
Universidade Federal de Pernambuco
Centro de Informática

Bacharelado em Engenharia da Computação

# Pesquisa sobre práticas de CI e CD em Recife

Mateus Valgueiro Teixeira

Trabalho de Graduação

Recife
05 de maio de 2021

Universidade Federal de Pernambuco
Centro de Informática

Mateus Valgueiro Teixeira

# Pesquisa sobre práticas de CI e CD em Recife

*Trabalho apresentado ao Programa de Bacharelado em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.*

Orientador: *Paulo Henrique Monteiro Borba*
Co-orientadora: *Klissiomara Lopes Dias*

Recife
05 de maio de 2021

*DIGITE A DEDICATÓRIA AQUI*

# Agradecimentos

DIGITE OS AGRADECIMENTOS AQUI

*DIGITE AQUI A CITAÇÃO*
—AUTOR (NOTA)

# Resumo

[WIP - Retirado da proposta TG]

As práticas de integração, entrega e implantação continuas (*Continuous Integration, Continuous Delivery* e *Continuous Deployment*) já estão presentes no cotidiano de grandes empresas de todo o mundo. E isto é devido, entre outras coisas, aos comprovados benefícios que a utilização destas técnicas trazem para a equipe e para o produto em desenvolvimento. Mesmo assim, poucos são os estudos que pesquisam a respeito do estado da arte destas práticas nas empresas, principalmente no contexto da cidade de Recife.

Há também um grande equívoco em relação a nomenclatura das práticas e a real utilização das mesmas em vários aspectos no processo de desenvolvimento software. A utilização de termos e softwares voltados para práticas que não são seguidas de fato por muito projetos que os utilizam levam a um ambiente não saudável de desenvolvimento.

Neste contexto, o presente trabalho objetiva a realização de uma pesquisa quantitativa para entender como as práticas de integração, entrega e implantação continuas foram implantadas no contexto recifense, assim como avaliar o entendimento dos termos utilizados com os desenvolvedores locais.

**Palavras-chave:** DIGITE AS PALAVRAS-CHAVE AQUI

# Abstract

This is the abstract

**Keywords:** DIGITE AS PALAVRAS-CHAVE AQUI

# Sumário

# Lista de Figuras

# Lista de Tabelas

CAPÍTULO 1

# Introdução

* Falar um pouco sobre técnicas de CI/CD * Motivar o leitor * Metodologia * um pouco dos resultados * Restante da estrutura do trabalho

[WIP - Retirado da proposta de TG]

As práticas de integração, entrega e implantação continuas (*Continuous Integration, Continuous Delivery* e *Continuous Deployment*) já são muito difundidas e utilizadas por empresas de tecnologia em todo o mundo. Segundo [1], 95% dos participantes reportaram que sua organização pratica metodologias ágeis e 55% utiliza a técnica de integração continua. Também nesta mesma pesquisa foi encontrado que 41% utilizam a técnica de *Continuous Delivery*, e 36%, *Continuous Deployment*.

Estes números elevados são causados principalmente pelos benefícios que a utilização destas técnicas trazem para a equipe e para o produto em desenvolvimento. Estudos comprovam que os desenvolvedores envolvidos se sentem mais produtivos quando usam as práticas ágeis [6] e o seu uso traz maior qualidade ao software que está sendo produzido [10].

Ainda há, no entanto, uma grande falta de estudos a respeito de como essas práticas se exportaram para as grandes empresas de Recife [9], um dos grandes polos tecnológicos do Brasil. Somente no Porto Digital, localizado na capital pernambucana, há cerca de 330 empresas e 11 mil trabalhadores com faturamento anual de R$ 2,3 bilhões em 2019 [3].

Percebe-se também uma quantidade notável de indivíduos equivocados a respeito dos conceitos e do uso das práticas supracitadas [4]. O estudo [7] identificou que 60% dos projetos não utilizam a técnica de *commits* contínuos, mesmo utilizando uma ferramenta que tem como objetivo implantar a prática de integração contínua. Os autores chamam este fenômeno de *Continuous Integration Theather* (Teatro da integração contínua), e comentam que este processo produz um ambiente não saudável de desenvolvimento ágil.

Neste contexto, este trabalho tem como objetivo entender melhor, através de uma pesquisa quantitativa, como as práticas de integração, entrega e implantação continuas foram implantadas no âmbito das empresas de Recife, assim como avaliar se os termos utilizados são corretamente compreendidos pelos desenvolvedores destas, procurando assim possíveis "Teatros" recifenses.

## 1.1 Estrutura do Trabalho

O trabalho está divido em 6 capítulos, segmentados da seguinte forma:

- O Cápitulo 2 contém a Fundamentação Teórica deste trabalho, abordando sobre as práticas de CI/CD e definindo termos que serão utilizados durante o trabalho, além de alguns

trabalhos relacionados

- O Capitulo 3 trás a motivação por trás deste trabalho
- O Capítulo 4 contém a metodologia utilizada para a montagem e execução das entrevistas, além de todos os passos da análise de dados
- No Capítulo 5 encontra-se os resultados obtidos pela entrevista qualitativa
- O Capitulo 6 contém a conclusão com uma análise final e perspectivas de trabalhos futuros.

CAPÍTULO 2

# Fundamentação Teórica

Este capítulo tem como objetivo apresentar conceitos importantes para a construção desta pesquisa....

CAPÍTULO 3

# Motivação

Neste cápitulo será abordada a motivação por trás da pesquisa produzida. Na primeira seção, será falado um pouco sobre a grande adoção das práticas ágeis pela indústria. Já na segunda seção, será abordada a falta de estudos voltados para o contexto Recifense. A terceira discorre sobre o artigo base que serviu de motivação para este trabalho. A última seção contém as perguntas de pesquisa que este trabalho tenta responder.

## 3.1 A grande adoção de CI e CD

As práticas de integração e *deployment* contínuos (*Continuous Integration* e *Continuous Deployment*) já são muito difundidas e utilizadas por empresas de tecnologia em todo o mundo. Segundo [1], 95% dos participantes reportaram que sua organização pratica metodologias ágeis de desenvolvimento e 55% utilizam a técnica de integração continua. Também nesta mesma pesquisa foi encontrado que 36% utilizam a técnica de *deployment* contínuos.

Estes números elevados são causados principalmente pelos benefícios que a utilização destas técnicas trazem para a equipe e para o produto em desenvolvimento. Ainda de acordo com [1], entre as razões para adoção de CI/CD, as principais são aceleração de entregas de software (71%), e aumentar a produtividade (51%) e a qualidade do software (42%).

Especificamente sobre integração contínua, a prática hoje em dia já é bastante estudada difundida na indústria. O estudo [6] comenta que desenvolvedores envolvidos se sentem mais produtivos quando utilizam a prática e dão mais valor aos testes automáticos. Já com relação a *deployment* contínuo, o estudo [10] mostra que seu uso traz maior qualidade ao software que está sendo produzido.

## 3.2 A falta de estudos no contexto Recifense

Ainda há, no entanto, uma grande falta de estudos a respeito de como essas práticas se exportaram para a maioria das empresas [9], inclusive as de Recife, um dos grandes polos tecnológicos do Brasil. Somente no Porto Digital, localizado na capital pernambucana, há cerca de 330 empresas e 11 mil trabalhadores com faturamento anual de R$ 2,3 bilhões em 2019 [3].

