Pesquisa sobre práticas de Integração e Deployment contínuos em Recife

Resumo

As práticas de *Continuous Integration* [CI], *Continuous Delivery* e *Continuous Deployment* [CD] 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 investigam o estado da arte destas práticas na maioria das empresas, principalmente no contexto da cidade de Recife. Com isso, o presente trabalho objetiva entender como as técnicas de integração, entrega e implantação contínuas foram importadas para as empresas recifenses, assim como identificar princípios e práticas adjacentes que governam a adoção destas técnicas. Para tanto, foi realizada uma pesquisa qualitativa por meio de entrevistas com 11 desenvolvedores de software de empresas de tecnologia sediadas na cidade do Recife. Foi descoberto que a amostra não segue o *Stairway to Heaven*, teoria definida pelo artigo base deste trabalho e que sugere que a adoção deve seguir uma sequência específica de práticas. Não obstante, a maioria entrevistada integra o código de uma nova funcionalidade para a *branch* principal apenas no final da *Sprint*, não diariamente, o que não é consistente com as definições mais comuns de CI. Além disso, a prática de Testes A/B não foi encontrada em nenhum dos times entrevistados, em razão da baixa quantidade de usuários ativa ou por não se aplicar ao contexto da aplicação.

Keywords

Integração Contínua, Implantação Contínua, Desenvolvimento Colaborativo, Engenharia de Software

ACM Reference Format:

. 2021. Pesquisa sobre práticas de Integração e Deployment contínuos em Recife. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introdução

As práticas de integração, entrega e implantação contínuos [7, 8] (*Continuous Integration* [CI], *Continuous Delivery* e *Continuous Deployment* [CD]) já são muito difundidas e utilizadas por empresas de tecnologia em todo o mundo. Segundo a pesquisa da empresa *digital.ia*[1], 55% dos participantes reportaram que sua organização utiliza a técnica de integração contínua. Também nesta mesma pesquisa foi encontrado que 41% utilizam a técnica de entrega contínua e 36%, implantação.

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 de CI/CD [10] e o seu uso traz maior qualidade ao software que está sendo produzido [18].

É possível, no entanto, perceber uma grande falta de estudos a respeito de como essas práticas foram importadas para as empresas de todo o mundo [17], incluindo Recife – as pesquisas são geralmente voltadas apenas para as corporações globais como Facebook [18] e Google [15]. A capital pernambucana é um dos grandes pólos tecnológicos do Brasil: somente no Porto Digital, localizado na cidade, há cerca de 330 empresas e 11 mil trabalhadores, com faturamento anual de R$ 2,3 bilhões em 2019 [2].

Um estudo a respeito de como as técnicas de CI/CD migraram para a indústria de Recife serve como ponto de partida para pesquisas relacionadas ao levantamento de dores sentidas pelos desenvolvedores locais que inibem a utilização destas técnicas.

Neste contexto, este trabalho tem como objetivo entender, através de uma pesquisa qualitativa, quais as práticas e técnicas subjacentes de integração, entrega e *deployment* contínuos adentraram nas empresas de Recife. Não obstante, o trabalho deseja descobrir que princípios e práticas subjacentes governam a adoção destas técnicas.

2 Motivação

Este trabalho foi motivado majoritariamente pela grande popularidade de integração e implantação contínuas, aliado a falta de estudos a este respeito no contexto da cidade de Recife.

2.1 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 a pesquisa realizada pela empresa *digital.ai* [1], 55% dos participantes reportaram que sua organização pratica a técnica de integração contínua. 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 [10] comenta que desenvolvedores envolvidos se sentem mais produtivos quando utilizam a prática e dão mais valor aos testes automáticos. Já com

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

relação a *deployment* contínuo, o estudo [18] mostra que seu uso traz maior qualidade ao software que está sendo produzido.

2.2 A falta de estudos no contexto Recifense

Ainda há, no entanto, uma grande falta de estudos a respeito de como essas práticas foram importadas para a maioria das empresas [17], 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 [2].

O presente trabalho procura entender sobre a utilização das práticas de CI/CD nas empresas de Recife para que esses dados sirvam como objeto de pesquisa para levantamento de possíveis dores sentidas pelos desenvolvedores locais que justifiquem a adoção ou não destas práticas. Um estudo a respeito de como as técnicas de integração e *deployment* contínuo migraram para a indústria 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 levantar possíveis discrepâncias entre empresas situadas na área e as grandes corporações.

2.3 O artigo base

Com o objetivo de entender um pouco mais sobre o estado da prática de CI/CD no contexto de Recife, nos baseamos no estudo [17], que busca entender como as práticas geralmente associadas a *Continuous Deployment* acharam o seu caminho nas indústrias européia e norte-americana. Nesse estudo os autores utilizaram um método misto de estudo empírico baseado em um pré-estudo na literatura, entrevistas com 20 participantes e um *survey* que recebeu 187 respostas. A ideia era questionar até que ponto o conhecimento na área estava dominado por peculiaridades de um pequeno grupo de grandes empresas, como Facebook e Google.

