Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Caracterização da Adoção de DevOps Utilizando Grounded Theory

Welder Pinheiro Luz

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Rodrigo Bonifácio de Almeida

Brasília
2018

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Pinheiro Luz, Welder
PL979c Uma Caracterização da Adoção de DevOps Utilizando Grounded Theory / Welder Pinheiro Luz; orientador Rodrigo Bonifácio de Almeida. -- Brasília, 2018.
66 p.

Dissertação (Mestrado - Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2018.

1. DevOps. 2. Grounded Theory. 3. Desenvolvimento de software. 4. Operações de software. I. Bonifácio de Almeida, Rodrigo, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Uma Caracterização da Adoção de DevOps Utilizando Grounded Theory

Welder Pinheiro Luz

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Rodrigo Bonifácio de Almeida (Orientador)
CIC/UnB

Prof.a Dr.a Edna Dias Canedo
FGA/UnB

Prof. Dr. Paulo Roberto Miranda Meirelles
FGA/UnB

Prof.a Dr.a Aletéia Patrícia Favacho de Araújo
Coordenadora do Programa de Pós-graduação em Computação Aplicada

Brasília, 13 de julho de 2018

Dedicatória

Eu dedico este trabalho inicialmente aos meus pais, na esperança de que ver um filho concluindo um curso de mestrado possa reduzir um pouco a dor por eles sentida neste momento tão difícil que todos nós estamos passando.

Dedico também à minha esposa Julia e ao meu filho Daniel que tão compreensivos tiveram que ser nesses últimos dois anos que estive tão distante.

Agradecimentos

Agradeço ao meu orientador, o professor Rodrigo, que por todo esse período me ofereceu orientação de altíssimo nível e de maneira extremamente respeitosa, demonstrando que não é necessário diminuir ninguém para se provar grande.

Ao professor Gustavo Pinto (UFPA) por todas as dicas e direcionamentos, poucas vezes na vida vi uma pessoa combinar tanto conhecimento e humildade.

A todos os professores do PPCA que de fato proporcionaram que a minha forma de pensar hoje seja bastante diferente do que era há dois anos atrás, no início do curso.

Ao pessoal do TCU que tanto me ajudou, especialmente à minha chefe Raquel, que sempre compreendeu as minhas ausências nos momentos em que precisei; ao secretário de TI, Rodrigo Felisdório, que me deu todo o respaldo necessário para disseminação da pesquisa no dia a dia do TCU; e ao pessoal do CPA e CPIC cuja colaboração técnica foi imprescindível.

À minha esposa Julia pela paciência de ouvir todas as minhas lamentações e pela força que sempre me deu para que eu não desistisse no meio do caminho.

Ao meu filho Daniel por fazer eu me sentir amado todos os dias e perceber que mesmo que tudo dê errado lá fora, aqui dentro sempre vai ter um abraço bem apertado e cheio de amor para mim.

Aos meus pais pela inspiração. Vê-los trabalhando duro, em situações tão adversas e sem baixar a cabeça nenhum minuto, me faz perceber o quão forte eu também posso ser.

Resumo

DevOps é um conjunto de práticas e valores culturais que visa reduzir as barreiras entre os times de desenvolvimento e operações durante o desenvolvimento de *software*. Devido ao seu crescente interesse e definições imprecisas, trabalhos de pesquisa recentes têm tentado caracterizar *DevOps*—tipicamente utilizando um conjunto de conceitos e práticas relacionadas. Todavia, pouco se sabe a respeito do *entendimento de praticantes* sobre os caminhos bem sucedidos para se adotar *DevOps*. A falta de tal entendimento pode impedir instituições de adotar práticas *DevOps*. Portanto, o objetivo aqui é apresentar uma teoria a respeito da adoção de *DevOps*, destacando a maneira como os principais conceitos relacionados têm contribuído para a sua adoção na indústria. Este trabalho utiliza uma abordagem multimétodo. Inicialmente, foi conduzido um estudo utilizando a variação clássica de *Grounded Theory*. Nesta etapa, profissionais que contribuíram para a adoção de *DevOps* em 15 companhias de diferentes domínios de negócio e de cinco países diferentes foram entrevistados. Com base nos resultados, um modelo foi produzido para melhorar tanto o entendimento como a orientação a respeito da adoção de *DevOps*. Na segunda etapa do estudo, o modelo foi introduzido na adoção de *DevOps* do Tribunal de Contas da União (TCU), momento em que um grupo focal foi conduzido para avaliar o estágio atual da adoção de *DevOps* e para validar a aplicabilidade e utilidade do modelo. O modelo incrementa a visão existente de *DevOps* explicando o papel e a motivação de cada categoria (e seus relacionamentos) no processo de adoção de *DevOps*. Este modelo foi organizado em termos de *categorias de facilitadores DevOps* e *categorias de saídas DevOps*. Concluiu-se que *colaboração* é a principal preocupação de *DevOps*, contrastando com um possível entendimento de que implantar ferramentas específicas para automatizar o *build*, o *deployment* e o provisionamento e gerenciamento da infraestrutura é suficiente para se implantar *DevOps*. Assim sendo, os resultados contribuem para (a) gerar um adequado entendimento a respeito da adoção de *DevOps*, a partir das perspectivas dos praticantes; e (b) auxiliar instituições, como o TCU, no processo de migração para adotar *DevOps*. Adicionalmente, as experiências coletadas durante a produção do modelo têm sido aplicadas durante a adoção de *DevOps* no TCU.

Palavras-chave: DevOps, desenvolvimento de software, operações de software, Grounded Theory, adoção de DevOps

Abstract

DevOps is a set of practices and cultural values that aims to reduce the barriers between development and operations teams during software development. Due to its increasing interest and imprecise definitions, existing research works have tried to characterize DevOps—mainly using a set of concepts and related practices. Nevertheless, little is known about the *practitioners understanding* about successful paths for DevOps adoption. The lack of such understanding might hinder institutions to adopt DevOps practices. Therefore, the goal here is to present a theory about DevOps adoption, highlighting how the main related concepts have contributed to its adoption in industry. This work uses a multi-method approach. Initially a Grounded Theory study was conducted using its classical variation. In this step, practitioners that contributed to the adoption of DevOps in 15 companies from different domains and across five countries were interviewed. Based on results, a model was produced to improve both the understanding and guidance of DevOps adoption. In the second step of the study, the model was introduced in the adoption of DevOps at the Brazilian Federal Court of Accounts (TCU), at which time a focus group was held to evaluate the current stage of adoption and to validate the applicability and utility of the model. The model increments the existing view of DevOps by explaining the role and motivation of each category (and their relationships) in the DevOps adoption process. This model was organized in terms of *DevOps enabler categories* and *DevOps outcome categories*. Was concluded that *collaboration* is the core DevOps concern, contrasting with a possible understanding that implanting specific tools to *automate building, deployment, and infrastructure provisioning and management* is enough to achieve DevOps. Altogether, the results contribute to (a) generating an adequate understanding of DevOps adoption, from the perspective of practitioners; and (b) assisting institutions, like TCU, in the migration path towards DevOps adoption. In addition, the experiences collected during the production of the model have been applied in adoption of DevOps at TCU.

Keywords: DevOps, software development, software operations, Grounded Theory, DevOps adoption

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

CAMS Cultura, Automação, Medição e Compartilhamento (*Sharing*).

CPA Comitê Permanente de Arquitetura.

CPIC Comitê Permanente de Integração Contínua.

GT *Grounded Theory*.

PDTI Plano Diretor de Tecnologia da Informação.

SINAP Serviço de Infraestrutura de Aplicações.

SLA Acordo de Nível de Serviço.

TCU Tribunal de Contas da União.

TI Tecnologia da Informação.

VM Máquina Virtual.

Capítulo 1

Introdução

O advento e a consolidação dos métodos ágeis de desenvolvimento de *software* possibilitaram uma maior eficiência na produção de *software* com entregas mais rápidas em um período de tempo menor [1]. Todavia, essa mudança gerou também um aumento na demanda da equipe de operações, tipicamente responsável pelas atividades relacionadas à publicação dos artefatos de *software*, e que não participava do processo de desenvolvimento ágil [2].

Visando reduzir as barreiras entre os times, trazendo as atividades típicas do time operações para um contexto de agilidade já presente durante o desenvolvimento, o conceito de *DevOps* emergiu na indústria de desenvolvimento de *software*. O termo *DevOps* é uma junção das palavras "*development*" e "*operations*" que foi usado pela primeira vez em 2009 [3].

Antes mesmo de o termo *DevOps* existir, artigos e apresentações já tratavam da aplicação dos princípios ágeis também às atividades de operações, da rediscussão sobre os papéis dos times de desenvolvimento e operações, e dos benefícios de uma mudança de paradigma com maior aproximação e colaboração entre estes times [3, 4, 5].

Os benefícios relacionados a *DevOps* incluem aumento na performance e na produtividade organizacional de TI, redução de custos no ciclo de vida de *software*, melhoria na eficiência e eficácia operacional e maior alinhamento com o negócio entre os times de desenvolvimento e operações [6].

O interesse em *DevOps* é crescente. O *State of DevOps Report* aponta que, após mais de 27000 respostas ao *DevOps survey* ao longo dos últimos seis anos, foi possível concluir que a utilização das práticas *DevOps* possibilita uma maior performance de TI e que *DevOps* contribui para melhorias nos ciclos de entrega de *software*, na qualidade e segurança de *software* e na capacidade de se obter *feedback* rápido sobre o desenvolvimento de produtos. Ademais, *DevOps* contribui para o atingimento das missões de qualquer tipo de organização, independente da indústria ou setor [7].

Já os resultados do *agile survey*, realizado entre entre agosto e dezembro de 2017, publicados no *State of Agile Report*, apontam que 71% dos 1492 respondentes ou estão implementando *DevOps* ou planejando adotar nos próximos doze meses [8].

Alinhado a esse contexto, o Tribunal de Contas da União (TCU) está buscando ampliar o uso da abordagem *DevOps* no desenvolvimento de suas aplicações corporativas. O entendimento é de que os rígidos procedimentos criados como paliativo para o problema da baixa colaboração entre os times de desenvolvimento e operações ocasionam atrasos na entrega de *software* do órgão que são considerados inadequados.

Os times de desenvolvimento e operações do TCU historicamente trabalham em silos. Metodologias ágeis são utilizadas durante o processo de desenvolvimento e a equipe de operações (Serviço de Infraestrutura de Aplicações (SINAP)) participa da produção de *software* apenas em momentos específicos, notadamente quando uma nova infraestrutura precisa ser provida para publicação de algum artefato de *software*, e também na publicação propriamente dita.

Essa estrutura em silos ocasionou alguns conflitos ao longo do tempo. Como exemplos de problemas ocorridos que levaram à manifestação destes conflitos, destacam-se: (1) o não funcionamento em ambiente de produção de um *software* que funciona adequadamente em outros ambientes, como ambiente de desenvolvimento, por exemplo; e (2) o atraso em entregas de *software* consideradas importantes ocasionado pelo não provisionamento tempestivo da infraestrutura necessária.

Como resposta aos conflitos ocorridos ao longo do tempo, o TCU investiu na padronização de procedimentos. O provimento de infraestrutura para publicação de novas aplicações passou a funcionar em conformidade com um Acordo de Nível de Serviço (SLA) que prevê prazos para conclusão por parte do time de operações. No tocante à publicação do *software*, foi criada uma padronização de procedimentos denominada *publicação programada*, que é vista como a maior manifestação atual da falta de colaboração entre os times de desenvolvimento e de operações do TCU. A publicação programada é um processo que prescreve um complexo encadeamento de ações com horários bem definidos visando reduzir o risco de problemas na publicação dos artefatos de *software*. A versão de todas as aplicações desenvolvidas é congelada no início de cada semana em horários preestabelecidos e existe uma grande quantidade de procedimentos formais que cada um dos times deve seguir para que no final da semana possa ocorrer a publicação em ambiente de produção.

A publicação programada e o SLA ajudaram a lidar com alguns dos problemas existentes previamente no TCU, todavia, alguns outros problemas de ordem prática passaram a existir e, por esse motivo, foi incluído o seguinte indicador no Plano Diretor de Tecnologia da Informação (PDTI) do TCU para o biênio 2017/2018: *o aprimoramento do uso de*

práticas ágeis em todas as equipes e ***ampliação do uso da abordagem DevOps***.

O TCU possui diversos comitês envolvidos no processo de desenvolvimento de suas aplicações corporativas. Um desses comitês se chama Comitê Permanente de Arquitetura (CPA) que é constituído por representantes de todas as equipes da área de TI e cuja função é orientar e padronizar as decisões tecnológicas e arquiteturais no âmbito do processo de desenvolvimento de *software*.

Inicialmente, para possibilitar o cumprimento do indicador constante no PDTI, o CPA deliberou no sentido da experimentação e realização de provas de conceito em ferramentas *DevOps*. Após um período de avaliação, o CPA avaliou que, embora houvesse competência técnica interna para implantação de ferramentas, existia uma carência de melhor entendimento a respeito de como se adotar *DevOps*. Formou-se então novo entendimento de que experiências bem sucedidas na adoção de *DevOps* no mercado deveriam ser buscadas através de contato direto e participações em eventos da comunidade de desenvolvimento de *software*. Essas experiências deveriam ser apresentadas internamente no sentido de pautar a adoção de *DevOps* no TCU.

1.1 Problema de Pesquisa

A rigidez procedimental existente no TCU ilustra o quão complexas podem se tornar soluções que visam contornar o problema da falta de colaboração entre os times de desenvolvimento e operações. Neste sentido, a adoção de *DevOps* é tida como necessária e faz parte dos indicadores do PDTI do órgão.

No entanto, a adoção de *DevOps* ainda é uma tarefa desafiadora, porque existe uma infinidade de informações, práticas e ferramentas relacionadas a *DevOps*, mas ainda não está claro como essa rica, porém ainda dispersa, quantidade de informação é organizada e estruturada em cenários reais e tidos como bem sucedidos de adoção de *DevOps*.

1.2 Justificativa

A adoção de *DevOps* no TCU está alinhada aos direcionadores estratégicos da área de TI do órgão e tem possibilitado uma mudança de paradigma necessária para tratar adequadamente os problemas decorrentes da baixa colaboração entre os times de desenvolvimento e operações.

A produção de um modelo para orientar a adoção de *DevOps* com base em experiências de mercado bem sucedidas, com implantação na adoção de *DevOps* no TCU, representa um incremento ao conhecimento existente sobre *DevOps*, com destaque para o melhor

entendimento de *como* se adotar *DevOps* com base no entendimento de profissionais envolvidos em cenários reais.

O intercâmbio de experiências e conhecimentos com praticantes do mercado durante o processo de adoção de *DevOps* no TCU possibilita uma redução das chances de o TCU incorrer em falhas comuns, já superadas em outros contextos.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho tem como objetivo geral investigar a adoção de *DevOps* em cenários reais e caracterizados como bem sucedidos, e utilizar os resultados e as experiências obtidas durante a investigação para fomentar a adoção de *DevOps* no desenvolvimento das aplicações corporativas do TCU.

1.3.2 Objetivos Específicos

Com o intuito de atingir o objetivo geral deste trabalho, os seguintes objetivos específicos foram definidos:

- Identificar as percepções sobre *DevOps* a partir do olhar crítico de profissionais que adotaram técnicas relacionadas em suas instituições;
- Propor um modelo para guiar a adoção de *DevOps* por novos praticantes com base nas percepções obtidas;
- Introduzir o modelo e as experiências obtidas durante a sua construção na adoção de *DevOps* no TCU; e
- Realizar um estudo empírico para avaliar o estágio atual da adoção de *DevOps* no TCU bem como para validar a relevância do modelo proposto neste contexto.

1.4 Metodologia

Para investigar a adoção bem sucedida de *DevOps* em cenários reais da prática de mercado foi utilizada a variação clássica do método *Grounded Theory* [9]. O uso de *Grounded Theory* se justifica por quatro motivos:

1. O método possibilita a construção de um entendimento independente e original, o que é adequado para coletar evidência empírica diretamente da prática de mercado

sem o viés de pesquisas anteriores, alinhado às necessidades do TCU. A evidência coletada só é reintegrada com a literatura existente após a construção da teoria [10, 11].

2. GT é uma metodologia consolidada em outras áreas de pesquisa, como sociologia médica [12], nutrição [13], educação [14] e administração [15, 16].

3. GT é considerado um método adequado para caracterizar cenários sob uma perspectiva pessoal daqueles envolvidos em uma disciplina ou atividade [11], que é exatamente o cenário que se almeja caracterizar aqui: quais são os caminhos seguidos por praticantes que adotaram *DevOps* de uma maneira bem sucedida?

4. GT tem sido cada vez mais aplicada para estudar tópicos de engenharia de *software* [17, 18, 11], com destaque para o recente trabalho de Hoda et al. [17] que caracterizou a adoção de métodos ágeis em uma maneira similar ao que se propõe fazer aqui em relação a *DevOps*.

O modelo para adoção de *DevOps* foi produzido com base na teoria que foi construída utilizando o método *Grounded Theory*.

Já a avaliação empírica foi realizada utilizando um grupo focal, pois este método é considerado adequado para obter novos *insights* sobre um assunto de um pequeno grupo de pessoas nele envolvido, de maneira rápida, ao invés de se buscar fornecer respostas quantificáveis a perguntas específicas obtidas a partir de uma amostra grande da população [19, 20].

1.5 Estrutura do Trabalho

O restante deste documento está organizado em quatro capítulos cujos conteúdos foram estruturados da seguinte forma:

- **Capítulo 2.** Introduz o tema *DevOps* por meio da apresentação de seu histórico, definições, elementos que o caracterizam e desafios relacionados;

- **Capítulo 3.** Apresenta o detalhamento da pesquisa utilizando *Grounded Theory (GT)* que possibilitou a caracterização da adoção de *DevOps* com base na percepção de praticantes do mercado;

- **Capítulo 4.** Contém detalhes e resultados da avaliação empírica a respeito da adoção de *DevOps* e uso do modelo no TCU;

- **Capítulo 5.** Contém as considerações finais do trabalho com designação das suas limitações bem como de oportunidades para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta uma revisão dos principais tópicos relacionados a *DevOps*, estando estruturado da seguinte forma: a Seção 2.1 apresenta a maneira como o desenvolvimento de *software* tipicamente é organizado quando não se utiliza *DevOps* como abordagem; a Seção 2.2 contém um breve histórico do surgimento do termo *DevOps*; na Seção 2.3 são apresentadas as principais definições de *DevOps* existentes na literatura. A Seção 2.4 contém um detalhamento dos principais elementos que fazem parte das caracterizações de *DevOps*. E, por fim, na Seção 2.5 são apresentados desafios relacionados a *DevOps*.

2.1 Abordagem "Não *DevOps*" de Desenvolvimento de *Software*

A questão de como o desenvolvimento de *software* deve ser organizado visando entregar soluções com maior velocidade, melhor qualidade e menor custo vem sendo discutida ao longo de décadas nos círculos de engenharia de *software* [21]. Entre as abordagens de desenvolvimento de *software* consideradas tradicionais, existe o modelo cascata, que divide o ciclo de vida de desenvolvimento de *software* em estágios distintos e lineares e que possui algumas desvantagens, tais como inflexibilidade para mudanças de requisitos, e um processo altamente cerimonioso, que tipicamente despreza a natureza e o tamanho dos projetos e as características individuais das pessoas envolvidas [22].

Diante da pressão por desenvolvimento acelerado de novos produtos e a contrastante exigência de que esses produtos sejam cada vez mais confiáveis, as organizações têm buscado novas maneiras de melhorar seu processo de desenvolvimento de *software* para acompanhar as demandas de negócios e de mercado.

Entre as formas de se trabalhar para desenvolver *software* em ritmo acelerado e de maneira confiável, os métodos ágeis de desenvolvimento de *software* emergiram como

alternativa aos modelos tradicionais (p.ex.: cascata) com a proposta de desburocratizar o processo, dispensando tudo que não seja essencial. Uma das consequências do uso de metodologias ágeis é a redução dos silos presentes no processo de desenvolvimento. As atividades que outrora eram executadas de maneira linear passaram a ser unificadas em um processo iterativo e contínuo de desenvolvimento [23].

Embora as metodologias ágeis tenham possibilitado uma aproximação das atividades de desenvolvimento de *software*, que passaram a ser executadas de maneira iterativa e contínua, em um primeiro momento as atividades de operações ficaram de fora do contexto de práticas ágeis, gerando um gargalo na entrega de valor do *software* [24].