O presente trabalho levantará as principais dores sentidas pelos desenvolvedores recifenses para que essas sejam pesquisadas com mais afinco para que sejam eliminadas. Um estudo a respeito de como as práticas de CI/CD migraram para a industria de Recife funciona ainda como um *benchmark* de como as empresas estão se portando em relação às novidades presentes na literatura nos últimos anos, assim como levanta possíveis discrepâncias entre empresas situadas

na área e as grandes corporações.

## 3.3 O artigo base

Com o objetivo de entender um pouco mais sobre o estado das práticas de CI/CD no contexto de Recife, nos baseamos no estudo [9], que busca entender como as práticas geralmente associadas a *Continuous Deployment* acharam o seu caminho na indústria européia e norte-americana. Para tal, foi utilizado um método misto de estudo empírico baseado em um pré-estudo na literatura, entrevistas qualitativas com 20 participantes e uma entrevista quantitativa que recebeu 187 respostas. A ideia era questionar até que ponto a sabedoria na area estava dominada por peculiaridades de um pequeno grupo de grandes empresas, como Facebook e Google. Saber mais sobre o problema que outras empresas não gigantes sofrem pode gerar mais estudos a respeito de como solucionar tais problemas, assim como mostrar possíveis oportunidades de melhoria e revisão dos processos utilizados.

Durante o trabalho os autores definem a *stairway to heaven* (escada para o céu, em tradução livre), presente na Figura 3.1. Ela tem como objetivo definir um caminho de evolução das empresas para um estágio de entregas sofisticado. A escada permeia práticas de integração contínua, *deployment* contínuo e entregas parciais.

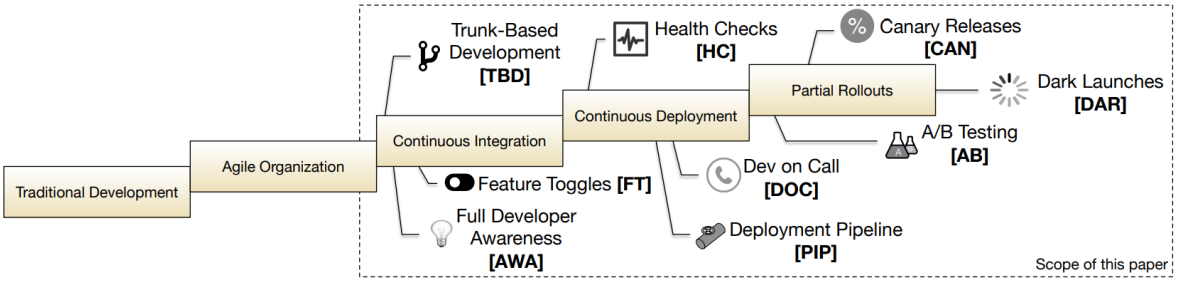


**Figura 3.1** A escada de evolução denominada "Stairway to Heaven"proposta pelo artigo base. Fonte: Schermman et al (2016)

Através desta metodologia os autores descobriram que, no contexto estudado, problemas arquiteturais são geralmente uma das maiores barreiras para a adoção de CD. Não obstante, a técnica de *Feature Toggles* como forma de realizar entregas parciais adiciona uma complexidade demasiada e não saudável ao código. Por fim eles concluem que os desenvolvedores necessitam também de um protocolo baseada em princípios que diga quando deve-se utilizar técnicas de entregas parciais, por exemplo, que funcionalidades e métricas devem ser testadas por um teste A/B.

Um trabalho que utilizasse a mesma metodologia aplicada a uma amostra de um outro contexto pode revelar discrepâncias e semelhanças entre os dois ambientes de estudo. As discrepâncias levantariam pontos de questionamento, pesquisa e até possíveis melhorias aos ambientes envolvidos. Já as semelhanças podem assegurar que as práticas utilizadas já acharam o

seu lugar na indústria e funcionam bem assim como a teoria propunha.

## 3.4 Perguntas de pesquisa

Assim, a grande adoção da indústria mundial das práticas de CI/CD, aliado a pequena quantidade de trabalhos a respeito no contexto recifense e a excelente metodologia proposta pelo artigo de Schermman et al [9] foram as principais motivações para o desenvolvimento deste trabalho. Para este as seguintes perguntas de pesquisa foram formadas:

1. Quais as práticas de CI/CD são utilizadas pelas empresas em Recife?
2. O cenário de CI/CD nas empresas em Recife segue o *stairway to heaven* proposto no artigo?
3. Quais são os princípios e práticas subjacentes que governam a adoção de CD na indústria?

Para responder as perguntas acima foi decido reaplicar a pesquisa qualitativa do artigo [9] com desenvolvedores recifenses, baseando-se também na mesma lista de definições de cada uma das práticas envolvidas no "stairway to heaven". A pesquisa qualitativa neste contexto foi considerada fundamental para garantir que os entrevistados entendessem claramente as perguntas e excluir possíveis entendimentos errados de termos em inglês, linguagem não nativa de todos os participantes. Vale salientar que não foi replicado a pesquisa quantitativa presente no artigo base, mas esta pode servir como um trabalho futuro a este.

Além disso, as perguntas 1 e 3 são bem parecidas com as perguntas de pesquisa do artigo base, mas incluindo a técnica de *Continuous Integration*. Não obstante, uma terceira pergunta foi adicionada (*RQ2*), que foca na escada de evolução proposta pelo artigo. Ela tenta responder se a *stairway to heaven* é seguida no contexto de Recife, visto que, no outro, os próprios autores já refutaram esta definição com os seus resultados.

CAPÍTULO 4

# Metodologia

Para o estudo, foi realizada uma pesquisa qualitativa através de entrevistas semi-estruturadas utilizando como base a entrevista desenvolvida pelo artigo base [9]. A primeira seção abordará uma visão geral sobre os primeiros passos da pesquisa. Já a segunda seção discorre sobre como as entrevistas foram conduzidas. Na terceira, encontra-se todo o processo utilizado para a análise dos dados obtidos na entrevista. Por último, a quarta seção mostra informações a respeito dos participantes da pesquisa.

## 4.1 Pré-estudo

Com o objetivo de entender melhor sobre como as práticas de CI/CD migraram para as empresas sediadas em Recife, Mateus Valgueiro (autor deste trabalho) e Heitor Samuel realizaram um pré-estudo. Primeiramente, os dois discutiram a respeito do artigo base: *An empirical study on principles and practices of continuous delivery and deployment* [9]. Na discussão foi definido que seria replicado apenas o estudo baseado em entrevistas semi-estruturadas para garantir o entendimento dos termos pelos entrevistados e a adequação com as definições do artigo base.

Após a discussão, os entrevistadores acessaram o apêndice online deste [2] para obter o guia de entrevista utilizado. Como forma de manter a consistência, foi utilizado o mesmo guia de entrevistas, assim como as mesmas definições utilizadas pelos autores. No início foi montado um arquivo com a definição de cada um dos termos envolvidos em língua portuguesa para ser utilizado como consulta caso haja alguma dúvida de definição de tópicos entre os entrevistadores.

Após isso, o guia de entrevistas do estudo base [2] também foi traduzido para português e utilizado durante as entrevistas. Um ponto importante a respeito da tradução é o fato de que alguns termos ainda foram mantidos na língua inglesa devido ao fato de serem conhecidos mundialmente nesta língua. Por exemplo, Continuous Integration, Canary Releases e Health check. O documento de consulta e a guia de entrevistas estão presentes no apêndice deste trabalho, ambos em sua versão traduzida.

