Otimização do Processo de Desenvolvimento de Software: Aplicação de Análise de Causa Raiz para Redução de Falhas em Produção

Júlio César Spanholi

[1]Instituto de Tecnologia - Universidade de Passo Fundo - UPF
Caixa Postal 611 - CEP: 99001-970 – Passo Fundo – RS – Brasil.

183154@upf.br

Abstract. *This study investigates the causes of production failures in software development within a financial squad, aiming to reduce the occurrence of these failures and discuss process improvements. Root Cause Analysis (RCA) techniques were applied, including the Five Whys Method and the Fishbone Diagram, using data extracted from Azure DevOps and structured interviews with the team. The results identified recurring issues such as lack of technical training, incomplete specifications, communication failures, inconsistencies in development environments, deadline pressures, and outdated documentation. Based on these analyses, the study suggests actions such as team training, documentation standardization, test automation, and the adoption of collaborative tools. Although the proposals require validation in the financial squad's context, the findings highlight the importance of structured processes to ensure stable and reliable systems.*

Keywords: *Root Cause Analysis; Software Development; Process Improvement.*

Resumo. *Este estudo investiga as causas de falhas em produção no desenvolvimento de software dentro de uma squad financeira, com o objetivo de reduzir a ocorrência dessas falhas e discutir melhorias nos processos. Para isso, foram aplicadas técnicas de Análise de Causa Raiz (RCA), incluindo o Método dos Cinco Porquês e o Diagrama de Causa e Efeito, utilizando dados extraídos do Azure DevOps e entrevistas estruturadas com a equipe. Os resultados identificaram problemas recorrentes, como falta de capacitação técnica, especificações incompletas, falhas de comunicação, inconsistências nos ambientes de desenvolvimento, pressão por prazos e documentação desatualizada. Com base nessas análises, o estudo sugere ações como capacitação da equipe, padronização da documentação, automação de testes e adoção de ferramentas colaborativas. Embora as propostas exijam validação no contexto da squad financeira, os resultados destacam a importância de processos estruturados para garantir sistemas estáveis e confiáveis.*

Palavras-chave: *Análise de Causa Raiz; Desenvolvimento de Software; Melhoria de Processos.*

1. Introdução

O desenvolvimento de software em squads financeiras enfrenta desafios recorrentes relacionados a falhas em produção, o que impacta negativamente a experiência do

usuário, eleva custos e compromete a confiança no produto [Chapin et al. 2001]. Em contextos financeiros, onde transações e dados sensíveis são geridos, essas falhas podem acarretar prejuízos financeiros, perda de credibilidade e até comprometer relações de negócios.

Desde junho de 2022, a squad financeira analisada registrou 115 falhas, demandando 876,5 horas para correções. Estudos indicam que desenvolvedores gastam, em média, 50% do tempo encontrando e corrigindo erros, gerando custos elevados e impactando a indústria de software devido a ineficiências e atrasos [Cambridge Judge Business School 2013]. Esses dados reforçam a necessidade urgente de otimizar o desenvolvimento, visando aumentar a eficiência e garantir a estabilidade do produto final.

Para abordar esses desafios, este estudo adota o *Método dos Cinco Porquês* e o *Diagrama de Causa e Efeito*, duas técnicas de *Análise de Causa Raiz* (RCA) amplamente empregadas para investigar as causas reais dos problemas, indo além dos sintomas superficiais [Andersen and Fagerhaug 2006]. A coleta de dados foi realizada por meio do Azure DevOps, já utilizado para monitoramento das atividades, além de entrevistas estruturadas com a equipe, proporcionando uma visão detalhada dos problemas e suas possíveis causas.

O objetivo principal deste estudo é identificar as origens das falhas em produção e propor melhorias para otimizar o processo de desenvolvimento de software, reduzindo a recorrência de falhas e aprimorando a eficiência operacional. Com base nas informações coletadas, será elaborado um Diagrama de Causa e Efeito, que organiza as principais causas dos problemas e sugere medidas para melhorias. Embora a implementação das soluções propostas não seja o foco imediato deste trabalho, as sugestões apresentadas oferecem direções para fortalecer o processo de desenvolvimento e garantir entregas mais confiáveis.

Este artigo está organizado de forma a facilitar a compreensão do estudo. Primeiramente, o Referencial Teórico apresenta conceitos como a Análise de Causa Raiz, o Método dos Cinco Porquês e o Diagrama de Causa e Efeito, além de práticas de melhoria contínua e uma revisão de literatura. Em seguida, o Estudo de Caso descreve o contexto da squad financeira analisada, os desafios enfrentados e exemplos de falhas críticas. Na Metodologia, são detalhados os métodos de coleta e análise de dados utilizados no estudo. Os Resultados e a Discussão exploram os fatores identificados, destacando as causas das falhas e sugerindo possíveis melhorias. Por fim, as Considerações Finais sintetizam as principais contribuições do trabalho e indicam direções para estudos futuros.

2. Referencial Teórico

Este trabalho busca otimizar o processo de desenvolvimento de software, com foco na redução de falhas em produção. Para isso, é utilizada a Análise de Causa Raiz (RCA), que investiga as causas dos problemas e propõe soluções duradouras. O Método dos Cinco Porquês complementa a RCA, facilitando a identificação das causas principais de forma direta e eficaz.

As informações obtidas por meio das análises serão estruturadas em um Diagrama de Causa e Efeito, que auxilia na visualização de falhas e padrões que precisam ser corrigidos. Além disso, a melhoria contínua é apoiada pelo Azure DevOps, utilizado para

monitorar e analisar as falhas, controlar o progresso das tarefas e identificar padrões recorrentes por meio de dashboards personalizados.

Nas seções seguintes, cada uma dessas abordagens será detalhada, destacando sua importância para o trabalho.

2.1. Análise de Causa Raiz (RCA)

A RCA é uma metodologia amplamente empregada para compreender as causas fundamentais de problemas, indo além dos sinais aparentes. Ela permite identificar o que aconteceu, como aconteceu e, principalmente, por que aconteceu, sendo uma base sólida para implementar soluções que evitem a repetição de falhas [Andersen and Fagerhaug 2006].