Neste cenário, a equipe de operações tipicamente participa do processo apenas em um último momento, que é o de disponibilizar o *software* em ambiente de produção. O time de desenvolvimento realiza diversas iterações, produz diversas versões intermediárias dos produtos de *software*, e o time de operações não tem participação ativa neste momento. Segundo Hüttermann [3], essa abordagem gera uma barreira organizacional e cultural que desencadeia três principais tipos de problemas: (1) cada time (desenvolvimento e operações) defende seus próprios interesses, ao invés de um interesse comum; (2) desenvolvimento de idiomas/vocabulários distintos pelos dois times, que trabalham em problemas isoladamente, sem a busca de um objetivo comum; e (3) não compartilhamento de conhecimento entre os times, por falta de confiança e receio de confrontamento ou de perda de poder.

Um outro problema típico dessa abordagem "não *DevOps*" é a existência de um *blame game* onde, na ocorrência de um problema, um lado acusa o outro de tê-lo causado e o foco deixa de ser a resolução do problema em si e passa a ser apenas em evitar a responsabilização [25].

2.2 A História de *DevOps*

O termo *DevOps* foi cunhado em 2009 na Bélgica [3]. Todavia, as origens de *DevOps* remetem para antes mesmo da existência do termo. Primeiro, em 2008, foi publicado o artigo "*Agile Infrastructure and Operations: How Infra-gile are You?*" [4] no qual Patrick Debois propôs analogias entre o sucesso com o qual os métodos ágeis se adaptam às constantes mudanças das necessidades de negócio e a maneira como a infraestrutura também poderia responder de maneira ágil à essas mudanças. Desafiando um suposto oxímoro no qual agilidade e infraestrutura não se encaixam, o autor relata experiências bem sucedidas da aplicação de técnicas ágeis na construção e gerenciamento da infraestrutura de três projetos reais. Na mesma conferência onde Debois apresentou suas ideias de aplicar técnicas ágeis a contextos de infraestrutura, Andrew Shafer propôs uma sessão

sobre infraestrutura ágil, momento em que os dois começaram a interagir a respeito do tema [26].

Pouco tempo depois, já em 2009, Hammond e Allspaw apresentaram "*10+ Deploys per Day: Dev and Ops Cooperation at Flickr*" [5] em uma conferência chamada *Velocity*, ilustrando não somente o alto número de *deploys* realizados na *Flickr*, mas também o poder revolucionário que a colaboração entre os times de desenvolvimento e operações demonstrou por lá.

E foi neste momento que o termo *DevOps* finalmente surgiu. Segundo J. Davis e K. Daniels [26], em um *tweet* de Andrew Shaffer convidando para a próxima edição da conferência *Velocity*, que iria se concentrar em infraestrutura ágil, continha a *hashtag* "*#devops*". Ao ler este *tweet*, Patrick Debois lamentou que não iria poder estar presente, pois estaria na Bélgica. Shaffer então teria sugerido a Debois que organizasse sua própria edição da conferência *Velocity*. Foi quando Debois, utilizando o texto da *hashtag*, organizou a primeira edição do evento *DevOpsDays* na Bélgica, iniciando a disseminação do termo *DevOps*.

2.3 O que é *DevOps*?

DevOps emergiu na indústria de desenvolvimento de *software* sem uma clara definição ou delimitação teórica.

Em *What is DevOps?* [27], Loukides tenta ilustrar o conceito de *DevOps* por meio da diferenciação entre o passado e o presente do trabalho de operações de TI. O texto ilustra os novos desafios existentes para os times de operações e destaca a importância da existência de colaboração com os times de desenvolvimento para superá-los. Todavia, não é apresentada uma definição final de *DevOps*.

Hüttermann afirma que "*DevOps* descreve práticas que simplificam o processo de entrega de *software*, enfatizando o aprendizado por meio da transmissão contínua de *feedback* da produção para o desenvolvimento". Quando comparada com outras, essa definição de *DevOps* como um conjunto de práticas relacionadas à entrega de *software*, demonstra carecer de uma maior ênfase no aspecto cultural. Por exemplo, de acordo com Walls [28], o termo foi introduzido justamente para definir uma cultura organizacional que as organizações podem buscar. De acordo com ele, *DevOps* é "um movimento cultural combinado com várias práticas relacionadas a *software* que possibilitam um desenvolvimento rápido dos produtos". O autor descreve quatro características culturais chave em um contexto de *DevOps*: comunicação aberta, incentivo e alinhamento de responsabilidades, respeito e confiança.

O aspecto cultural de *DevOps* é também ressaltado por Davis e Daniels [26] que o definem como um movimento cultural que altera como os indivíduos pensam sobre seu trabalho, valoriza a diversidade, apóia processos que aceleram o ritmo da obtenção de valor e medem o efeito da mudança social e técnica.

Como ressaltado por Hüttermann [3], *DevOps* é um termo multifacetado e de difícil definição. A seguir, são apresentadas três definições de *DevOps* obtidas em estudos que dedicaram-se a investigar essa questão em específico.

Por meio de uma revisão de literatura, o trabalho de J. Smeds et al. [29] define *DevOps* como um conjunto de capacidades do processo de engenharia de *software* que é sustentado por certos facilitadores culturais e tecnológicos. Os autores enfatizam que as *capacidades* definem processos que uma organização deve ser capaz de executar, enquanto que os *facilitadores* possibilitam uma maneira fluente, flexível e eficiente de se trabalhar.

Dyck et al. [30], por sua vez, definem *DevOps* como uma abordagem organizacional que enfatiza a empatia e a colaboração interfuncional dentro e entre as equipes - especialmente o desenvolvimento e operações de TI - em organizações de desenvolvimento de *software*, para operar sistemas resilientes e acelerar a entrega de mudanças.

Por fim, França et al. [6] definem *DevOps* como um neologismo que representa um movimento de profissionais de tecnologia da informação e comunicação buscando uma diferente atitude no que se refere à entrega de *software*, através da colaboração entre o desenvolvimento de sistemas de *software* e as funções de operações, com base em um conjunto de práticas e princípios, tais como cultura, automação, medição e compartilhamento.

2.4 Elementos de *DevOps*

As caracterizações de *DevOps* tipicamente utilizam enumerações e detalhamentos de elementos relacionados. Nesta seção são apresentados os principais elementos de *DevOps* identificados na literatura relacionada. O termo "elemento" é aqui usado para unificar o vocabulário da literatura relacionada no qual constam termos distintos, como "conceitos", "dimensões" e "princípios" de *DevOps*.

Em 2010, John Willis propôs uma caracterização de *DevOps* [31] que se tornou influente. De acordo com esta proposta, *DevOps* é constituído de quatro elementos cujas iniciais compõem o acrônimo CAMS: cultura, automação, medição e compartilhamento (*sharing*, em inglês). Esse modelo é conhecido como CAMS *Framework*.

O trabalho de Hamunen [32], após investigar quais são os componentes-chave de *DevOps*, identificou que diversas fontes adicionaram *Lean* como novo componente, culminando na formação do que foi chamado de CALMS *Framework*.

Uma revisão de literatura de Erich et al. [33] identificou oito conceitos principais relacionados a *DevOps*, os quatro presentes no CAMS *framework* acrescidos de (1) serviços, (2) garantia da qualidade, (3) estruturas e (4) padrões.

Posteriormente, Lwakatare et al. [34] apresentaram uma caracterização de *DevOps* como um *framework* conceitual que contém quatro dimensões: colaboração, automação, medição e **monitoramento**, este último sendo o único elemento novo, em comparação com os estudos anteriores. Para identificar as dimensões de *DevOps* foi utilizada uma revisão de literatura combinada com entrevistas a quatro praticantes de três companhias de um mesmo grupo empresarial. Uma extensão deste trabalho foi publicada posteriormente em [35], onde foi acrescentada uma nova dimensão chamada de *colaboração*. Nessa extensão foi conduzida uma revisão de literatura multivocal, utilizando dados da literatura cinza.

Em *Characterizing DevOps by Hearing Multiple Voices*, França et al. [6] organizam os elementos de *DevOps* em seis categorias de princípios: (1) automação, (2) garantia da qualidade, (3) compartilhamento, (4) medição, (5) **aspectos sociais** e (6) ***leanness***. Apenas a categoria *aspectos sociais* representa um elemento novo em relação aos já apresentados aqui. Para chegar a estes resultados os autores conduziram uma revisão de literatura multivocal, e os dados foram analisados usando técnicas de codificação de *Grounded Theory (GT)*.

Por fim, por meio de uma nova revisão sistemática de literatura, Erich et al. [36] identificaram sete agrupamentos de conceitos presentes na literatura sobre *DevOps*: (1) cultura de colaboração, (2) automação, (3) medição, (4) compartilhamento, (5) serviços, (6) garantia da qualidade e (7) **governança**. Apenas *governança* não aparece em nenhum dos trabalhos apontados previamente.

É possível identificar elementos de *DevOps* que aparecem em vários trabalhos bem como outros que aparecem apenas em algum deles, a Tabela 2.1 lista os elementos identificados e aponta em quais trabalhos cada um deles aparece. Após comparação dos estudos, foi identificado que os elementos *cultura*, *cultura de colaboração*, *colaboração* e *aspectos sociais* são similares e, para efeitos de comparação, foi designada uma única linha na tabela, a linha 1 (cultura de colaboração).

A Tabela 2.1 reforça a relevância dos elementos do CAMS *framework*. Apenas *compartilhamento* deixa de aparecer em um dos estudos citados, os demais aparecem em todos. Com relação aos elementos que não fazem parte do *framework*, *garantia da qualidade* aparece em três estudos, *serviços*, e *leanness* aparecem em dois, enquanto que os demais aparecem apenas em um único estudo.

Nas subseções a seguir, cada um dos elementos é detalhado. Alguns dos trabalhos relacionados citam práticas relacionadas aos elementos. Quando aplicável, as práticas

também são listadas. Os elementos *estruturas* e *padrões* foram citados apenas em [33] e não foi apresentado um detalhamento sobre suas características. O mesmo grupo de autores retirou esses elementos dos resultados apresentados a partir de uma nova revisão de literatura realizada em [36]. Assim sendo, esses elementos não fazem parte do detalhamento apresentado a seguir.

2.4.1 Cultura de Colaboração

Uma parte essencial de *DevOps* é a mudança de cultura organizacional de uma coleção de silos para uma forma de trabalho colaborativa. Isso implica envolver o pessoal de operações no processo de projeto e transição de uma aplicação. A cultura *DevOps* desfoca a linha entre os papéis dos desenvolvedores e da equipe de operações. Em alguns casos, esse desfoque elimina totalmente a distinção [32].

No CAMS *framework* [31], este elemento é chamado apenas de *cultura* e é apresentado de maneira sucinta como um direcionamento de que pessoas e processos devem vir primeiro, sob pena de os demais esforços relacionados a *DevOps* serem infrutíferos.

Já no trabalho de Lwakatare et al. [34, 35], este elemento é apresentado primeiramente apenas como *colaboração*, e em um segundo momento como duas dimensões distintas: *cultura* e *colaboração*. Ao tratar do elemento como *colaboração*, os autores mencionam especificamente que *DevOps* envolve uma *cultura de colaboração*. Neste trabalho não é apresentada uma caracterização do que seria essa *cultura de colaboração*, apenas é posto

Tabela 2.1: Elementos de *DevOps*

	Elemento	CAMS *Framework* [31]	Hamunen [32]	Erich et al. [33]	Lwakatare et al. [35]	França et al. [6]	Erich et al. [36]
1	Cultura de Colaboração	X	X	X	X	X	X
2	Automação	X	X	X	X	X	X
3	Medição	X	X	X	X	X	X
4	Compartilhamento	X	X	X		X	X
5	Serviços			X			X
6	Garantia da Qualidade			X		X	X
7	Estruturas			X			
8	Padrões			X			
9	Monitoramento				X		
10	*Leanness*		X			X	
11	Governança						X

que ela é reforçada por meio de compartilhamento de informações, ampliação de qualificações e transferência de responsabilidades entre as duas equipes, além de incutir um senso de responsabilidade compartilhada. Após o acréscimo de *cultura* como elemento *DevOps*, não é também apresentada uma diferenciação entre essas duas dimensões que são apresentadas separadamente. Sobre *cultura*, apenas é dito que muitas das práticas *DevOps* envolvem uma mudança de cultura e mentalidade para incentivar a empatia, o apoio mútuo e um bom ambiente de trabalho para os envolvidos no desenvolvimento de *software* e nos processos de entrega.

No trabalho de França et al. [6], este elemento é apresentado como um princípio denominado *aspectos sociais*. De acordo com os autores, apesar de existirem diversos princípios técnicos, muitas das características de *DevOps* estão associadas a aspectos sociais entre as equipes de desenvolvimento de *software* e de operações. A **cultura *DevOps*** reconhece a confiança como uma característica relevante para influenciar a mudança organizacional exigida por *DevOps*.

Erich et al. [36] limitam-se a afirmar que organizações que praticam *DevOps* tentam remover a barreira cultural entre o pessoal de desenvolvimento e de operações.

Em síntese, o elemento *cultura de colaboração*, consiste na existência de uma maneira de se organizar as atividades de construção de *software* que reforça a colaboração entre os times de desenvolvimento e operações, buscando-se evitar exatamente que se organizem em silos com comunicação burocrática, sem confiança mútua e buscando objetivos distintos.

Práticas Relacionadas

Como práticas relacionadas ao elemento *cultura de colaboração*, são apresentadas as seis seguintes em [35]:

1. Aumentar o escopo de responsabilidades;
2. Intensificar a cooperação e o envolvimento no trabalho diário de cada um dos times;
3. Tanto desenvolvedores como operadores são considerados responsáveis para lidar com incidentes;
4. Integrar o desenvolvimento em pós-mortem de produção;
5. Tornar a comunicação entre desenvolvimento e operações menos formal e sem adversidades;
6. Respeito mútuo, apoio e vontade de trabalhar juntos e compartilhamento de responsabilidades.

Enquanto que em [6], são apresentadas mais oito práticas: (1) avaliação coletiva de desempenho, (2) cultura de confiança, (3) comunicação efetiva, (4) aprendizagem mútua, (5) abertura para mudanças, (6) responsabilidade pessoal, (7) relevância de aspectos culturais e (8) respeito entre os membros dos times.

2.4.2 Automação

Organizações praticando *DevOps* almejam ter um maior nível de automação [36]. O objetivo por trás das práticas de automação em *DevOps* está em alcançar *feedback* rápido e baixos tempos de espera [32].

No CAMS *framework* [31], é explicado que as ferramentas *DevOps* possibilitam uma ampla gama de possibilidades para automação. Ferramentas para gerenciamento de versões, provisionamento, gerenciamento de configuração, integração de sistemas, monitoramento e controle e orquestração são tratadas como peças importantes na construção de uma estrutura *DevOps*.

Para acompanhar o ritmo ágil das práticas de desenvolvimento de *software*, os processos de operações precisam ser flexíveis, repetíveis e rápidos, e a automação é vista como um meio para viabilizar isso. Em ambientes complexos, é difícil e demorado implantar funcionalidades e gerenciar configurações de infraestrutura de *software* manualmente. Além disso, existe automação de testes visando garantir a qualidade das funcionalidades implantadas [34, 35].

França et al. [6] acrescentam que automação é um dos princípios fundamentais de *DevOps* pois reduz os esforços repetitivos e acelera a entrega de *software*, melhorando não só a velocidade de entrega, como também a consistência da infraestrutura, a produtividade das equipes e a repetibilidade das tarefas.

Práticas Relacionadas

Como práticas relacionadas a automação, têm-se (1) infraestrutura como código e (2) automação do processo de *deployment*, apresentadas em [35], enquanto que em [6] são apresentadas as seguintes: (1) aceleração por meio de automação, (2) automatização de tarefas repetitivas, (3) automatização da coleta de métricas, (4) automatização da infraestrutura e (5) consistência da infraestrutura.

2.4.3 Medição

Organizações que praticam *DevOps* tentam usar métricas que envolvem disciplinas de desenvolvimento e operações, ao invés de métricas separadas para cada um dos times [36]. As métricas são importantes para que os times tenham uma clara visão do que

está acontecendo com as aplicações em qualquer momento do tempo e para auxiliar na descoberta de gargalos no processo [32].

O processo de melhoria contínua requer a coleta de métricas, uma implementação bem sucedida de *DevOps* irá medir tudo o que for possível com a maior frequência possível, há a coleta de métricas de desempenho, de processo e até mesmo de pessoas [31].

A *medição* em *DevOps* é obtida quando os esforços vão além do controle básico de qualidade, há medições usando dados de desempenho e uso em tempo real das funcionalidades do *software* em ambiente de produção. Com uma *medição* efetiva, os esforços de desenvolvimento de *software* são efetivamente medidos [34]. *DevOps* envolve a coleta de métricas eficientes para apoiar a tomada de decisões no ciclo de vida de desenvolvimento e operações de *software* [6].

Práticas Relacionadas

As quatro práticas a seguir são apresentadas em [35]:

1. As equipes de operações e de desenvolvimento são incentivadas e recompensadas pelas mesmas métricas;
2. Tanto desenvolvimento quanto operações se concentram no valor de negócio como a unidade essencial de medição;
3. O progresso no desenvolvimento é medido em termos de sistema funcionando em ambiente de produção;
4. Os desenvolvedores usam *feedback* de produção para orientar decisões, melhorias e alterações nos sistemas.

Enquanto que em [6] são apresentadas as quatro seguintes práticas: (1) medição para entender o código e o ambiente, (2) monitoramento do negócio por meio de métricas, (3) monitoramento de operações e (4) monitoramento de equipes por meio de métricas.

2.4.4 Compartilhamento

Segundo Hamunen [32], uma cultura transparente com mecanismos eficazes de disseminação de conhecimento entre as equipes é essencial para derrubar as barreiras entre desenvolvimento e operações. As organizações que praticam *DevOps* tentam facilitar o desenvolvimento e as operações por meio do uso de práticas para compartilhar o conhecimento [36].

O compartilhamento é considerado o *loopback* no ciclo do CAMS *framework*. Segundo Willis, criar uma cultura onde as pessoas compartilham ideias e problemas é fundamental

pois isso gera um cenário de *feedback* aberto que no final possibilita melhorias na cultura [31].

DevOps envolve cenários onde a informação e o conhecimento são disseminados entre os indivíduos para promover o intercâmbio de aprendizado pessoal e informações sobre projetos. Nesse sentido, os indivíduos divulgam informações relevantes, por exemplo, sobre como implementar e executar práticas recomendadas. Além disso, as informações sobre mudanças ou novas características de produtos devem ser distribuídas entre os envolvidos tanto da equipe de desenvolvimento quanto da de operações e isso promove a visibilidade do fluxo de trabalho [6].

Práticas Relacionadas

Como práticas relacionadas a *compartilhamento*, apenas são apresentadas as práticas (1) compartilhamento do aprendizado pessoal e (2) compartilhamento de informações de projeto, em [6].

2.4.5 Serviços

Este elemento está presente em [33] e em [36]. Neste último estudo, é explicado que considerar *serviços* como elemento *DevOps* consiste em considerar uma característica das organizações que se estruturam em torno de serviços, ao invés de em torno de disciplinas (como desenvolvimento e operações, por exemplo) e que organizações que praticam *DevOps* tipicamente usam serviços em nuvem e a arquitetura de microsserviços.

2.4.6 Garantia da Qualidade

As organizações que praticam *DevOps* tentam incorporar o controle de qualidade ao processo de construção de *software*, visando garantir que os produtos e serviços possuam uma qualidade adequada [36].

A busca por *garantia da qualidade* suporta a implementação de práticas *DevOps* uma vez que vincula diferentes partes interessadas (desenvolvimento, operações, suporte e clientes) para realizar atividades de forma eficiente e confiável. Ademais, representa o esforço para garantir que os produtos e serviços atendam aos padrões de qualidade estabelecidos [6].

Práticas Relacionadas

Em [6] é apresentada uma única prática que possui pouca expressividade e detalhamento: *preocupações de garantia de qualidade*.

2.4.7 Monitoramento

A proposta de *monitoramento* como elemento *DevOps* foi indicada como uma das principais contribuições do primeiro trabalho de Lwakatare et al. [34] e mantida na extensão deste trabalho, publicada em [35]. Segundo os autores, a equipe de operações é tipicamente responsável por monitorar os sistemas e a infraestrutura subjacente para determinar a atribuição apropriada de recursos e para detectar, relatar e corrigir os problemas que ocorrem nos sistemas. Como contraponto, *DevOps* propicia uma resposta aos desafios enfrentados na realização de um monitoramento efetivo, por meio da ênfase em colaboração entre desenvolvedores e operadores. Adicionalmente, as análises são usadas para integrar os dados de desempenho da infraestrutura e do sistema com o comportamento de uso do cliente. As informações coletadas são fornecidas como *feedback* aos desenvolvedores e ao gerenciamento de produtos para serem utilizadas em aprimoramentos e personalização dos produtos de *software*.