## 4.2 Estrutura da Entrevista

O guia de entrevistas se baseia na estratégia de evitar perguntas diretas (exemplo: “determinada prática está sendo utilizada?”). Esse modelo é essencial para garantir que a comparação entre participantes a respeito do uso ou não de práticas está sendo feita de maneira concisa, e não baseada nos conhecimentos prévios de cada participante. Assim, através de perguntas sobre

o processo utilizado pelo entrevistado é possível, com uma certa margem de erro associada, afirmar que práticas ele utiliza. A margem de erro surge do fato de o autor ter que refletir sobre as informações recebidas e as definições para inferir o uso ou não de certa prática.

A entrevista é dividida em 5 sessões:

1. Processo de entrega no geral
2. Papéis/Responsabilidades
3. Garantia da Qualidade (Quality Assurance)
4. Gerenciamentos de Problemas
5. Avaliação de Entrega

No guia, todas as sessões iniciam com uma questão aberta. As entrevistas seguiram os tópicos abordados em cada uma delas, mas sem ordem específica, respeitando o desenrolar da conversa. A primeira entrevista foi guiada pelos dois pesquisadores para assegurar que as próximas seriam feitas de forma semelhante pelos dois. Esta entrevista foi considerada como válida na análise de dados visto que não houve nenhuma mudança no questionário de entrevistas; o próprio já tinha sido validado no artigo utilizado de base para este trabalho. As outras 10 - totalizando 11 entrevistas feitas - foram guiadas por apenas um entrevistador: 5 conduzidas por Heitor e outras 5 conduzidas por Mateus.

Todas as entrevistas ocorreram de forma online, através do Google Meet, e aconteceram entre os meses de Setembro e Outubro de 2020 em português brasileiro. As entrevistas levaram entre 30 e 50 minutos, duração bem parecida com os tempos obtidos no artigo base (35 a 60 minutos). Cada uma delas foi gravada pela plataforma para futura análise e 4 delas foram transcritas para o uso em citações neste trabalho, escolhidas através da relevância da entrevista e da forma como certos termos e processos foram apresentados pelo entrevistado. Foi deixado claro em cada uma das conversas a respeito da gravação e que estas seriam utilizadas apenas pelos entrevistadores, respeitando assim o anonimato de informações pessoais como nome ou empresa da qual o profissional se referia.

## 4.3 Análise dos Dados

A Figura 4.1 apresenta uma visão geral de como funcionou o processo de coleta e análise de dados. A análise foi feita apenas pelo autor deste artigo, Mateus Valgueiro. O processo escolhido foi baseado nas fases de *Coding* e *Thematic Analysis* da metodologia *Grounded Theory* [8]. No processo de Coding a ideia é levantar rótulos ou tags relevantes para o texto e, tradicionalmente, é feito baseado na transcrição das entrevistas. No entanto, neste trabalho o autor gerou códigos através da escuta das entrevistas. Então, como um exemplo, a seguinte citação:

> *"Como somos uma equipe muito pequena, todos os desenvolvedores são meio que DEVOPS. Quando tem que tomar alguma decisão nós entramos em discussão e definimos por nós."*
>
> —P5

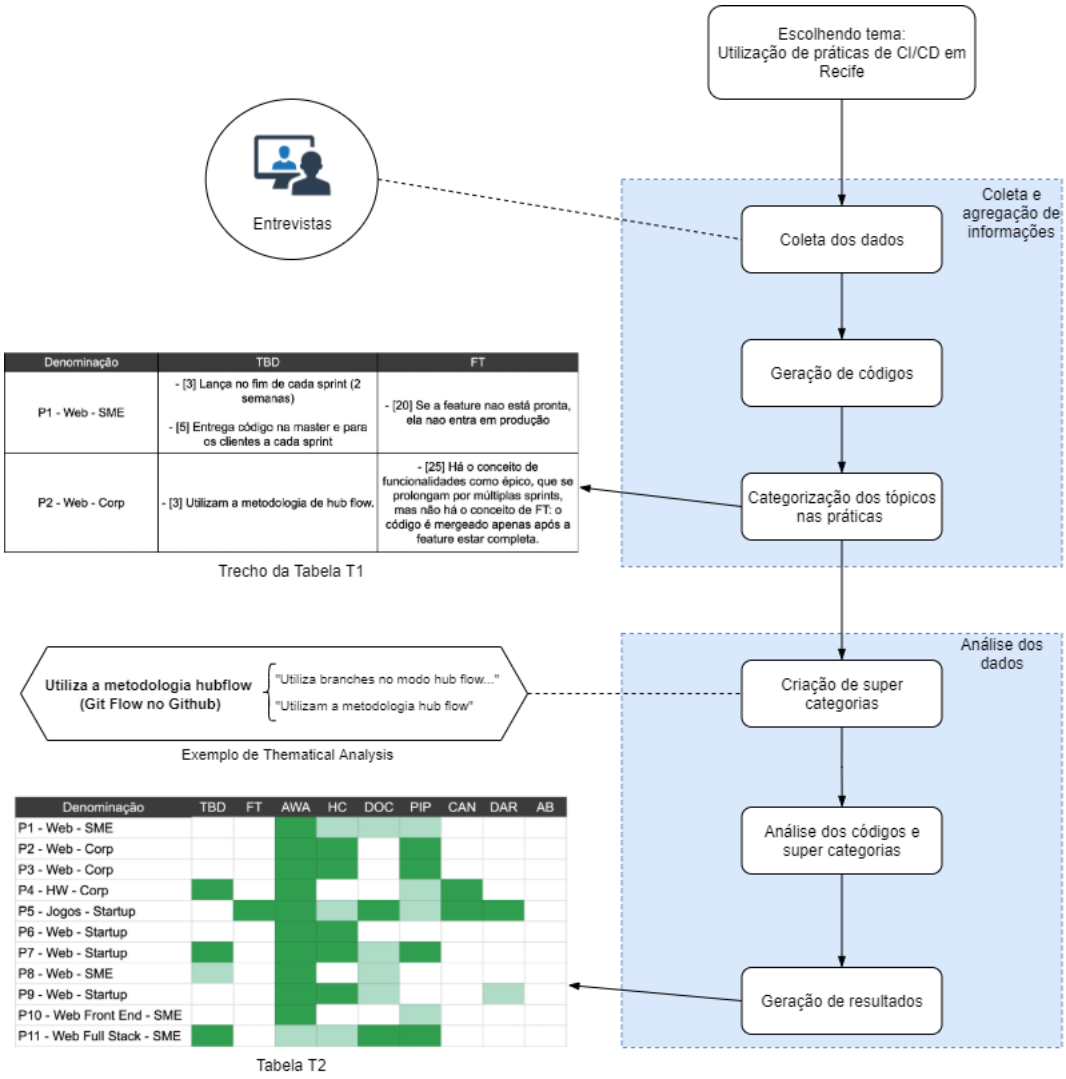


**Figura 4.1** Demonstra como funcionou o processo de coleta e análise de dados.

Gerou o código: *"Todos os desenvolvedores são devops."* - P5_15

É importante salientar que os códigos gerados sofreram um certo enviesamento visto que este trabalho é uma replicação de um estudo, então o autor tinha em mente que assuntos estavam sendo procuradas na fala durante o levantamento de códigos.

Então, após levantados todos os códigos de uma entrevista, estes foram revisados para garantir semântica e sintaxe adequadas. Alguns códigos nessa fase foram eliminados por redundância, enquanto outros foram quebrados em múltiplos. Depois, eles foram agrupados, quando compatíveis, em cada uma das 9 práticas descritas pelo artigo base e foi escolhida uma nota entre 0 e 2, representando não utiliza, utiliza parcialmente e utiliza completamente baseado nas definições do artigo base traduzidas. Este processo foi então replicado para cada uma das 11 entrevistas.

