado através das

Figura 29 – Ciclo de vida - Setor B



Fonte: Autoria Própria.

atividades de revisão, refinamento e estimativa pelo PO, com base no levantamento das necessidades de negócio, o *Product Owner* deve propor o projeto, caso seja menor que 150 horas, se não as tarefas podem ser alocadas diretamente na *sprint*. Nessa fase o líder técnico trabalha na resolução de possíveis impedimentos e apoio ao time, após o encerramento da *sprint* é feita a homologação e a publicação da versão.

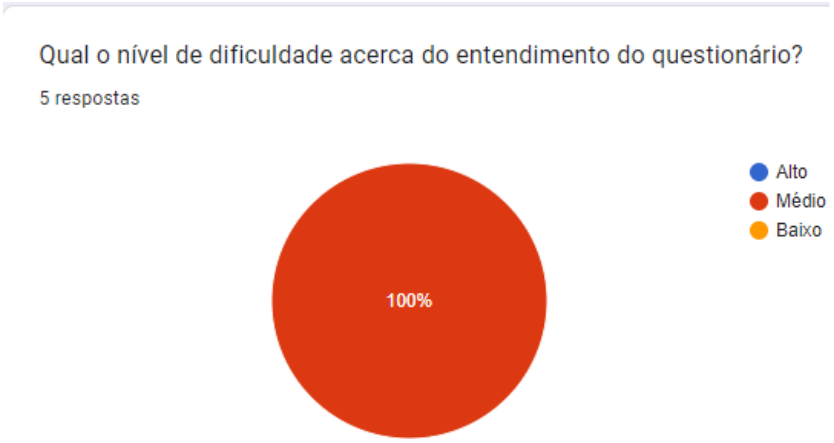
A cultura DevOps é encontrada em ambas as fábricas de forma informal e formal. Contando com uma equipe DevOps em cada setor, responsáveis pela parte da integração continua, criação de *build*, analisar auditorias e publicar versões. Utilizando como principais ferramentas o *Jira* para o controle das atividades, *Freshdesk* para o suporte e GitLab para o controle de integrações.

5.0.14 RESULTADO COLETA DE *FEEDBACKS* ACERCA DO MODELO DE MATURIDADE

Com o proposito de coletar *feedbacks* acerca do modelo aplicado disponibilizou-se um formulário respondido pelas lideranças contendo duas questões, a primeira “Qual o nível de dificuldade acerca do entendimento do questionário? ” e a segunda se tinham “Alguma sugestão de melhoria? ”. A primeira questão buscou medir o nível de dificuldade de compreensão das práticas presentes e a segunda visando coletar sugestões de melhorias. O *feedback* foi coletado por 5 lideres de equipes, os mesmos que responderam o modelo de maturidade aplicado.

- Na Figura 30, é possível visualizar o gráfico gerado a partir das respostas da primeira questão. De forma unanime elencou-se um nível de dificuldade média para entendimento do modelo elaborado.

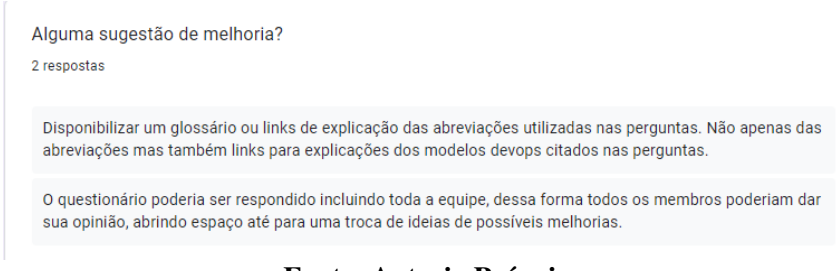
Figura 30 – Resultado Questionário Nível de Dificuldade Modelo Maturidade



Fonte: Autoria Própria.

- Houveram duas respostas para a segunda questão conforme pode ser visualizado na Figura 31. Notam-se sugestões de informações complementares para a pesquisa disponibilizadas de forma que seu entendimento atinja um nível baixo de dificuldade além, de abrir espaço para uma abrangência maior de respondentes.

Figura 31 – Resultado Questionário Sugestão de Melhoras



Fonte: Autoria Própria.

6 CONCLUSÃO

Esse Capítulo visa apresentar a discussão das informações obtidas, na Seção 6.1 e, em seguida, na Seção 6.2, são apresentadas as conclusões acerca do assunto.

6.1 DISCUSSÃO

Ao longo da revisão da literatura, foi possível observar que durante a última década houve avanços na criação de soluções metodológicas e ferramentas para avaliação da cultura DevOps. No entanto, as soluções propostas possuem alto grau de heterogeneidade, pois não há um consenso claro sobre as definições e conceitos associados ao DevOps.

Contudo, apesar da discrepância dos modelos existentes, foi possível notar que a avaliação da maturidade dos processos é de grande valia, pois permite saber o quanto a empresa está madura. Além disso, permite identificar o motivo pelo qual determinados setores produzem mais que outros, identificar lacunas entre os níveis, além de possibilitar definir qual deve ser o ponto de início do ciclo de melhoria.