No desenvolvimento de software, a RCA desempenha um papel crucial ao tratar problemas complexos, como falhas em produção, que muitas vezes possuem múltiplas origens. Conforme explicado pela AWS, concentrar-se apenas nos sintomas não resolve os problemas de forma eficaz. A análise das causas subjacentes possibilita a criação de ações corretivas que reduzam impactos futuros e fortaleçam a qualidade dos sistemas [Amazon Web Services 2024].

2.2. Método dos Cinco Porquês

O *Método dos Cinco Porquês* é uma técnica simples, porém poderosa, utilizada para identificar as causas mais profundas de um problema ao perguntar ”Por que?”repetidamente até que a verdadeira causa seja descoberta. Essa abordagem evita soluções superficiais e possibilita uma compreensão mais completa do problema, promovendo a implementação de ações corretivas eficazes [Rooney and Vanden Heuvel 2004]. Neste trabalho, o método será aplicado em entrevistas com a equipe de desenvolvimento para investigar a origem das falhas em produção, com o objetivo de identificar falhas no processo e propor melhorias com base em evidências concretas.

2.3. Diagrama de Causa e Efeito

O *Diagrama de Causa e Efeito*, também conhecido como *Diagrama de Ishikawa*, é uma ferramenta que categoriza as causas de um problema em grupos como métodos, pessoas, ferramentas, materiais, ambiente e medidas. Desenvolvido por Kaoru Ishikawa, o diagrama oferece uma visualização clara das relações entre diferentes fatores e os problemas observados [Ishikawa 1982].

Essa técnica promove discussões mais direcionadas e ajuda a evitar soluções que não tratam as causas principais. Neste trabalho, o diagrama será utilizado para organizar as informações obtidas pelo Método dos Cinco Porquês, permitindo que a equipe visualize as origens das falhas de forma estruturada e tome decisões mais assertivas e baseadas em evidências [Donato 2024].

2.4. Melhoria Contínua

Nas metodologias ágeis, a Melhoria Contínua permite ajustes constantes que otimizam processos e garantem a qualidade do produto. A RCA e o Método dos Cinco Porquês são fundamentais para identificar e resolver as causas dos problemas de forma definitiva, evitando correções superficiais [Rooney and Vanden Heuvel 2004].

Neste contexto, o *Azure DevOps* é utilizado como uma ferramenta de controle de informações, através de dashboards que acompanham a classificação das falhas. Esses dashboards permitem visualizar o status e a categorização das falhas em tempo real, facilitando o monitoramento do progresso e o ajuste das estratégias com base nos dados obtidos [Jabbari et al. 2016].

2.5. Revisão de Literatura

Esta revisão de literatura identifica estudos que abordam técnicas de redução de falhas em produção e práticas de melhoria contínua nos processos de desenvolvimento de software. A pesquisa foi realizada em bases de dados reconhecidas, como *IEEE Xplore*, *ScienceDirect* e o *International Journal of Science and Research (IJSR)*, com foco na seleção de artigos atuais e relevantes para o tema. Foram selecionados trabalhos que atendem aos seguintes critérios:

- Estudos que apresentem estratégias para otimizar processos de desenvolvimento de software, com foco na redução de falhas em produção.
- Trabalhos que explorem metodologias de RCA, frameworks de qualidade de software ou práticas de melhoria contínua.
- Artigos que discutam os desafios e as oportunidades no ciclo de vida do desenvolvimento de software, incluindo o uso de ferramentas e métodos para controle de qualidade.

2.5.1. Extração de Dados

Para a extração de dados, foram utilizadas palavras-chave como ”Gestão de Defeitos”, ”Análise de Causa Raiz”e ”Qualidade de Software”, selecionando estudos relevantes para a otimização no desenvolvimento de software. A busca foi limitada a artigos atuais e alinhados ao escopo deste trabalho. Durante o processo, os resultados foram filtrados, excluindo estudos fora do foco do estudo ou com metodologias pouco claras.

2.5.2. Trabalhos Relacionados

O artigo ”Defect Management and Root Cause Analysis: Pillars of Excellence in Software Quality Engineering”, de Shravan Pargaonkar [Pargaonkar 2023], destaca a importância da gestão de defeitos e da análise de causa raiz como componentes essenciais para a melhoria contínua no processo de desenvolvimento de software. O autor explica que a gestão de defeitos é crucial para identificar, monitorar e corrigir problemas durante todo o ciclo de vida do desenvolvimento. Ao se concentrar na gestão de defeitos, as equipes podem evitar que falhas cheguem aos usuários finais, o que resulta em produtos mais estáveis e de maior qualidade. Já a análise de causa raiz vai além do simples diagnóstico de falhas, buscando entender as causas fundamentais dos defeitos. Isso permite a implementação de ações corretivas que evitam a recorrência desses problemas no futuro. Pargaonkar conclui que a combinação dessas práticas não só torna o processo de desenvolvimento mais eficiente, mas também resulta em produtos mais confiáveis e alinhados às expectativas dos clientes. Esse ponto está em consonância com o objetivo deste

trabalho, que também visa otimizar o processo de desenvolvimento, reduzir falhas em produção e, ao focar nas causas principais, garantir um produto final de maior qualidade.

O estudo "Risk Assessment on Software Development using Fishbone Analysis", de Muhammad Talha Riaz et al. [Riaz et al. 2019], propõe uma abordagem prática para identificar e analisar os riscos no desenvolvimento de software, utilizando o Diagrama de Espinha de Peixe (ou Diagrama de Causa e Efeito). Essa ferramenta é aplicada para descobrir as causas raiz dos problemas identificados durante o processo de desenvolvimento. Os autores conduziram suas pesquisas em várias empresas de software, empregando uma combinação de avaliações qualitativas e quantitativas para classificar e priorizar os riscos observados. Entre os principais problemas identificados estão estimativas de tempo erradas e alocação inadequada de recursos, que resultaram em atrasos e sobrecarga de trabalho. O estudo mostrou que o uso do Diagrama de Espinha de Peixe foi crucial para organizar e priorizar as causas dos riscos, permitindo a criação de estratégias mais eficazes para mitigar esses problemas. Essa abordagem complementa diretamente o objetivo deste trabalho, que também busca reduzir as falhas em produção ao atacar suas causas principais. A utilização do Diagrama de Espinha de Peixe reforça a análise de causa raiz deste estudo, pois proporciona uma visão clara e estruturada das interações entre diferentes fatores, facilitando a compreensão dos problemas e suas soluções.