Os autores também definem a chamada *stairway to heaven* (escada para o céu, em tradução livre), presente na Figura 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.

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* [11] 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 baseado em princípios que estabeleçam quando deve-se utilizar técnicas de entregas parciais, por exemplo, que funcionalidades e métricas devem ser testadas por um teste A/B [13].

Aprender mais sobre as dificuldades que outras empresas não globais sofrem no processo de adoção de práticas de CI/CD pode gerar mais estudos a respeito de como solucionar tais problemas, assim como mostrar possíveis oportunidades de melhoria e revisão dos processos utilizados. Um trabalho que utilize uma adaptação da metodologia aplicada a uma amostra de um outro contexto pode revelar ainda 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.

Assim, a grande adoção da indústria mundial das práticas de CI/CD, aliado a falta de trabalhos a respeito no contexto recifense, além da metodologia já validada proposta pelo artigo foram as principais motivações para o desenvolvimento deste trabalho

2.4 Perguntas de pesquisa

Com o intuito de entender como as técnicas de integração, entrega e *deployment* contínuos foram importadas para as empresas recifenses, 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 CI/CD na indústria?

Para responder às perguntas acima foi decidido aplicar a pesquisa qualitativa do artigo [17] 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 devido à falta de tempo hábil para tal, mas esta pode servir como um trabalho futuro a este.

Além disso, as perguntas 1 e 3 são uma adaptação das perguntas 1 e 2 do artigo base [17], respectivamente, 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.

3 Definições

Para este trabalho foram utilizados as mesmas definições definidas pelo artigo base. Neste capítulo serão abordados todas as definições das práticas pesquisadas no trabalho.

3.1 Integração Contínua

A técnica de *Continuous Integration* (integração contínua ou CI), de acordo com [8], têm como principal característica a integração de código no repositório compartilhado múltiplas vezes por dia. Para garantir a qualidade do software que está sendo integrado, cada *merge* passa por testes automáticos pré-programados que garantem que cenários de usabilidade estão de acordo com os requisitos definidos.

Algumas práticas foram definidas dentro da técnica para definir processos chaves que produzem um CI eficiente. Entre elas, temos o *trunk based development* [TBD] [6], onde todo o time contribui para uma única *branch* no repositório de versionamento, chamada usualmente de *main*. Esta técnica é comumente utilizada para diminuir

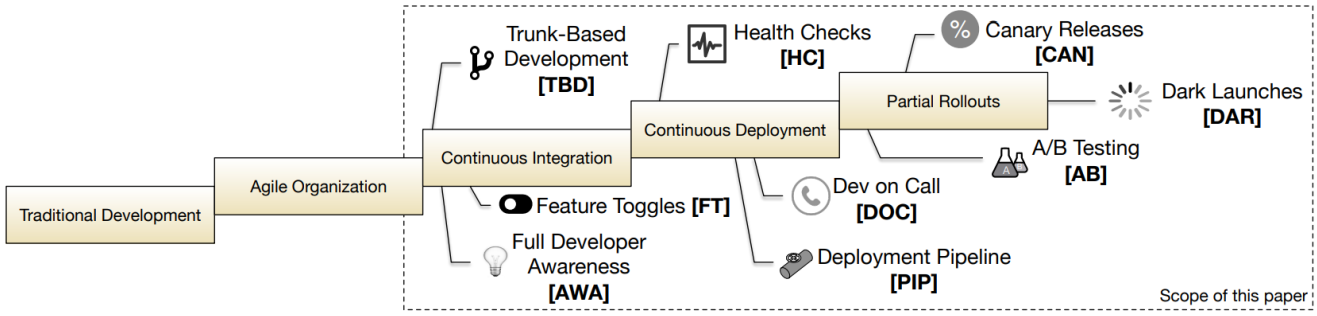

Figura 1: A escada de evolução denominada *Stairway to Heaven* proposta pelo artigo base. Fonte: Schermman et al [17]

a quantidade de conflitos entre linhas de códigos que estão sendo concorrentemente modificadas por desenvolvedores diferentes.

Para garantir que o *trunk based development* funcione como esperado, deve-se ter uma ferramenta que possibilite o chaveamento de partes de código que ainda não foram finalizadas. A implementação comum desta técnica é *feature toggle* [FT] [11]. A técnica define, através de condicionais adicionadas no código fonte, que blocos devem ser executados ou ignorados no ambiente de produção.

Outra prática relacionada a técnica de integração contínua é o chamado *developer awareness* [AWA] [16], que prega a quebra de silos entre os desenvolvedores e processos de *release*, o status dos ambientes e informações gerais da infraestrutura do sistema que está sendo desenvolvido. A ideia é não depender de um time específico de fora para cuidar de questões arquiteturais e do processo de entrega, mas sim difundir o conhecimento entre todos os envolvidos no desenvolvimento. Este é um dos conceitos recomendados na cultura *DevOps* comentada anteriormente.