Observou-se também a alta resposta de desempenho das empresas ao utilizar práticas DevOps. Diferente dos modelos tradicionais, a cultura DevOps trabalha harmonizando as equipes, fluindo melhor a comunicação entre os desenvolvedores de software e o setor operacional. A qualidade também aumenta, pelo fato das ferramentas serem implantadas simultaneamente em diversas plataformas, dessa forma ao produzir um produto de qualidade é visível o retorno monetário com a vinda de novos clientes. Os resultados das organizações que utilizam DevOps, indicam que a conforme as práticas são amadurecidas, os programas também amadurecem, resultando em métricas importantes para o negócio, com melhorias em diversas áreas além da redução do tempo gasto em implementações e manutenções.

A partir dos resultados levantados e das informações obtidas com a revisão narrativa notou-se também que, atualmente, ainda existem barreiras à adoção de práticas DevOps. Esta situação está relacionada as dúvidas relativas a sua aplicação, ainda que exista uma infinidade de informações, práticas e ferramentas relacionadas a DevOps, os parâmetros que norteiam sua implementação real em empresas e curva de adoção em nível de maturidade ainda estão sendo estudados.

Outra situação é a relutância a mudança, baixo conhecimento de suas práticas e o pensamento de que apesar dos benefícios são demandados altos investimentos. Percebe-se, também, que a adesão a modelos de melhoria de processos não é uma tarefa simples, pois além dos recursos financeiros necessários, existe a necessidade de conscientização de todos os membros da empresa, logo que não somente um produto de qualidade é feito somente através de ferramentas, profissionais especializados são parte do conjunto para sua construção.

A partir do resultado da aplicação do *guideline* de maturidade durante o estudo de caso, percebeu-se a disparidade de conhecimento acerca de processos comuns para todas as equipes

como a coleta de requisitos. As práticas realizadas nesse processo possuem níveis de maturidade bem diferentes com equipes utilizando práticas de “Melhorias Contínuas” enquanto outras coletando requisitos com práticas de nível Inicial, constatando-se a falta de um padrão.

De forma semelhante é observado a falta de padrões na validação e verificação da qualidade do código, com algumas equipes utilizando práticas de alto nível, e outras de nível dois, “Consciente”, essa situação ocorre também nas áreas de configuração como código, gestão de *releases e* pessoas e cultura. Este desnivelamento demonstra a falta de processos bem estruturados, com práticas e conhecimentos não compartilhados entre as equipes onde existe alguma organização, porém não há padronização.

Há uma lacuna a ser preenchida, com oportunidades para desenvolvimento técnico das equipes assim como melhorias em infraestrutura e processos internos. Tendo como base o *guideline* proposto como documento de referência, pode-se criar ciclos de avaliações periódicos para possibilitar melhorias nas áreas deficitárias, buscando um nivelamento mais homogêneo de todas as equipes.

6.2 CONCLUSÕES

DevOps, deriva da combinação de Desenvolvimento e Operações, é uma nova forma de pensar no domínio da engenharia de software, a qual está em ascendência. Culturalmente busca aproximar as equipes de desenvolvimento e operações, tendo apoio de diferentes práticas e pilares, com o intuito de agilizar entregas com mais qualidade. Para uma adoção de sucesso é demandado esforços, mudanças e colaboração de todos. Todavia, por se tratar de uma abordagem emergente, as empresas ainda estão lapidando os seus processos, no que tange a aplicação dos conceitos de DevOps necessitando de um modelo de avaliação que permita avaliar o nível de maturidade das suas práticas.

Considera-se que os objetivos propostos no início deste trabalho foram alcançados, sendo a proposta de um modelo de avaliação de maturidade de práticas DevOps e um *guideline* para avaliação e nivelamento, com sua validação por meio de um estudo de caso.

Dessa forma, este trabalho apresentou uma proposta de modelo de maturidade elaborado através da harmonização de 3 propostas de modelos diferentes para avaliação DevOps, de forma que a partir de um único modelo pode-se abranger diferentes processos diminuindo assim as dúvidas na escolha de uma forma de avaliação. O modelo proposto é direcionado principalmente para pequenas e médias empresas, com aplicação por meio de um *guideline* simplificado para referência.

A aplicação desse estudo foi realizada dentro de uma empresa de desenvolvimento que aplica DevOps em produtos legados e novas criações. Adicionalmente o modelo demonstrou que a empresa está com práticas conflitantes, com processos mais maduros em algumas áreas e equipes e menos maduros em outros.