O trabalho "Not all bugs are the same: Understanding, characterizing, and classifying bug types", de Gemma Catolino et al. [Catolino et al. 2019], realiza uma análise aprofundada de uma grande quantidade de relatórios de falhas de projetos de código aberto, como Mozilla, Apache e Eclipse. Os autores desenvolveram uma classificação detalhada, com nove tipos principais de falhas, baseada em uma análise minuciosa desses relatórios. Além disso, foi criado um modelo automatizado para identificar esses tipos de falhas, o que acelera o processo de triagem e facilita a designação do desenvolvedor mais adequado para resolvê-las. Com uma classificação clara dos tipos de falhas, as equipes de desenvolvimento podem priorizar e abordar os problemas de maneira mais eficaz, evitando a recorrência de falhas semelhantes no futuro. Esse estudo complementa a análise de causa raiz realizada neste trabalho, pois, ao entender melhor os diferentes tipos de falhas, torna-se mais fácil implementar soluções mais específicas e direcionadas. A classificação proposta pelos autores oferece uma visão mais organizada dos fatores que contribuem para os problemas, ajudando a melhorar a qualidade do produto e a eficiência do processo de correção.

3. Estudo de Caso

3.1. Contexto e Propósito da Squad

O estudo de caso é focado em uma squad financeira de uma instituição de ensino, que tem como principal objetivo gerenciar a inadimplência dos alunos ativos e inativos. Essa equipe é composta por uma *Agile Master (AM)*, responsável por facilitar os ritos ágeis e garantir a fluidez das entregas; um *Product Owner (PO)*, que prioriza as demandas do backlog e garante o alinhamento com as necessidades do negócio; três *desenvolvedores*, que implementam as soluções e sustentam o sistema; um *Quality Assurance (QA)*, que realiza testes e valida a qualidade do produto entregue; e uma *analista financeira*, que dá suporte técnico e estratégico relacionado às demandas específicas de gestão de inadimplência.

Essa composição multidisciplinar permite que a squad atue de forma colaborativa, cobrindo desde o levantamento de requisitos até a sustentação e evolução dos sistemas financeiros. As principais frentes de atuação da squad incluem:

- **Sustentação de Ferramentas de Negociação**: Interface usada pelos atendentes para negociar débitos com os estudantes.
- **Integração com APIs de Cobrança**: Comunicação entre o sistema financeiro da instituição e as assessorias de cobrança externas.
- **Cadastro e Gestão de Campanhas de Cobrança**: Configuração de campanhas e gerenciamento de recebimentos, facilitando a recuperação de créditos.
- **Serviço de Baixa Automática de Acordos**: Automatização do processo de baixa de acordos financeiros.
- **Sustentação do Programa de Bolsas**: Manutenção do sistema de gestão de bolsas de estudo para os estudantes.

3.2. Descrição e Impacto do Problema

Desde junho de 2022, a squad financeira registrou um aumento expressivo no número de falhas em produção, contabilizando 115 ocorrências até o momento. A correção dessas falhas exigiu aproximadamente 876,5 horas de trabalho, sobrecarregando a equipe e aumentando os custos operacionais. Essas falhas frequentemente ocorrem em ambiente de produção, impactando diretamente a confiabilidade do sistema e a experiência dos usuários. Entre os problemas, destacam-se interrupções de serviço, erros em cálculos financeiros e inconsistências na comunicação com assessorias de cobrança.

A análise dos problemas revelou que a equipe dedica um tempo considerável a atividades de correção, desviando o foco de iniciativas estratégicas e da implementação de novas funcionalidades. Essa situação destaca a necessidade de uma investigação aprofundada das causas das falhas para orientar ações corretivas e reduzir seu impacto.

A alta frequência de falhas em produção impacta diretamente a operação da squad e resulta em consequências como:

- **Interrupção de Serviços**: As falhas causam paradas temporárias, impedindo o acesso a recursos críticos para negociações e acordos financeiros.
- **Aumento de Custos**: O tempo investido na correção das falhas reduz a produtividade da equipe e aumenta os custos indiretos do projeto.
- **Experiência do Usuário Comprometida**: Erros em cálculos e inconsistências na comunicação com as assessorias de cobrança afetam a confiança dos usuários no sistema.

Esses impactos reforçam a necessidade de implementar melhorias no processo de desenvolvimento para reduzir a recorrência de falhas e garantir a estabilidade e confiabilidade do sistema.

3.3. Exemplos Práticos de falhas Críticas

As falhas críticas apresentadas a seguir ilustram problemas recorrentes que geraram impactos significativos no ambiente de produção da squad financeira. Essas falhas evidenciam como questões relacionadas à especificação, manipulação de dados e lógica de codificação podem comprometer a confiabilidade do sistema, gerando custos adicionais e dificuldades para os usuários.

- **Especificação Ausente ou Incompleta**: Um exemplo ocorreu na funcionalidade de ”Negociar”, onde a falta de detalhamento na aplicação das taxas de parcelamento gerou cálculos incorretos. Em outro caso, a ausência de pacotes necessários impediu o cancelamento de negociações, gerando mensagens de erro no ambiente de produção. Esses problemas demonstram a importância de especificações completas e detalhadas para evitar falhas na implementação.
- **Manipulação de Dados Incorreta**: O sistema falhou ao exibir débitos de alunos ao tentar remover títulos da cobrança, devido a um erro na lógica de manipulação dos dados. Isso impediu a conclusão do processo, causando impacto no fluxo de trabalho. A análise revelou que a lógica de dados não estava corretamente implementada para lidar com as diferentes condições do sistema, resultando em falhas de exibição de informações.
- **Conteúdo ou Codificação Errada**: Erros na configuração do parcelamento de pagamentos impediram a formalização de promessas de pagamento via cartão de crédito à vista. A lógica da aplicação permitia apenas parcelas predefinidas, desconsiderando a possibilidade de uma única parcela com taxas ajustadas. Esse erro de codificação limitou a funcionalidade e gerou frustração entre os usuários.