3.2 Entrega e Implantação Contínuos

A técnica de *Continuous Delivery* (entrega contínua ou CD) define que o software tem que estar a qualquer momento pronto para ser enviado para produção no repositório principal [7]. Para que a técnica de entrega contínua funcione como o esperado, é necessário principalmente que o software esteja pronto para *deploy* durante todo o ciclo de vida e que o time sempre priorize deixar o código pronto para produção ao invés de priorizar novas funcionalidades.

Muitas vezes a técnica de *Continuous Delivery* é confundida com a de *Continuous Deployment* (implantação contínua). No entanto, esta última significa que o código é automaticamente colocado em produção sempre que possível, resultando em múltiplos *deployments* por dia. Com a entrega contínua, a empresa pode escolher ter um processo de entregas mais devagar, mesmo tendo o software sempre pronto para utilização. Outro ponto importante é que para se ter Implantação contínua, é necessário ter entrega contínua.

A técnica de *Continuous deployment* foi bastante difundida através do exemplo de empresas como a *Flickr*, que em 2009 comentou sobre seus mais de 10 *deploys* diários [3], antes mesmo o termo *DevOps* ser inventado.

Algumas práticas definem alguns dos passos necessários para utilizar a técnica de CD. Podemos citar a *deployment pipeline* (pipeline de implantação) [PIP] [14], que define o conjunto de passos que qualquer mudança de código tem que passar para chegar em produção. Esses passos podem tratar da compilação do código, da execução de testes em diferentes ambientes, entre outras coisas e pode ser totalmente ou parcialmente automatizada. Idealmente, o conjunto de passos deve ser automatizado o máximo possível para garantir rapidez e confiabilidade dos processos envolvidos.

Depois da entrega de uma nova versão, *health checks* [HC] [14] são necessários para garantir que o produto está funcionando corretamente. O sistema tem uma série de parâmetros definidos pela equipe que mostram se há algum problema no ambiente de produção, por exemplo, falha na conexão com o banco de dados, serviços inativos, certificados HTTPS expirados, entre outros. Muitas vezes o sistema tem a funcionalidade de envio de mensagens para os responsáveis quando algo de errado ocorre no ambiente.

Para garantir ainda mais uma colaboração maior entre desenvolvedores e o time de operações, surgiu a prática de *developer on call* [DOC] [6]. Esta sugere que o desenvolvedor responsável pela funcionalidade recém lançada fique disponível por tempo extra após o lançamento em produção. Caso haja algum erro, ele será a pessoa mais propícia a resolvê-lo o mais rápido possível.

3.3 Entregas Parciais

A prática de *partial rollouts* (ou entregas parciais) pode ser definida como um processo de garantia de qualidade e validação de requisitos que ocorrem em tempo de execução [17]. Dentre as técnicas ligadas, podemos citar *canary releases* [CAN] [12], que define o ato de enviar versões apenas para uma parte dos usuários ativos. Isto serve para testar mudanças no software primeiramente com uma parcela pequena de clientes.

Outra prática relacionada a técnica é a de Testes A/B [AB] [13]. Ela se baseia no teste concorrente de duas ou mais versões rodando paralelamente, que se diferem em um ponto isolado do sistema. A ideia é avaliar através de medidas estatísticas de uso e performance do sistema quais das versões é a mais pertinente aos requisitos do software. O teste pode mostrar desde versões que representam um tempo de resposta mais rápido para o usuário, até o lugar que um botão deve aparecer para vender mais produtos.

A prática de *dark launches* [DAR] [6] também se enquadra dentro das técnicas de entregas parciais. Ela é utilizada para testes de funcionalidades no ambiente de produção, mas sem a necessidade

de habilitar a mesma para os usuários. Geralmente é utilizado para testar situações encontradas especificamente apenas em produção.

4 Metodologia

Para responder as perguntas de pesquisa foi realizada uma pesquisa qualitativa através de entrevistas semi-estruturadas utilizando como ponto de partida o roteiro desenvolvido pelo artigo base [17]. AS entrevistas foram feitas com desenvolvedores de empresas sediadas em Recife.

4.1 Pré-estudo

Com o objetivo de entender melhor sobre como as práticas de CI/CD migraram para as empresas sediadas em Recife, essa fase do estudo contou com a participação de um segundo pesquisador para discussão do artigo base [17]. Na discussão foi definido que seria aplicado 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 pesquisadores acessaram o apêndice do trabalho 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 também foi traduzido para português e utilizado durante as entrevistas [1]. 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*.

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) Gerenciamento 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, sendo 5 feitas por cada um dos dois pesquisadores.

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 65 minutos, duração bem parecida com os tempos obtidos no artigo base (35 a 60 minutos), totalizando 7 horas e 15 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 do participante e confidencialidade de demais informações pessoais, bem como da empresa onde o participante trabalha.