Convém ressaltar, que as descrições contidas nos demais trabalhos para o elemento *medição*, tipicamente englobam aspectos de *monitoramento*. Por exemplo, entre as práticas apontadas em [6] estão *monitoramento de operações* e *monitoramento de equipes por meio de métricas*. Essa separação de *monitoramento* como um elemento autônomo em relação a *medição* apenas foi realizada por Lwakatare et al.

2.4.8 *Leanness*

De acordo com França et al. [6], algumas práticas *DevOps* são baseadas nos princípios de *pensamento enxuto*[1]. Considerando que *DevOps* visa garantir um fluxo contínuo para desenvolver e entregar *software* regularmente, em pequenas e incrementadas mudanças, o processo precisa ser enxuto (*lean*).

Já Hamunen [32], cita a redução de desperdícios, a limitação do trabalho em andamento e a redução de tamanho das entregas para ilustrar os motivos de as metodologias "*Lean*" serem um importante elemento de *DevOps*.

Práticas Relacionadas

França et al. [6] apresentam seis práticas relacionadas ao elemento *Leanness*: (1) fluxo contínuo, (2) melhoria contínua, (3) eliminação de desperdícios, (4) *feedback* rápido, (5) visão holística ou sistêmica e (6) simplicidade.

[1]Lean Thinking [37]

2.4.9 Governança

Este elemento foi apontado apenas no trabalho de Erich et al. [36] no qual é posto que se refere ao modo como as organizações que praticam *DevOps* são dirigidas. Foi apontado que a integração de *DevOps* com padrões como os presentes na *Information Technology Infrastructure Library (ITIL)* é uma questão relevante, todavia, as conclusões do estudo apontam que este é um aspecto implícito de como as organizações gerenciam o pessoal, as equipes e os departamentos.

2.5 Desafios na Adoção de *DevOps*

Implementar *DevOps* é reconhecidamente uma atividade desafiadora. Nesta seção são apresentados os principais desafios relatados na literatura relacionada a respeito da adoção de *DevOps*.

Riungu-Kalliosaari et al. [38] apontam quatro desafios relacionados à adoção de *DevOps*. O primeiro é a existência de *comunicação insuficiente* entre os times de desenvolvimento e operações. Em segundo lugar, como *DevOps* requer uma mudança cultural, existe o desafio da existência de *culturas profundamente arraigadas*. O terceiro desafio apontado é a possibilidade de *restrições legais* relacionadas, por exemplo, ao acesso, para execução de testes, de dados protegidos. Por fim, o quarto desafio é que *DevOps* ainda não é completamente compreendido mas continua evoluindo rapidamente. Este quarto desafio é endossado por Smeds et al. [29], que afirmam que a definição e os objetivos de se adotar *DevOps* não são claros, e também por Hamunen [32] que observou uma falta de maturidade para o conceito de *DevOps*.

Mais dois desafios foram identificados tanto em [29] quanto em [32]: *DevOps* ser visto como uma *buzzword* e a distribuição geográfica dos times.

Adicionalmente, Hamunen [32] aponta que a falta de suporte tanto a nível de gerência quanto a nível de time, bem como a falta de confiança e as dificuldades para implementar tecnologias *DevOps* e para adaptar os processos organizacionais são os outros principais agrupamentos de desafios relacionados a *DevOps*.

Enquanto que Smeds et al. [29] apontam ainda como desafios para *DevOps*: dificuldades relacionadas a estruturas organizacionais não compatíveis, onde clientes não vêem valor em *DevOps*; falta de interesse de algum dos times; e dificuldades técnicas como a existência de múltiplos ambientes de produção e o fato de ambientes como desenvolvimento e teste não refletirem o ambiente de produção.

Capítulo 3

Uma *Grounded Theory* Sobre a Adoção de *DevOps* no Mercado

Neste capítulo, apresenta-se um estudo utilizando a variação clássica da metodologia *Grounded Theory (GT)* [9] para se caracterizar a adoção de *DevOps* em organizações de mercado que foram bem sucedidas neste processo.

A teoria aqui apresentada foi construída com base na percepção de praticantes de quinze companhias de cinco países que foram bem sucedidas na adoção de *DevOps*. A teoria torna claro que praticantes interessados em adotar *DevOps* devem focar na construção de uma ***cultura de colaboração***, o que previne pontos de falha comuns relacionados a focar em ferramental ou em automação. O papel dos demais elementos que fazem parte da adoção de *DevOps* é explicado por meio da definição de dois grandes agrupamentos de categorias de conceitos, denominados *Facilitadores DevOps* e *Saídas DevOps*.

3.1 *Grounded Theory* (GT)

Grounded Theory (GT) é um método de pesquisa, desenvolvido originalmente pelos sociólogos B. Glaser e A. Strauss, que possibilita a geração sistemática de uma teoria a partir de dados analisados em um rigoroso processo [9]. O objetivo de um estudo em GT é entender como se dá a ação em uma determinada área de conhecimento a partir do ponto de vista dos atores envolvidos [39].

Ressalte-se que, de acordo com Strauss e Corbin [40], no âmbito de *Grounded Theory*, uma teoria é "um conjunto bem desenvolvido de categorias (incluindo conceitos e temas) que são sistematicamente inter-relacionadas através de **sentenças de relacionamentos** para formar um *framework* teórico que explica algum fenômeno relevante".

Hoda et al. [17] explicam que um estudo utilizando *Grounded Theory* é mais que apenas um conjunto de categorias. Ele deve descrever os principais relacionamentos entre

as categorias, e que essas "sentenças de relacionamentos" são aqui chamadas de hipóteses, sem prejuízo para o uso deste mesmo termo com significado distinto em abordagens que usam métodos estatísticos para confirmar ou refutar hipóteses pré-estabelecidas.

Segundo Locke [16], o diferencial de *Grounded Theory* é seu compromisso com a "descoberta" através do contato direto com o mundo social de interesse, juntamente com uma limitação, a priori, do contato com a teorização existente. Os estudos utilizando GT não começam com uma estrutura conceitual, ao invés disso pretendem culminar em uma [41]. São estudos em que uma cobertura aprofundada da literatura é deliberadamente adiada até que surjam as direções da análise de dados coletados diretamente na fonte de interesse [42]. Stol et al. [11] explicam que a principal razão para se limitar a exposição à literatura é previnir que o pesquisador se concentre em pensar em termos de conceitos pré-estabelecidos, testando teorias existentes, ao invés de desenvolver uma. De fato, GT é um método de desenvolvimento de teoria, e não de teste de teoria. Jantunen e Gause [43], utilizando a classificação de abordagens de pesquisa proposta por Jarvinen [44], apontam que GT é uma abordagem de desenvolvimento de teoria do grupo estudos empíricos da realidade, conforme a área em cinza da Figura 3.1.

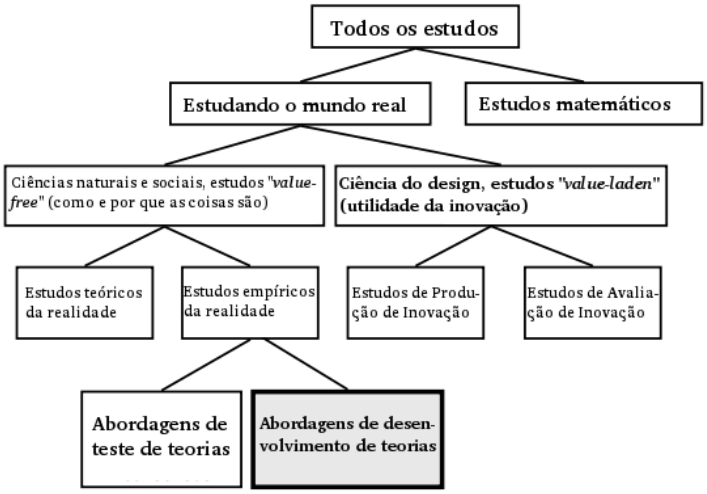

Figura 3.1: Classificação de GT nas Abordagens de Pesquisa (Fonte: [43]).

No entanto, conforme destaca Glaser [45], essa postura de limitar a exposição à literatura é parte da abordagem apenas no início da pesquisa. Quando a teoria proposta parece suficientemente amadurecida, então o pesquisador pode começar a revisar a literatura no campo substantivo e relacioná-la com seu próprio trabalho.

3.1.1 Variações de GT

Desde a publicação da versão original de GT, diversas modificações e variações foram propostas ao método, ocasionando a existência de pelo menos sete diferentes versões de *Grounded Theory* (GT) [46]. Segundo Stol et al. [11], as principais versões são três: (1) GT de *Glaser* (GT clássica ou *Glaseriana*), (2) GT de *Strauss* e *Corbin* (GT *Straussiana*) e (3) GT de *Charmaz* (GT construtivista), e qualquer estudo utilizando GT deve especificar qual versão foi utilizada.

As principais diferenças entre as versões de GT estão relacionadas à maneira de utilizar questões de pesquisa, ao papel da literatura, às técnicas de codificação, aos tipos de questionamentos que devem ser levantadas durante a análise dos dados e aos critérios de avaliação da teoria construída. A Tabela 3.1, adaptada de [11] sintetiza as principais diferenças entre as três principais versões.

Tabela 3.1: Principais diferenças entre as versões de GT

Elemento	**Clássica**	***Straussiana***	**Construtivista**
Questão de pesquisa	Não é definida *a priori*, emerge da pesquisa. O pesquisador inicia com uma área de interesse.	Pode ser definida antecipadamente, derivada da literatura ou sugerida por um colega, é frequentemente ampla e aberta.	A pesquisa inicia com questões de pesquisa iniciais, que evoluem ao longo do estudo.
Papel da literatura	Uma revisão abrangente da literatura só deve ser conduzida após a literatura possuir estágio avançado de desenvolvimento, para evitar a influência de conceitos existentes na teoria emergente.	A literatura pode ser consultada ao longo do processo para algumas necessidades específicas, como na formualação de perguntas para coleta de dados, para obter sugestões de áreas para exemplificação teórica, etc.	Embora concorde com os motivos de *Glaser* recomendar evitar contato com a literatura no início da pesquisa, defende uma consulta adaptada da literatura para adequar o propósito do estudo.
Técnicas de codificação	*Open coding*, *selective coding* e *theoretical coding*.	*Open coding*, *axial coding* e *selective coding*.	*Initial coding*, *focused coding* e *theoretical coding*.
Questões levantadas durante a análise	Estes dados são um estudo de quê? Que categoria ou propriedade este incidente indica? O que está acontecendo atualmente nos dados?	Questões do estilo quem, quando, onde, como, com quais consequências ou sob quais condições algum fenômeno ocorreu, ajudam a descobri ideias importantes para a teoria.	Estes dados são um estudo de quê? O que este dado sugere? Sob o ponto de vista de quem? A qual categoria teórica esta unidade de dado indica pertencer?
Critérios de avaliação	As categorias geradas devem estar alinhadas aos dados, a teoria deve funcionar, a teoria deve ter relevância para a ação da área, e a teoria deve ser modificável quando novos dados aparecerem.	São definidos sete critérios para o processo de pesquisa, tais quais, informação na seleção de exemplos, principais categorias, hipóteses derivadas e discrepâncias. E oito critérios a respeito do crescimento empírico, tais quais "conceitos foram gerados?", "existe variação construída na teoria?"	Credibilidade, originalidade, utilidade e se a teoria faz sentido para os participantes.

Neste trabalho foi utilizada a variação clássica, principalmente pela inexistência de questões de pesquisa no início, exatamente como sugerido nessa versão. O início da pesquisa estava ancorado em investigar uma área de interesse: a adoção de *DevOps* em cenários reais e tidos como bem sucedidos. Adicionalmente, os trabalhos de pesquisa atu-

almente existentes na área de engenharia de *software* usam predominantemente a versão clássica [11] e, portanto, existe um maior referencial que pode ser utilizado como base em um novo estudo.

3.1.2 Abordagem GT

A Figura 3.2, adaptada da proposta por Adolph et al. [47], ilustra uma abordagem de como conduzir um estudo utilizando GT como método. Esta abordagem foi utilizada no estudo aqui realizado.

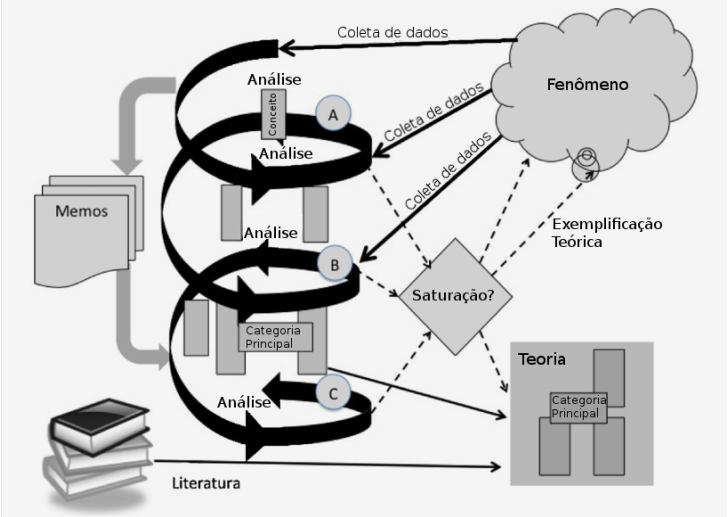

Figura 3.2: O método GT (Fonte: [47]).

(A) Inicialmente, foi observado o fenômeno da adoção de *DevOps* em companhias que o adotaram de maneira bem sucedida. Os dados foram coletados e analisados simultaneamente, em um processo iterativo. O dado bruto foi analisado por meio da busca de padrões de incidentes que indicam *conceitos*, e estes conceitos foram agrupados em *categorias*. Esta primeira etapa onde todo o dado bruto é analisado, gerando novos conceitos e categorias, é chamada de *codificação aberta*.

(B) As categorias vão sendo desenvolvidas por meio da *comparação constante* dos novos incidentes com os prévios. Todo estudo utilizando *grounded theory* deve identificar uma "*categoria principal*" [11]. A categoria principal é responsável por possibilitar a integração das demais categorias e estruturar os resultados em uma teoria densa e consolidada [43]. A identificação da categoria principal representa o fim da *codificação aberta* e o início da *codificação seletiva*. Na codificação seletiva, são codificadas apenas variáveis específicas que são diretamente relacionadas com a categoria principal, visando a produção de uma teoria harmônica [48, 49]. A *codificação seletiva*

termina quando é atingida a *saturação teórica*, o que ocorre quando os novos dados contém mais evidência e exemplos do que já está desenvolvido, mas não contém novos conceitos ou categorias [9].

(C) Após a *saturação teórica* é iniciada uma nova etapa denominada *codificação teórica* cujo propósito é construir uma teoria que explica como conceitos, categorias e relacionamentos se encaixam em uma unidade conceitual. Adicionalmente, a teoria produzida pode ser reintegrada com a literatura no final do processo por meio da comparação de resultados. Ou seja, quando se utiliza *grounded theory*, uma revisão de literatura só deve ser conduzida nos estágios finais da pesquisa, visando evitar influências externas na concepção da teoria [10].

(D) Durante todo o processo, *memos* são escritos com o objetivo de registrar as ideias e pensamentos do pesquisador durante a análise dos dados; os *memos* são consultados para ajudar na descoberta de conceitos, no agrupamento de conceitos em categorias e nas explicações a respeito das relações entre as categorias [10].

3.2 Coleta de Dados

Na coleta de dados, foram conduzidas entrevistas semi-estruturadas com praticantes de quinze companhias do Brasil, Espanha, Estados Unidos, Irlanda e Portugal. Estes praticantes contribuíram para a adoção de *DevOps* em suas companhias.

Os participantes das entrevistas foram recrutados de três maneiras: (1) através de chamadas gerais postadas em grupos de *DevOps* em redes sociais (Facebook, Slack e Telegram); (2) através de mensagens diretas para praticantes com perfil na rede social LinkedIn; e (3) através de contato direto no evento *DevOpsDays* que ocorreu em Brasília em novembro de 2017.

Aos praticantes que demonstravam interesse em participar da entrevista, foram feitos breves questionamentos para assegurar que possuíam participação efetiva em processos consolidados de adoção de *DevOps* em suas respectivas companhias.

A Tabela 3.2 apresenta as características dos participantes que foram entrevistados. Para manter anonimidade, em conformidade com as diretrizes éticas, daqui em diante os participantes serão referenciados como P1–P15 (primeira coluna). Todos os entrevistados são do sexo masculino.

As entrevistas foram conduzidas entre agosto de 2017 e maio de 2018 por meio de chamadas utilizando a ferramenta *skype*. As entrevistas duraram em média 30,93 minutos com duração mínima de 17 minutos, máxima de 50 minutos e mediana de 31 minutos. A realização das entrevistas e a análise dos respectivos dados ocorreram de maneira iterativa,

Tabela 3.2: Perfil dos Participantes

P#	**Cargo**	**SX**	**DX**	**CN**	**Domínio**	**CS**	**Idade**	**Formação**
P1	*DevOps Developer*	9	2	IR	TI	S	29	Graduação
P2	Consultor *DevOps*	9	3	BR	TI	M	35	Pós-graduação
P3	*DevOps* Developer	8	1	IR	TI	S	28	Graduação
P4	Técnico em Computação	10	2	BR	Saúde	S	30	Graduação
P5	*Systems Engineer*	10	3	SP	Telecom	XL	30	Graduação
P6	*Developer*	3	1	PO	TI	S	27	Graduação
P7	Analista de Suporte	15	2	BR	Telecom	L	34	Pós-graduação
P8	*DevOps Engineer*	20	9	BR	Marketing	M	39	Graduação
P9	Gerente de TI	14	8	BR	TI	M	32	Pós-graduação
P10	Administrador de Redes	15	3	BR	IT	S	33	Graduação
P11	Supervisor de *DevOps*	6	4	BR	TI	M	31	Graduação
P12	*Cloud Engineer*	9	3	US	IT	L	27	Técnico
P13	Gerente de Tecnologia	18	6	BR	Food	M	35	Pós-graduação
P14	Gerente de TI	7	2	BR	TI	S	28	Graduação
P15	Desenvolvedor	3	2	BR	TI	S	22	Graduação

SX = anos de experiência com desenvolvimento de *software*;
DX = anos de experiência com *DevOps*;
CN = nome do país (BR=Brasil, IR=Irlanda, SP=Espanha, PO=Portugal e US=Estados Unidos);
CS = tamanho da companhia (S<100; M<1000; L<5000; XL>5000).

de modo que os dados coletados e analisados serviram de insumo para guiar a realização das novas entrevistas por meio de adaptação do roteiro. As questões evoluíram de acordo com o progresso da pesquisa. No início haviam apenas quatro questões *open-ended*: (1) O que motivou a adoção de *DevOps*? (2) O que adotar *DevOps* significa no contexto da sua companhia? (3) Como *DevOps* foi adotado em sua companhia? E (4) Quais foram as principais dificuldades e como foram superadas?

Conforme a análise evoluiu, novas perguntas relacionadas aos conceitos, categorias e relacionamentos identificados nas entrevistas anteriores foram adicionadas ao roteiro de entrevistas. Exemplos de questões adicionadas incluem: (1) Qual é a relação entre automatizar o *deployment* e a adoção de *DevOps*? (2) É possível adotar *DevOps* sem automação? (3) Foi formada uma cultura de colaboração? Como? O que faz parte dessa cultura?

3.3 Análise de Dados

Para analisar os dados, as entrevistas foram gravadas, transcritas e analisadas. A primeira fase da análise, chamada de *codificação aberta*, iniciou imediatamente após a transcrição da primeira entrevista.

A codificação aberta durou até que não restassem dúvidas a respeito de qual era a *categoria principal* do estudo. Similarmente ao descrito por Adolph et al. [10], no início da análise, uma categoria demonstrou potencial para ser a categoria principal, mas não se consolidou e foi alterada depois. A primeira candidata a categoria principal foi ***automação*** que aparecia de maneira recorrente nos relatos de adoção de *DevOps*. Todavia, foi notado que muitos dos padrões identificados nos dados não eram facilmente explicados em torno de ***automação***. Por exemplo, o senso de responsabilidade compartilhada entre os times para resolução de problemas, ou a noção de formação de uma visão de produto durante o processo de desenvolvimento de *software*, eventos comumente descritos nas entrevistas, não aparentavam se relacionar com ***automação***. Passou-se então a observar que a formação de uma ***cultura de colaboração*** era outra categoria recorrente na análise dos dados e com maior potencial para explicar os eventos. Assim sendo, perguntas tratando especificamente a respeito do papel da ***automação*** em uma adoção de *DevOps* e sobre a formação de uma ***cultura de colaboração*** foram adicionadas ao roteiro de entrevista.