Ao final do processo ainda haviam 3 ligações entre a prática de *Dark Launch* e entrevistados que não tinham recebido nenhum código em comum. Para estes casos, foi enviado um email diretamente para cada um dos entrevistados com a definição do artigo base da técnica e foi perguntado se o entrevistado utilizava ou não a mesma, o que gerou mais 3 códigos. Ao final, 292 códigos surgiram ao todo.

O agrupamento entre códigos e práticas gerou a Tabela T1, que mostra uma visão geral de cada relação entre prática e entrevistado, contendo a nota dada e os códigos que justificam a nota. Esta tabela contém 147 códigos.

De posse da tabela T1 foi gerada a Tabela T2, que contem uma visão reduzida da primeira, contendo apenas as notas de cada uma das práticas para cada um dos entrevistados com as colunas ordenadas de acordo com o *Stairway to Heaven* descrito no artigo base. Com a Tabela T2, também foi criada uma nova Tabela T3 com o mesmo formato da primeira, mas com as colunas ordenadas pela utilização de cada prática em ordem decrescente. Esta tem como objetivo replicar a Tabela 1 do artigo base [9].

Por fim, para identificar padrões e abstrações nos códigos agrupados na Tabela T1, foi feito um trabalho de agrupamento semântico, gerando por fim a tabela T4 apresentando as novas super categorias geradas no processo de *Thematic Analysis*. Neste, 17 super categorias foram geradas.

É importante salientar que ainda durante o processo de levantamento de códigos surgiram tópicos relevantes que não se relacionavam diretamente com as práticas descritas, mas que são perguntados pelo questionário e relevantes para o tema tratado. Surgiram então 2 novas colunas que agregavam códigos sobre as práticas de *Code Review* e de testes automáticos. Para estas foram relacionados 29 códigos, e duas super categorias foram geradas.

## 4.4 Participantes

No total foram entrevistados 11 desenvolvedores (P1 a P11) de 7 empresas diferentes sediadas em Recife. Destes, 3 eram mulheres. Pode-se ver a distribuição dos participantes entre gênero na Figura 4.2. Dentro desse grupo, 9 trabalhavam com aplicações Web, enquanto 1 trabalhava com sistemas embarcados e o último, com jogos.

A escolha dos entrevistados foi feita baseado na rede de conhecidos dos entrevistadores, com o propósito de agregar pessoas de que trabalhavam em empresas de tamanhos distintos para garantir uma variedade de parâmetros envolvidos. Como é possível perceber na Figura 4.3, com relação a esse aspecto a amostra está bem distribuída.

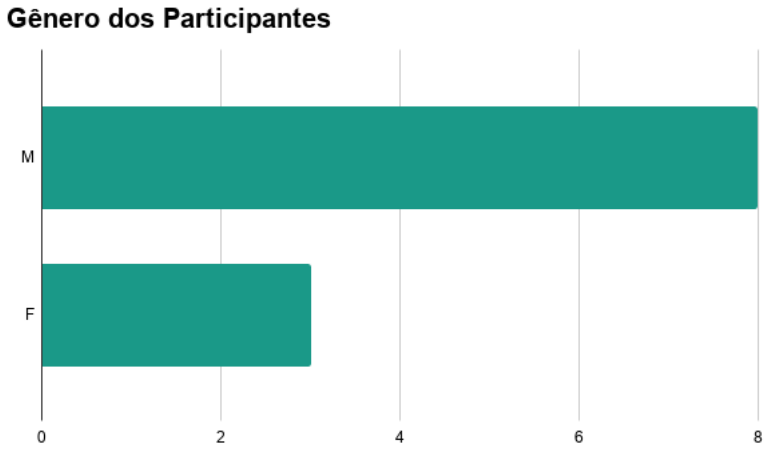


**Figura 4.2** Distribuição dos Participantes por gênero

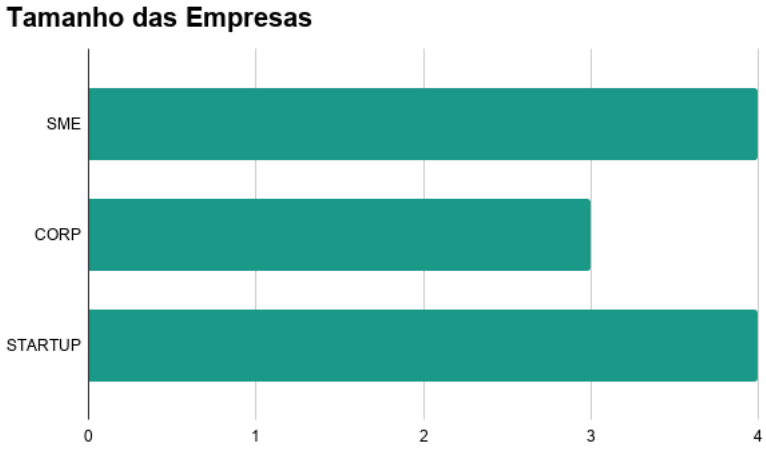


**Figura 4.3** Distribuição dos Participantes por tamanho da empresa.

CAPÍTULO 5

# Resultados

Neste capítulo serão apresentados os resultados encontrados com as entrevistas baseado na análises de dados apresentada no capítulo anterior. Primeiramente será focado o estudo do predomínio das práticas em Recife, seguido da análise de coerência do stairway to heaven. Por fim, serão apresentados princípios e práticas subjacentes que governam a adoção de CI/CD na amostra.

## 5.1 Estudo de Predomínio das Práticas

Com o intuito de responder a pergunta de pesquisa RQ1 - *Quais as práticas de CI/CD são utilizadas pelas empresas em Recife?* - foi montada a tabela T3, presente na Figura 5.1, demonstrando a utilização de cada uma das práticas definidas para cada um dos participantes. Nesta, é demonstrado através da coloração da célula o grau de utilização de determinada prática por determinado participante. Assim, o verde mais escuro significa que o participante utiliza totalmente, enquanto o verde claro significa utilização parcial e, por fim, o branco denota a não utilização. A ideia desta tabela é replicar a Tabela 1 do artigo base, presente na Figura 5.2.

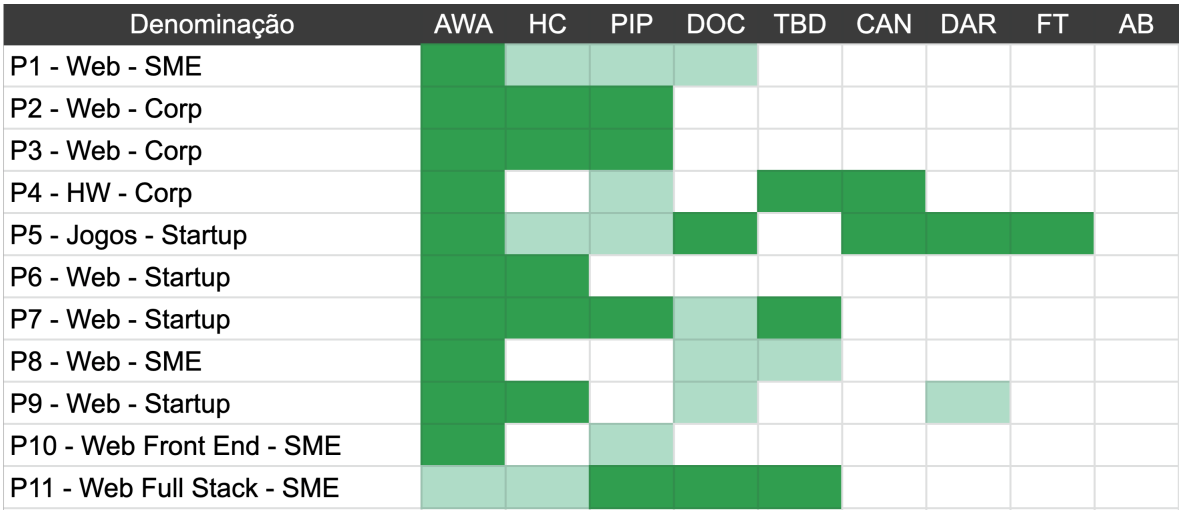


**Figura 5.1** Tabela T3 com o nível de utilização de cada uma das práticas, com as colunas ordenadas em ordem decrescente de uso.