4.3 Participantes

No total foram entrevistados 11 desenvolvedores (P1 a P11) de 7 empresas diferentes sediadas em Recife. A Figura 2 mostra todos os dados demográficos obtidos de cada um dos entrevistados. Esses dados foram obtidos pelos entrevistadores no começo de cada entrevista e agregados nesta figura. O tamanho da empresa é definido pela quantidade de colaboradores: *Startup* define empresas com menos de 50 trabalhadores; *SME* são empresas que contém entre 50 e 500 colaboradores; e *CORP* são empresas com mais de 500 colaboradores.

Da amostra, 3 eram mulheres. Pode-se ver a distribuição dos participantes por gênero na Figura 3. Dentro desse grupo, 9 trabalhavam com aplicações Web, enquanto 1 trabalhava com sistemas embarcados e o último, com jogos.

ID	Empresa	Tipo de Aplicação	Tamanho da Empresa	Experiência (Anos)	Sexo
P1	E1	Web	SME	2,5	M
P2	E2	Web	CORP	22	M
P3	E2	Web	CORP	2	F
P4	E7	Hardware	CORP	3	F
P5	E5	Games	STARTUP	2	M
P6	E6	Web	STARTUP	1	M
P7	E3	Web	STARTUP	0,58	M
P8	E1	Web	SME	2	M
P9	E3	Web	STARTUP	1	M
P10	E4	Web	SME	1	F
P11	E4	Web	SME	8	M

Figura 2: Detalhamento dos entrevistados da amostra, contendo todos os dados demográficos coletados.

A escolha dos entrevistados foi feita baseado na rede de conhecidos dos entrevistadores, com o propósito de agregar pessoas que trabalhavam em empresas de tamanhos distintos para garantir uma

[1] O guia de entrevistas traduzido está disponível em http://bit.ly/ArtigoQuestoesEntrevista

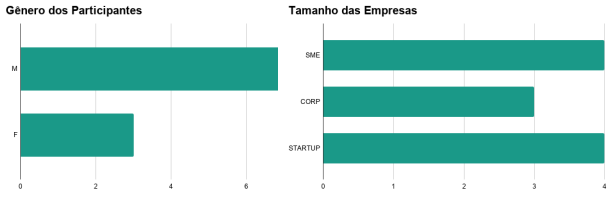

Figura 3: Distribuição dos Participantes por gênero e tamanho da empresa

variedade de parâmetros envolvidos. Como é possível perceber também na Figura 3, com relação a esse aspecto a amostra está bem distribuída.

4.4 Análise dos Dados

A Figura 4 apresenta uma visão geral de como funcionou o processo de coleta e análise de dados. A análise foi feita apenas por um dos autores deste estudo. O processo escolhido foi baseado nas fases de *Coding* e *Thematic Analysis* da metodologia *Grounded Theory* [9]. 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

Gerou o código: *"Todos os desenvolvedores são devops."* – P5_15, onde P5 identifica o número do participante do qual a entrevista gerou o código, e o 15 define a ordem de criação deste, além de servir como identificador único.

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 a 2, representando não utiliza, utiliza parcialmente e utiliza completamente baseado nas definições do artigo base traduzidas. Por exemplo, na Figura 4 pela etapa de "Categorização dos tópicos nas práticas". Este processo foi então replicado para cada uma das 11 entrevistas.

Ao final do processo ainda haviam 3 entrevistas que não geraram nenhum código relacionado a prática de *Dark Launch*. 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 [2].

Após a etapa de geração e revisão dos códigos, estes foram então agrupados, quando válido, na prática específica com a qual aquele código se relacionava. Este agrupamento gerou a Tabela Entrevistado-Pratica-Codigos [3], que mostra uma visão geral de cada relação entre prática e entrevistado, contendo a nota dada e os códigos que justificam a nota. A nota é demonstrada pela cor presente em cada célula: branco significa não utiliza, verde claro mostra que o participante utiliza parcialmente, e verde mais escuro, totalmente. A tabela contém 147 dos 292 códigos gerados e foi utilizada como base para a maioria das figuras geradas neste trabalho. A Figura 5 apresenta um trecho da Tabela Entrevistado-Pratica-Codigos.

Para identificar padrões e abstrações nos códigos agrupados na Tabela Entrevistado-Pratica-Codigos, foi feito um trabalho de agrupamento semântico, gerando por fim 17 novas super categorias[4] através do processo de *Thematic Analysis* [9]. Esse processo é ilustrado pela etapa de "Criação de super categorias" na Figura 4. As super categorias tinham como objetivo agrupar códigos semelhantes ou de mesma semântica para facilitar na reflexão dos dados. Um exemplo de super categoria gerada para a prática de *Trunk Based Development* [TBD] pode ser encontrado na Figura 6.

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 novos tópicos que agregam códigos sobre as práticas de *Code Review* e de testes automáticos. Para estas foram relacionados 29 códigos dos 292 já mencionados, e duas super categorias foram geradas. A tabela com todos os códigos relacionados está presente na Tabela Entrevistado-Pratica-Codigos juntamente com as outras práticas.