Considera-se que a principal contribuição deste trabalho é oferecer um *guideline* para avaliação de maturidade para empresas que utilizam práticas DevOps, com a possibilidade de ser também um guia de referência para empresas que desejam implementar práticas DevOps, apresentando as etapas para a sua implantação e os processos que seguem desde a transformação cultural onde toda a equipe envolvida no projeto deve trabalhar em formato de cooperação, passando pela codificação e revisão do código, pelos testes automatizados que visam buscar falhas garantindo a qualidade do produto, pela integração que visa agilizar a detecção de defeitos e bugs entre todas as partes desenvolvidas, pela implantação que deve ser feita em conjunto com as esquipes de desenvolvimento e operação até o monitoramento do projeto.

A partir da realização deste trabalho, será possível desenvolver propostas de melhorias nas categorias identificadas como deficitárias dentro das equipes da empresa avaliada, medindo resultados e apoiando o desenvolvimento técnico.

E como trabalhos futuros, espera-se aplicar o *guideline* proposto em outras empresas, para que se possa fazer um mapeamento da maturidade das práticas DevOps das empresas de desenvolvimento de software em diversos contextos regionais.

REFERÊNCIAS

ABES. Mercado Brasileiro de Software: panorama e tendências. 2021. Disponível em: https://abessoftware.com.br/dados-do-setor/. Acesso em: 24/04/2022.

ABRAHAMSSON, P. *et al.* New Directions on Agile Methods: A Comparative Analysis. **Proceedings of ICSE'03**, p. 244 – 254, 2003.

ALVES, J. S. O. Cultura organizacional e adoção de práticas DevOps : um estudo de caso na Companhia de Tecnologia da Informação do Estado de Minas Gerais - PRODEMGE. 2020.

ANISETT, M.; ARDAGNA, C. A. A Continuous Certification Methodology for DevOps. **Università degli Studi di Milano Milano, Italy**, 2019.

ANTONIO MUNIZ; ANALIA IRIGOYEN. **Jornada DevOps 2a edição (Jornada Colaborativa)**. [*S.l.*]: Brasport, 2020.

ARAUJO, M. Grafos de Maturidade DevOps. **Medium**, 2018. Disponível em: https://medium.com/@matheus.saraujo/grafos-de-maturidade-devops-6b8055579328. Acesso em: 02/11/2022.

ATAÍDES, A. da C.; GUEDES, L. G. de R. **Um método para acompanhamento e controle da implantação do CMMI**. 2006. Dissertação (Mestrado), 2006.

BATRA, P.; JATAIN, A. Measurement Based Performance Evaluation of DevOps. **International Conference on Computational Performance Evaluation (ComPE) North-Eastern Hill University, Shillong, Meghalaya, India. Jul 2-4**, 2020.

BECK, K. *et al.* Manifesto for Agile Software Development. 2001. Disponível em: agilemanifesto.org.

CMMI® for Development, Version 1.3. **Software Engineering Process Management Program**, 2010.

CRESWELL, J. W.; CRESWELL, J. D. **Projeto de pesquisa**: Métodos qualitativo, quantitativo e misto. 2. ed. [*S.l.*]: Penso, 2021.

FERNANDES, T. C. M. *et al.* Influência das práticas do DevOps nos processos de gestão de TI conforme o modelo COBIT 51. **Navus: Revista de Gestão e Tecnologia**, 2018.

GARCIA, I. Os desafios enfrentados e os benefícios conquistados com a cultura e as práticas DevOps. Universidade Federal Fluminense, Niterói, 2020.

GASPARAITE, M.; RAGAIŠIS, S. Comparison of devops maturity models. **Institute of Computer Science Vilnius University**, 2019.

GOOGLE CLOUD. Explore DORA's research program. 2014. Disponível em: https://www.devops-research.com/research.html. Acesso em: 15/11/2022.

HUMBLE, J. *et al.* **The DevOps Handbook**: How to Create World-Class Agility, Reliability, & Security in Technology Organizations. [*S.l.*]: It Revolution Press, 2021.

HUMBLE, J.; FARLEY, D. Continuous Delivery: Reliable Software Release Through Build, Test and Deployment Automation. 2010.

HUSSAINI, S. W. Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through systems approach. 17th International IEEE Conference on Intelligent Transportation Systems, 2014.

JUCÁ JUNIOR, A. da S.; CONFORTO, E. C.; AMARAL, D. C. Maturidade em gestão de projetos em pequenas empresas desenvolvedoras de software do Polo de Alta Tecnologia de São Carlos. **Gestão & Produção**, 2010.

KIM, G. **Manual de DevOps**: como obter agilidade, confiabilidade e segurança em organizações tecnológicas. [*S.l.*]: Alta Books, 2018.

LEVITA, C. de A. Proposta de Modelo para Avaliação da Maturidade DevOps. **Research, Society and Development, v. 7, n. 2,p. 01-33, e672128**, 2018.

MACHADO, F. B.; AMÊNDOLA, R. B. Análise Comparativa das Certificações de Qualidade CMM e ISSO 9000: Um Estudo de Caso da IBM Brasil. **VII SEMEAD**, São Paulo, 2004.