Considerando as adaptações no roteiro e as análises de novos dados em um processo de *comparação constante*, levando em conta as análises prévias e os respectivos *memos* escritos durante todo o processo, após a décima entrevista concluiu-se que ***cultura de colaboração*** era inequivocamente a *categoria principal* sobre como *DevOps* foi adotado de maneira bem sucedida nas companhias investigadas.

Neste momento, a *codificação aberta* foi encerrada e a *codificação seletiva* foi iniciada. A codificação passou a ser restrita àquelas variáveis diretamente relacionadas à categoria principal e seus relacionamentos. Realizando mais três entrevistas e respectiva análise, percebeu-se que os novos dados forneciam cada vez menos conteúdo à teoria que estava emergindo. Ou seja, as explicações a respeito de como a categoria ***cultura de colaboração*** é desenvolvida em uma adoção de *DevOps* mostravam sinais de saturação. Assim sendo, foram conduzidas mais duas entrevistas para concluir que a *saturação teórica* havia sido atingida.

Seguindo a abordagem descrita anteriormente, iniciou-se a última etapa da análise que é a *codificação teórica* cujo objetivo é encontrar uma forma de se integrar todos os conceitos, categorias e *memos* em uma teoria coesa e homogênea. Percebeu-se neste momento que as categorias identificadas desempenhavam os papés de *saídas* ou *facilitadores DevOps*. Esses papéis serão melhor detalhados na seção 3.8.

Para ilustrar com maior precisão os detalhes de como foi realizado o processo de codificação, que resultou na descoberta dos conceitos e categorias relacionados à adoção de *DevOps*, a seguir é apresentado um exemplo da evolução de um trecho de transcrição de entrevista (dado bruto) até uma categoria.

Durante as entrevistas, diversos detalhes acessórios são citados pelos entrevistados. Isso resulta em dados brutos (transcrições) cheios de ruídos. Assim sendo, inicialmente foram removidos os ruídos do dado bruto, resultando na identificação dos *pontos-chave*. Pontos-chave são a síntese obtida dos trechos de entrevistas [50]. Por exemplo:

Dado bruto: *"Então, aqui nós adotamos este tipo de estratégia que é a infraestrutura como código, consequentemente, nós temos um versionamento de toda a nossa infraestrutura em uma linguagem comum, de tal maneira que qualquer pessoa, um desenvolvedor, um arquiteto, o cara de operações, ou mesmo o gerente, ele olha e consegue descrever que a configuração da aplicação x é y. Então, isto agrega muito valor para nós exatamente com mais transparência".*

Ponto-chave: *"Infraestrutura como código contribui para transparência por possibilitar o versionamento da infraestrutura em uma linguagem comum para todos os profissionais".*

Após a identificação dos pontos-chave, um ou mais códigos foram então atribuídos a cada um deles. Um código é uma frase que sumariza o ponto-chave, e um ponto-chave pode levar a diversos códigos [17].

Código: *Infraestrutura como código contribui para transparência.*

Código: *Infraestrutura como código provê uma linguagem comum.*

Neste exemplo, o conceito que emergiu foi o de "*infraestrutura como código*". O trecho de dado bruto acima foi retirado da décima segunda entrevista, momento em que este conceito já existia como resultado das entrevistas anteriores. Abaixo é apresentado um exemplo de *memo* escrito neste momento da análise.

Memo: *O conceito de infraestrutura como código já apareceu diversas vezes nas análises anteriores, esse é mais um exemplo. É mais um sinal de que os conceitos estão apenas se repetindo e pode-se estar próximo da saturação teórica. Este exemplo também reforça o que já estava sendo cogitado de que infraestrutura como código está mais relacionado a transparência do que a automação.*

O *memo* descrito acima foi útil para posteriormente ajudar a concluir que a saturação teórica havia sido atingida. Também foi utilizado como auxílio no processo de agrupamento de conceitos na categoria que foi denominada ***Compartilhamento e Transparência***. Em um estudo utilizando GT, uma categoria é um agrupamento de conceitos relacionados, um novo nível de abstração. A Figura 3.3 ilustra o agrupamento dos conceitos que formaram a referida categoria.

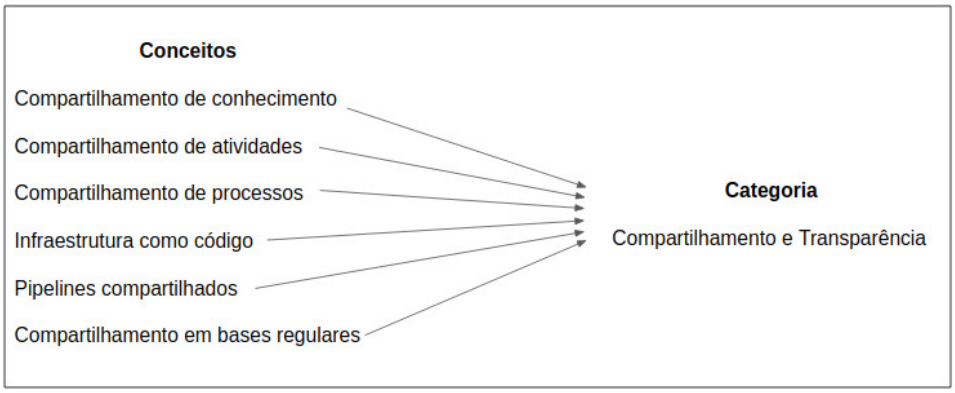

Figura 3.3: Codificação: Construindo Categorias.

A expressão correspondente ao conceito de *infraestrutura como código* veio direto do dado bruto, mas isso não é uma regra. É comum que o conceito seja uma abstração, sem que tenha emergido diretamente de uma expressão contida na transcrição de entrevista. Por exemplo, abaixo é apresentado como o conceito de "*empoderamento do desenvolvimento de software*" emergiu sem que esta expressão específica esteja presente no dado bruto.

Dado bruto: *"Nós temos diversas pessoas trabalhando no desenvolvimento, a quantidade de desenvolvedores é realmente impressionante. Não era viável a gente ter tantos desenvolvedores produzindo artefatos e parando o trabalho deles para aguardar outro time completamente separado publicar. Ou também precisar de um ambiente de teste e ter que esperar o time de operações prover isto quando possível. Essas atividades tinham que estar disponíveis para servir rapidamente ao time de desenvolvimento. Com DevOps nós conseguimos suprir essa necessidade de liberdade e mais poder para executar algumas tarefas que são intrisicamente relacionadas ao trabalho deles".*

Ponto-chave: *"Faz parte da adoção de DevOps a incorporação de maior liberdade e poder para o time de desenvolvimento executar tarefas que são intrisicamente relacionadas ao seu trabalho, como publicação de artefatos e provimento de ambientes de testes".*

Código: Facilitar a publicação de artefatos empodera o desenvolvimento de *software*.

Código: Facilitar o provimento de ambientes de testes empodera o desenvolvimento de *software*.

Conceito: Empoderamento do desenvolvimento de *software*.

A análise dos dados resultou na geração de 34 conceitos relacionados à adoção de *DevOps*, que foram agrupados em sete categorias. Os conceitos e categorias são enumerados na Tabela 3.3 e detalhados nas seções 3.4 a 3.7. Durante estas seções, são exibidos trechos de transcrições de entrevistas que foram selecionados por ilustrar de maneira clara as ideias envolvidas na caracterização da adoção de *DevOps*.

	Categoria		**Conceito**
1	Cultura de Colaboração	1	Operações no dia a dia de desenvolvimento
		2	Empoderamento do desenvolvimento de *software*
		3	Pensamento de produto
		4	Comunicação direta
		5	Responsabilidade compartilhada
		6	*Blameless*
2	Automação	1	Automação do *deployment*
		2	Automação de testes
		3	Automação do provimento da infraestrutura
		4	Automação do gerenciamento da infraestrutura
		5	Serviços autônomos
		6	Conteinerização
		7	Automação do monitoramento
		8	Automação da recuperação
3	Compartilhamento e Transparência	1	Compartilhamento de conhecimento
		2	Compartilhamento de atividades
		3	Compartilhamento de processos
		4	Infraestrutura como código
		5	Compartilhamento em bases regulares
		6	*Pipelines* compartilhados
4	Agilidade	1	Integração contínua
		2	Provimento contínuo da infraestrutura
		3	*Deployment* contínuo
5	Resiliência	1	Auto *scaling*
		2	Automação da recuperação
		3	Zero *down-time*
6	Medição Contínua	1	Monitoramento de *logs* de aplicações
		2	Monitoramento contínuo de aplicações
		3	Monitoramento contínuo de infraestrutura
7	Garantia da Qualidade	1	Ramificação de código
		2	Serviços coesos
		3	Teste contínuo
		4	Análise estática de código fonte
		5	Paridade entre ambientes

Tabela 3.3: Enumeração de Categorias e Conceitos Relacionados

3.4 A Categoria Principal: Cultura de Colaboração

A categoria principal da adoção de *DevOps* é ***cultura de colaboração***. A ***cultura de colaboração*** visa essencialmente remover os silos entre os times de desenvolvimento e operações, o que até certo ponto se confunde com os próprios objetivos de *DevOps*. Inicialmente, uma ***cultura de colaboração*** envolve que as tarefas tipicamente de operações — como *deployment*, provisionamento e gerenciamento de infraestrutura e monitoramento — devem ser consideradas atividades regulares, parte do dia a dia de desenvolvimento de *software*. Os entrevistados relatam que nas suas organizações não são mais aguardados momentos específicos para performar essas atividades, elas são executadas continuamente. Isso leva ao primeiro conceito relacionado à categoria principal: **operações no dia a dia de desenvolvimento**.

> "*Uma etapa muito importante foi trazer o deployment para dentro do dia a dia de desenvolvimento, sem ter que ficar esperando um dia específico da semana ou do mês. Nós queríamos fazer deploy o tempo inteiro, mesmo que em um primeiro momento não fosse em produção, um ambiente de staging era suficiente. [...] O que a gente queria era incorporar o deployment ao desenvolvimento. É claro que para que a gente pudesse fazer o deployment continuamente, a gente tinha que prover toda a infraestrutura necessária no mesmo ritmo.*" (P14, Gerente de TI, Brasil)

Sem *DevOps*, um cenário comumente descrito é a existência de um processo acelerado de desenvolvimento de *software* sem preocupações relacionadas a operações. No final, quando o time de desenvolvimento tem um mínimo produto viável, ele o envia para o time de operações para publicação. Conhecendo poucas coisas sobre a natureza do *software* e como ele foi produzido, o time de operações tem que criar e configurar o ambiente necessário e publicar o *software*. Neste cenário, a entrega de *software* tipicamente atrasa e os conflitos entre os times se manifestam. Quando uma ***cultura de colaboração*** é fomentada, os times colaboram para executar as tarefas desde o primeiro dia do desenvolvimento do *software*. Com o constante exercício das práticas de provisionamento, gerenciamento, configuração e *deployment*, a entrega de *software* se torna mais natural, reduzindo os atrasos e, consequentemente, os conflitos entre os times.

> "*Nós trabalhamos usando uma abordagem ágil, com sprints de quinze dias, onde a gente foca em produzir software e gera novas versões em altíssima frequência. Mas, na hora da entrega do software é que as complicações começavam a aparecer. O trabalho de construir todo o ambiente e fazer o deploy não fazia parte das sprints, a gente focava apenas em codificar a aplicação. As entregas frequentemente atrasavam,*

e a gente tinha que entregar com atrasos de semanas, o que não era bom nem para nós e nem para os clientes." (P6, *Developer*, Portugal)

Um dos resultados da construção de uma ***cultura de colaboração*** é que o time de desenvolvimento não precisa mais parar o seu trabalho aguardando pela disponibilização de um servidor de aplicação, pela execução de um *script* na base de dados, ou ainda pela publicação de uma nova versão do *software* em um ambiente de testes. Todos os envolvidos precisam conhecer a maneira como todas essas coisas são feitas e, com a colaboração do time de operações, isso pode ser executado de maneira regular. Se uma tarefa pode ser executada pelo time de desenvolvimento e há confiança entre os times, essa tarefa será incorporada ao processo de desenvolvimento de uma maneira natural, manifestando o segundo conceito relacionado à categoria ***cultura de colaboração***: **empoderamento do desenvolvimento de *software***.

"*Não era viável a gente ter tantos desenvolvedores produzindo artefatos e parando o trabalho deles para aguardar outro time completamente separado publicar. Ou também precisar de um ambiente de teste e ter que esperar o time de operações prover isto quando possível. Essas atividades tinham que estar disponíveis para servir rapidamente ao time de desenvolvimento. Com DevOps nós conseguimos suprir essa necessidade de liberdade e mais poder para executar algumas tarefas que são intrisicamente relacionadas ao trabalho deles.*" (P5, *Systems Engineer*, Espanha)

Uma ***cultura de colaboração*** requer **pensamento de produto**, em substituição a pensamento de desenvolvimento ou de operações. O time de desenvolvimento precisa compreender que o *software* é um produto que não se encerra após o "*push*" do código para o repositório de código-fonte, e o time de operações precisa entender que o processo também não se inicia quando um artefato é recebido para publicação. **Pensamento de produto** é o terceiro conceito relacionado à categoria principal.

"*Nós alteramos o perfil profissional buscado em nossas contratações. Nós queríamos contratar pessoas que tivessem uma visão de produto. Pessoas que eram capazes de olhar para um problema e pensar na melhor solução para ele. Mas não apenas pensar em uma solução de software, pensar também no momento em que essa aplicação vai ser publicada. Nós também reunimos os desenvolvedores para reforçar que todos deviam atuar dessa maneira. Todos deviam pensar no produto e não apenas em seu código ou em sua infraestrutura.*" (P12, *Cloud Engineer*, Estados Unidos)

Deve haver uma **comunicação direta** entre os times. Sistemas de *ticket* são citados como um meio típico e inadequado de comunicação entre os times de desenvolvimento

e operações. A comunicação face a face é a melhor opção, mas considerando que nem sempre é viável, o uso contínuo de ferramentas como *Slack*[1] e *Hipchat*[2] é citado como opção apropriada. **Comunicação direta** é o quarto conceito relacionado à categoria principal.

> "*Nós também usamos essa ferramenta (Hipchat) como uma maneira de facilitar a comunicação entre os times de desenvolvimento e operações. O ritmo de trabalho é bastante acelerado, e por isso não é viável ter uma comunicação burocrática. Pra agilizar as coisas, a gente usa bastante o Hipchat, todos estão sempre atentos às mensagens, as respostas são rápidas e a gente tem um controle bem adequado por lá. Isso deu muita liberdade nas tarefas de desenvolvimento, em caso de qualquer dúvida, a equipe de operações está ao alcance de uma mensagem.*" (P5, *Systems Engineer*, Espanha)

Existe uma **responsabilidade compartilhada** de identificar e corrigir os problemas de um ***software*** ao fazer a transição para produção. A estratégia de fugir da responsabilidade deve ser evitada. A equipe de desenvolvimento não deve afirmar que uma determinada questão é um problema na infraestrutura, então é responsabilidade da equipe de operações. Ou o contrário, a equipe de operações não deve afirmar que uma determinada falha foi motivada por um problema na aplicação, então é responsabilidade da equipe de desenvolvimento. Um contexto de ***blameless*** deve ser fomentado. Os times precisam focar na resolução dos problemas e não em encontrar um culpado e fugir da responsabilidade. O contexto de **responsabilidade compartilhada** envolve não apenas a resolução de problemas, mas também qualquer outra responsabilidade inerente ao produto de *software* deve ser compartilhada. ***Blameless*** e **responsabilidade compartilhada** são os conceitos restantes da categoria principal.

> "*Como consequência dessa busca contínua por melhoria da qualidade, em um momento já avançado do processo, quando nós já tínhamos um bom nível de colaboração, automação e tudo mais, nós identificamos um ponto de melhoria na nossa cultura. Nós percebemos que algumas pessoas tinham medo de cometer erros. Nossa cultura não era forte o suficiente para fazer com que todos se sentissem à vontade para inovar e experimentar sem medo de errar. Nós fizemos um grande esforço para espalhar essa ideia de que não há culpados por qualquer problema que possa ocorrer. Nós fazemos todo o possível para evitar falhas, mas elas vão acontecer, e*

[1] https://slack.com/
[2] https://www.hipchat.com/

apenas sem essa busca por culpados nós vamos ser capazes de resolver os problemas rapidamente." (P8, *DevOps Engineer*, Brasil)

À primeira vista, considerar a criação e o fortalecimento da ***cultura de colaboração*** como o passo mais importante para a adoção de *DevOps* pode parecer um tanto óbvio, mas os próprios entrevistados citaram alguns equívocos que consideram recorrentes em não priorizar esse aspecto na adoção de *DevOps*:

"*Na adoção de DevOps, há uma questão cultural muito forte que os times muitas vezes não estão adaptados. Relacionado a isso, uma coisa que me incomoda muito e que eu vejo acontecer muito é que as pessoas tomam DevOps exclusivamente por ferramentas ou automação*" (P9, Gerente de TI, Brasil)

"*DevOps envolve ferramental, mas DevOps não é ferramental. Ou seja, as pessoas muitas vezes focam no uso de ferramentas que são chamadas de 'ferramentas DevOps', acreditando que DevOps é isto. Eu sempre insisto que DevOps não é uma ferramenta, DevOps envolve o uso de ferramentas para melhorar os procedimentos de desenvolvimento de software.*" (P2, Consultor *DevOps*, Brasil)

Além da categoria principal (***cultura de colaboração***), foram identificados três outros conjuntos de categorias: os *facilitadores* da adoção de *DevOps*, as *saídas* da adoção de *DevOps*, e as categorias que são tanto facilitadores como saídas. Uma explicação mais abrangente sobre esses dois papéis desempenhados pelas categorias de conceitos em uma adoção de *DevOps* será apresentada na seção 3.8. Nas próximas seções as categorias são descritas por meio dos seus conceitos relacionados.

3.5 Facilitadores *DevOps*

Nesta seção serão detalhadas as duas categorias que sustentam a adoção das práticas *DevOps*: ***automação*** e ***compartilhamento e transparência***.

3.5.1 Automação

Essa é a categoria que apresenta o maior número de conceitos relacionados. Isso ocorre porque procedimentos manuais são considerados fortes candidatos para propiciar a formação de um silo, dificultando a criação de uma ***cultura de colaboração***. Se uma tarefa é manual, uma pessoa ou um time será responsável por executá-la. Apesar de

compartilhamento e transparência poderem ser usados para garantir a colaboração mesmo em tarefas manuais, com a ***automação***, os pontos onde os silos podem surgir são minimizados.

> "*Quando um desenvolvedor precisava criar uma nova aplicação, o workflow antigo exigia que ele criasse um ticket para a equipes de operações, que então avaliava e resolvia manualmente o problema solicitado. Essa tarefa podia levar muito tempo e não havia visibilidade entre os times sobre o que estava acontecendo [...]. Hoje, esses silos não existem mais dentro da empresa, em particular porque não é mais necessário executar todas essas tarefas manualmente, tudo foi automatizado.*" (P12, *Cloud Engineer*, Estados Unidos)

Além de contribuir para transparência, a ***automação*** de procedimentos também é considerada importante para garantir *reprodutibilidade* de tarefas, reduzindo retrabalho e risco de falha humana. Consequentemente, ***automação*** aumenta a confiança entre os times o que é um aspecto importante da ***cultura de colaboração***.

> "*Antes de nós adotarmos DevOps, havia muito trabalho manual. Por exemplo, se você precisasse criar um esquema no banco de dados, era um processo manual; se você precisava criar um servidor de banco de dados, era um processo manual; se você precisasse criar instâncias EC2*[a] *adicionais, mais uma vez um processo manual. Este trabalho manual era demorado e muitas vezes causava erros e retrabalho.*" (P1, *DevOps Developer*, Irlanda)
>
> [a] *Amazon Elastic Compute Cloud*

> "*Nossa principal motivação para adotar DevOps foi basicamente reduzir o retrabalho. Quase toda semana a gente tinha que basicamente construir novos servidores e iniciá-los manualmente, o que era muito demorado.*" (P4, Técnico em Computação, Brasil)

Os oito conceitos da categoria ***automação*** são detalhados a seguir. Todas as entrevistas continham explicações sobre (1) **automação do *deployment***, como parte da adoção de *DevOps*. A entrega de *software* é a manifestação mais clara da entrega de valor no desenvolvimento de *software*. Em caso de problemas no *deployment*, a expectativa de entregar valor ao negócio pode rapidamente gerar conflitos e manifestar a existência de silos. Desta forma, a ***automação*** normalmente aumenta a agilidade e a confiabilidade. Alguns outros conceitos de automação giram exatamente em torno da automação do *deployment*.