4. Metodologia

Este estudo adota uma abordagem mista, combinando métodos quantitativos e qualitativos para identificar e entender as causas das falhas em produção. Essa combinação permite observar tanto a frequência e características das falhas quanto os fatores humanos e organizacionais envolvidos.

A primeira etapa consistiu na análise do dashboard de qualidade no Azure DevOps, mantido pela equipe e utilizado para registrar dados sobre as falhas, incluindo backlog, classificação e controle de atividades. Com base nesses dados, foram realizadas entrevistas estruturadas com a equipe de desenvolvimento, aplicando o método dos Cinco Porquês para investigar profundamente as causas dessas falhas. As respostas obtidas fundamentaram a elaboração de um Diagrama de Causa e Efeito, que facilita a visualização das conexões entre fatores e a priorização de ações corretivas.

4.1. Coleta de Dados

A coleta de dados envolveu a extração de informações do dashboard de qualidade no Azure DevOps e a realização de entrevistas estruturadas com membros da equipe. Esses métodos complementares foram essenciais para capturar tanto os aspectos técnicos quanto as percepções dos profissionais diretamente envolvidos no processo de desenvolvimento.

O *dashboard* do *Azure DevOps* (Figura 1), já em uso pela equipe, forneceu dados quantitativos essenciais sobre o histórico das falhas, incluindo informações como classificação, prioridade, ambiente de ocorrência, tipo e tempo de resolução. Essas informações ajudaram a identificar padrões de recorrência e áreas mais problemáticas no sistema, orientando a formulação das perguntas utilizadas nas entrevistas.

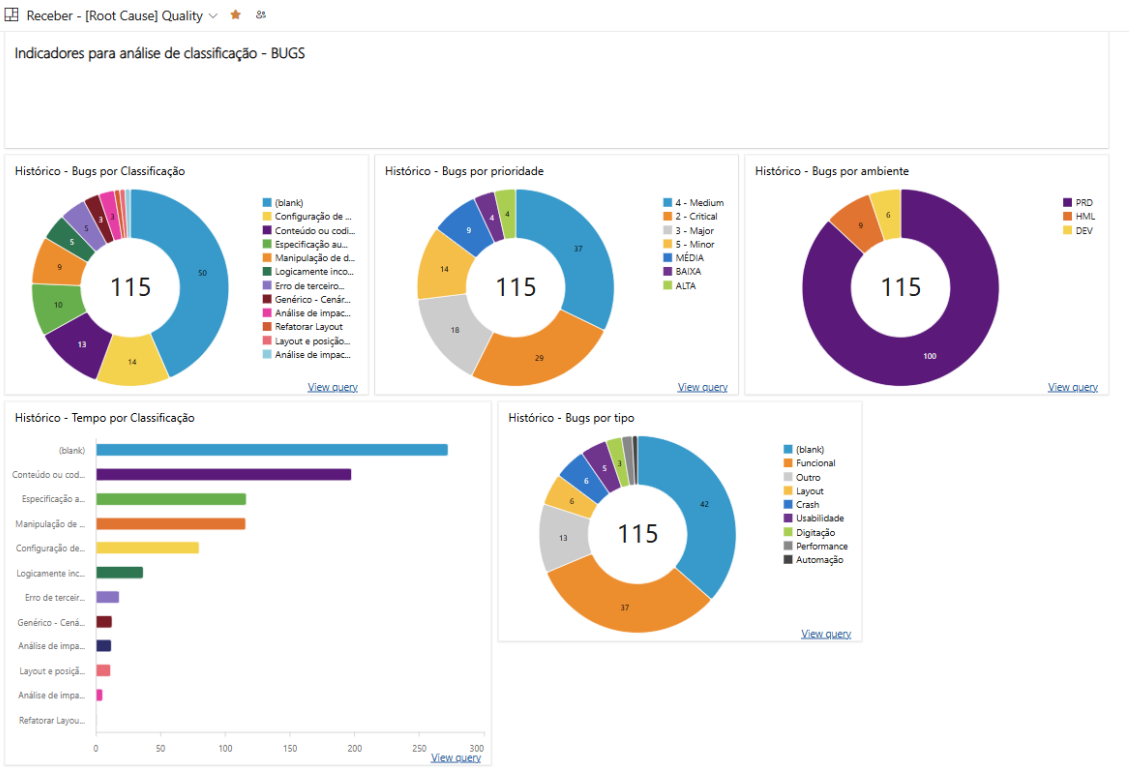

Figura 1. Dashboard de Qualidade no Azure DevOps, destacando a análise de classificações e tempos de resolução de falhas

A análise inicial apontou que a classificação ”Branco”foi a mais recorrente, com 46 falhas registradas e 268 horas de resolução, seguida por ”Configuração de Parâmetro Errada”(14 falhas, 80 horas), ”Conteúdo ou Codificação Errada”(13 falhas, 197,5 horas) e ”Especificação Ausente ou Incompleta”(10 falhas, 116,25 horas). Essas classificações, que apresentam alta frequência e demandam tempo significativo de resolução, serviram como referência para o desenvolvimento das perguntas direcionadas à equipe, buscando explorar as causas e compreender as áreas mais problemáticas.

As entrevistas foram realizadas com o *Product Owner* (PO), dois desenvolvedores (um deles estagiário) e o líder técnico, garantindo uma variedade de perspectivas sobre o processo de desenvolvimento. O PO abordou questões relacionadas aos requisitos de negócio e alinhamento, enquanto os desenvolvedores trouxeram perspectivas sobre as dificuldades encontradas na implementação das soluções. O líder técnico, por sua vez, forneceu uma visão estratégica e destacou pontos relacionados ao fluxo de trabalho e à supervisão das atividades.