MEZZENA, B.; ZWICKER, R. Benefits and difficulties of the CMM model for software process improvement. **REGE Revista De Gestão**, 2007.

MOLESKY, J.; HUMBLE, J. Why enterprises must adopt devops to enable continuous delivery. **Cutter IT Journal, v. 24, n. 8**, p. 7 – 12, 2011.

MUNIZ, A. *et al.* **Jornada Azure DevOps:**: unindo teoria e prática com o objetivo de acelerar o aprendizado do Azure DevOps para quem está iniciando. [*S.l.*]: Brasport, 2021.

NEUBRAND, T.; HAENDLER, T. Development of a GQM-based Technique for Assessing DevOps Maturity. **University of Applied Sciences BFI**, Vienna, Austria, 2020.

NORMAN, F.; JAMES, B. **Software Metrics**: A Rigorous and Practical Approach, Third Edition. [*S.l.*]: CRC Press, 2020.

OROZCO, C. E.; CALVACHE, C. J. P. Modelo de métricas para apoyar la evaluación de DevOps en empresas de desarrollo de software. ResearchGate, 2022.

PEREIRA, B. **DORA metrics**: métricas inovadoras para medir a maturidade de DevOps. Elven Works, 2022. Disponível em: https://elven.works/dora-metrics-metricas-inovadoras-para-medir-a-maturidade-de-devops/. Acesso em: 15/11/2022.

PERERA, P.; BANDARA, M.; PERERA, I. Evaluating the Impact of DevOps Practice in Sri Lankan Software Development Organizations. **2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)**, Department of Computer Science & Engineering, University of Moratuwa Sri Lanka, 2016.

PONTES, T. B.; ARTHAUD, D. D. B. Metodologias ágeis para o desenvolvimento de softwares. **Ciência E Sustentabilidade**, 2018.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software:**: uma abordagem profissional. 9. ed. [*S.l.*]: AMGH, 2021. 1767 p.

RADSTAAK, J. Developing a DevOps Maturity Model: A validated model to evaluate the maturity of DevOps in organizations. **EEMCS: Electrical Engineering, Mathematics and Computer Science,**, 2019.

RANDELL, B.; NAUR, P. Software engineering: Report of a conference sponsored by the nato science committee, garmisch, Germany, 7th to 11th October 1968. 1969.

RONG, G.; ZHANG, H.; SHAO, D. CMMI Guided Process Improvement for DevOps Projects: An Exploratory Case Study. **IEEE/ACM International Conference on Software and System Processes**, 2016.

SANTOS, P. H. C. Métricas de Software para a segurança da informação: Definição e Aplicação em sistemas Web. **Centro Universitário de Anápolis-UniEvangélica**, 2019.

SILVEIRA, V. N. S. Os modelos multiestágios de maturidade: um breve relato de sua história, sua difusão e sua aplicação na gestão de pessoas por meio do People Capability Maturity Model (P-CMM). **Revista de Administração Contemporânea**, scielo, 2009.

SMITH, D. *et al.* State of DevOps 2021. **[S.l.]**, 2021. Disponível em: https://services.google.com/fh/files/misc/state-of-devops-2021.pdf. Acesso em: 17/11/2022.

SOARES, M. dos S. Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. 2004. Disponível em: https://infocomp.dcc.ufla.br/index.php/infocomp/article/view/68.

SOMMERVILLE, I. **Software Engineering**. 9th. ed. USA: Addison-Wesley Publishing Company, 2011.

SOUSA, L. DevOps – estudo de caso. **Instituto Politécnico de Coimbra**, Coimbra, 2019.

SOUSA, S. C. C. O impacto do cmm/cmmi na qualidade do software: um estudo sobre a percepção dos profissionais de tic. **Universidade Federal da Bahia**, 2009.

SRIVASTAVA, S. *et al.* Developer Velocity: How software excellence fuels business performance. 2020. Disponível em: https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/developer-velocity-how-software-excellence-fuels-business-performance. Acesso em: 26/04/2022.

TOMAS, N.; JINGYUE, L. An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps. **Software Institute and Department of Computer Science Nanjing University**, 2019.

VASCONCELOS, A.; MORAIS, L. Modelos de Maturidade para Processos de Software: CMMI e MPS.BR. *In*: ____. [*S.l.*]: UFPE, 2012. cap. 8.

VICTORINO, M.; BRÄSCHER, M. Organização da Informação e do Conhecimento, Engenharia de Software e Arquitetura Orientada a Serviços: uma Abordagem Holística para o Desenvolvimento de Sistemas de Informação Computadorizados. DataGramaZero - Revista de Ciência da Informação, 2009.

WETTINGER, J.; ANDRIKOPOULOS, V.; LEYMANN, F. Automated Capturing and Systematic Usage of DevOps Knowledge for Cloud Applications. **IEEE International Conference on Cloud Engineering**, 2015.

YIN, R. K. **Estudo de caso:**: planejamento e métodos. Porto Alegre: Bookman, 2015.