É importante observar que a ocorrência frequente de *deployments* bem sucedidos não é suficiente para garantir a geração de valor para o negócio. Certamente, a qualidade do *software* é mais relevante. Portanto, para que possam fazer parte do *pipeline* do *deployment*, as verificações de qualidade também precisam ser automatizadas, como é o caso da (2) **automação de testes**. Além disso, para automatizar o *deployment* de aplicações, o ambiente em que elas serão executadas precisa estar disponível. Portanto, a (3) **automação do provimento da infraestrutura** também deve ser considerada no processo. Além de estar disponível, o ambiente precisa ser configurado adequadamente, incluindo a quantidade de memória e CPU disponibilizada, as versões corretas de bibliotecas e a estrutura do banco de dados. Se a configuração de algum desses aspectos não tiver sido automatizada, o *deployment* automatizado pode não funcionar. Portanto, a (4) **automação do gerenciamento da infraestrutura** é outro conceito da categoria ***automação***.

Software moderno é tipicamente construído em torno de serviços. Microsserviços foram comumente citados como um aspecto da adoção de *DevOps*. Para Fowler e Lewis [51], no estilo arquitetural de microsserviços, os serviços precisam ser independentemente implantáveis por mecanismos de *deployment* totalmente automatizados. Essa parte das características de microsserviços relacionada a ***automação*** foi aqui denominada de (5) **serviços autônomos**. Os relatos sobre a adoção de *DevOps* tipicamente citam a (6) **conteinerização** como uma maneira de automatizar o provisionamento do ambiente onde esses serviços autônomos são executados: os contêineres. (7) **Automação do monitoramento** e (8) **automação da recuperação** são os conceitos restantes. O primeiro refere-se à capacidade de monitorar as aplicações e a infraestrutura subjacente sem intervenção humana. Um exemplo clássico é o uso generalizado de ferramentas para enviar mensagens relatando alarmes - por meio de SMS, *Slack* / *Hipchat*, ou até mesmo chamadas de celular - em caso de incidentes relacionados às aplicações detectados automaticamente. E o segundo está relacionado à capacidade de substituir um componente que não está funcionando adequadamente ou reverter uma falha no *deployment* sem intervenção humana.

3.5.2 Compartilhamento e Transparência

Esta categoria representa o agrupamento de conceitos emergidos nas entrevistas a respeito de atividades que ajudam a disseminar conhecimento técnico e procedimental entre os times, de modo a incrementar a colaboração entre eles. Ações de treinamento interno e externo, palestras, discussões em grupos e *round tables* são exemplos desses eventos. Criar canais usando ferramentas de comunicação é outra alteranativa recorrentemente citada no ***compartilhamento e transparência*** ao longo do processo de adoção de *DevOps*.

De acordo com o conteúdo e o meio onde há o compartilhamento, foram identificados inicialmente três conceitos para esta categoria:

1. **Compartilhamento de conhecimento**: os profissionais entrevistados mencionaram que existe uma ampla gama de habilidades técnicas e culturais que precisaram adquirir durante a adoção de *DevOps*. Como meio para suavizar a curva de aprendizagem existente foi mencionada a realização de eventos de compartilhamento de conhecimento, tais quais, ações estruturadas de treinamento (cursos) com participação de profissionais dos dois times e participação em eventos da comunidade de desenvolvimento de *software*.

2. **Compartilhamento de atividades**: esse conceito trata de ações onde o foco é em compartilhar a maneira como tarefas simples foram realizadas, por exemplo, como um erro específico foi corrigido, ou detalhes de configuração de alguma ferramenta específica. As próprias ferramentas de comunicação, junto com existência de comitês e a realização de *round tables* foram citados como fóruns adequados para compartilhamento deste tipo de conteúdo.

3. **Compartilhamento de processos**: aqui, o foco é em compartilhar um processo de trabalho como um todo. Por exemplo: como configurar um *pipeline* no *Jenkins* utilizando *Jenkinsfile*. O conteúdo é mais abrangente do que no compartilhamento de atividades. A realização de apresentações e palestras foi citada como meio mais comum para o compartilhamento de processos.

> "*Hoje em dia eu vejo as pessoas na empresa muito preocupadas em que todo mundo saiba o que ela está fazendo e como ela está fazendo. Por isso a gente tem essas ações estruturadas que eu te falei, se alguma pessoa quer repassar algum conteúdo relevante para o restante da equipe, ela tem total liberdade para reservar a sala e realizar a própria tech talk. Antigamente tinha um negócio de meio que fazer as coisas às escondidas para o cara até mesmo tentar valorizar o passe dele, 'ah, só fulano sabe como fazer isso'. Agora com esse investimento em horizontalizar a cultura, o cara sabe que tudo tem que ser compartilhado e transparente para todos, e isso gera um ciclo positivo que aumenta o senso de colaboração e faz ele querer compartilhar ainda mais o que ele faz.*" (P7, Analista de Suporte, Brasil)

Estes conceitos de compartilhamento contribuem com a ***cultura de colaboração***. Por exemplo, todos os membros dos times ganham uma melhor compreensão sobre todo o

processo de produção de *software*, com um sólido entendimento de que as responsabilidades devem ser compartilhadas. Um vocabulário compartilhado também tende a se formar por meio dessas ações de compartilhamento e isso facilita a comunicação.

O uso de **infraestrutura como código** foi recorrentemente citado como um meio de se garantir que todos saibam como o ambiente de execução de aplicação é provido e gerenciado. Abaixo, é apresentada um trecho de transcrição de entrevista que resume bem este conceito:

> "*Então, aqui nós adotamos este tipo de estratégia que é a infraestrutura como código, consequentemente, nós temos um versionamento de toda a nossa infraestrutura em uma linguagem comum, de tal maneira que qualquer pessoa, um desenvolvedor, um arquiteto, o cara de operações, ou mesmo o gerente, ele olha e consegue descrever que a configuração da aplicação x é y. Então, isto agrega muito valor para nós exatamente com mais transparência.*" (P12, *Cloud Engineer*, Estados Unidos)

Em relação a ***compartilhamento e transparência***, foi também identificado o conceito de **compartilhamento em bases regulares**, que sugere que as ações de compartilhamento devem ser incorporadas no processo de desenvolvimento de *software*, de modo a contribuir de maneira eficaz para a transparência. Conforme será detalhado no conceito de *integração contínua* da categoria ***agilidade*** (subseção 3.6.1), uma maneira comum de se integrar todas as tarefas é um *pipeline*. Aqui existe o conceito de ***pipelines* compartilhados**, que indica que o código dos *pipelines* deve ser acessível a todos, buscando fomentar a transparência.

> "*O código de como a infraestrutura é feita é aberto aos desenvolvedores e os sysadmins precisam conhecer alguns aspectos de como o código da aplicação é construído. O código dos nossos pipelines é acessível a todos na empresa para saberem como as atividades estão automatizadas.*" (P13, Gerente de Tecnologia, Brasil)

3.6 *Saídas* da Adoção de *DevOps*

Nesta seção são detalhadas as categorias que correspondem às consequências esperadas do processo de adoção de *DevOps*: ***agilidade*** e ***resiliência***.

3.6.1 Agilidade

A maior *agilidade* dos times foi frequentemente descrita como um dos principais resultados da adoção do *DevOps*. Com mais colaboração entre os times, a **integração contínua**

com execução de *pipelines* multidisciplinares é possível e este é um conceito relacionado a agilidade que é frequentemente explorado. Esses *pipelines* podem conter provimento de infraestrutura, testes automatizados, análise de código, *deployment* automatizado e qualquer outra tarefa considerada importante no processo de desenvolvimento para que possa ser executada continuamente.

> "*O nosso pipeline tem etapas com toda uma estrutura para criar e destruir contêineres de maneira rápida e gerar ambientes rapidamente utilizando Docker. Ele também passa pelo Sonar*[a] *para analisar o código utilizando os plugins e bibliotecas dele [...]. Tem também os testes automatizados e em qualquer dessas etapas que estou falando o pipeline pode ser interrompido com envio de mensagem para que o desenvolvedor ajuste algum aspecto no código. Tem também pontos específicos de uso do Nessus*[b] *para checagem de vulnerabilidade, então, se a aplicação consegue passar por todas essas etapas do pipeline, a gente não tem motivos para se preocupar em fazer o deployment dessa versão.*" (P7, Analista de Suporte, Brasil)
>
> [a] https://www.sonarqube.org/
> [b] *Scanner* de Vulnerabilidades

Estes *pipelines* ocasionam dois outros conceitos da categoria ***agilidade***: **provisionamento contínuo de infraestrutura** e ***deployment* contínuo**. Este último é um dos conceitos mais recorrentes identificados na análise das entrevistas. Sem *DevOps*, o *deployment* é visto como um grande evento com alto risco de *down-time* e falhas. Com *DevOps*, a sensação de risco no *deployment* diminui e essa atividade torna-se mais natural e frequente. Alguns praticantes afirmam realizar dezenas de *deployments* diariamente.

> "*Quando a gente adotou DevOps, a nossa curva de ganho foi muito grande. Então a gente foi de um deploy a cada 15 dias para 40 deploys por dia. Então, é uma curva muito gigante, a entrega de valor para a empresa foi gigante.*" (P12, *Cloud Engineer*, Estados Unidos)

3.6.2 Resiliência

Também relacionada à uma saída esperada do processo de adoção de *DevOps*, a categoria ***resiliência*** refere-se à capacidade que as aplicações desenvolvidas possuem de se adaptar rapidamente a situações adversas. O primeiro conceito relacionado é **auto *scaling***, que indica a presença de mecanismos que possibilitam a alocação automática de mais ou menos recursos para aplicações que aumentam ou diminuem sua demanda de acesso em

momentos específicos. Outro conceito relacionado à categoria de ***resiliência*** é **automação da recuperação**, que é a capacidade de as aplicações e infraestrutura subjacente se recuperarem em caso de falhas. Foram relatados dois casos típicos de automação de recuperação: (1) em casos de alguma instabilidade no ambiente de execução de uma aplicação (um contêiner, por exemplo) ocorre uma reinicialização automática desse ambiente; e (2) em casos de implantação de nova versão, se ela não funcionar adequadamente, a anterior é automaticamente restaurada. Essa restauração automática de uma versão anterior diminui a chance de inatividade devido a erros em versões específicas, que é o conceito de **zero *down-time***, que é o último da categoria de ***resiliência*** e indica que uma aplicação que esteja executanto corretamente não sofrerá indisponibilidade por conta de falhas que possam ser evitadas.

> "*Quando era necessário dar um deploy em alguma das aplicações que a gente tinha sempre acontecia um downtime de alguns minutos e, obviamente, se tinha o downtime e o deploy não dava certo, o downtime era ainda maior. Mas com a adoção de DevOps a gente conseguiu justamente diminuir para, na verdade a gente diminuiu para quase nada, acho que era em torno de um minuto ou menos e posteriormente a gente conseguiu eliminar qualquer downtime, utilizando o Kubernetes*[a]" (P1, *DevOps Developer*, Irlanda)
>
> [a] https://kubernetes.io/

3.7 Categorias que são tanto *Facilitadores* como *Saídas* da Adoção de *DevOps*

Finalmente, aqui são detalhadas as categorias que aparecem tanto como *facilitadores* quanto como *saídas* no processo de adoção de *DevOps*: ***medição contínua*** e ***garantia da qualidade***.

3.7.1 Medição Contínua

Como a responsabilidade pela execução das atividades de medição e monitoramento é tida como atribuição típica do time de operações, à medida que ela passa a ser executada continuamente e de maneira transparente, manifesta-se como um *facilitador DevOps* pois fomenta a ***cultura de colaboração***. Ademais, a coleta contínua de métricas reforça a confiança entre os times pois há um incremento na proatividade, o que também é uma característica importante da ***cultura de colaboração***.

"*Antes, a gente tinha só aquelas olhadas esporádicas no zabbix*[a] *para verificar se estava tudo OK. No máximo, alguém parava para verificar o consumo de memória e CPU. Para manter a qualidade dos serviços, expandimos essa questão da coleta de métricas para que ela se tornasse parte do produto de software. Depois, a gente começou a coletar métricas continuamente e com responsabilidades compartilhadas. Por exemplo, se acontecer um overflow no número de conexões do banco de dados, todos recebem um alerta e são responsáveis por buscar soluções para esse problema. Este (número de conexões de banco) é um exemplo interessante de métrica que todos começaram a ficar mais atentos, não só o time de operações.*" (P3, *DevOps Developer*, Irlanda)

[a] `https://www.zabbix.com/`

Já considerando as *saídas DevOps*, a coleta contínua de métricas das aplicações e da infraestrutura é tida pelos entrevistados como uma consequência requerida da adoção de *DevOps*. Isto ocorre porque a agilidade resultante do processo aumenta o risco de algo dar errado. Os times devem ser capazes de reagir rapidamente em caso de problemas, e a medição contínua possibilita essa proatividade e resiliência.

"*Hoje é viável que a gente faça o deploy o tempo todo e, naturalmente, houve a necessidade de maior controle do que estava acontecendo. Então, nós usamos grafana*[a] *e prometheus*[b] *para acompanhar tudo o que está acontecendo na infraestrutura e nas aplicações. Nós temos um painel completo em tempo real, extraímos relatórios e, quando algo dá errado, somos os primeiros a saber.*" (P10, Administrador de Redes, Brasil)

[a] `https://grafana.com/`
[b] `https://prometheus.io/`

A ***medição contínua*** envolve (1) **monitoramento de logs de aplicações**, um conceito que corresponde ao uso do *log* produzido pelas aplicações e pela infraestrutura como fonte de dados. O conceito de (2) **monitoramento contínuo de infraestrutura** indica que o monitoramento não é realizado por uma pessoa ou time específicos em um momento específico. A responsabilidade de monitorar a infraestrutura é compartilhada e é executada continuamente. Já o **monitoramento contínuo de aplicações** refere-se à instrumentação para fornecer métricas que são usadas para avaliar o comportamento das aplicações em execução e, muitas vezes, direcionar decisões de evolução ou de negócios. Todas essas medições/monitoramentos podem ocorrer de forma automatizada, o conceito de **automação do monitoramento** já foi descrito na subseção 3.5.1.

3.7.2 Garantia da Qualidade

Da mesma forma que a ***medição contínua***, a ***garantia de qualidade*** é uma categoria que pode funcionar tanto como *facilitador* quanto como *saída* do processo de adoção de *DevOps*. Como *facilitador*, porque um aumento na qualidade é descrito como responsável por gerar mais confiança entre os times, o que, no final, gera um ciclo virtuoso de colaboração. Como *saída*, o princípio é de que não é viável criar um cenário de entrega contínua de *software* sem um controle rigoroso da qualidade dos produtos e seus respectivos processos de produção.

Os entrevistados apontaram para a necessidade de um controle sofisticado a respeito de quais partes de código devem fazer parte das entregas que são realizadas continuamente. O *Git Flow*[3] foi recorrentemente citado como modelo utilizado para atender à essas necessidades de (1) **ramificação de código**, o primeiro conceito de ***garantia de qualidade***. Em uma seção anterior, foi explorada a face de automação relacionada a microsserviços e testes. Esses elementos também têm uma face de garantia de qualidade. Uma característica do estilo arquitetural de microsserviços é a necessidade de os serviços serem de pequeno porte com foco em fazer apenas uma coisa. Esses pequenos serviços são mais fáceis de dimensionar e estruturar, o que manifesta um conceito de garantia de qualidade: (2) **serviços coesos**. Em relação aos testes, a outra face mencionada é o (3) **teste contínuo**. Para garantir a qualidade dos produtos de *software*, foi identificado que os testes (bem como outras verificações de qualidade) devem ocorrer continuamente. Executar os testes de maneira contínua é considerado uma tarefa desafiadora sem o uso de automação, e isso reforça a necessidade de testes automatizados.

Os outros dois conceitos citados como parte da ***garantia de qualidade*** na adoção de *DevOps* são o uso de (4) **análise estática de código fonte** para calcular e avaliar continuamente métricas de qualidade no código-fonte e **paridade entre os ambientes** para reforçar a transparência e a colaboração durante o desenvolvimento de *software*.

> "*Com essa questão do deploy automatizado a gente tinha que se preocupar bastante com testar direito a aplicação, foi inevitável a gente automatizar os testes. [...] aí também a gente começou a usar o Git Flow para acabar com as inconsistências que aconteciam, era commit de muita gente no mesmo repositório, sem pull request, sem nada e às vezes isso comprometia o processo de entrega, tinha que ficar fazendo uma branch separada só para release.*" (P6, Developer, Portugal)

[3] https://nvie.com/posts/a-successful-git-branching-model/

3.8 Uma Teoria Sobre a Adoção de *DevOps*

Os resultados de um estudo utilizando *grounded theory*, como o próprio nome do método sugere, são fundamentados nos dados coletados, de modo que as hipóteses emergem dos dados, ao invés de serem definidas no início da pesquisa e validadas utilizando algum modelo estatítico. Hoda et al. [17] explicam que uma teoria produzida utilizando *grounded theory* deve conter um conjunto de hipóteses inter-relacionadas e que o termo *hipótese* neste contexto se refere às descrições das relações-chave entre as categorias que compõem uma determinada teoria.

Nas próximas seções serão apresentadas as hipóteses que descrevem as principais relações entre as categorias que fazem parte da adoção de *DevOps*. Estas hipóteses constituem a teoria produzida e indicam que a adoção de *DevOps* é explicada por meio de uma rede de categorias que contribuem para o desenvolvimento da ***cultura de colaboração***, os *facilitadores*. E que algumas outras categorias relacionadas são consideradas resultados esperados do processo de adoção de *DevOps*, as *saídas DevOps*. É ainda explicado que algumas categorias em alguns momentos podem ser consideradas facilitadores e em outros saídas.

3.8.1 Uma Abordagem Geral para a Adoção de DevOps

Aqui é apresentado um possível caminho que pode ser utilizado por novos praticantes que desejam adotar *DevOps*. Esta abordagem foi construída com base nas análises realizadas conforme detalhado na subseção 3.1.2. Conforme já mencionado anteriormente, a categoria principal na adoção de *DevOps* é ***cultura de colaboração***. Isto implica que a principal preocupação na adoção de *DevOps* deve ser a formação e o desenvolvimento de uma ***cultura de colaboração*** entre as equipes de desenvolvimento e operações de *software*. De acordo com as análises, as outras categorias, muitas das quais também estão presentes em outros estudos que investigaram *DevOps*, só podem ser consideradas como efetivas para a adoção de *DevOps* se as práticas e conceitos relacionados a elas contribuírem para o nível de ***cultura de colaboração*** ou levarem às conseqüências esperadas de uma ***cultura de colaboração***. Esse entendimento induz algumas hipóteses, como discutido a seguir.

> **Hipótese 1:** *Existe um conjunto de categorias relacionadas à adoção de DevOps que apenas fazem sentido se usadas para aumentar o nível da* **cultura de colaboração**. *Esse conjunto de categorias é chamado de* ***facilitadores DevOps***.

De modo geral, os *facilitadores DevOps* são os meios comumente usados para aumentar o nível da ***cultura de colaboração*** em um processo de adoção de *DevOps*. Conforme

detalhado nas seções 3.5 e 3.7, foram identificadas quatro categorias que atuam como *facilitadores DevOps*: ***automação***, ***medição contínua***, ***garantia da qualidade*** e ***compartilhamento e transparência***.

Com base nesta primeira hipótese, a maturidade da adoção de *DevOps* não avança em situações em que apenas uma equipe é responsável por entender, adaptar ou evoluir algum aspecto. Por exemplo, um sofisticado grau de automação, mesmo quando ela suportar diferentes atividades, como implantação, provisionamento de infraestrutura ou monitoramento, não contribui para *DevOps* se não contribuir para o incremento da ***cultura de colaboração***. O mesmo vale para as outras categorias de *facilitadores*. Ou seja, nas situações em que ***compartilhamento e transparência***, ***garantia da qualidade*** e ***medição contínua*** não contribuem para a ***cultura de colaboração***, também não possibilitam um avanço na adoção de *DevOps* como um todo. Alguns exemplos que suportam essa primeira hipótese incluem:

> "*Olhe, dentro do setor de operações havia algum grau de automação. O cara tinha bash scripts na máquina dele que ajudavam na criação de um servidor ou de uma nova instância de banco de dados. Só que isso na minha visão, não havia DevOps porque não tinha relação intrínseca dessa automação com o processo de desenvolvimento*" (P11, Supervisor de *DevOps*, Brasil)

> "*Manter a cultura viva continua sendo um desafio para nós e isso é muito importante. Aqui na empresa, por exemplo, temos tech talks que são conversas mensais que temos com as equipes. O objetivo destes Tech Talks é compartilhar conhecimentos sobre tecnologias e processos de trabalho aumentando a transparência de como tudo funciona. Nós também temos um canal Slack chamado DevOps como cultura, onde discutimos as coisas da cultura DevOps. A ideia é não deixar a cultura morrer, estamos sempre alimentando-a com alguma coisa, porque isso é a essência de DevOps para nós.*" (P12, *Cloud Engineer*, Estados Unidos)

Hipótese 2: *Há um grupo de categorias relacionadas à adoção de DevOps que não são mencionadas por contribuem para aumentar o nível da* **cultura de colaboração**, *mas que são apontados como relacionadas à adoção de DevOps, porque elas emergem como um consequência da adoção. Essas categorias representam o grupo de* ***saídas DevOps***.