Para incentivar respostas sinceras e detalhadas, foi garantida a confidencialidade dos participantes. As orientações para responder ao questionário foram enviadas por e-mail, com um prazo estabelecido entre 08/10 e 11/10, assegurando que os participantes tivessem tempo suficiente para refletir sobre as questões propostas.

4.2. Análise das respostas

A análise das respostas foi realizada em duas etapas complementares. A primeira etapa utilizou os dados do dashboard para identificar padrões de falhas e as áreas mais

afetadas. Essas informações ajudaram a entender quais aspectos do processo de desenvolvimento estavam apresentando mais problemas, o que permitiu direcionar as entrevistas para investigar mais a fundo as causas dessas falhas. A partir dos dados quantitativos, foi possível identificar as falhas mais recorrentes e os pontos críticos em que a equipe estava enfrentando maiores dificuldades.

Na segunda etapa, as respostas dos participantes foram agrupadas em temas principais e analisadas quantificadamente. Esse processo permitiu identificar os fatores mais impactantes para o processo de desenvolvimento. Os problemas foram divididos em duas categorias: organizacionais e técnicos. Entre os problemas organizacionais, os mais citados envolvem falhas na comunicação e no alinhamento entre as equipes, especialmente no que diz respeito ao entendimento claro sobre os requisitos e o escopo das tarefas. A falta de um alinhamento adequado entre o PO e a equipe de desenvolvimento dificultou a definição precisa dos critérios de aceitação e das prioridades do backlog. Já os problemas técnicos envolveram falhas nas especificações, que muitas vezes eram incompletas ou mal detalhadas, além de dificuldades na integração entre sistemas e a complexidade do processo de desenvolvimento, o que aumentava o tempo necessário para concluir as tarefas.

Os dados mostraram que os problemas organizacionais e técnicos estão interligados e afetam tanto o desempenho da equipe quanto a qualidade das entregas. A comunicação ineficaz e o alinhamento inadequado das expectativas de desenvolvimento, além das especificações incompletas, foram identificados como fatores críticos, frequentemente resultando em retrabalho. Esses fatores foram organizados em temas principais, com base na frequência e importância atribuída pelos entrevistados. A Tabela 1 apresenta um resumo desses fatores, destacando as áreas que mais precisam de atenção para aprimorar o processo de desenvolvimento.

Tabela 1. Distribuição de Temas e Frequências nas Respostas

Tema/Subtema	**Frequência**
Falta de alinhamento	13
Infraestrutura limitada para testes	13
Revisão de código superficial	12
Poucas reuniões de refinamento	11
Validação das especificações	10
Falta de padronização na documentação	6
Automação de testes incompleta	5
Prazos apertados para entrega	3
Dependência de uma única pessoa	1
Especificações incompletas	1

Com base nesses temas, foi desenvolvido um Diagrama de Causa e Efeito (Figura 2), que organiza os fatores identificados em seis categorias principais: métodos, pessoas, ferramentas, materiais, ambiente e medidas. Esse diagrama permite visualizar as interdependências entre os fatores, auxiliando na identificação de prioridades para ações corretivas.

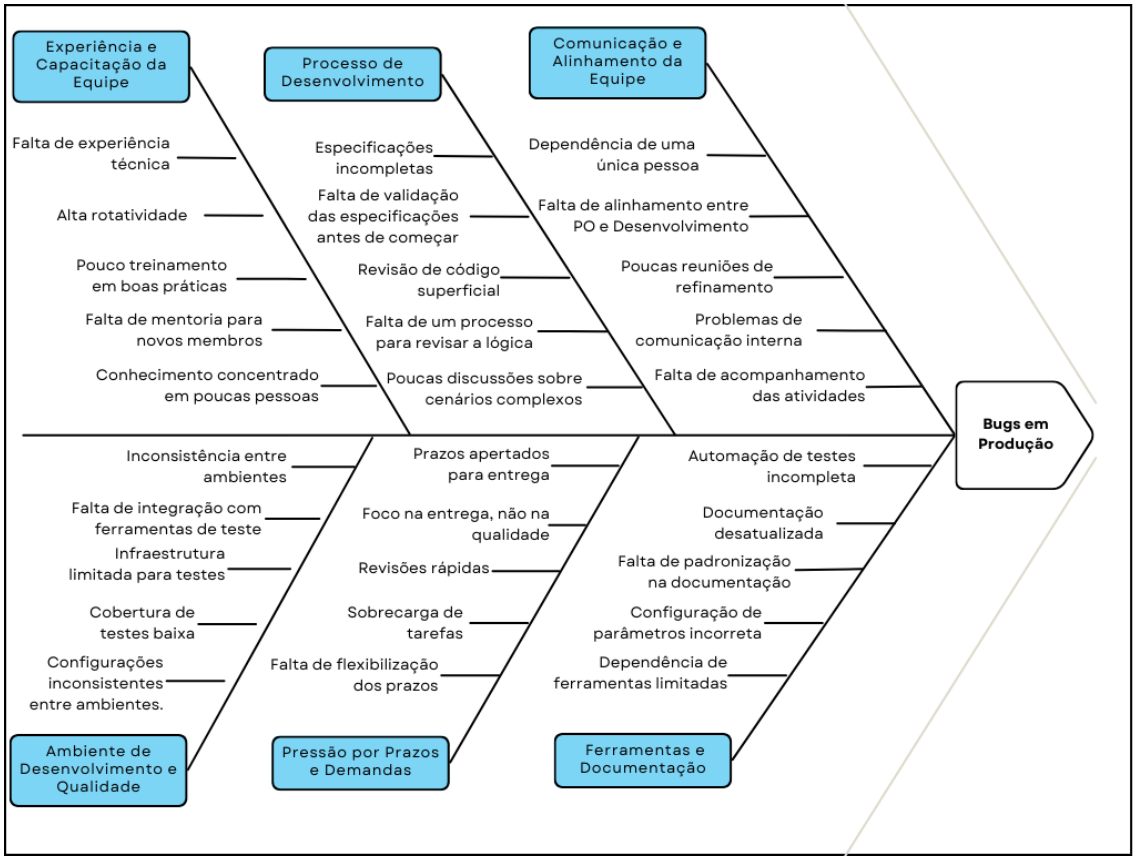

Figura 2. Diagrama de Causa e Efeito

A integração de análises qualitativas e quantitativas permitiu construir uma visão clara e estruturada das causas mais relevantes para as falhas em produção. Esse esforço fornece uma base sólida para as propostas de melhoria apresentadas, direcionando ações que podem mitigar problemas recorrentes e fortalecer o processo de desenvolvimento.