ZAROUR, M. *et al.* A research on DevOps maturity models. **International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-3, September**, 2019.

APÊNDICE A – QUESTIONÁRIO PESSOAS E CULTURA

APÊNDICE A – QUESTIONÁRIO PESSOAS E CULTURA

Abaixo estão descritos processos relacionados a cultura DevOps entre as equipes de desenvolvimento e operações. Seleciona a opção que mais se enquadra nas práticas da sua equipe.

Opção 5

- Todos os processos são unificados entre Dev e Ops e as informações compartilhadas por meio de ferramentas integradas.
- Os princípios e as práticas DevOps são institucionalizados em toda a empresa, com qualificação das equipes e alinhamento sobre a cultura DevOps.
- A metodologia de entrega está alinhada fim-a-fim em toda a infraestrutura, com práticas de gestão do conhecimento compartilhadas.

Opção 4

- Dev e Ops usam o mesmo conjunto de processos, mas alguns pontos ainda não estão unificados e as ferramentas são compartilhadas entre Dev, QA e Ops, mas não há um único fluxo entre elas.
- Acontecem trocas de conhecimento regulares entre Dev e Ops e um consenso dentro da empresa acerca dos princípios básicos do Dev Ops.
- Vários times têm total experiência e habilidades em DevOps e a maioria das equipes estão motivadas e alinhadas sobre a cultura DevOps, com a estrutura organizacional parcialmente adaptada para DevOps e o processo ágil é executado fim-a-fim, até a produção, por quase todas as equipes.

Opção 3

- Alguns elementos do processo são compartilhados entre Dev e Ops e as ferramentas são parcialmente compartilhadas, com compartilhamento moderado de conhecimento.
- Existe um consenso acerca de criações de atividades a respeito dos princípios básicos de DevOps e como aplicálos e vários times têm alguma experiência e habilidades em práticas DevOps.
- A empresa possui alguns entusiastas das práticas DevOps, com um portfólio de aplicações DevOps funcionando separadamente, com aplicação de metodologias ágeis nos processos de QA, mas os processos de *releases* não são ágeis.

Opção 2

- Dev e Ops usam processos distintos, mas as interfaces são alinhadas e compartilhadas, com pouca integração de ferramentas e algum compartilhamento de status, via painéis de controle.

APÊNDICE B – QUESTIONÁRIO QUALIDADE DO CÓDIGO

APÊNDICE B – QUESTIONÁRIO QUALIDADE DO CÓDIGO

Abaixo estão descritos práticas relacionados para garantir a qualidade do código. Seleciona a opção que mais se enquadra nas práticas da sua equipe.

Opção 5

- A qualidade do código é verificada no momento em que são gerados os *commits*, com os defeitos e não-conformidades sendo resolvidos de forma diária.
- Informações, métricas e indicadores de qualidade de código são compartilhados publicamente.
- As revisões de código são efetuadas de forma automatizada e integrada com o gerenciamento de build.
- São efetuados testes unitários de forma automatizada em torno de 80 a 90%.

Opção 4

- Análises e revisões de código são realizadas diariamente de forma automatizada, onde são analisados o nível de performance, segurança e usabilidade.
- São efetuados testes unitários de forma automatizada em torno de 40 a 50%.

Opção 3

- A revisão de código é realizada de forma manual e em pares, mas sua monitorização é realizada através de ferramentas, além disso são gerados alertas ao identificar defeitos na análise da qualidade do código.
- A verificação da qualidade do código ocorre em ambientes direcionados à integração.
- Os testes unitários são realizados de forma manual conforme acordado com o cliente.

Opção 2

- A revisão de código é realizada de forma manual e em pares.
- Início de automações de qualidade em ambientes de desenvolvimento.
- Testes unitários são realizados de forma manual, sendo levado em consideração a performance de cada desenvolvedor ao realizá-lo.

Opção 1

- Não ocorrem revisões de código e uso de padrões de codificação.
- Não há aplicabilidade de testes unitários, com a qualidade dependendo de cada desenvolvedor.

Selecione a opção que mais se enquadra nas práticas de sua equipe.

APÊNDICE C – QUESTIONÁRIO CONFIGURAÇÃO CÓDIGO

APÊNDICE C – QUESTIONÁRIO CONFIGURAÇÃO CÓDIGO

Abaixo estão descritas as práticas de configuração como código. Selecione a opção que mais se enquadra nas práticas da sua equipe.

Opção 5

- Utilização de infra-estrutura como código, de modo a automatizar o processo de criação dos ambientes de trabalho necessários e estimular a cooperação dos times de desenvolvimento e operações.
- As aplicações são hospedadas respeitando as políticas de definições de redes, assim como os serviços em nuvem.
- É encorajado a utilização de serviços SaaS, desde que o fornecedor garanta segurança e qualidade do serviço.
- Criação de ambientes de testes e desenvolvimento em máquinas virtuais de forma flexível e controlada, com as versões de software iguais às de produção.