Com a tabela T3 (Figura 5.1) é possível perceber que *Developer awareness [AWA]* foi a

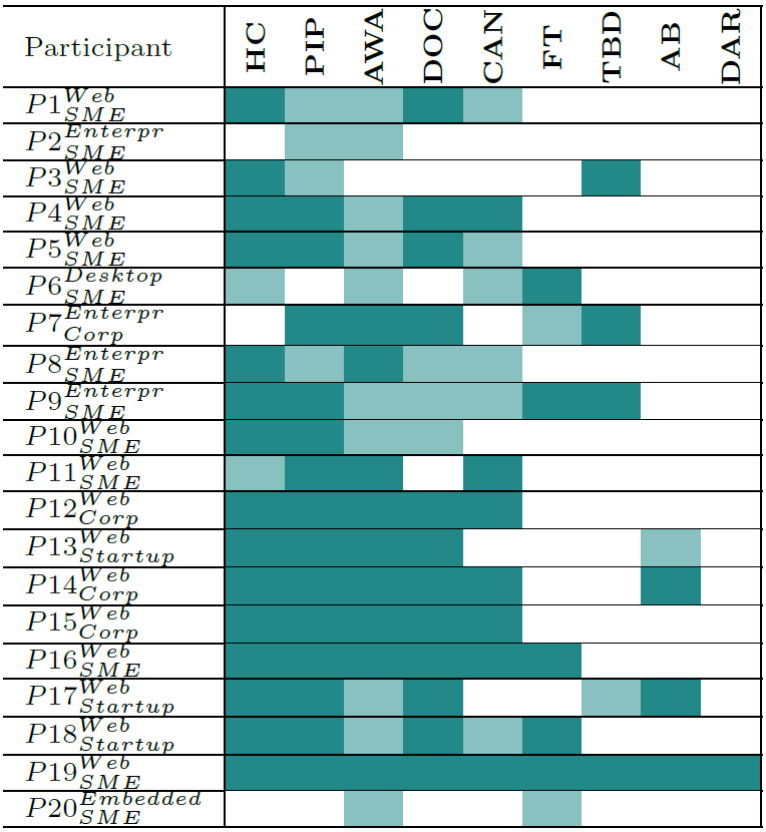


**Figura 5.2** Utilização das práticas pelos participantes do artigo base. Fonte: Schermman et al (2016)

prática mais encontrada em toda a amostra, sendo, no mínimo, parcialmente utilizada por todos os entrevistados. Logo após, pode-se encontrar as práticas de *Deployment Pipeline [PIP]* e *Health Check [HC]* na segunda e na terceira colocação, respectivamente. No geral, também pode-se inferir que as técnicas de *Partial Rollouts* são ainda muito pouco utilizadas, com as 3 práticas do grupo entre as quatro últimas colocadas. Vale a pena citar que nenhum dos entrevistados utiliza, mesmo que precariamente, a técnica de Testes A/B [AB].

Quando comparamos a Tabela T3 (Figura 5.1) com a tabela 1 do artigo base (Figura 5.2), podemos perceber que há diferenças de posição entre práticas, mas que não há mudanças exorbitantes. Com a ajuda da Figura 5.3, é possível perceber que há uma diferença de no máximo 2 posições na ordem de predomínio, ao comparar os resultados obtidos neste trabalho com os do artigo base.

## 5.2 Stairway to heaven

Com o objetivo de responder a pergunta de pesquisa RQ2 - *O cenário de CI/CD nas empresas em Recife segue o "stairway to heaven"proposto no artigo?* - foi produzida a Tabela T2, presente na Figura 5.4. Esta contém a visualização de utilização das práticas, utilizando a mesma

| Prática/Trabalho | Contexto Recifense | Artigo Base | Diferença de Posição |
|---|---|---|---|
| AWA | 1 | 3 | -2 |
| HC | 2 | 1 | 1 |
| PIP | 3 | 2 | 1 |
| DOC | 4 | 4 | 0 |
| TBD | 5 | 7 | -2 |
| CAN | 6 | 5 | 1 |
| DAR | 7 | 9 | -2 |
| FT | 8 | 6 | 2 |
| AB | 9 | 8 | 1 |

**Figura 5.3** Diferença entre o artigo base e este trabalho a respeito da ordem de predomínio das práticas

técnica de cores aplicada na Tabela T3 (Figura 5.1) e explicada na seção anterior, mas com as colunas ordenadas pela escada definida no *Stairway to Heaven*.

| Denominação | TBD | FT | AWA | HC | DOC | PIP | CAN | DAR | AB |
|---|---|---|---|---|---|---|---|---|---|
| P1 - Web - SME | | | | | | | | | |
| P2 - Web - Corp | | | | | | | | | |
| P3 - Web - Corp | | | | | | | | | |
| P4 - HW - Corp | | | | | | | | | |
| P5 - Jogos - Startup | | | | | | | | | |
| P6 - Web - Startup | | | | | | | | | |
| P7 - Web - Startup | | | | | | | | | |
| P8 - Web - SME | | | | | | | | | |
| P9 - Web - Startup | | | | | | | | | |
| P10 - Web Front End - SME | | | | | | | | | |
| P11 - Web Full Stack - SME | | | | | | | | | |

**Figura 5.4** Tabela T2 com o nível de utilização de cada uma das práticas, com as colunas ordenadas na ordem do *Stairway to Heaven*.

Com a Tabela T2 (Figura 5.4) é possível perceber que a amostra deste estudo também não segue a evolução proposta pelos autores do artigo base [9]. Isso fica claro quando percebe-se que não há, em nenhum dos entrevistados, uma relação clara entre a coloração da coluna com a sua anterior. É possível notar também que há um vão nas duas primeiras práticas, seguido de uma grande utilização da terceira, confirmando a tese de que a *Stairway to Heaven* não é coerente na amostra. É interessante destacar que a amostra do próprio artigo base também tem a mesma falha de coerência.

## 5.3 Analise das Práticas

Nesta seção será abordado, para cada uma das práticas, as principais curiosidades encontradas na amostra. Esta tem como objetivo responder a pergunta RQ3: *Quais são os princípios e práticas subjacentes que governam a adoção de CD na indústria?*. As seguintes subseções agrupam as práticas marginais através das práticas gerais definidas no *Stairway to Heaven*, inclusive seguindo a ordem deste.