5. Resultados

Com base na metodologia, as entrevistas estruturadas foram analisadas para identificar e organizar as causas principais das falhas em um Diagrama de Causa e Efeito. Essa análise ajudou a apontar fatores críticos, categorizados de acordo com a frequência e a relevância atribuída pelos entrevistados.

A análise também contou com a experiência prática no time, que ajudou a contextualizar as respostas e destacar áreas problemáticas. Por exemplo, comentários sobre a falta de treinamento e a dependência de uma única pessoa para decisões importantes foram conectados a situações recorrentes no time, reforçando a necessidade de medidas específicas.

As respostas foram agrupadas em padrões comuns, levando em conta repetições diretas e variações que apontavam para problemas similares. Temas como experiência da equipe, comunicação, infraestrutura e demandas do processo de desenvolvimento se destacaram como os mais relevantes. Essa análise integrada forneceu uma visão clara dos pontos críticos e serviu de base para as propostas de melhoria.

- **Experiência e Capacitação da Equipe:** Os entrevistados apontaram a falta de experiência técnica e a carência de treinamentos específicos como razões frequentes para as falhas em produção. Comentários sobre rotatividade de pessoal, falta de mentoria e concentração de conhecimento em poucos membros também foram recorrentes, sugerindo uma necessidade de capacitação e suporte mais efetivos para novos integrantes.
- **Processo de Desenvolvimento:** Falhas no fluxo de desenvolvimento foram citadas, especialmente aquelas relacionadas a especificações incompletas e ausência de validações antes da codificação. A falta de tempo para revisões detalhadas e a qualidade limitada das especificações durante o refinamento das histórias foram identificadas como fatores que aumentam as chances de erros e retrabalho.
- **Comunicação e Alinhamento da Equipe:** Diversas respostas indicaram problemas de comunicação, incluindo a dependência de uma única pessoa para decisões críticas e o desalinhamento entre o PO e os desenvolvedores. Essa falta de alinhamento foi associada a mal-entendidos e falhas na entrega, indicando a necessidade de uma comunicação mais estruturada e de reuniões regulares de alinhamento.
- **Ambiente de Desenvolvimento e Qualidade:** Foram mencionadas inconsistências entre os ambientes de desenvolvimento, homologação e produção, além de limitações na infraestrutura de testes. A ausência de integração robusta entre ambientes foi apontada como um problema que impede a detecção de falhas antes de chegarem à produção, destacando a importância de uma infraestrutura mais consistente.
- **Pressão por Prazos e Demandas:** A pressão por prazos e a urgência em algumas entregas foram apontadas como fatores que comprometem a qualidade do trabalho. Esse cenário reduz o tempo para preenchimento de documentações e execução de testes detalhados, aumentando o risco de falhas passarem despercebidas e reforçando a necessidade de um equilíbrio entre qualidade e prazos.
- **Ferramentas e Documentação:** A documentação desatualizada e o preenchimento inconsistente do campo ”causa raiz”no Azure DevOps foram identificados como obstáculos para entender e documentar as causas dos problemas. Manter a documentação atualizada e padronizar o preenchimento do campo ”causa raiz”no Azure DevOps foi considerado essencial para análises futuras. A automação de testes também foi recomendada para garantir cobertura dos parâmetros críticos e aumentar a eficiência do time.

Essas categorias foram definidas considerando as observações feitas durante a análise, a frequência e o impacto das respostas obtidas nas entrevistas, além do contexto prático vivenciado pela equipe. O Diagrama de Causa e Efeito elaborado no estudo oferece uma visão estruturada das relações entre os fatores identificados, orientando ações para mitigar falhas recorrentes e fortalecer os processos de desenvolvimento, com foco em entregas mais estáveis e confiáveis, sujeitas a validação em estudos futuros.

6. Discussão

Nesta seção, são apresentadas as principais reflexões sobre os problemas identificados durante o estudo e as possíveis abordagens para enfrentá-los. As sugestões propostas não devem ser vistas como soluções definitivas, mas como direções a serem exploradas em estudos futuros. As propostas foram desenvolvidas com base nas ferramentas

disponíveis e nos desafios específicos da squad financeira, mas sua aplicação e eficácia precisam ser avaliadas de forma sistemática.

Um dos problemas levantados foi a falta de experiência técnica na equipe e a necessidade de treinamentos mais direcionados. Isso resulta em sobrecarga para alguns membros, enquanto outros não possuem autonomia suficiente para lidar com as demandas do dia a dia. Esse ponto refere-se a questões de equilíbrio técnico e autonomia da equipe. Por meio de conceitos e práticas em Engenharia de Software, uma possível abordagem seria explorar temas como a divisão de responsabilidades e papéis no desenvolvimento de software [Suhanda and Pratami 2021], frameworks de treinamento para capacitação progressiva (como o Miro) [Miro 2024], além de mentoria técnica especializada. Essas práticas podem ajudar na distribuição equilibrada do conhecimento técnico, reduzindo a sobrecarga e aumentando a eficiência da equipe.