Opção 4

- *Scripts* para criação de ambientes de máquinas virtuais começam a ser utilizados e desenvolvidos.
- Aproximação dos times de desenvolvimento e operações ao utilizar infraestrutura como código.
- Utilização de serviços em nuvem PaaS para hospedagem e implementação de hardware e software.
- Criação de ambientes de teste e desenvolvimento em máquinas virtuais flexíveis, mas nem sempre igual ao de produção.

Opção 3

- Utilização de *scripts* para a maioria dos trabalhos repetitivos de forma compartilhada entre os times e organizados em repositórios.
- Utilização do modelo de serviço de computação em nuvem IaaS.
- Para a criação dos ambientes de testes e desenvolvimento em máquinas virtuais é necessária a intervenção da equipe de operações.

Opção 2

- *Scripts* de automação começam a ser criados e compartilhados entre times de desenvolvimento, qualidade e operações.
- Não são utilizados serviços SaaS (*Software as a Service*).
- Ambientes de testes e desenvolvimento são geralmente diferentes dos ambientes de produção.

APÊNDICE D – QUESTIONÁRIO GESTÃO DE *BUILDS*

APÊNDICE D – QUESTIONÁRIO GESTÃO DE *BUILDS*

Abaixo estão descritas as práticas de gestão de *build*. Seleciona uma opção que se enquadre nas práticas da sua equipe.

Opção 5

- Gerenciamento de códigos e controle de versão é realizado por meio de sistemas compartilhados de código-fonte.
- Débito técnico é medido de forma contínua a cada *check-in*.
- Processo de integração de código, gerenciamento de build, gerenciamento de defeitos, configuração e testes unitários efetuados de forma automatizada e contínua. Além disso, o sistema de gerenciamento de configuração é integrado com desenvolvimento, teste e implantação.

Opção 4

- *Builds* abrangem obrigatoriamente automação de testes, configuração de ambiente, qualidade código, geração de dados, geração de tags, documentação técnica, entre outros.
- O sistema de gerenciamento de configuração de testes não é integrado com o sistema SCM.
- Débito técnico é medido a cada build e o seu monitoramento é realizado através de ferramentas com logs.
- As mudanças no ciclo de desenvolvimento são controladas por meio de ferramentas de gerenciamento.

Opção 3

- *Builds* incorporam alguns processos de qualidade tais como automação de testes, configuração de ambientes, qualidade de código, geração de tags e documentação técnica.
- As fases de produção e teste não são integradas com o sistema de SCM.
- Mudanças durante o ciclo de desenvolvimento são gerenciadas e monitoradas através de um processo manual de gerenciamento de requisitos.

Opção 2

- Geração da *build* é realizada em ambientes propícios com gestão centralizada de dependências e compilação do código-fonte.
- O gerenciamento de códigos e o controle de versões é efetuado por meio de um controlador instalado em um servidor.
- Produção e testes não são integrados com o sistema SCM.

APÊNDICE E – QUESTIONÁRIO GESTÃO DE *RELEASES*

APÊNDICE E – QUESTIONÁRIO GESTÃO DE *RELEASES*

Abaixo estão descritas as práticas de gestão de *release*. Seleciona uma opção que se enquadre nas práticas da sua equipe.

Opção 5

- Automação da implantação incorpora infraestrutura como código (IAC) sendo realizado de forma automática sendo possível implantar uma ou mais vezes por dia usando ferramentas DevOps.
- Os processos de implantação são totalmente replicáveis, medidos e melhorados continuamente, implantados em larga escala.
- O processo de *release* é totalmente automatizado por ferramentas DevOps.
- Os itens de configuração são monitorados através de relatórios automáticos usando integração de ferramentas DevOps assim como o provisionamento completamente virtualizado.

Opção 4

- Processo de definição da infraestrutura começa a ocorrer com micro-contêineres (ex.*Docker*).
- Implantação automatizada incorpora o provisionamento de ambientes com uso de conteinerização.
- O processo de gerenciamento de *release* é realizado por meio de ferramentas como (JIRA, RTC), mas sem rastreabilidade até a produção, e com novas funcionalidades implantadas pelo menos uma vez por semana.
- Controle de *release* é em grande parte automatizado, mas ainda é necessário algum envolvimento manual.

Opção 3

- Automação dos processos de aprovação entre ambientes de desenvolvimento, homologação e produção.
- Implantação baseada em *scripts*, sem utilização de ferramentas de implantação de *builds*.
- Gerenciamento de *release* realizado utilizando planilhas Excel, com rastreabilidade.
- Novas implantações em produção são realizadas pelo menos uma vez a cada duas semanas, para as implantações em homologação ocorre sempre que um build é gerado.

Opção 2

- Automação das cópias dos *builds* entre ambientes.

APÊNDICE F – QUESTIONÁRIO GESTÃO DE TESTES

APÊNDICE F – QUESTIONÁRIO GESTÃO DE TESTES

Abaixo estão descritas as práticas de gestão de testes. Seleciona a opção que mais se enquadra nas práticas da sua equipe.