Para poder definir padrões encontrados na amostra o autor, após a fase de agrupamento semântico, juntou os todas as categorias e super categorias ligadas a cada uma das práticas baseando-se principalmente na nota de utilização. Com a leitura e reflexão deste conjunto, o autor procurou por princípios que governam a adoção ou não de de cada prática. Foi feito também um estudo comparativo com os resultados obtidos pelo artigo base [9] para analisar discrepâncias e congruências entre os dois contextos de estudo.

### 5.3.1 Integração Contínua

Com a Figura 5.1 é possível perceber que, apesar da grande disseminação a respeito de informações relacionadas a infraestrutura e manutenção de software - ligadas à prática de *Developer Awareness* - estar em primeiro lugar, as outras técnicas que compõem o conjunto de *Continuous Integration* ainda não estão disseminadas na nossa amostra.

Nesta subseção, serão abordadas a seguir cada uma das práticas ligadas ao processo de Integração contínua: *Trunk Based Development* [TBD], *Feature Toggles* [FT] e *Developer Awareness* [AWA].

#### 5.3.1.1 Trunk Based Development [TBD]

Na amostra é possível perceber que a técnica de *Trunk Based Development* [TBD] não é tão amplamente adotado, visto que apenas 4 entrevistados utilizam ao menos parcialmente. Destes, 2 comentam que entregam novas versões aos clientes baseados em novas funcionalidades, e não em *sprints*. Já entre os que não utilizam, 5 integram o código apenas no final da *sprint*. Deste grupo, 2 utilizam a metodologia *Git Flow* [5], que define uma maneira de manusear várias branches ao mesmo tempo de modo que os desenvolvedores deparem com o mínimo de conflitos possível e que software seja entregue em versões bem definidas.

#### 5.3.1.2 Feature Toggles [FT]

No estudo foi possível perceber que, na amostra, a prática de *Feature Toggles* [FT] é raramente utilizada, assim como no artigo base [9]. Esta técnica foi encontrada apenas na equipe do participante P5, que a utilizava para esconder funcionalidades enquanto testes manuais ainda estavam sendo feitos. É interessante notar que este participante também foi um dos poucos que utilizava a técnica de *Canary Releases* [CAN].

Entre o grupo dos que não utilizavam, 7 só enviam código novo para produção quando a funcionalidade está concluída. Deste grupo, 2 comentaram que fazem uso de conceito de épicos, onde uma história de usuário se prolonga por mais do que apenas uma sprint. É também

interessante notar que P3 está em vias de utilizar esta técnica para reduzir conflitos de *merge* e *rebase* devido ao grande número de desenvolvedores em seu time, como podemos ver na citação:

> *"O meu time tem 23 [pessoas]. [...] a gente tá trabalhando em funcionalidades muito distintas, então às vezes acontece de termos um paralelismo de branches muito grande. [...] é muito complicado 'mergear' e fazer rebase de tudo. Realmente dá muitos conflitos"*
>
> —P3 - WEB - CORP

#### 5.3.1.3 Developer Awareness [AWA]

Na amostra é possível identificar que a prática de *Developer Awareness* [AWA] é amplamente adotado nas companhias, assim como na amostra do artigo base [9]. Em 6 entrevistas foi possível perceber que o time de desenvolvimento era o mesmo responsável pela entrega e manutenção da aplicação. Jà outras 2, há um time específico de *DevOps*, mas não havia grandes silos entre este e a equipe de desenvolvimento. Um caso interessante foi o de P11, que, apesar de um conhecimento espalhado dentro da equipe, há receio e insegurança por parte de alguns a respeito de questões de infraestrutura no geral.

### 5.3.2 Deployment Contínuo

Nesta subseção, serão abordadas as práticas ligadas ao processo de *Deployment* contínuo: *Health Checks* [HC], *Developer on Call* [DOC] e *Deployment Pipeline* [PIP]. É possível inferir, baseado principalmente na Figura 5.1, que as técnicas ligadas ao processo estão presentes na maioria das equipes da amostra: as 3 estão nas quatro primeiras colocações.

#### 5.3.2.1 Health Checks [HC]

Sobre a prática de *Health Checks* [HC], é possível perceber que, apesar de não ser o mais adotado - como foi no artigo base, ainda tem destaque entre as outras, presente na segunda colocação. Na amostra, 7 entrevistados continham pelo menos uma forma rudimentar de checagem e alertas, e destes, 2 continham apenas verificações não muito complexas. Um ponto interessante que surgiu foi o fato de P4 achar que não era necessário utilizar esta técnica por estar em fase de prototipação.

#### 5.3.2.2 Developer on Call [DOC]

Na amostra é possível perceber que a técnica de *Developer on Call* [DOC] é mais adotada de forma implícita do que de fato definida - 3 entrevistados estão em times que funcionam desta forma. Contudo, 5 pessoas do grupo não utilizam esta prática: alguns comentaram que a confiança nos testes automáticos faz com que não utilizem a prática, enquanto outro comentou que a prática é inclusive mal vista pela empresa.

Uma dicotomia interessante foi encontrada entre os resultados da amostra deste trabalho e o do artigo base [9]. Este último comenta que a prática já esta sendo largamente aceita nas organizações atualmente, e inclusive um dos entrevistados comenta que essa responsabilidade de ficar até mais tarde para resolver problemas leva os desenvolvedores a escrever e testar seus códigos mais veemente. Tal argumentação tem como base a seguinte citação retirada do artigo e traduzida:

> *"Se você não tem testes suficientes e faz deploy de um código ruim isso vai se voltar contra você pois você estará de plantão e terá que dar suporte a isto."*
>
> —P14 (DO ARTIGO BASE) - WEB - CORP

Contudo, com a seguinte citação de P2, é possível perceber que há um sentimento contrário: confia-se no processo de qualidade e, por isso, não necessitam de plantão.

> *"Temos o ciclo de QA, se encontrar alguma coisa a gente vê [. . . ] não precisa isso de plantão não."*
>
> —P2 - WEB - CORP

#### 5.3.2.3 Deployment Pipeline [PIP]

Em relação à prática de *Deployment Pipeline* [PIP] é possível inferir que ela é amplamente adotada pela amostra, visto que apenas 2 entrevistados obtiveram nota 0 (não utiliza). No artigo base [9] é possível perceber um resultado semelhante a este. Uma grande parte dos entrevistados segue o mesmo padrão para qualquer tamanho da mudança. Outros 2 tinham alguns processos automatizados, mas a *pipeline* era diferente dependendo do tamanho da mudança.

É legal perceber ainda que alguns times ainda pecam na falta de automação dos processos da *pipeline*. Isto acontece em decorrência da complexidade da automação, devido às tecnologias utilizadas, ou da falta de prioridade do time para tal.

### 5.3.3 Entregas Parciais

Nesta subseção, serão abordadas as práticas ligadas ao processo de Entregas Parciais: *Canary Releases* [CAN], *Dark Launches* [DAR] e Testes A/B [AB]. Na amostra foi possível perceber que todas as técnicas são muito pouco utilizadas: todas estão entre as 4 últimas colocações. É valido também notar que as 3 mantiveram a mesma ordem de utilização do processo do artigo base [9].

#### 5.3.3.1 Canary Releases [CAN]

A técnica de *Canary Releases* [CAN] apareceu nos contextos de jogos e no de sistemas embarcados. Na equipe do entrevistado P5, que trabalha no domínio de jogos eletrônicos, os

principais jogadores - conhecidos como "baleias- são escolhidos para participar de um *early access* de novas funcionalidades. Esses testes levam em torno de 1 semana.