Por fim, foram geradas algumas tabelas derivadas da Tabela Entrevistado-Pratica-Codigos, ilustrado no passo de "Geração de resultados" na Figura 4, para facilitar a visualização e análise dos dados agregados, mostrando um mapa de utilização das práticas na amostra de diferentes formas. Essas tabelas serão apresentadas na próxima seção.

5 Resultados

Neste capítulo serão apresentados os resultados encontrados com as entrevistas baseadas no processo de análise de dados apresentada no capítulo anterior. Primeiramente será abordado como foi feito o agrupamento dos códigos gerados com as práticas. Após isso, 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?* – de forma restrita aos 11 entrevistados das 7 empresas da amostra, foi montada a Tabela 1, demonstrando a utilização de cada uma das práticas definidas para cada um dos participantes. Nesta, é demonstrada 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

[2] A lista de todos os códigos está disponível em http://bit.ly/ArtigoCodigos
[3] A Tabela Entrevistado-Pratica-Codigos está disponível em http://bit.ly/ArtigoTabela
[4] A tabela com todas as super categorias geradas pode ser encontrada em http://bit.ly/ArtigoSuperCategorias

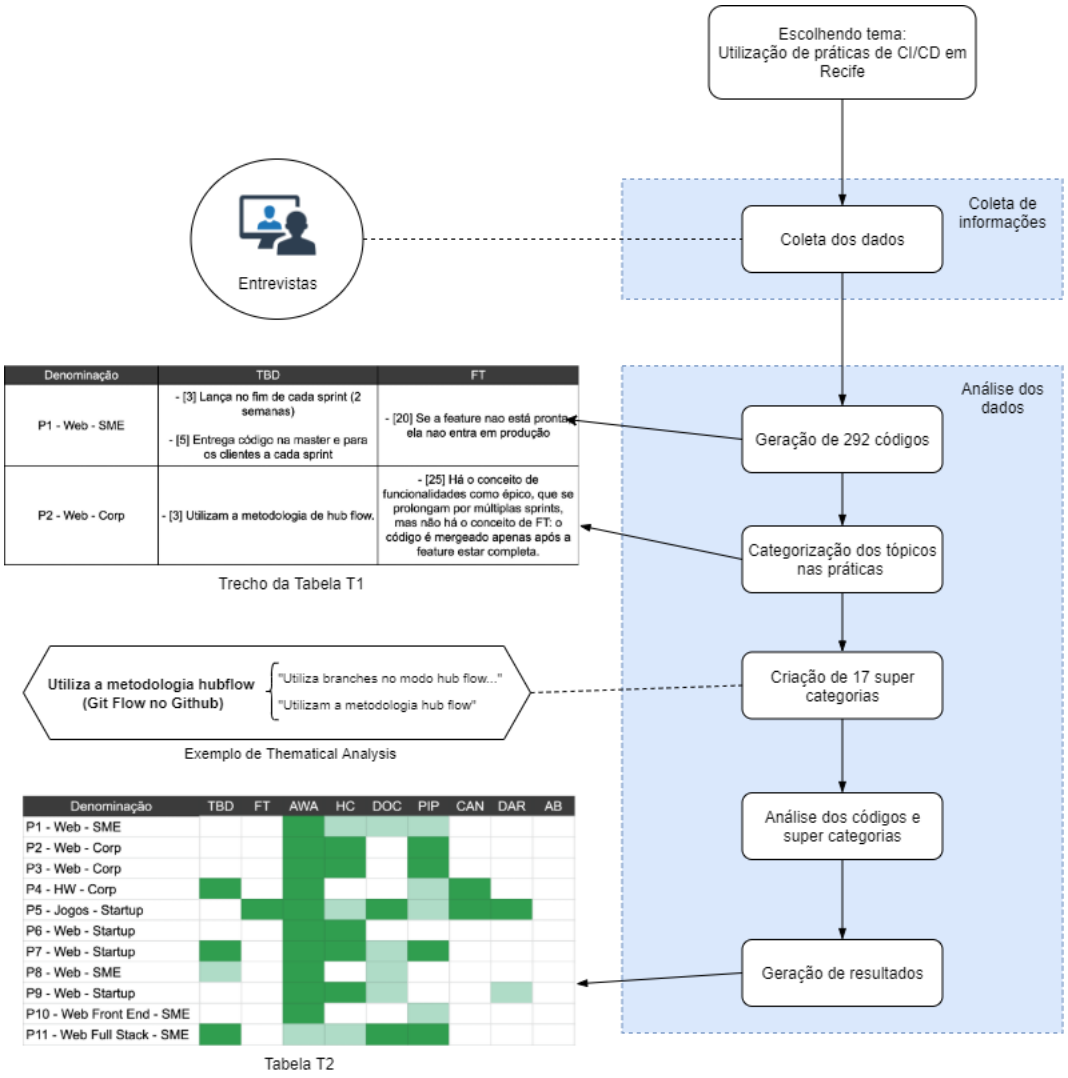

Figura 4: Visão geral dos processos de coleta e análise de dados.

utilização. Esta foi produzida a partir da Tabela Entrevistado-Pratica-Codigos, retirando-se os códigos agrupados e ordenando as colunas pela utilização de determinada prática. A ideia é replicar a Tabela 1 do artigo base [17], presente na Figura 7.