Em um primeiro momento, o simples fato de que uma equipe é mais ágil na entrega de *software*, ou produz aplicações mais resilientes na recuperação de falhas, não é apontado

como responsável por contribuir diretamente para aproximar as equipes de operações das equipes de desenvolvimento. No entanto, um sinal de uma adoção madura de *DevOps* é um aumento da capacidade de entregar *software* continuamente (de maneira mais ágil) e de construir aplicações e infraestruturas resilientes.

Em síntese, as *saídas DevOps* são o conjunto de categorias que não produzem primariamente o efeito esperado de um *facilitador*. Como já detalhado nas seções 3.6 e 3.7, foram identificadas quatro categorias que podem aparecer como saídas da adoção de *DevOps*: ***agilidade***, ***resiliência***, ***garantia da qualidade*** e ***medição contínua***.

Convém destacar ainda que, em algumas situações, os potenciais gerados pela adoção de *DevOps* podem não ser completamente explorados em um primeiro momento devido a decisões de negócio. Por exemplo, um dos entrevistados citou que a sua companhia não permitiu uma entrega contínua em ambiente de produção:

> "*Nós tínhamos condições e nos sentíamos seguros para publicar continuamente também em produção, mas, no começo os gerentes ficaram meio assustados e decidiram que a publicação deveria acontecer apenas semanalmente.*" (P9, Gerente de TI, Brasil)

Hipótese 3: *As categorias* **medição contínua** *e* **garantia da qualidade** *são relacionadas à adoção de DevOps tanto como* ***facilitadores*** *como quanto* ***saídas***.

Os conceitos relacionados à medição são citados como responsabilidades típicas da equipe de operações. Ao mesmo tempo que compartilhar essa responsabilidade entre os times reduz os silos, também é mencionado que a ***medição contínua*** é uma consequência necessária da adoção de *DevOps*. Particularmente porque a entrega contínua de *software* requer maior controle, que é fornecido por meio da utilização dos conceitos relacionados à esta categoria. A mesma premissa é válida para a categoria ***garantia da qualidade***. À primeira vista, ***garantia da qualidade*** aparece como uma resposta ao contexto de agilidade nas operações decorrente da adoção do *DevOps*. Mas, os esforços na garantia de qualidade de produtos de *software* também são responsáveis por aumentar a confiança entre as equipes de desenvolvimento e operações, aumentando o nível da ***cultura de colaboração***.

Hipótese 4: *Não há precedência entre os facilitadores em um processo de adoção de DevOps.*

Foi identificado que o processo de adoção de *DevOps* pode não ter que priorizar algum dos facilitadores. Há casos de adoção de *DevOps* com maior ênfase em automação e

outros com maior ênfase em garantia da qualidade. Assim sendo, uma organização que visa adotar *DevOps* deve começar com os facilitadores que parecem mais apropriados em termos das suas especificidades. Assim, não foi encontrada nenhuma evidência de que um facilitador é mais eficiente que outro para fomentar a ***cultura de colaboração***. ***Automação***, por exemplo, é a categoria que aparece com maior frequência na análise, todavia, vários participantes mencionaram que consideram um equívoco associar *DevOps* à automação.

> "*Eu penso que a expansão da colaboração entre as equipes envolveu outras coisas, não foi apenas automação. É preciso ter um alinhamento com as necessidades de negócio. (...) Eu acho que DevOps possibilitou inclusive um entendimento mais amplo da produção do software como um todo e a gente percebeu exatamente que não se trata de sair automatizando tudo. (...) Então, vejo com cautela uma suposta visão que automatizar as coisas pode ser a maneira de implementar DevOps.*" (P7, Analista de Suporte, Brasil)

> "*Embora atualmente a gente use automação em um número até razoável de cenários, nós conseguimos desenvolver nossa cultura significativamente com coisas que não envolvem automação e eu penso que você pode sim conseguir um bom nível de DevOps com pouco ou talvez até nada de automação.*" (P8, *DevOps Engineer*, Brasil)

3.8.2 Um Modelo para Adoção de *DevOps*

Com base nas hipóteses H1-H4 apresentadas, foi construído um modelo de três etapas para guiar a adoção de *DevOps*. A Figura 3.4 representa graficamente o modelo, cujas etapas são descritas na enumeração a seguir:

1. Na primeira etapa, a companhia interessada em adotar *DevOps* deve compreender e disseminar que o objetivo principal é o estabelecimento de uma ***cultura de colaboração*** entre os times de desenvolvimento e operações.

2. Na segunda etapa, devem ser selecionados e desenvolvidos os facilitadores mais adequados para o contexto da organização. Os facilitadores são meios tipicamente utilizados para desenvolver a ***cultura de colaboração*** e seus conceitos relacionados.

3. Por fim, na terceira etapa, a organização deve verificar as saídas que o processo está produzindo, visando alinhá-las com a prática de mercado e explorá-las de acordo com a sua necessidade.

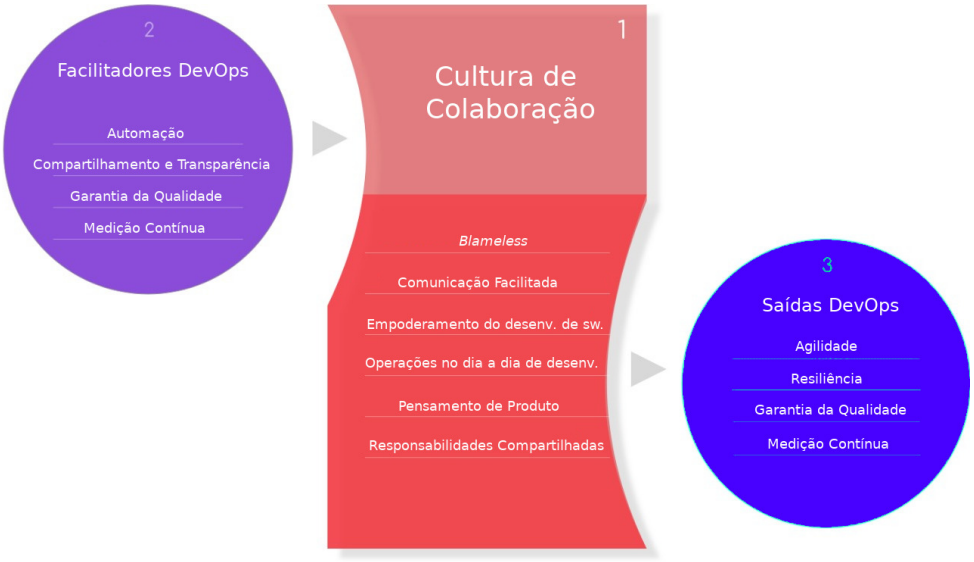

Figura 3.4: Modelo para Adoção de *DevOps*.

3.8.3 Trabalhos Relacionados

Seguindo os preceitos da metodologia *Grounded Theory*, o contato com a literatura existente sobre *DevOps* foi aprofundado apenas ao final da produção da teoria apresentada nas seções anteriores. Aqui são apresentados os principais trabalhos relacionados com indicação das principais semelhanças e diferenças entre o trabalho aqui apresentado e os demais.

O trabalho de Smeds et al. [29] propôs a existência de *enablers* técnicos e culturais que compõem a definição de *DevOps*, e *capabilities* resultantes do uso das técnicas *DevOps*. Estes *enablers* e *capabilities* se assemelham aos *facilitadores* e *saídas DevOps* apresentados aqui. Ademais, alguns conceitos coincidem: (1) automação de testes, *deployment*, monitoramento, infraestrutura e recuperação; (2) integração, testes e *deployment* contínuos; (3) recuperação de falhas em serviços sem *delay*; e (4) comunicação constante e *effortless*. Todavia, algumas outras diferenças também podem ser identificadas: (1) o não agrupamento de conceitos em categorias, a maior parte dos facilitadores são relacionados a automação; (2) apresentam os aspectos culturais como responsáveis por contribuir com a formação de capacidades *DevOps* e não como a preocupação principal de *DevOps*; (3) o foco da parte empírica do estudo, onde foram conduzidas entrevistas semi-estruturadas com 13 funcionários de uma única companhia cujo processo de adoção de *DevOps* estava em um estágio inicial, foi em construir uma lista de possíveis impedimentos na adoção de *DevOps* e não em prover orientações para novos praticantes a respeito de *como* se adotar *DevOps*.

Já Lwakatare et al. [34, 35] propuseram um *framework* conceitual para explicar "*DevOps* como um fenômeno". O *framework* foi construído por meio de uma revisão de literatura multivocal, utilizando dados da literatura cinza, combinada com entrevistas a praticantes de três companhias que estavam aplicando práticas *DevOps*; organizado em torno de cinco dimensões (colaboração, automação, cultura, monitoramento e medição); e estas dimensões foram apresentadas com práticas relacionadas. O foco deste trabalho também não foi em investigar a adoção de *DevOps* na prática. A principal semelhança identificada foi a presença de todas as dimensões aqui também, embora *cultura* e *colaboração* sejam aqui apresentadas como uma única abstração (cultura de colaboração) e os conceitos relacionados a monitoramento estejam aqui presentes como parte da categoria de ***medição contínua*** e não em duas categorias separadas. Ademais, infraestrutura como código foi lá apresentada como um conceito relacionado a automação enquanto que aqui está relacionado a ***compartilhamento e transparência***.

França et al. [6], por sua vez, realizaram uma revisão multivocal de literatura, utilizando dados de diversas fontes, incluindo a literatura cinza, com procedimentos de análise qualitativa de *grounded theory* para prover uma definição para *DevOps* e identificar práticas *DevOps*, habilidades requeridas, características, benefícios, e problemas motivando a sua adoção. O uso de *grounded theory* representa uma semelhança entre os estudos, embora, neste caso tenham sido utilizados apenas os procedimentos de análise e não a abordagem como um todo. Os resultados também apresentam algumas semelhanças: (1) automação, compartilhamento, medição e garantia da qualidade são apresentados em ambos os estudos como categorias; (2) a categoria *aspectos sociais* apresentada lá é similar à categoria *cultura de colaboração* apresentada aqui. Como diferenças, destacam-se: (1) o trabalho não realizou entrevistas para coletas de dados; (2) os resultados lá são apresentados como um conjunto de princípios *DevOps* enquanto que aqui são conjuntos de facilitadores e saídas *DevOps*; e (3) a categoria *leanness* é apresentada lá e não aqui, enquanto que *resiliência* é apresentada aqui e não lá.

A primeira parte do trabalho de Erich et al. [36] se concentrou em identificar elementos que compõem *DevOps* por meio de uma revisão de literatura. Cinco das sete categorias apresentadas como resultado dessa parte do trabalho também foram apresentadas aqui: cultura de colaboração, automação, medição, compartilhamento e garantia da qualidade. Uma segunda parte deste trabalho se concentrou em investigar como *DevOps* é implementado na prática por meio de entrevistas com praticantes de seis companhias. O propósito inicial e a utilização de entrevistas para coleta de dados são aspectos similares aos aqui apresentados, todavia, as percepções foram lá apresentadas de maneira individual para cada companhia, não direcionando os pontos comuns com o intuito de formular um modelo geral de como *DevOps* pode ser adotado.

Já no trabalho de Hamunen [32], a primeira parte foi dedicada a identificar os componentes chave de *DevOps* por meio de consultas à literatura. Como resultado foi apresentado o modelo "CALMS": cultura, automação, *lean*, medição e compartilhamento. À exceção de *lean*, os demais componentes aparecem como categorias da adoção de *DevOps* aqui também. A segunda parte buscou identificar os principais desafios que as organizações enfrentam ao lidar com *DevOps*, onde foram realizadas nove entrevistas com profissionais com experiência em iniciativas *DevOps* no mercado finlandês. Embora o foco deste trabalho tenha sido em identificar os desafios, há uma sobreposição quando, mesmo que secundariamente, são providos meios de se superar estes desafios. Por exemplo, é mencionado que o principal avanço tecnológico em *DevOps* é a criação de um *pipeline* automatizado de entrega contínua, que contém aspectos dos facilitadores *DevOps* presentes aqui. Destaca-se também que o estudo aponta como necessária para superar os desafios de *DevOps* a adaptação dos processos organizacionais, em uma linha similar ao proposto na categoria aqui denominada de cultura de colaboração.

O estudo de Feijter et al. [52] resultou na proposição de um modelo de maturidade para adoção de *DevOps*. Este estudo possui diversas semelhanças com o aqui realizado: (1) o foco foi exatamente em explicar como evoluir no nível de maturidade de *DevOps* em alguma organização; (2) foram realizadas entrevistas com praticantes para obtenção desse modelo de maturidade; (3) a motivação para a pesquisa envolveu a necessidade de uma companhia de desenvolvimento de *software* específica; (4) os procedimentos de análise de dados envolveram um modelo de comparação constante que em muito se assemelha com os procedimentos de codificação de *grounded theory*; e (5) foram realizados *workshops* com especialistas da organização para avaliar níveis de maturidade de *DevOps* e validar a aplicabilidade do modelo. Os estudos diferem quanto ao método de pesquisa utilizado, enquanto lá foi utilizado o *Information Systems Research Framework* da *ciência do design*, aqui foi utilizada *Grounded Theory*. Enquanto que aqui a teoria foi construída utilizando como base unicamente as percepções obtidas nas entrevistas, lá foi utilizado um modelo de competências previamente existente no âmbito da empresa interessada na pesquisa e, das 14 entrevistas realizadas, 12 foram com profissionais da própria empresa. Em termos de resultados, inicialmente lá são apresentados seis *DevOps drivers* que servem como base para a construção do modelo de maturidade: criar uma (1) cultura de colaboração, (2) agilidade e alinhamento de processos, (3) automação, (4) maior qualidade, (5) desenvolvimento e implantação de aplicações baseadas em nuvem, e (6) melhoria contínua. Os três primeiros são similares às categorias ***cultura de colaboração***, ***agilidade*** e ***automação*** aqui apresentadas. A partir desses *DevOps drivers*, e de uma versão aprimorada do modelo de competências já existente anteriormente na empresa, foi então apresentado um modelo de maturidade *DevOps*. No modelo de competências constam as perspectivas de

stakeholders internos e externos e três áreas focais de *DevOps*: (1) fundação, (2) produto, processo e qualidade e (3) cultura e colaboração. Cada uma dessas áreas focais possui *capacidades* relacionadas, às quais foram atribuídos diferentes níveis de maturidade. Por exemplo, *comunicação* é uma *capacidade* que faz parte da área focal *cultura e colaboração*. No modelo de maturidade foram atribuídos 5 possíveis níveis de maturidade para comunicação. A quantidade de possíveis níveis de maturidade varia de acordo com cada *capacidade*, de modo que podem existir 3, 4, 5 ou 6 níveis. Entre as 16 capacidades lá listadas, 6 possuem conceitos similares também identificados aqui: (1) comunicação, (2) compartilhamento de conhecimento, (3) melhoria da qualidade do desenvolvimento, (4) automação de testes, (5) automação do *deployment* e (6) infraestrutura. Os níveis de maturidade lá apresentados podem ser utilizados aqui para aprimorar a compreensão dos elementos.

Considerações Gerais a Respeito dos Trabalhos Relacionados

Em relação aos trabalhos cujo propósito é apresentar elementos *DevOps* e práticas relacionadas, a teoria apresenta detalhes que ajudam a compreender como podem ser respondidas algumas questões práticas a respeito da adoção de *DevOps* que permaneciam em aberto: (1) Existe um caminho recomendado para se adotar *DevOps*? (2) Já que *DevOps* é constituído de múltiplos elementos, eles possuem a mesma relevância quando se adota *DevOps*? (3) Qual é o papel desempenhado por cada um desses elementos - tais como medição, compartilhamento e automação - em uma adoção de *DevOps*?

Dos onze elementos *DevOps* listados no capítulo 2, cinco foram aqui confirmados. Os elementos *serviços* e *leanness* aparecem como aspectos de outras categorias, não foram identificados padrões de conceitos que culminassem na geração de categorias próprias para eles. O elemento *monitoramento* aqui faz parte da categoria ***medição contínua***. Já quanto aos demais elementos (estruturas, padrões e governança) não foram aqui identificadas características relevantes.

De maneira geral, é possível destacar que: (1) nenhum dos estudos utilizou *grounded theory* para investigar o processo de adoção de *DevOps*; (2) exceto o trabalho de Feijter et al. [52], os demais focaram em caracterizar *DevOps* de maneira geral ou desafios relacionados a *DevOps* e não em produzir orientações relacionadas ao processo de adoção em si; (3) entre os trabalhos que realizaram entrevistas, o que maior número de companhias distintas ouviu foi o de Hamunen [32], com 9 companhias, enquanto que aqui foram coletadas experiências de 15 companhias.

Capítulo 4

Adoção de *DevOps* no TCU

Neste capítulo é detalhada a realização de um grupo focal [19, 20] para identificar a percepção de profissionais envolvidos tanto a respeito da adoção de *DevOps* em geral, quanto da aplicabilidade e utilidade do modelo proposto na adoção de *DevOps* no TCU. Este capítulo contém ainda detalhamento das ações citadas no grupo focal que foram desenvolvidas no TCU após o intercâmbio de experiências ocorrido durante a elaboração do modelo.

4.1 Metodologia

Para obtenção da avaliação empírica sobre a adoção de *DevOps* no TCU, bem como da aplicabilidade e utilidade do modelo descrito, proposta como objetivo deste trabalho de pesquisa, foi utilizado o método qualitativo *grupo focal* [19].

A população atualmente envolvida no processo de adoção de *DevOps* no TCU é relativamente pequena (em torno de 12 desenvolvedores e 6 profissionais do time de operações) e há um maior número de profissionais nos times de desenvolvimento do que no time de operações. Para que as percepções obtidas na avaliação não possuíssem o viés do time de desenvolvimento, foi adotada a estratégia de ouvir um número similar de profissionais de cada um dos times. Com essa limitação do número de profissionais do time de desenvolvimento que poderiam ser ouvidos na pesquisa, o número total de participantes na amostra (estimado entre 8 e 12 pessoas) mostrou-se pequeno para realização de um questionário.

Ao mesmo tempo, a realização de entrevistas, além de alongar a obtenção do *feedback* desejado, poderia gerar um grande número de respostas repetitivas, uma vez que são todos profissionais envolvidos no mesmo processo de adoção de *DevOps*. Optou-se, portanto, por se realizar um grupo focal.

Grupo focal emergiu como método de pesquisa nas ciências sociais nos anos 1950 e atualmente é amplamente utilizado, por exemplo, em estudos sociológicos, pesquisas de

mercado, planejamento de produtos e estudos de usabilidade de sistemas [20]. Morgan [53] define grupo focal como uma técnica de pesquisa que coleta dados através da interação de grupo em um tópico específico determinado pelo pesquisador.

Segundo F. Shull et al. [20], grupos focais tipicamente possuem entre três e doze participantes, são projetados para obter percepções pessoais de membros de um ou mais grupos envolvidos em uma área definida de interesse de pesquisa e possuem como benefícios a produção de informações cândidas, muitas vezes perspicazes, com um baixo custo e rápida execução. Estas características tornam o grupo focal uma alternativa adequada aos propósitos desta pesquisa. Ainda segundo os autores, a discussão é guiada e facilitada por um pesquisador-moderador que segue uma estrutura predefinida de questionamentos.

O grupo focal realizado no TCU contou com a participação de quatro profissionais, cujo perfil está descrito na Tabela 4.1.

P#	**Time de atuação**	**Formação**	**Experiência**
P1	Desenvolvimento	Graduação	Atua há 3 anos no desenvolvimento de *software* do TCU e possui 9 anos de experiência prévia
P2	Desenvolvimento	Pós-graduação	Atua há 6 anos no desenvolvimento de *software* do TCU e possui 7 anos de experiência prévia
P3	Operações	Graduação	Atua há 3 anos no time de operações do TCU e possui 8 anos de experiência prévia
P4	Operações	Graduação	Atua há 3 anos no time de operações do TCU e possui 10 anos de experiência prévia

Tabela 4.1: Participantes do Grupo Focal

Seguindo uma estrutura similar ao realizado por Lehtola et al. [54], o grupo focal foi conduzido da seguinte forma: o pesquisador-moderador atuou como facilitador do grupo focal fornecendo aos participantes três tópicos de discussão, listados na Tabela 4.2. No início da discussão de cada tópico, as perguntas foram apresentadas aos participantes que escreveram suas ideias e palavras-chave em notas *post-it*. Depois disso, as notas foram postas em um quadro branco e serviram como ponto de partida para a realização de discussões sobre o respectivo tópico com o propósito de se obter conclusões a respeito da respectiva pergunta. Os resultados das discussões de cada tópico são apresentados na próxima seção.