Outro ponto identificado foi a ausência de especificações detalhadas e validações consistentes durante o desenvolvimento, o que aumenta o risco de falhas. Para enfrentar esse problema, a implementação de testes automatizados surge como uma abordagem válida. Testes de software visam garantir a qualidade do produto, identificando erros antes da entrega final [Blascke et al. 2023]. Ferramentas como o *Cypress* [Cypress.io 2024] permitem verificar automaticamente se os requisitos estão sendo cumpridos, reduzindo a dependência de validações manuais. Além disso, integrar esses testes a pipelines utilizando o *Azure DevOps* [Microsoft 2024a], que já faz parte das ferramentas adotadas pela empresa, pode otimizar o processo, automatizando execuções e fornecendo feedback imediato sobre possíveis problemas. Essa abordagem contribui diretamente para a confiabilidade e consistência das entregas.

A dependência de uma única pessoa para decisões críticas e o desalinhamento entre o PO e a equipe foram questões destacadas. Essas situações podem ser abordadas com práticas recomendadas no Scrum, como o *Sprint Planning*, que promove a colaboração entre o PO e os desenvolvedores para definição de metas claras e prioridades [Sutherland and Schwaber 2020]. Essa prática ajuda a estabelecer um entendimento comum, criando um ponto de partida organizado para as atividades do time.

Eventos como o *Daily Scrum* e a *Sprint Retrospective* também desempenham um papel importante no alinhamento e na melhoria contínua [Sutherland and Schwaber 2020]. Enquanto o *Daily Scrum* proporciona uma visão rápida do andamento do trabalho, a *Sprint Retrospective* permite ajustes mais profundos no processo. No contexto do desenvolvimento ágil, essas práticas podem favorecer a criação de um ambiente de trabalho mais sustentável, promovendo maior autonomia e comunicação eficiente entre os membros da equipe.

As diferenças entre os ambientes de desenvolvimento, homologação e produção foram identificadas como um desafio relevante, dificultando a detecção de falhas antes que elas impactem os usuários finais. Inconsistências nesses ambientes podem gerar comportamentos inesperados, frequentemente não detectados nas etapas iniciais do desenvolvimento. Humble e Farley (2010) destacam que a padronização dos ambientes contribui para entregas mais previsíveis e para a redução de problemas relacionados a discrepâncias [Humble and Farley 2010].

No contexto das ferramentas já utilizadas pela empresa, o *Azure Test Plans*, in-

tegrante da suíte do Azure, pode ser explorado para criar pipelines de validação automatizada. Essa solução, integrada ao ecossistema existente, auxilia na execução de testes padronizados em diferentes ambientes e fornece relatórios detalhados para identificar inconsistências [Microsoft 2024c]. A implementação de práticas de validação regular, como as aplicadas na entrega contínua (*Continuous Delivery*), pode favorecer maior estabilidade no processo de desenvolvimento ao integrar testes de forma consistente no fluxo de trabalho [Humble and Farley 2010].

Um dos desafios enfrentados pela equipe é a pressão por prazos apertados, que muitas vezes prejudica a qualidade das entregas. Esse problema é agravado pela falta de clareza sobre o esforço real necessário para concluir as tarefas e pela dificuldade em estimar prazos de forma precisa. Para lidar com essa questão, práticas como o *Planning Poker*, bastante comuns em metodologias ágeis, podem ajudar a criar estimativas mais confiáveis, envolvendo toda a equipe no processo de planejamento. Essa abordagem estimula a colaboração e o alinhamento entre os membros do time, reduzindo diferenças de expectativas e contribuindo para prazos mais realistas [Cohn 2006].

Adicionalmente, o *Azure DevOps*, também parte do conjunto de ferramentas já adotadas pela organização, permite acompanhar o progresso das tarefas em tempo real e identificar possíveis desvios no cronograma. Recursos como gráficos burndown e relatórios de desempenho ajudam a identificar problemas antecipadamente, permitindo ajustes antes que atrasos comprometam as entregas [Microsoft 2024b]. A integração contínua dessas ferramentas reforça a padronização dos processos e otimiza os fluxos de trabalho da squad.

Por fim, a documentação desatualizada e o preenchimento inconsistente do campo ”causa raiz”dificultam a análise de problemas recorrentes. A automatização desse processo por meio de templates padronizados pode organizar as informações de maneira mais eficiente, garantindo a consistência dos dados. Ferramentas como o *Miro*, por exemplo, têm sido utilizadas para facilitar a visualização e interpretação dos dados, promovendo maior alinhamento e colaboração entre os membros da equipe [Miro 2024]. Embora essa abordagem mostre grande potencial, é necessário avaliá-la cuidadosamente para entender seu impacto na eficiência e na precisão das análises realizadas pelo time. Práticas de automação e organização, como o uso de templates e ferramentas colaborativas, podem melhorar significativamente a produtividade e a qualidade do processo de desenvolvimento, criando um ambiente mais colaborativo e eficiente para a equipe [Cohn 2006].

6.1. Limitações

Este estudo apresenta algumas limitações que podem ter influenciado os resultados e as sugestões propostas. Um dos principais pontos é o número reduzido de participantes nas entrevistas, o que pode ter limitado a diversidade de opiniões e excluído perspectivas importantes, como as de integrantes externos ou de outras squads.

Outra limitação está nos dados quantitativos extraídos do Azure DevOps. Embora sejam fundamentais para a análise, sua precisão depende do preenchimento adequado pelos desenvolvedores, o que pode ter gerado inconsistências, especialmente na classificação das causas raízes. A ausência de padronização nesse preenchimento pode ter dificultado análises mais detalhadas.

Além disso, as propostas apresentadas foram elaboradas considerando as carac-

terísticas específicas da squad financeira estudada. Por isso, algumas ideias podem não ser totalmente aplicáveis em outros contextos, principalmente em equipes com dinâmicas e objetivos distintos.

Por fim, a análise qualitativa das respostas, embora detalhada, foi limitada pela quantidade de dados e pela amostra de falhas utilizadas, o que restringe a abrangência dos resultados. Pesquisas futuras podem expandir a coleta de informações, aumentar o número de participantes e adotar métodos complementares, como grupos focais ou observação direta, para oferecer uma visão mais abrangente e precisa.

Apesar dessas restrições, as melhorias propostas neste estudo representam um avanço relevante na compreensão das causas raízes das falhas em produção, servindo como referência para adaptações e soluções em outros cenários do desenvolvimento de software.