Com a Tabela 1 é possível perceber que *Developer awareness [AWA]* foi a prática mais encontrada em toda a amostra, sendo totalmente utilizada pela grande maioria dos entrevistados; apenas um deles utiliza parcialmente. Logo após, pode-se encontrar as práticas de *Health Check* [HC] e Pipeline de Implantação [PIP] 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 1 com a Tabela 1 do artigo base (Figura 7), podemos perceber que há diferenças de posição entre práticas, mas que não há mudanças exorbitantes. Com a ajuda da Figura 2, 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. É possível perceber que as práticas de AWA, TBD, DAR e FT tiveram mudanças de 2 posições, enquanto HC, PIP, CAN e AB tiveram apenas diferença de 1 unidade de posição. Apenas *developer on call* permaneceu no mesmo local dentro das duas amostras. Por fim, é interessante perceber que o conjunto das três primeiras práticas é o mesmo, mas em posições trocadas nos dois estudos.

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 3. Esta contém a visualização de utilização das práticas, utilizando a mesma técnica

Denominação	HC	DOC
P1 - Web - SME	- [17] Não há HC rodando atualmente, mas já houve (está quebrado). Há apenas um check sobre o certificado https.	- [9] Geralmente não fica de plantão depois do deploy para algum problema, apenas quando vai haver uma apresentação para clientes.
P2 - Web - Corp	- [14] Utiliza Health check para monitorar se a aplicação está funcionando como esperado e todas suas conexões (banco de dados e cache)	- [21] Não é necessário o uso de DEV on call, eles confiam bastante no processo de garantia de qualidade.

Figura 5: Trecho da Tabela Entrevistado-Pratica-Codigos, com os códigos derivados das entrevistas agrupados por prática. As cores de cada célula representam a nota dada para determinada prática e foram dadas com base nos códigos presentes nela. Em algumas células há mais de uma evidência.

Denominação	AWA	HC	PIP	DOC	TBD	CAN	DAR	FT	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									

Tabela 1: Nível de utilização de cada uma das práticas, com as colunas ordenadas em ordem decrescente de uso. Práticas: AWA: *Developer Awareness*; HC: *Health Check*; PIP: Pipeline de Implantação; DOC: *Developer on Call*; TBD: *Trunk Based Development*; CAN: *Canary Releases*; DAR: *Dark Launches*; FT: *Feature Toggles*; AB: Testes A/B.

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

Tabela 2: Diferença entre o artigo base e este trabalho a respeito da ordem de predomínio das práticas. Práticas: AWA: *Developer Awareness*; HC: *Health Check*; PIP: Pipeline de Implantação; DOC: *Developer on Call*; TBD: *Trunk Based Development*; CAN: *Canary Releases*; DAR: *Dark Launches*; FT: *Feature Toggles*; AB: Testes A/B.

de cores aplicada na Tabela 1 e explicada na seção anterior, mas com as colunas ordenadas pela escada definida pela sequência de práticas do *Stairway to Heaven* apresentada na Figura 1 (Capítulo 3).

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									

Tabela 3: Nível de utilização de cada uma das práticas, com as colunas ordenadas na ordem do *Stairway to Heaven*. Práticas: AWA: *Developer Awareness*; HC: *Health Check*; PIP: Pipeline de Implantação; DOC: *Developer on Call*; TBD: *Trunk Based Development*; CAN: *Canary Releases*; DAR: *Dark Launches*; FT: *Feature Toggles*; AB: Testes A/B.

Com a Tabela 3 é possível perceber que a amostra deste estudo, assim como a amostra do estudo original, não segue a evolução proposta pelos autores do artigo base. 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 uma lacuna nas duas primeiras práticas, seguido de uma grande utilização da terceira, confirmando a tese de que a *Stairway to Heaven* não foi identificada na amostra. O mais próximo dos participantes é P5, mas ainda há nele a falta do pilar TBD do primeiro degrau da escada. É interessante destacar que na amostra do próprio artigo base também não foi possível identificar a escada de evolução, como é possível visualizar na Figura 7.

5.3 Análise 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 CI/CD na indústria?* As seguintes subseções agrupam as práticas emergentes através das práticas gerais definidas no *Stairway to Heaven*, inclusive seguindo a ordem deste, conforme explicado no Capítulo 4, Seção 4.4.

Para poder definir padrões encontrados na amostra, o autor, após a fase de agrupamento semântico, juntou todos os códigos 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 cada prática. Foi feito também um estudo comparativo com os resultados obtidos pelo artigo base para analisar discrepâncias e congruências entre os dois contextos de estudo.

5.3.1 Integração Contínua

Com a Tabela 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*

Categories	Codes
Utiliza a metodologia hubflow (Git Flow no Github)	P2 - [3] Utilizam a metodologia de hub flow.
	P5 - [11] Utiliza branches no modo hub flow. Apenas entra na branch master aquilo que está preparado para ir para produção.