Já na equipe de P4, que trabalha com sistemas embarcados, a prática era necessária devido ao contexto de atuação e às tecnologias utilizadas. Como o sistema que está em fase de prototipação servirá para o contexto médico, vários testes de campo deveriam ser feitos para garantir que todas as funcionalidades estivessem de acordo com o esperado. Para os testes, o cliente que contratou a empresa de P4 escolhia a quantidade de pessoas e local que serviria como validação de funcionalidades.

Entre os entrevistados qe não utilizam a prática de *Canary Releases*, 3 têm um ambiente de homologação para testes e validação de requisitos, mas este utiliza dados diferentes dos de produção. Outros 3 comentam que as features são sempre entregues para todos os usuários ao mesmo tempo.

#### 5.3.3.2 Dark Launches [DAR]

Do grupo das práticas associadas a Entregas Parciais, *Dark Launches* [DAR] foi a segunda mais utilizada, mas a menos conhecida entre os entrevistados. Isto se confirma com o fato de que 6 entrevistados disseram nunca ter utilizado, e 3 destes disseram especificamente que não conheciam a técnica. Importante notar que no estudo base [9] esta é a menos utilizada do grupo de práticas.

*Dark Launches* foi utilizado totalmente por P5 no contexto de jogos para testes manuais, e apenas parcialmente no contexto de WEB por P9 para validações de alguns cenários que dependiam de dados de produção.

> *"A gente implementa a funcionalidade mas condiciona a não aparecer para o usuário até que a gente queira."*
>
> —P5 - JOGOS - STARTUP

#### 5.3.3.3 Testes A/B [AB]

A prática de Testes A/B [AB] não foi identificada em nenhum dos participantes. Entre as principais causas levantadas para tal, 4 entrevistados comentaram que não utilizam pela baixa quantidade de usuários ativos no sistema. Outros 2 comentaram que a técnica não se aplicava ao contexto da aplicação. É interessante notar que esses dois motivos estão presentes no artigo base [9], contudo, ao contrário deste, ninguém na amostra comentou sobre problemas na arquitetura como causa para não utilização.

> *"O sistema da gente - apesar de lidar com uma massa de dados muito grande - não têm tantos usuários, então não faz muito sentido [utilizar testes A/B]...."*
>
> —P2 - WEB - CORP

Outro motivo importante foi levantado por P8, que comenta que aparentemente não existe

na empresa o interesse em investir nesta prática. P3 comenta ainda que o time está mais focado em entregar novas funcionalidades, pois a demanda por parte do cliente está muito grande.

## 5.4 Descobertas adicionais

Esta subseção abordará sobre os dois tópicos marginais que foram levantados durante a análise dos dados obtidos pelas entrevistas: *Code Review* e testes automáticos.

### 5.4.1 Code Review

A prática de *Code Review* é bem vista pela maioria dos entrevistados, considerado uma técnica importante e essencial para o controle de qualidade de código. As razões para o uso são diversas, desde de impor boas práticas - por P7 - até o compartilhamento e nivelamento do conhecimento - por P11.

> *"... a partir do momento que o sistema vai crescendo, o próprio desenvolvedor não tem a noção de que aquela sua mudança não é a melhor forma de fazer e que pode quebrar outras partes do sistema..."*
>
> —P2 - WEB - CORP

Ainda na amostra, 3 entrevistados acreditam que a prática funciona principalmente para troca de conhecimento. Sobre isso, a entrevistada P10 comenta que agrega muito valor para ela como novata no time, mas acha que adiciona um tempo desnecessário na entrega de funcionalidades por pessoas mais experiente, visto que - para ela - a técnica não faria sentido neste caso.

Outro ponto importante foi a distinção entre dois relatos a respeito do engajamento do time com a prática. Enquanto P3 comenta que a equipe dela é bem aberta ao debate e está engajada com o processo, P7 fala que o processo funciona "mais ou menos", dependendo do humor dos revisores.

### 5.4.2 Testes Automáticos

Testes automáticos são, no geral, bem vistos e extremamente recomendados pela maioria dos entrevistados como forma de prevenir erros em tempo de execução. Contudo, o participante P2 comenta que ainda é contrário a depender somente deles como garantia de qualidade.

> *"... automação [de testes] não resolve todos os problemas [relacionados a garantia de qualidade], ele vai identificar muita coisa, mas tem várias outras que precisamos do olhar de um testador..."*
>
> —P2 - WEB - CORP

Na amostra, 5 entrevistados acreditam que o aumento da cobertura de testes automáticos pode diminuir a quantidade de bugs em produção. Dentro desse grupo, o participante P9 comenta que isto poderia, no entanto, diminuir a velocidade de entregas do time. P5 adicionou

ainda que sente que a *sprint* é muito curta e não consegue tempo dentro destas para adicionar testes automáticos.

CAPÍTULO 6

# Conclusão

Com os resultados encontrados, é possível perceber que, apesar de algumas poucas diferenças a amostra deste trabalho se comportou de forma condizente àquela encontrada no artigo base [9]. Desde de o ranking de utilização de cada prática até as barreiras enfrentadas pelos desenvolvedores, houve uma congruência razoável entre os dois resultados.

Sobre o processo de integração contínua, pode-se inferir que as técnicas ligadas ao *merge* contínuo estão pouco presentes, apesar de *Developer Awareness* ser a prática mais utilizada da amostra. A maioria entrevistada integra o código na *branch* principal apenas no final da *sprint*.

É possível perceber ainda que todas as práticas ligadas à *Deployment* contínuo estão muito presentes na amostra. Mesmo com a prática de *Developer on Call* sendo mais utilizada de forma implícita do que de fato definida, e com algumas empresas interpretando-a como uma má prática, ela está em 4º lugar no ranking de utilização montado e é a de menor colocação do grupo.

Por fim, foi possível perceber que a amostra utiliza muito pouco as práticas de entregas parciais. Foi possível identificar técnicas desconhecidas por um grupo razoável de entrevistados - *Dark Launches* - e até técnicas que não foram utilizados por nenhum dos participantes - Testes A/B.

## 6.1 Ameaças a validade

Mesmo com a metodologia definida e seguida utilizando métodos conhecidos pela comunidade científica, é necessário levantar possíveis ameaças à validade dos resultados encontrados neste trabalho. Uma ameaça plausível é a quantidade relativamente pequena de entrevistas feitas.

Outra ameça que deve ser levada em conta é o processo de *coding* realizado. Ele foi feito baseando-se no áudio das entrevistas, e não nos texto transcritos, como geralmente é feito [8]. Isso pode tornar os códigos enviesados ou até mesmo significar a falta de códigos importantes que poderiam ter sido levantados pelo processo original.

## 6.2 Trabalhos Futuros

Como trabalhos futuros, pode ser feito o mesmo estilo de entrevistas com um grupo maior de pessoas. A reaplicação poderia levantar novas barreiras e até identificar lacunas que não puderam ser vistas com a amostra deste trabalho. Outro ponto que também seria de grande valia é a pesquisa sobre o mesmo tema, mas de forma quantitativa, com o objetivo de entender melhor que práticas estão sendo utilizadas na indústria de Recife com um grupo maior de

entrevistados.

APÊNDICE A

# Questionário Traduzido

## A.1 Demografia

- Qual seu cargo no trabalho atualmente?
- Qual o tamanho da empresa para o qual você trabalha?
- Qual é o tamanho típico das equipes dentro da empresa que você trabalha?
- Quantos anos de experiência profissional em <cargo do entrevistado> você tem?
- Qual o domínio (contexto: educação, saúde, etc) da empresa ou projeto no qual você trabalha?