7. Considerações Finais

Este estudo destacou a importância da RCA no contexto do desenvolvimento de software, especialmente em squads financeiras, onde a confiabilidade e a eficiência são essenciais. Os métodos aplicados, como o Método dos Cinco Porquês e o Diagrama de Causa e Efeito, atenderam ao objetivo de identificar as causas subjacentes das falhas em produção, permitindo explorar os fatores críticos que contribuem para sua recorrência.

A partir dos dados coletados e analisados, foram propostas estratégias que vão ao encontro do objetivo de sugerir melhorias no processo de desenvolvimento de software. Entre essas estratégias, destacam-se a padronização de documentações, automação de testes, integração de pipelines e promoção de treinamentos técnicos. Essas propostas têm potencial para reduzir a recorrência de falhas e aprimorar a eficiência operacional, mesmo que ainda demandem validação prática no contexto estudado.

O estudo apresenta algumas limitações, como a dependência de dados do Azure DevOps, que podem conter inconsistências no preenchimento, e o número reduzido de participantes nas entrevistas, restringindo a diversidade de perspectivas. Além disso, as propostas foram elaboradas com foco na realidade da squad analisada, podendo necessitar de ajustes para aplicação em outros contextos.

Para trabalhos futuros, recomenda-se implementar as estratégias de forma gradual, permitindo ajustes contínuos e avaliação do impacto de cada medida. Essa abordagem facilita a adaptação às mudanças e aumenta as chances de sucesso. Além disso, ampliar a análise para outras squads e contextos organizacionais pode ajudar a identificar padrões comuns de falhas e desenvolver soluções adaptáveis a diferentes realidades.

Em resumo, este estudo atingiu os objetivos propostos ao identificar as causas raiz das falhas em produção e sugerir ações concretas para otimizar os processos de desenvolvimento. As reflexões apresentadas oferecem caminhos para a melhoria contínua, fomentando um ambiente mais eficiente e colaborativo para a entrega de software com maior qualidade e confiabilidade.

Referências

Amazon Web Services (2024). O que são análises de causas-raiz (rca)? Disponível em: https://aws.amazon.com/pt/what-is/root-cause-analysis/.

Acessado em: 18 nov. 2024.

Andersen, B. and Fagerhaug, T. (2006). *Root Cause Analysis: Simplified Tools and Techniques, Second Edition*. ASQ Quality Press, Milwaukee, Wisconsin.

Blascke, F. F. M., Farina, R. M., and Florian, F. (2023). Testes de software como garantia de qualidade e eficiência: estudo de caso com uso do selenium. *Revista Científica Multidisciplinar Núcleo do Conhecimento*, 3(9):26–55. Acessado em: 23 out. 2024. Disponível em: https://www.nucleodoconhecimento.com.br/engenharia-da-computacao/testes-de-software.

Cambridge Judge Business School (2013). Research by cambridge mbas for tech firm undo finds software bugs cost the industry $316 billion a year. Disponível em: https://www.jbs.cam.ac.uk/2013/research-by-cambridge-mbas-for-tech-firm-undo-finds-software-bugs-cos Acesso em: 23 out. 2024.

Catolino, G., Palomba, F., Zaidman, A., and Ferrucci, F. (2019). Not all bugs are the same: Understanding, characterizing, and classifying bug types. *Journal of Systems and Software*, 152:165–181.

Chapin, N., Hale, J., Khan, K., Fernandez-Ramil, J., and Tan, W.-G. (2001). Types of software evolution and software maintenance. *Journal of Software Maintenance*, 13:3–30.

Cohn, M. (2006). *Agile estimating and planning*. Prentice Hall, Upper Saddle River.

Cypress.io (2024). Cypress - next generation testing tool. Acessado em: 23 out. 2024. Disponível em: https://www.cypress.io.

Donato, L. (2024). Diagrama de ishikawa: o que é, para que serve e os 6ms. Criado em: 19 ago. 2021. Atualizado em: 27 fev. 2024. Acesso em: 22 out. 2024. Disponível em: https://blog.aevo.com.br/diagrama-de-ishikawa/.

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

Ishikawa, K. (1982). *Guide to Quality Control*. Asian Productivity Organization, Tokyo.

Jabbari, R., bin Ali, N., Petersen, K., and Tanveer, B. (2016). What is devops? a systematic mapping study on definitions and practices. In *Proceedings of XP2016 Workshops*, New York, NY, USA. Association for Computing Machinery.

Microsoft (2024a). Acessado em: 23 out. 2024. Disponível em: https://azure.microsoft.com/en-us/services/devops/.

Microsoft (2024b). Azure devops documentation. Acesso em: 22 nov. 2024. Disponível em: https://learn.microsoft.com/en-us/azure/devops/.

Microsoft (2024c). Azure test plans documentation. Acesso em: 22 nov. 2024. Disponível em: https://learn.microsoft.com/en-us/azure/devops/test/overview?view=azure-devops.

Miro (2024). Miro - collaborative whiteboard platform. Acessado em: 23 out. 2024. Disponível em: https://miro.com.

Pargaonkar, S. (2023). Defect management and root cause analysis: Pillars of excellence in software quality engineering. *International Journal of Science and Research (IJSR)*, 12:53–55.

Riaz, M. T., Jahan, M. S., Arif, K. S., and Butt, W. H. (2019). Risk assessment on software development using fishbone analysis. In *2019 International Conference on Data and Software Engineering (ICoDSE)*, pages 1–6.

Rooney, J. J. and Vanden Heuvel, L. N. (2004). Root cause analysis for beginners. *Quality Progress*, 37(7):45–56.

Suhanda, R. and Pratami, D. (2021). Raci matrix design for managing stakeholders in project case study of pt. xyz. *International Journal of Innovation in Enterprise System*, 5.

Sutherland, J. and Schwaber, K. (2020). The scrum guide: The definitive guide to scrum: The rules of the game. Disponível em: `https://scrumguides.org/`. Acessado em: 12 nov. 2024.