Opção 5

- Práticas de injeção de falhas são inicializadas.
- Testes canários(testes A/B) são incorporados nas *releases*.
- Os ambientes de teste são idênticos aos ambientes de produção, incluindo a configuração de hardware e de software, sendo realizados continuamente de forma ágil, ocorrendo testes de sanidade em ambientes de produção.Casos de teste unitários são automatizados de ponta a ponta e um painel de controle da cobertura do código o qual é disponibilizado para todos os releases.
- O provisionamento do ambiente é totalmente automatizado por meio de ferramentas DevOps.
- Utilização de infraestrutura como código para definição dos ambientes para desenvolvimento, teste, homologação e produção.
- Requisitos de teste não-funcional são obtidos na fase inicial e modificados durante o ciclo de desenvolvimento.

Opção 4

- Processos de *build* possuem suítes de automação de testes robustas e com excelente cobertura de código.
- Os ambientes de teste são parcialmente semelhantes ao ambiente de produção, definições de software são idênticas, mas a de hardware não.
- Os casos de teste unitário são executados manualmente pelos desenvolvedores de forma individual posteriormente um relatório consolidado é compartilhado antes de cada *release.* Planos, casos e *scripts* de teste são criados antes do desenvolvimento, posteriormente os testes são feitos por *releases* iterativas.
- Testes funcionais são parcialmente automatizados, pois nem tudo é coberto, para os não funcionais, são automatizados processos de segurança ou usabilidade.

Opção 3

- Automação de testes ocorre para testes funcionais e testes de unidade, testes manuais possuem apoio de ferramentas de gerenciamento de teste, em relação aos testes contínuos existem muitos desafios devido à disponibilidade de recursos.
- A cobertura automatizada do código é monitorada, automação de são realizados ao nível de UI para aplicações, com cobertura média.
- Alguns ambientes de teste são idênticos aos de produção e outros similares.

APÊNDICE G – QUESTIONÁRIO GESTÃO DE CONFIGURAÇÃO

APÊNDICE G – QUESTIONÁRIO GESTÃO DE CONFIGURAÇÃO

Abaixo estão descritas as práticas de gestão de configuração. Seleciona a opção que mais se enquadra nas práticas da sua equipe.

Opção 5

- São criados projetos piloto para a absorção e internalização de novas tecnologias a serem utilizadas, com foco na melhoria contínua quantitativa do processo.
- Todos os binários de componentes e bibliotecas fazem parte da gestão de configuração.
- Gestão da configuração incorpora ambientes virtualizados e trata a infraestrutura como código (IAC), com processos sendo mensurados e controlados.
- Preocupação em tratar causas comuns de variação de processos de modo a melhorar o desempenho e satisfazer os objetivos quantitativos.

Opção 4

- Gerência de configuração generalizada com ferramentas e metodologias padronizadas.
- Grande parte dos binários de componentes e bibliotecas fazem parte da gestão de configuração.
- Políticas de controle de código-fonte incorporam barreiras sólidas de qualidade, sendo estas automatizadas.
- Processos de produção são mensurados e controlados, além de medições iniciais de desempenho e qualidade, com os processos sendo baseados em termos estatísticos e gerenciados ao longo da vida.

Opção 3

- Processos gerenciais e técnicos básicos bem definidos.
- Gestão da configuração começa a abranger binários de automação de tarefas.
- Automações de políticas de controle de código-fonte começam a ser experimentadas.
- Padrões, descrições de processo e procedimentos para um projeto são adaptados a partir de um conjunto de processos padronizados da organização que se ajustam a um projeto específico.

Opção 2

- Processo gerenciado, infraestrutura básica e versionamento de código com controle automatizado.
- Utilização de ferramentas de gestão de configuração de código (SCM), além de políticas de controle.

APÊNDICE H – QUESTIONÁRIO TESTE DE CARGA

APÊNDICE H – QUESTIONÁRIO TESTE DE CARGA

Abaixo estão descritas as práticas de testes de carga. Seleciona a opção que mais se enquadra nas práticas da sua equipe.

Opção 5

- Testes de carga fornecem políticas automatizadas de infraestrutura de código para responder a picos de utilização em ambientes de produção.
- Disponibilidade do sistema a maior parte do tempo em produção.
- O início da execução dos testes é comunicado de forma automatizada, possuindo painéis de controle e relatório de monitoração, com captura de defeitos feita de forma automatizada.
- Equipe de desenvolvimento cria e fornece as bases de dados de teste, que se assemelha ao de produção.

Opção 4

- Testes de carga, estresse e maturidade incorporados aos processos de *builds* e *releases*. As bases de teste são criadas pela equipe de teste, alinhado com os dados de produção.
- O planejamento de capacidade fornece diretrizes de provisionamento de ambiente em produção.
- A comunicação do início da execução dos testes é realizada de forma automatizada com uma lista de distribuição.
- Os defeitos são capturados utilizando ferramentas, mas sem métricas automatizadas, pois são geradas manualmente, usando planilhas.
- Requisitos identificados através de planilhas.