## A.2 Processo de entrega em geral

Dado uma funcionalidade recém-implementada, como é o processo de entrega da sua empresa a partir do commit da funcionalidade até chegar ao ambiente de produção e, portanto, aos clientes?

Possíveis questões complementares:

- O processo é o mesmo para toda mudança no código, isto é, é o mesmo para uma funcionalidade totalmente nova bem como para uma mudança pequena no código? Quem define/decide sobre esse "impacto" das mudanças?
- As mudanças são lançadas diretamente para todos os usuários?
- O quão automatizado é esse processo?
- Quanto tempo leva desde o commit de uma funcionalidade até que ela esteja disponível para os usuários?
- Qual a frequência típica de entrega da sua empresa? A cada commit, uma vez por dia, uma vez por semana, etc? Como isso se relaciona com os processos de desenvolvimento de software? Por exemplo, você está fazendo entregas "sempre" ou só no fim de uma sprint?

- Como os empregados (incluindo desenvolvedores) ficam informados acerca das entregas, por exemplo, qual versão está atualmente implementada, quais versões/funcionalidade estão sob teste em certos ambientes? Como essa comunicação é gerenciada entre equipes e nas equipes?
- Como você julga pessoalmente seu processo de entrega? O que funciona bem, e o que não funciona?

## A.3 Papéis/Responsabilidades

Quais são, tipicamente, os papéis envolvidos no processo de entrega da sua empresa e quem é responsável pelo que?

É uma abordagem mais colaborativa com times contendo pessoal de operações e pessoal de desenvolvimento, ou existem equipes dedicadas, por exemplo: o time de operações assume a partir do momento que as mudanças feitas pelo time de desenvolvimento passaram os testes e estão prontas para serem entregues?

Possíveis questões complementares:

- Quem é responsável por monitorar a entrega uma vez que ela chegou em produção?
- Como você julga, pessoalmente, as definições de papéis na sua equipe? Elas fazem sentido? Algo está faltando, ou não está muito claro?
- Caso haja papel de DevOps: Como DevOps é realizado dentro da sua empresa?

## A.4 Garantia da Qualidade

Quando é feito o commit de uma mudança no sistema de controle de versão (git, por exemplo) da sua empresa ou do seu projeto, como vocês garantem que commits errôneos ou commits que não seguem certas orientações/padrões não cheguem em produção? Como vocês garantem a qualidade do software na empresa?

Possíveis questões complementares:

- Caso seja um processo em estágios, quais estágios existem?
- Caso seja em estágios: Testes críticos são selecionados e executados em estágios iniciais para obter feedback mais rapidamente?
- É feito o build do software exatamente uma vez durante o processo todo, ou cada estágio requer uma build com configuração diferente? Tais arquivos de configuração são gerenciados pelos sistemas de controle de versão?
- Existem estágios/barreiras de qualidade que exigem uma aprovação manual explícita?
- Existe um ambiente parecido com o de produção ou alguma forma para que vocês tenham a certeza de que as mudanças vão funcionar em produção também?

- As revisões de código (code reviews) são parte do seu processo de análise de qualidade(testes)? Quais as razões para que isso seja feito/não seja feito?
- Você acha que o processo de análise de qualidade de vocês funciona bem? O que poderia ser melhorado, na sua opinião?

## A.5 Gerenciamento de Problemas

Suponha que uma nova funcionalidade ou mudança chegou em produção, mas não se comporta como esperado(*). Quais são os passos executados e por quem são executados para gerenciar problemas em caso de:

1. Bugs devidos a erros no código
2. Defeitos devido a desvios dos requisitos, resultando em, por exemplo, performance ruim, mantenabilidade ruim, usabilidade ruim, etc.
3. Problemas com a disponibilidade de certas funcionalidades/serviços/partes da sua aplicação.

Adicionalmente: Como vocês identificam ou mensuram se o sistema está exibindo um comportamento inesperado?

Possíveis questões complementares:

- Tais problemas são comuns?
- Como são os problemas típicos que ocorrem em tempo de execução?
- O lançamento de uma funcionalidade requer que os desenvolvedores envolvidos estejam em plantão para lidar com tais problemas?
- Como tais problemas são identificados/descobertos? Existe suporte para isso fazendo uso de alguma ferramenta?
- Reversões (rollbacks) são utilizados? Caso afirmativo, quanto a compatibilidade de versões, como vocês lidam com problemas quando revertendo para versões incompatíveis?
- Você tem quaisquer sugestões para como prevenir uma boa parcela de problemas em tempo de execução?
- Em caso de uma arquitetura baseada em serviço: O quão comum é que novas versões de certos serviços sejam incompatíveis com outros serviços já existentes?

## A.6 Avaliação da entrega

Dado a entrega contínua de novas versões, como vocês comparam se a versão mais recente é "melhor que" ou traz benefícios comparada com suas predecessoras? Como vocês avaliam se uma certa feature/mudança:

- Está satisfazendo as demandas dos usuários
- Tem o impacto desejado no lucro, performance, mantenabilidade, usabilidade, ... ?

Suponha que você quer comparar duas versões da sua aplicação que foram entregues e estão em execução. Quais as métricas que você consideraria?

Possíveis questões complementares caso live testing (Canary, A/B, ...) for usado:

- Qual a duração de tais testes?
- Qual a quantidade? Quantos testes rodam em paralelo?
- Cada um desses experimentos testa exatamente uma feature?
- Como esses testes são avaliados, quando, e em quais intervalos?
- Quem define as métricas e as limiares que se deve observar?
- Como você se certifica que os experimentos paralelos não estão influenciando um ao outro?
- Qual o escopo dos testes? Quais usuários são selecionados para tais testes e como eles são selecionados?

## A.7 Finalização

Caso não tenha sido perguntado ou mencionado durante a entrevista:

- Que tipo de arquitetura tem o sistema? Monolítica, baseada em serviços (microsserviços)?
- Caso teste A/B não seja utilizado: Quais são as razões para não utilizar técnicas como teste A/B?
- Como você lida com funcionalidades que ainda não estão prontas para serem entregues, especialmente se elas necessitam de mudanças mais complexas através da base de código? Você faz uso de condicionais no código que previne tais funcionalidades de serem executadas até estarem prontas?
- Você tem processos de entrega diferentes para projetos diferentes?
- Na sua opinião, o quão automatizado os processos de entrega devem ser? Por exemplo, deve haver uma aprovação manual antes de fazer uma reversão automática para uma versão anterior (rollback) quando certos limiares não são atingidos?

# Referências Bibliográficas


[1] 14th annual state of agile report, 2020.

[2] Continuous deployment study - online appendix, 2020.

[3] O que É o porto digital, 2020.

[4] Adam Debbiche, Mikael Dienér, and Richard Berntsson Svensson. Challenges when adopting continuous integration: A case study. In *International Conference on Product-Focused Software Process Improvement*, pages 17–32. Springer, 2014.

[5] Abhishek et al DWARAKI. Gitflow: Flow revision management for software-defined networks. 2015.

[6] Michael Hilton et Al. Continuous integration (ci) needs and wishes for developers of proprietary code. 2016.

[7] Wagner Felidré et Al. Continuous integration theater. 2019.

[8] Barney G. Glaser. Basics of grounded theory analysis: Emergence vs. forcing. 1992.

[9] et al. Schermann, Gerald. An empirical study on principles and practices of continuous delivery and deployment. 2016.

[10] Mitchel Douglas Tony Savor and Michael Gentili. Continuous deployment at facebook and oanda. 2016.