A gravação do grupo focal foi proposta aos participantes e recusada. Como os times de desenvolvimento e operações do TCU atualmente são partes de secretarias diferentes, foi levantado pelos participantes que a gravação necessitava ser tratada diretamente com

ambos os secretários. Ademais, foi apontado que a gravação também limitaria a exposição de opiniões e ideias pelo receio existente de gerar alguma interpretação indevida.

Foi então acordado que seria produzida uma ata de reunião cotendo o detalhamento das discussões, e que essa ata deveria ser enviada a todos os participantes para aprovação. Assim sendo, os resultados apontados nas seções 4.2 a 4.4 foram validados pelos participantes por meio da ratificação da ata que a todos foi enviada.

4.2 Estágio Atual da Adoção de *DevOps* no TCU

O primeiro tópico de discussão no grupo focal foi o estágio atual da adoção de *DevOps* no TCU, onde foram discutidas as ações já desenvolvidas e quais problemas essas ações resolveram. A seguir são enumerados os resultados da discussão neste tópico.

Disponibilização de VMs para Ferramentas de Desenvolvimento

A primeira ação indicada e discutida foi a disponibilização de ambientes (VMs) para instalação de ferramentas que são relacionadas ao trabalho de desenvolvimento. Essa ação foi exemplificada com as recentes instalações bem sucedidas das ferramentas Elasticsearch[1] e

[1]https://www.elastic.co/

	Tópico	**Perguntas**
1	Estágio atual da adoção de *DevOps* no TCU	1. Quais ações já desenvolvidas no TCU você considera que fazem parte da adoção de *DevOps*? 2. Quais problemas existentes anteriormente foram resolvidos por essas ações?
2	Aplicabilidade e utilidade do modelo proposto	1. Você considera que este modelo que foi aprovado no CPA tem contribuído para a adoção de *DevOps* no TCU? 2. Caso positivo, quais as principais contribuições?
3	Desafios enfrentados e próximos passos na adoção de *DevOps*	1. Quais os principais desafios que a adoção de *DevOps* enfrenta atualmente no TCU? 2. Quais os próximos passos para a adoção de *DevOps* no TCU?

Tabela 4.2: Tópicos do Grupo Focal

Kafka[2]. O problema existente anteriormente era que quando o desenvolvedor necessitava de uma ferramenta desse tipo, inerente ao trabalho que realizava e necessária para atender de maneira adequada problemas específicos, dependia de abrir uma solicitação para que o time de operações a provesse, com prazos muitas vezes tão dilatados que inviabilizavam o uso da solução mais adequada. Com o provisionamento de VMs e cooperação entre os dois times, essas ferramentas ficaram disponíveis rapidamente para uso e são administradas de maneira conjunta. Este é um claro exemplo de aplicação dos conceitos de **empoderamento do desenvolvimento de *software*** e **responsabilidade compartilhada** da categoria principal ***cultura de colaboração***.

Execução Conteinerizada de Aplicações

Na sequência do grupo focal, a utilização de contêineres para execução de aplicações foi debatida. O primeiro problema que essa ação solucionou, no entendimento dos participantes, foi a anterior falta de paridade entre os ambientes (desenvolvimento, aceite, pré-produção e produção). Foram relembrados os problemas recorrentes de aplicações que funcionavam em ambiente de desenvolvimento mas apresentavam problemas em produção, o que deixou de acontecer a partir do uso dos contêineres.

Em seguida foi apontado que a disponibilização do *Dockerfile*[3] no repositório de código-fonte de cada projeto possibilita que tanto os desenvolvedores como o pessoal de operações possam ter uma primeira ideia sobre o ambiente de execução de cada aplicação.

Ainda sobre o uso de contêineres, foi debatida a exploração dos recursos providos pelo *Kubernetes*[4] nas aplicações. Com *Kubernetes* foi possível avançar ainda mais na disponibilização de infraestrutura como código. Foi relembrado que o Comitê Permanente de Integração Contínua (CPIC) do TCU determinou que as aplicações que executem em contêiner devem possuir um diretório chamado de *k8s* na raiz do repositório de código-fonte, onde devem constar os arquivos de configuração do *Kubernetes*. A aplicação do conceito de *deployment* da ferramenta tem permitido a configuração de mecanismos para escalabilidade horizontal, alocação de recursos para os contêineres e alta disponibilidade. Foi também citado que a publicação de aplicações sem *down-time*, possibilitada pelos mecanismos do *Kubernetes*, resolveu o problema de interromper o trabalho das secretarias estaduais de controle externo que possuem fuso horário em relação a Brasília.

[2]https://kafka.apache.org/

[3]Arquivo que descreve os passos a serem executados para construção de uma imagem *Docker* que, ao ser executada, instancia um contêiner

[4]Ferramenta de orquestração de contêineres: https://kubernetes.io/

Foi ainda relembrado que as experiências coletadas durante a pesquisa ajudaram na decisão sobre qual ferramenta de orquestração de contêineres utilizar, haja vista que havia uma dúvida entre *Kubernetes* e *Cattle*[5].

Neste ponto da discussão, foi possível identificar a aplicação de sete conceitos: (1) paridade entre ambientes, (2) automação do provimento da infraestrutura, (3) serviços autônomos, (4) conteinerização, (5) auto *scaling*, (6) automação da recuperação e (7) zero *down-time*.

Desburocratização da Comunicação

A experiência obtida em outras organizações que adotaram *DevOps* possibilitou o direcionamento de evitar o uso de meios burocráticos de comunicação. Durante este tópico foi relembrado o processo de comunicação extremamente cerimonioso contido no âmbito da publicação programada. Há uma diretriz atual de se evitar o uso do *servicedesk* para resolução de problemas simples. O uso do *Slack* tem sido institucionalizado e facilitado o contato entre os dois times.

Aqui, é possível notar a aplicação do conceito de **comunicação facilitada** da categoria ***cultura de colaboração***.

Pipelines Multidisciplinares

As aplicações mais recentes do TCU executam *pipelines* multidisciplinares cujas etapas envolvem desde o *build*, passando por testes automatizados e análise estática de código fonte, execução dos contêineres utilizando *Kubernetes* e publicação de maneira isonômica nos diversos ambientes (desenvolvimento, aceite e produção). Estes *pipelines* são acessíveis por meio de um *Jenkinsfile* na raiz do repositório de código-fonte de cada projeto e são automaticamente acionados a partir de cada *push* executado. A Figura 4.1 contem um exemplo de *pipeline* em execução em uma aplicação real do TCU. Na visão dos participantes do grupo focal, a implementação efetiva destes *pipelines* possibilita a entrega contínua de *software* solucionando diversos problemas contidos na publicação programada, principalmente as recorrentes demoras para publicação em produção de funcionalidades já prontas. No caso de falha em qualquer das etapas do *pipeline*, ele é interrompido e uma mensagem é enviada automaticamente por meio do *Slack*.

A utilização de *Jenkinsfile* como maneira de se implementar um *pipeline* compartilhado é outro exemplo de ação que decorreu diretamente do intercâmbio de experiências com praticantes de *DevOps* durante a condução deste trabalho de pesquisa.

[5]https://github.com/rancher/cattle

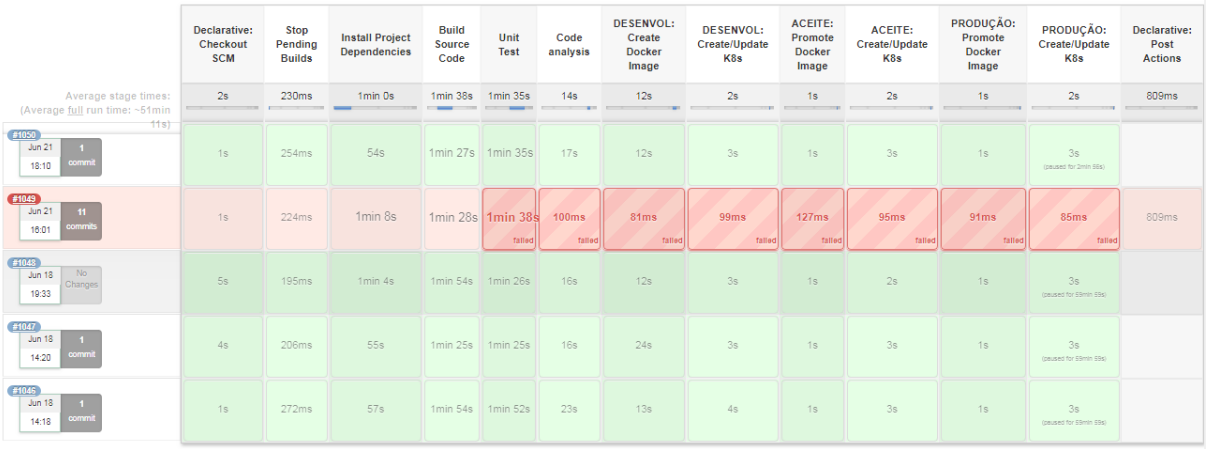

Figura 4.1: *Pipeline* em Execução no TCU.

Aqui, é possível identificar a aplicação dos seguintes conceitos presentes no modelo sobre a adoção de *DevOps*: (1) operações no dia a dia de desenvolvimento, (2) automação de testes, (3) automação do *deployment*, (4) automação do provimento da infraestrutura, (5) conteinerização, (6) infraestrutura como código, (7) *pipelines* compartilhados, (8) integração contínua, (9) provimento contínuo da infraestrutura, (10) *deployment contínuo*, (11) teste contínuo e (12) análise estática de código-fonte.

Versionamento de Bancos de Dados Relacionais

Nesse momento do debate, foi relembrado que o uso de ferramentas de versionamento de bancos de dados havia sido anteriormente declarado como incompatível com o processo de desenvolvimento de *software* do TCU. Apenas com a aproximação dos times, foi possível gerar uma compreensão de quais procedimentos eram executados quando se solicitava a alteração de uma estrutura de banco de dados. Essencialmente, foi identificado que as alterações de bancos de dados geravam atualizações nos mecanismos de auditoria executados pelo time de operações. Com essa compreensão, foi possível automatizar essas atualizações de auditoria viabilizando o uso da ferramenta para versionamento de bancos de dados escolhida, que é o *Flyway*[6]. Atualmente, quando se necessita, por exemplo, criar ou alterar uma tabela no banco de dados, basta incluir o *script* equivalente no diretório adequado do repositório de código fonte. Anteriormente era necessário abrir uma solicitação no *servicedesk* e sincronizar os horários e procedimentos para que pudessem ser executados na publicação programada.

O uso do *Flyway* é um exemplo da aplicação do conceito de **automação do gerenciamento da infraestrutura** da categoria ***automação***.

[6]https://flywaydb.org/

Monitoramento Automatizado e Contínuo

A respeito do monitoramento das aplicações, especificamente quando da ocorrência de erros em tempo de execução, a solução construída de monitoramento automatizado e contínuo foi destacada como parte das ações já desenvolvidas e que resolveu o problema da falta de proatividade na correção de erros existente anteriormente. A obtenção de acesso aos *logs* de aplicações anteriormente precisava ser solicitada pelos desenvolvedores quando necessário, tipicamente no momento da resolução de algum incidente.

Esta solução também foi implementada como fruto direto da experiência coletada sobre a adoção de *DevOps* em uma das companhias entrevistadas durante a condução desta pesquisa. A solução envolve a execução de um procedimento utilizando a ferramenta *Fluentd*[7] para coleta dos *logs* de aplicações e carga no banco de dados não relacional *Elasticsearch*. Após essa carga, a ferramenta *ElastAlert*[8] executa buscas semânticas pré-configuradas e, em caso de ocorrência de algum padrão nos *logs* de aplicações (mensagem de erro, por exemplo), envia automaticamente uma mensagem aos times de desenvolvimento e operações por meio do *Slack*. A Figura 4.2 ilustra a solução implementada no TCU.

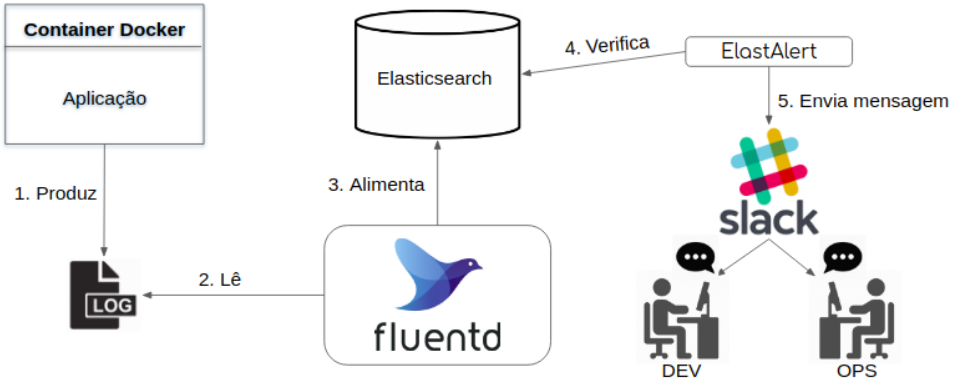

Figura 4.2: Solução do TCU para Monitoramento Contínuo.

Trata-se da aplicação de dois dos conceitos existentes no modelo: (1) automação do monitoramento e (2) monitoramento de *logs* de aplicações.

4.3 Aplicabilidade e Utilidade do Modelo Proposto

Todos os participantes do grupo focal concordaram que o modelo proposto possui grande utilidade para a adoção de *DevOps* do TCU. Relembraram que boa parte das ações dis-

[7]https://www.fluentd.org/
[8]http://elastalert.readthedocs.io/

cutidas no tópico anterior foram fruto direto do desenvolvimento deste modelo e que, portanto, consideram que a sua aplicação já está sendo efetiva e produzido resultados satisfatórios no sentido de colaborar para a ampliação do uso de *DevOps* no desenvolvimento das aplicações corporativas do TCU. A seguir, são apresentados os dois principais benefícios da utilização do modelo, discutidos durante o grupo focal.

Compreensão Institucional sobre *DevOps*

Como resposta à pergunta sobre as principais contribuições do modelo para o TCU, inicialmente foi apontado que, durante provas de conceito realizadas anteriormente (que foi o primeiro encaminhamento dado pelo CPA para cumprimento do indicador do PDTI sobre *DevOps*), ficou nítido que o simples uso das ferramentas não estava aproximando os times, que alguns desenvolvedores estavam agindo como se houvesse um salvo conduto para "passar por cima" dos procedimentos do time de operações e que o pessoal de operações estava bastante preocupado em delimitar formalmente as fronteiras de responsabilidade quanto à administração das ferramentas. O avanço da discussão mostrou que todos os participantes concordaram com o entendimento de que fomentar a cultura de colaboração não era um fator levado em conta pela maioria dos profissionais envolvidos, e que ver, através do modelo, que a adoção de *DevOps* no mercado passa principalmente por este ponto, tem possibilitado uma mudança de postura das equipes, no sentido de colaborar mais ao invés de tentar defender interesses próprios de cada uma das equipes.

Na sequência, foi discutido o conteúdo da nota *post-it* que continha uma anotação a respeito da "ampla gama de práticas e experiências" presentes no modelo. Foi mais uma vez relembrado que diversas práticas já foram implementadas usando como insumo experiências de mercado coletadas durante a produção do modelo. O modelo também foi apontado como uma ferramenta para avaliação de práticas que o TCU ainda não adota, fornecendo um *roadmap* robusto para guiar os próximos passos.

Experiências de Mercado

Por fim, foi destacado que o modelo foi construído levando em conta experiências bem sucedidas no mercado e que isso representa grande valor para o TCU. No entendimento do grupo, embora o TCU possua muitas das peculiaridades presentes em ambientes governamentais, a busca pela inovação tecnológica faz parte do mapa estratégico do órgão e não pode ser alcançada olhando-se apenas para cenários similares ao atual. Foi ressaltado que o mercado é um importante ator na definição de novas tecnologias que, adaptadas em maior ou menor grau, podem ser plenamente aplicáveis aos órgãos governamentais, como o TCU. Segundo o entendimento formado, o fato de um modelo construído com base em

experiências de mercado estar sendo efetivamente aplicado, é mais uma constatação de que essa premissa é verdadeira.

4.4 Desafios Enfrentados e Próximos Passos na Adoção de *DevOps*

As discussões do último tópico do grupo focal se concentraram em identificar os desafios enfrentados na evolução do uso de *DevOps* no TCU, bem como nos próximos passos para superar os desafios e institucionalizar *DevOps* como abordagem de desenvolvimento.

Maturidade do Entendimento Interno sobre *DevOps*

Inicialmente foi debatida a percepção apontada por um dos participantes de que ainda há muita *hype* em torno do que seria a adoção de *DevOps*. Que muitos desenvolvedores ainda pensam que isso possibilita a tomada de iniciativas técnicas sem consultar outros profissionais e que algumas pessoas de operações ainda não se sentem confortáveis com essa mudança de paradigma, pois entendem que *DevOps* pode provocar uma desorganização em um ambiente que já possuia estabilidade. Foi mencionado que o modelo apresentado ajuda a lidar com esse desafio, mas que é necessária uma melhor conscientização de todos os profissionais sobre fomentar a colaboração entre os times e não apenas sair querendo resolver tudo de acordo com convicções pessoais.

Foi apontada também a dificuldade de disseminar o conhecimento relacionado às novas ferramentas e processos que vieram junto com a adoção de *DevOps*. Ações para mitigação desse desafio foram discutidas, incluindo a ampliação das palestras internas do TCU que são chamadas internamente de bate-bola técnico, a participação nos eventos tais qual o *DevOpsDays*, e os estímulos que o TCU já dá para os servidores, como licença capacitação, reembolso de treinamentos e disponibilização da plataforma *safari books*. Nesse sentido, formou-se o entendimento de que é um dos próximos passos a ampliação da capacitação técnica das equipes nos temas relativos à modernização das ferramentas e processos.

Segurança da Informação

Aqui, foi debatido que a adoção de *DevOps* aumentou consideravalmente a superfície de vulnerabilidades que o TCU possui. Foi apontado que o time de operações tem grande preocupação com a segurança da informação, que estão avaliando algumas das ferramentas implantadas e que irão propor modificações. Foi então alinhado que este debate não pode ficar apenas no time de operações, pois isso é exatamente uma manifestação da falta de colaboração.

Por iniciativa de um dos participantes, foi então debatido que essas preocupações ampliam um pouco do escopo de *DevOps*, indo para um contexto de *DevSecOps*, quando as atividades de segurança também são integradas ao processo de desenvolvimento. Tem-se, então, mais um próximo passo, que é a ampliação da perspectiva de *DevSecOps*.

Coleta de Métricas em Aplicações

Nessa parte do debate, foi apontado que a solução de monitoramento contínuo atual se restringe aos erros das aplicações, e que o modelo contém ideias a respeito da coleta de métricas em aplicações para fomentar evoluções e decisões de negócio. Foi pontuado por um dos participantes que a mesma solução pode ser ampliada, desde que as aplicações sejam instrumentadas para gerar *logs* de quaisquer outras métricas. Por conseguinte, a coleta contínua de outras métricas de aplicações foi apontado como mais um dos próximos passos da adoção de *DevOps* do TCU.

Portarias de Responsabilidade

Aqui, foi apontado que, embora o modelo tenha possibilitado a compreensão de que o mais importante deve ser fomentar a ***cultura de colaboração***, muitos profissionais do órgão ainda pensam de maneira mais formal e a portaria na qual consta a estrutura organizacional do TCU estabelece que as responsabilidades por questões relacionadas à infraestrutura de aplicações são do SINAP, o que dificulta a consolidação de um senso de responsabilidade compartilhada.

Não houve um consenso a respeito de qual a melhor solução para resolver as restrições contidas nas portarias de estruturação organizacional. Alguns (P1 e P4) entendem que seria adequado que o setor de operações (SINAP) fosse transferido para a Secretaria de Soluções de TI (atualmente é parte da Secretaria de Infraestrutura de TI); outros (P2 e P3) demonstraram o entendimento de que basta uma alteração na portaria para definir que há responsabilidade compartilhada por questões relacionadas a infraestrutura de aplicações. Foi lembrado que há um grupo de trabalho constituído no intuito de propor modificações nas portarias para ajustar essas atribuições ao cenário de *DevOps*.