Figura 6: Exemplo de super categoria gerada durante o processo de *Thematic Analysis*.

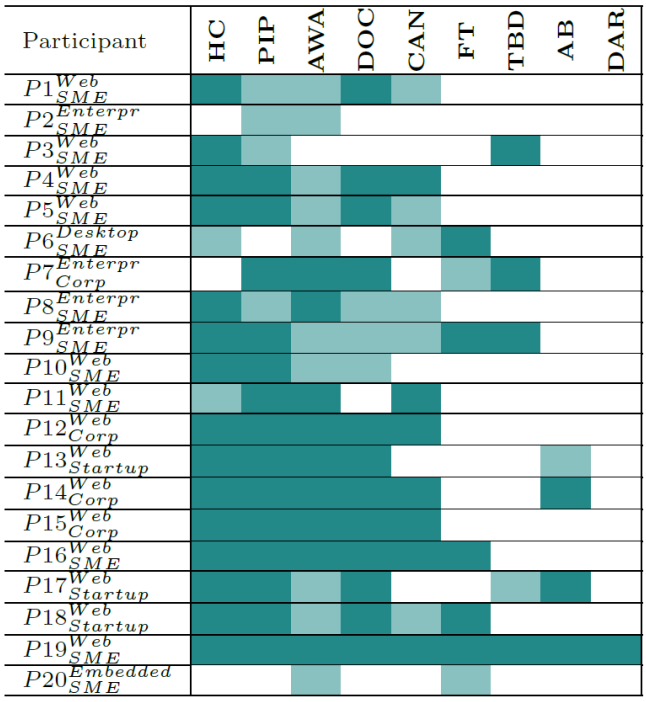

Figura 7: Utilização das práticas pelos participantes do artigo base. Fonte: Schermman et al [17]. Práticas: AWA: *Developer Awareness*; HC: *Health Check*; PIP: Pipeline de Implantação; DOC: *Developer on Call*; TBD: *Trunk Based Development*; CAN: *Canary Releases*; DAR: *Dark Launches*; FT: *Feature Toggles*; AB: Testes A/B.

[16] – 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, como explicado no Capítulo 2: *Trunk Based Development* [TBD] [6], *Feature Toggles* [FT] [11] 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 adotada, 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-se 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. Esta técnica foi encontrada apenas na equipe do participante P5, que a utilizava para esconder funcionalidades enquanto testes manuais ainda estavam sendo feitos. Geralmente esta técnica é utilizada para auxiliar a integração de códigos ainda não finalizados quando a equipe utiliza técnicas como o *trunk based development*, mas este não é o caso de P5. 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 conflito" — P3 – Web – CORP

5.3.1.3 Developer Awareness [AWA]

Na amostra é possível identificar que a prática de *Developer Awareness* [AWA] é amplamente adotada nas companhias, assim como na amostra do artigo base. Em 6 entrevistas foi possível perceber que o time de desenvolvimento era o mesmo responsável pela entrega e manutenção da aplicação. Em 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.

5.3.1.4 Deployment Contínuo

Nesta subseção, serão abordadas as práticas ligadas ao processo de *Deployment* contínuo, como explicado no Capítulo 2: *Health Checks* [HC] [14], *Developer on Call* [DOC] [6] e Pipeline de Implantação [PIP] [14]. É possível inferir, baseado principalmente na Tabela 1, que as técnicas ligadas ao processo estão presentes na maioria das equipes da amostra: as 3 estão nas quatro primeiras colocações.

5.3.1.5 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 verificação 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.

5.3.1.6 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. Este último comenta que a prática já está 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 escreverem e testarem seus códigos mais veementemente. 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

5.3.1.7 Pipeline de Implantação [PIP]

Em relação à prática de Pipeline de Implantação [PIP] é possível inferir que ela é amplamente adotada pela amostra, visto que apenas 2 entrevistados obtiveram nota 0 (não utiliza). No artigo base 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 interessante perceber que alguns times ainda demonstram falta de automação dos processos da *pipeline*. Isto acontece em decorrência da complexidade da automação, devido às tecnologias e ferramentas utilizadas, ou da falta de prioridade do time para tal.

5.3.1.8 Entregas Parciais

Nesta subseção, serão abordadas as práticas ligadas ao processo de Entregas Parciais, como explicado no Capítulo 2: *Canary Releases* [CAN] [12], *Dark Launches* [DAR] [6] e Testes A/B [AB] [13]. Na amostra foi possível perceber que todas as técnicas são muito pouco utilizadas: todas estão entre as 4 últimas colocações. É válido também notar que as 3 mantiveram a mesma ordem de utilização do processo do artigo base [17].

5.3.1.9 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 que 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.

5.3.1.10 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 [17] 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

5.3.1.11 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 [17], 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 nessa prática. P3 comenta ainda que o time está mais focado em entregar novas funcionalidades, pois a demanda por parte do cliente é 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.