Opção 3

- Variações dos testes de carga como testes de performance, estresse e testes de maturidade começam a ser adotadas, com o time de testes criando sua própria base de dados, mas com baixo nível de semelhança com o de produção.
- Problemas de indisponibilidade são menores na produção.
- A comunicação do início da execução dos testes é realizada a partir de e-mails automáticos.
- Os defeitos são identificados em planilhas, mas não são geradas métricas.

Opção 2

- Padrões de qualidade são inseridos no desenvolvimento juntamente com a revisão por pares.

APÊNDICE I – QUESTIONÁRIO MONITORAÇÃO DE APLICAÇÃO

APÊNDICE I – QUESTIONÁRIO MONITORAÇÃO DE APLICAÇÃO

Abaixo estão descritas as práticas de monitoração de aplicação. Seleciona a opção que mais se enquadra nas práticas da sua equipe.

Opção 5

- Utilização de ferramentas DevOps para a monitoração, de forma integrada com os processos e ferramentas de desenvolvimento.
- Gerenciamento de incidentes e eventos são realizados através de ferramentas DevOps, de forma integrada com processos de *release* e desenvolvimento
- Limites de desempenho são monitorados de forma automatizada, com rápido tempo de resposta para a escalada de ambiente.
- Monitoração de erros de desempenho, alertas e erros através de ferramentas DevOps, com *feedback* automatizado para o time de desenvolvimento.

Opção 4

- Monitoramento de aplicativos incorporando práticas de telemetria os quais coletam dados de uso para gerar aprendizados de negócio para as equipes de desenvolvimento.
- Os problemas na produção são informados às equipes por meio de ferramentas que geram alertas de forma automatizada.
- Monitoração de desempenho automatizada.
- Monitoração de erros, desempenho e alertas através de ferramentas DevOps, mas sem integração com as ferramentas de desenvolvimento.

Opção 3

- Monitoramento de aplicativos em produção em formato contínuo.
- Informações de disponibilidade, carga e performance são regularmente repassados para as equipes de desenvolvimento.
- Bancos de dados são sincronizados usando scripts para a criação de ambientes iguais.
- O gerenciamento de incidentes e eventos são realizados através de ferramentas como (Remedy ou CQ), mas de forma não automatizada com o processo de *release*.

Opção 2

- Aplicações monitoradas não geram *feedback* acionáveis para times de desenvolvimento.
- Problemas na produção são informados às equipes por meio de solicitações/reclamações dos usuários finais por telefone, *e-mail*, *help desk* ou gerentes de conta.

APÊNDICE J – QUESTIONÁRIO REQUISITOS

APÊNDICE J – QUESTIONÁRIO REQUISITOS

Abaixo estão descritas as práticas de requisitos de software. Seleciona uma opção que se enquadre nas práticas da sua equipe.

Opção 5

- Requisitos funcionais e de negócio são coletados junto ao cliente através de técnicas e ferramentas DevOps de forma integrada no processo de desenvolvimento de Dev e Ops.
- São utilizados *backlogs* compartilhados para controle de trabalhos a serem desenvolvidos e quadros Kanban abrangendo diferentes equipes.
- A priorização dos requisitos é feita de forma ágil, através de ferramentas de gerenciamento de projetos com recursos compartilhados e integração com outras ferramentas, com métricas bem definidas com a coleta de dados e análise feitas através de ferramentas DevOps.
- Medição e modelagem de processos.

Opção 4

- Requisitos funcionais e de negócio são coletados junto ao cliente através de ferramentas DevOps e as tarefas a serem realizadas juntamente com os times são coletadas através de *backlogs*.
- Requisitos são priorizados de forma ágil utilizando uma ferramenta de gerenciamento de requisitos, mas sem integração com outras ferramentas e as métricas para gerenciamento são coletadas de forma parcial através de ferramentas DevOps.
- Os requisitos são identificados e rastreados para a *release* do sistema de produção através de planilhas excel, de forma muito bem planejada e rastreada, mas as atualizações são efetuadas de forma manual.

Opção 3

- Requisitos funcionais e de negócio são coletados junto ao cliente através de ferramentas de documentação de requisitos e as tarefas serem feitas junto ao time são documentadas através de *backlog* do time, com uso de quadro Kanban físico ou virtual ou através de Ticket/ Incidente.
- Requisitos são priorizados de forma ágil sem o uso de ferramentas para gerenciamento de requisitos e o planejamento dos projetos são realizados de forma ágil através de planilhas Excel.
- Métricas são coletadas e analisadas utilizando métodos manuais através de planilhas e os relatórios são gerados por meio da automação no Excel e os requisitos são rastreados utilizando planilhas Excel ou ferramentas próprias, sem planos de dados disponíveis ou não há indicadores de dados suficientes disponíveis.

Opção 2