Distanciamento Físico das Equipes

A última dificuldade apontada foi que as equipes de desenvolvimento e operações atualmente atuam em salas separadas. A distância física foi posta como um fator que dificulta a comunicação e atrapalha a formação da cultura de colaboração. Os participantes concordaram que a aproximação física das equipes passa pela questão da reestruturação das portarias discutidas acima. Caso o setor de operações seja incorporado à Secretaria de

Soluções de TI, é provável que a aproximação ocorra, caso contrário, é necessário buscar uma outra solução viável.

4.5 Considerações Gerais

O grupo focal não se concentrou especificamente na avaliação do modelo proposto nesta pesquisa porque a adoção de *DevOps* no TCU não envolve apenas esta pesquisa. O modelo é mais um mecanismo na busca pela maturação do uso de *DevOps* no desenvolvimento das aplicações corporativas do TCU.

Mesmo não tendo sido o único ponto de debate, considerações a respeito do modelo permearam todos os tópicos debatidos, foram destacadas ações práticas que só se materializaram devido ao intercâmbio de experiências ocorridas durante a pesquisa. Dos 34 conceitos apresentados no modelo, 23 foram visualizados durante os debates do grupo focal. Ademais, foi identificada uma conscientização a respeito da cultura de colaboração que não existia anteriormente, as ações têm sido compreendidas como parte dos esforços para se fomentar a cultura de colaboração.

Por fim, cabe ressaltar que a versão final do modelo foi introduzida há pouco tempo (em torno de 1 mês) no TCU e que uma percepção mais robusta do seu impacto requer mais tempo para se formar. Duas ações específicas foram realizadas para divulgação: (1) apresentação em um bate-bola técnico (*tech talk*) para todos os profissionais; e (2) apresentação para o Comitê Permanente de Arquitetura (CPA) em uma das suas reuniões quinzenais, oportunidade em que foi decidido que a adoção de *DevOps* no TCU deve ser pautada por este modelo.

Capítulo 5

Conclusões

Neste capítulo são apresentadas as considerações finais do trabalho de dissertação contendo uma retrospectiva da pesquisa (seção 5.1) e a indicação de limitações e possíveis trabalhos futuros (seção 5.2).

5.1 Retrospectiva da Pesquisa

Os problemas existentes no TCU decorrentes da baixa colaboração entre seus times de desenvolvimento e operações durante o desenvolvimento de *software* culminaram na inclusão de um indicador no PDTI do órgão tratando especificamente da ampliação do uso de *DevOps* no desenvolvimento das suas aplicações corporativas. Decorrente da necessidade de cumprimento deste indicador, o CPA, comitê responsável por orientar e padronizar as decisões tecnológicas e arquiteturais do TCU, direcionou que esse processo de adoção de *DevOps* deveria ser conduzido pautado no intercâmbio de experiências com praticantes de mercado que foram bem sucedidos na adoção de *DevOps*.

Alinhado com as necessidades do TCU, neste trabalho de pesquisa foram ouvidos 15 praticantes que contribuíram para a adoção de *DevOps* em suas companhias e, por meio da utilização da metodologia *grounded theory*, foi produzida uma teoria que explica como *DevOps* foi adotado nessas companhias e um modelo para guiar novas adoções de *DevOps* com base nessas experiências. Por meio de uma análise da literatura realizada após a condução do estudo, foi identificado que a teoria e o modelo representam uma contribuição acadêmica para a compreensão de *DevOps*, uma vez que: (1) não foi identificado outro estudo que tenha investigado a adoção de *DevOps* utilizando *grounded theory* como metodologia; (2) o número de companhias estudadas é maior que o presente nos demais trabalhos, ampliando os cenários dos quais foram extraídas compreensões sobre a adoção de *DevOps*; e (3) as explicações a respeito de como os elementos *DevOps* se relacionam em um processo de adoção, constitui um acréscimo à compreensão previamente existente.

Os resultados da pesquisa ilustram que a adoção de *DevOps* envolve 34 conceitos agrupados em sete categorias: ***agilidade***, ***automação***, ***compartilhamento e transparência***, ***cultura de colaboração***, ***medição contínua***, ***garantia da qualidade*** e ***resiliência***. A categoria principal da adoção de *DevOps* é ***cultura de colaboração***. Algumas das categorias (***automação*** e ***compartilhamento e transparência***) atuam como *facilitadores* para formação desta ***cultura de colaboração***. Outras categorias (***agilidade*** e ***resiliência***) são explicadas como *saídas* esperadas da adoção de *DevOps*. Por fim, outras duas categorias (***medição contínua*** e ***garantia da qualidade***) atuam tanto como *facilitadores* quanto como *saídas* na adoção de *DevOps*. Essencialmente, este modelo simplifica o entendimento de como se organiza o complexo conjunto de elementos que são parte da adoção de *DevOps*, possibilitando que o processo seja mais direto e com menores chances de focar em coisas erradas.

As experiências coletadas durante a produção deste estudo já têm sido efetivamente aplicadas no processo de adoção de *DevOps* do TCU. Já o modelo em si, foi introduzido por meio da apresentação em uma *tech talk* para os profissionais envolvidos no desenvolvimento de *software* do TCU e apresentado ao CPA para análise. O CPA entendeu que o processo de adoção de *DevOps* do TCU deve se pautar no modelo apresentado. Embora a formação de percepções mais concretas sobre o papel efetivo do modelo durante o processo de adoção de *DevOps* requeira tempo, as percepções de profissionais do TCU tanto a respeito do estado atual da adoção de *DevOps* como um todo, quanto da aplicabilidade e utilidade do modelo foram coletadas por meio da realização de um grupo focal. No grupo focal foram identificadas (1) as principais ações já desenvolvidas no âmbito da adoção de *DevOps* no TCU; (2) quais as contribuições que já podem ser notadas a respeito da utilização do modelo, e (3) os desafios enfrentados e próximos passos na adoção de *DevOps*.

5.2 Limitações e Sugestões para Trabalho Futuro

Embora tenha-se buscado a experiência de praticantes em companhias localizadas em outros países, a busca por interessados em contribuir com uma pesquisa acadêmica é uma tarefa ainda mais desafiadora quando se vai além das fronteiras do Brasil. O resultado disso é que dois terços dos entrevistados são de companhias brasileiras. Assim sendo, uma primeira sugestão de trabalho futuro é a expansão da quantidade de entrevistas em outros países e contextos para validar se a saturação teórica aqui atingida foi afetada por questões geográficas.

Durante o trabalho buscou-se investigar organizações cujo processo de adoção de *DevOps* foi bem sucedido. Todavia, a percepção de o processo ter sido bem sucedido é subjetiva e individual dos praticantes, não há uma maneira quantificável de se verificar.

Ademais, embora os entrevistados tenham fornecido algumas ideias sobre caminhos incorretos que podem comprometer a adoção de *DevOps*, para se identificar orientações a respeito do que não deve ser feito, o ideal seria investigar organizações onde a adoção de *DevOps* não foi bem sucedida, o que representa mais uma oportunidade para trabalho futuro, onde poderia inclusive se avaliar se o modelo aqui proposto teria sido útil nesses cenários.

Existe uma considerável sobreposição de resultados com os de trabalhos relacionados. Esse é um risco que os estudos utilizando *grounded theory* enfrentam, por conta da limitação de exposição à literatura no início do trabalho. Embora a sobreposição de fato exista, os resultados foram reintegrados à literatura mostrando as semelhanças e ilustrando os pontos em que se complementam.

Organizações possuem particularidades. Isso dificulta a geração de um modelo mais prescritivo, com detalhamento de práticas que precisam necessariamente ser aplicadas. Enquanto que em algumas organizações é possível existir um alto grau de automação, com experimentação e entrega contínua, em outras o cenário pode ser diferente. O resultado dessa limitação é que o modelo proposto contém alguns aspectos que são decididos no contexto de cada organização, como em quais facilitadores investir para fomentar a cultura de colaboração ou de que maneira explorar as saídas do processo.

A única evidência de utilidade e aplicabilidade do modelo proposto foi obtida durante a realização de um grupo focal no TCU. A participação do pesquisador como moderador do grupo focal certamente pode ter influenciado a opinião dos demais profissionais no intuito de considerar que o modelo é útil. Todavia, as próprias ações concretas apontadas, atenuam essa possibilidade. Isto posto, trabalhos futuros podem conduzir investigações a respeito da aplicabilidade e utilidade do modelo em cenários distintos.

Embora *grounded theory* ofereça procedimentos rigorosos para coleta e análise de dados, as pesquisas qualitativas em geral estão sujeitas a conterem algum grau de viés do pesquisador. Certamente outros pesquisadores podem formar uma interpretação e uma teoria diferentes depois de analisar os mesmos dados, porém, acredita-se que ao menos as principais percepções seriam preservadas.

Os estudos utilizando *grounded theory* em geral não reinvidicam ser definitivos, a teoria resultante deve ser modificável em outros contextos [55]. A implicação disso é que não se reivindica que a teoria aqui apresentada seja absoluta ou final. São bem-vindas extensões da teoria baseadas em aspectos não percebidos, detalhes mais sutis das categorias e conceitos atuais, ou ainda potencial descoberta de novas dimensões ou conceitos a partir de estudos futuros.

Referências

[1] Highsmith, J. e A. Cockburn: *Agile software development: the business of innovation.* Computer, 34(9):120–127, Sep 2001, ISSN 0018-9162. 1

[2] Humble, Jez e David Farley: *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation.* Addison-Wesley Professional, 1st edição, 2010, ISBN 0321601912, 9780321601919. 1

[3] Hüttermann, Michael: *DevOps for Developers.* Apress, Berkely, CA, USA, 1st edição, 2012, ISBN 1430245697, 9781430245698. 1, 7, 9

[4] Debois, P.: *Agile infrastructure and operations: How infra-gile are you?* Em *Agile 2008 Conference*, páginas 202–207, Aug 2008. 1, 7

[5] *10+ deploys per day: Dev and ops cooperation at flickr.* `https://conferences.oreilly.com/velocity/velocity2009/public/schedule/detail/7641`. Accessed: 2018-06-01. 1, 8

[6] França, Breno B. Nicolau de, Helvio Jeronimo, Junior e Guilherme Horta Travassos: *Characterizing devops by hearing multiple voices.* Em *Proceedings of the 30th Brazilian Symposium on Software Engineering*, SBES '16, páginas 53–62, New York, NY, USA, 2016. ACM, ISBN 978-1-4503-4201-8. `http://doi.acm.org/10.1145/2973839.2973845`. 1, 9, 10, 11, 12, 13, 14, 15, 16, 44

[7] Labs, Puppet, DevOps Research e DORA Assessment: *2017 state of devops report.* Relatório Técnico, 2017. Retrieved June, 2018 from `https://puppet.com/resources/whitepaper/state-of-devops-report`. 1

[8] One, Version: *12th annual state of agile report.* Relatório Técnico, Version One, Tech. Rep, 2018. Retrieved June, 2018 from `https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report`. 2

[9] Glaser, Barney G. e Anselm L. Strauss: *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Observations (Chicago, Ill.). Aldine Publishing Company, 1967, ISBN 9780202300283. 4, 18, 21

[10] Adolph, Steve, Philippe Kruchten e Wendy Hall: *Reconciling perspectives: A grounded theory of how people manage the process of software development.* 85, junho 2012. 5, 21, 23

[11] Stol, K. J., P. Ralph e B. Fitzgerald: *Grounded theory in software engineering research: A critical review and guidelines*. Em *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, páginas 120–131, May 2016. 5, 19, 20, 21

[12] Charmaz, Kathy: *Discovering chronic illness: using grounded theory*. Social science & medicine, 30(11):1161–1172, 1990. 5

[13] Benoliel, Jeanne Quint: *Grounded theory and nursing knowledge*. Qualitative Health Research, 6(3):406–428, 1996. 5

[14] Hutchinson, Sally A: *Education and grounded theory*. Journal of Thought, páginas 50–68, 1986. 5

[15] Kenealy, G: *Management research and grounded theory: A review of grounded theory building approach in organisational and management research*. The Grounded Theory Review, 7(2):95–117, 2008. 5

[16] Locke, Karen: *Grounded theory in management research*. Sage Publications, 2001. 5, 19

[17] Hoda, R. e J. Noble: *Becoming agile: A grounded theory of agile transitions in practice*. Em *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, páginas 141–151, May 2017. 5, 18, 24, 39

[18] Waterman, Michael, James Noble e George Allan: *How much up-front? A grounded theory of agile architecture*. Em *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015*, páginas 347–357, 2015. 5

[19] Edmunds, H.: *The Focus Group Research Handbook*. McGraw-Hill Education, 2000, ISBN 9780071394536. `https://books.google.com.br/books?id=vGa5szorHEAC`. 5, 48

[20] Shull, Forrest, Janice Singer e Dag IK Sjøberg: *Guide to advanced empirical software engineering*. Springer, 2007. 5, 48, 49

[21] Dybå, Tore e Torgeir Dingsøyr: *Empirical studies of agile software development: A systematic review*. Information and Software Technology, 50(9):833 – 859, 2008, ISSN 0950-5849. `http://www.sciencedirect.com/science/article/pii/S0950584908000256`. 6

[22] Huo, Ming, J. Verner, Liming Zhu e M. A. Babar: *Software quality and agile methods*. Em *Proceedings of the 28th Annual International Computer Software and Applications Conference, 2004. COMPSAC 2004.*, páginas 520–525 vol.1, Sept 2004. 6

[23] Dingsøyr, Torgeir, Sridhar Nerur, VenuGopal Balijepally e Nils Brede Moe: *A decade of agile methodologies*. J. Syst. Softw., 85(6):1213–1221, junho 2012, ISSN 0164-1212. `http://dx.doi.org/10.1016/j.jss.2012.02.033`. 7

[24] Virmani, M.: *Understanding devops & bridging the gap from continuous integration to continuous delivery*. Em *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, páginas 78–82, May 2015. 7

[25] Wahaballa, A., O. Wahballa, M. Abdellatief, H. Xiong e Z. Qin: *Toward unified devops model*. Em *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, páginas 211–214, Sept 2015. 7

[26] Davis, Jennifer e Katherine Daniels: *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*. O'Reilly Media, Inc., 1st edição, 2016, ISBN 1491926309, 9781491926307. 8, 9

[27] Loukides, M.: *What is DevOps?* O'Reilly Media, 2012, ISBN 9781449339111. 8

[28] Walls, M.: *Building a DevOps Culture*. O'Reilly Media, 2013, ISBN 9781449368364. 8

[29] Smeds, Jens, Kristian Nybom e Ivan Porres: *Devops: A definition and perceived adoption impediments*. Em Lassenius, Casper, Torgeir Dingsøyr e Maria Paasivaara (editores): *Agile Processes in Software Engineering and Extreme Programming*, páginas 166–177, Cham, 2015. Springer International Publishing, ISBN 978-3-319-18612-2. 9, 17, 43

[30] Dyck, Andrej, Ralf Penners e Horst Lichter: *Towards definitions for release engineering and devops*. Em *Proceedings of the Third International Workshop on Release Engineering*, RELENG '15, páginas 3–3, Piscataway, NJ, USA, 2015. IEEE Press. http://dl.acm.org/citation.cfm?id=2820690.2820694. 9

[31] Willis, John: *What devops means to me*. https://blog.chef.io/2010/07/16/what-devops-means-to-me/, 2010. Accessed: 2018-06-01. 9, 11, 13, 14, 15

[32] Hamunen, Joonas: *Challenges in adopting a devops approach to software development and operations*. Tese de Mestrado, 2016. 9, 11, 13, 14, 16, 17, 45, 47

[33] Erich, Floris, Chintan Amrit e Maya Daneva: *Cooperation between information system development and operations: A literature review*. Em *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '14, páginas 69:1–69:1, New York, NY, USA, 2014. ACM, ISBN 978-1-4503-2774-9. http://doi.acm.org/10.1145/2652524.2652598. 10, 11, 15

[34] Lwakatare, Lucy Ellen, Pasi Kuvaja e Markku Oivo: *Dimensions of devops*. Em Lassenius, Casper, Torgeir Dingsøyr e Maria Paasivaara (editores): *Agile Processes in Software Engineering and Extreme Programming*, páginas 212–217, Cham, 2015. Springer International Publishing, ISBN 978-3-319-18612-2. 10, 11, 13, 14, 16, 44

[35] Lwakatare, Lucy Ellen, Pasi Kuvaja e Markku Oivo: *An exploratory study of devops extending the dimensions of devops with practices*. ICSEA'16, páginas 91–99, 2016, ISBN 978-1-61208-498-5. 10, 11, 12, 13, 14, 16, 44

[36] A., Erich F. M., Amrit C. e Daneva M.: *A qualitative study of devops usage in practice*. Journal of Software: Evolution and Process, 29(6):e1885, 2017. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1885, e1885 smr.1885. 10, 11, 12, 13, 14, 15, 17, 45

[37] Womack, J.P. e D.T. Jones: *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*. Free Press, 2010, ISBN 9781439135952. 16

[38] Riungu-Kalliosaari, Leah, Simo Mäkinen, Lucy Ellen Lwakatare, Juha Tiihonen e Tomi Männistö: *Devops adoption benefits and challenges in practice: A case study*. Em Abrahamsson, Pekka, Andreas Jedlitschka, Anh Nguyen Duc, Michael Felderer, Sousuke Amasaki e Tommi Mikkonen (editores): *Product-Focused Software Process Improvement*, páginas 590–597, Cham, 2016. Springer International Publishing, ISBN 978-3-319-49094-6. 17

[39] Glaser, Barney G.: *Doing grounded theory: Issues and discussions*. Sociology Press, 1998. 18

[40] Corbin, J. e A. Strauss: *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 2014, ISBN 9781483315683. https://books.google.com.br/books?id=hZ6kBQAAQBAJ. 18

[41] Miles, M.B., A.M. Huberman, M.A. Huberman e P.M. Huberman: *Qualitative Data Analysis: An Expanded Sourcebook*. Sage Publications, 1994, ISBN 9780803955400. 19

[42] Punch, Keith F: *Introduction to social research: Quantitative and qualitative approaches*. Sage Publications, 2013. 19

[43] Jantunen, Sami e Donald C. Gause: *Using a grounded theory approach for exploring software product management challenges*. Journal of Systems and Software, 95:32 – 51, 2014, ISSN 0164-1212. http://www.sciencedirect.com/science/article/pii/S0164121214000776. 19, 21

[44] Järvinen, Pertti: *Mapping research questions to research methods*. Em Avison, David, George M. Kasper, Barbara Pernici, Isabel Ramos e Dewald Roode (editores): *Advances in Information Systems Research, Education and Practice*, páginas 29–41, Boston, MA, 2008. Springer US, ISBN 978-0-387-09682-7. 19

[45] Glaser, Barney G: *Basics of grounded theory analysis: Emergence vs forcing*. Sociology press, 1992. 19

[46] Denzin, Norman K: *Grounded theory and the politics of interpretation*. The Sage handbook of grounded theory, páginas 454–471, 2007. 19

[47] Adolph, Steve, Wendy Hall e Philippe Kruchten: *Using grounded theory to study the experience of software development*. Empirical Software Engineering, 16(4):487–513, Aug 2011, ISSN 1573-7616. https://doi.org/10.1007/s10664-010-9152-6. 20

[48] Coleman, Gerry e Rory O'Connor: *Using grounded theory to understand software process improvement: A study of irish software product companies*. Information and Software Technology, 49(6):654 – 667, 2007, ISSN 0950-5849. http://www.sciencedirect.com/science/article/pii/S0950584907000134, Qualitative Software Engineering Research. 21

[49] Hoda, Rashina, James Noble e Stuart Marshall: *The impact of inadequate customer collaboration on self-organizing agile teams*. Information and Software Technology, 53(5):521 – 534, 2011, ISSN 0950-5849. http://www.sciencedirect.com/science/article/pii/S0950584910001941, Special Section on Best Papers from XP2010. 21

[50] Georgieva, Svetla e George Allan: *Best practices in project management through a grounded theory lens*. 6, janeiro 2008. 24

[51] Lewis, James e Fowler Martin: *Microsservices*. http://martinfowler.com/articles/microservices.html, 2014. Accessed: 2018-05-22. 32

[52] Feijter, Rico, Rob Vliet, Erik Jagroep, Sietse Overbeek, Sjaak Brinkkemper *et al.*: *Towards the adoption of devops in software product organizations: A maturity model approach*, 2017. 45, 47

[53] Morgan, David L: *Focus groups*. Annual review of sociology, 22(1):129–152, 1996. 49

[54] Lehtola, Laura, Marjo Kauppinen e Sari Kujala: *Requirements prioritization challenges in practice*. Em Bomarius, Frank e Hajimu Iida (editores): *Product Focused Software Process Improvement*, páginas 497–508, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg, ISBN 978-3-540-24659-6. 49

[55] Hoda, Rashina, James Noble e Stuart Marshall: *Developing a grounded theory to explain the practices of self-organizing agile teams*. Empirical Software Engineering, 17(6):609–639, 2012. 61