5.4.0.1 Code Review

A prática de *Code Review* [4] 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 experientes, 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.

5.4.0.2 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.

5.5 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.

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

Outra ameaça que deve ser levada em conta é o processo de *coding* realizado. Ele foi feito baseando-se no áudio das entrevistas, e não nos textos transcritos, como geralmente é feito [9]. Isso pode tornar os códigos enviesados ou até mesmo significar a falta de códigos importantes que poderiam ter sido levantados pelo processo original. É válido citar que há algumas vertentes que defendem a codificação através do áudio, por preservar a entonação e a intenção do usuário, mas ainda devem ser feitos trabalhos quantitativos na área para validação e melhor definição do processo [19].

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 [17]. Desde o ranking de utilização de cada prática até as barreiras enfrentadas pelos desenvolvedores, houve uma congruência razoável entre os dois resultados. Esta congruência, principalmente dentro do grupo das práticas mais utilizadas, serve como indicador de que elas acharam seu lugar na indústria e, assim como a teoria propunha, funcionam de forma eficiente.

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*. Este comportamento pode trazer vários problemas de integração de código quando atualizado concorrentemente, causa principal para a criação da prática de *Trunk Based Development* [6]. É válido um trabalho mais quantitativo na região que busque razões específicas para a não utilização desta, aliado a uma divulgação maior dos benefícios que essa técnica pode trazer para times de desenvolvimento.

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. Isso demonstra principalmente que, na amostra, há um trabalho bem evoluído de

otimização e automação do processo de entrega, além de utilização de ferramentas para garantir que a aplicação está rodando corretamente em produção.

Por fim, foi possível perceber que os entrevistados utilizam muito pouco as práticas de entregas parciais. Foi possível identificar técnicas desconhecidas por um grupo razoável de participantes – *Dark Launches* – e até técnicas que não foram utilizadas por nenhum deles – Testes A/B. É necessário um trabalho maior de disseminação de como realizar e das vantagens que essas técnicas podem trazer para produtos produzidos em Recife.

Todos esses pontos demonstram uma certa congruência da amostra de Recife com a amostra européia e norte-americana do artigo base. Assim, podemos afirmar que a amostra está no mesmo patamar no sentido de utilização de outras partes do mundo, mas é necessário uma pesquisa quantitativa para poder generalizar esse achado para a capital pernambucana.

6.1 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, permitindo assim a generalização dos achados para a capital.

Referências

[1] [n.d.]. 14th Annual State of Agile Report. https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494. Acessado em 23/04/2021.

[2] [n.d.]. O que é o porto digital. https://www.portodigital.org/parque/o-que-e-o-porto-digital. Acessado em 07/09/2021.

[3] John Allspaw and Paul Hammond. 2009. 10+ deploys per day: Dev and ops cooperation at Flickr. (2009).

[4] Bacchelli et al. 2013. Expectations, outcomes, and challenges of modern code review. (2013). https://doi.org/10.1109/icse.2013.6606617

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

[6] D. G. Feitelson et al. 2013. Development and Deployment at Facebook. *IEEE Internet Computing* 17(4):8-17 (2013), 8–17.

[7] Martin Fowler. [n.d.]. Continuous Delivery. https://martinfowler.com/bliki/ContinuousDelivery.html. Acessado em 20/04/2021.

[8] Martin Fowler. [n.d.]. Continuous Integration. https://martinfowler.com/articles/continuousIntegration.html. Acessado em 20/04/2021.

[9] Barney G. Glaser. 1992. Basics of Grounded Theory Analysis: Emergence Vs. Forcing. (1992).

[10] Michael Hilton et al. 2016. Continuous Integration (CI) Needs and Wishes for Developers of Proprietary Code. (2016).

[11] Pete Hodgson. [n.d.]. Feature Toggles. https://martinfowler.com/articles/feature-toggles.html. Acessado em 17/04/2021.

[12] Jez Humble and David Farley. 2010. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation.* Addison-Wesley Professional.

[13] Ron Kohavi et al. Agosto 2007. Practical Guide to Controlled Experiments on the Web: Listen to Your Customers not to the HiPPO. (Agosto 2007), 956–967.

[14] Ingo Weber Len Bass and Liming Zhu. 2015. *DevOps: A Software Architect's Perspective.* Addison-Wesley Professional.

[15] John Micco. [n.d.]. Tools for Continuous Integration at Google Scale. https://www.youtube.com/watch?v=KH2_sB1A6lA. Acessado em 20/04/2021.

[16] Akond Ashfaque Ur Rahman et al. 2015. Synthesizing continuous deployment practices used in software development. (2015).

[17] Gerald Schermann et al. 2016. An empirical study on principles and practices of continuous delivery and deployment. (2016).

[18] Mitchel Douglas Tony Savor and Michael Gentili. 2016. Continuous Deployment at Facebook and OANDA. (2016).

[19] Jade L. Ozawa-kirk Uma D. Parameswaran and Gwen Latendresse. 2020. To live (code) or to not: A new method for coding in qualitative research. (2020), 630–644.