Walysson Carlos dos Santos Oliveira

Aplicando análise qualitativa para caracterização dos princípios DevOps em uma empresa do Setor Público

São Luís - Maranhão

14 de dezembro de 2018

Walysson Carlos dos Santos Oliveira

Aplicando análise qualitativa para caracterização dos princípios DevOps em uma empresa do Setor Público

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia da Computação da UFMA, como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação, Centro de Ciência Exatas e Tecnológicas da Universidade Federal do Maranhão.

Universidade Federal do Maranhão

Centro de Ciências Exatas e Tecnologia

Engenharia da Computação

Orientador: Prof. Dr. Davi Viana dos Santos

São Luís - Maranhão

14 de dezembro de 2018

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Núcleo Integrado de Bibliotecas/UFMA

Oliveira, Walysson Carlos dos Santos.
Aplicando análise qualitativa para caracterização dos princípios DevOps em uma empresa do Setor Público / Walysson Carlos dos Santos Oliveira. - 2018.
48 p.

Coorientador(a): Walysson Carlos dos Oliveira.
Orientador(a): Davi Viana dos Santos.
Curso de Engenharia da Computação, Universidade Federal do Maranhão, São Luís - MA, 2018.

1. DevOps. 2. Estudo de Caso. 3. Método Grounded Theory. 4. Setor público. I. Oliveira, Walysson Carlos dos. II. Santos, Davi Viana dos. III. Título.

Walysson Carlos dos Santos Oliveira

Aplicando análise qualitativa para caracterização dos princípios DevOps em uma empresa do Setor Público

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia da Computação da UFMA, como requisito parcial para a obtenção do grau de Bacharel em Engenharia da Computação, Centro de Ciência Exatas e Tecnológicas da Universidade Federal do Maranhão.

Trabalho aprovado. São Luís - Maranhão, 19 de dezembro de 2018:

Prof. Dr. Davi Viana dos Santos
Orientador

Prof. Dr. Sérgio Souza Costa
Universidade Federal do Maranhão

Prof. Dr. Luis Jorge Enrique Rivero Cabrejos
Universidade Federal do Maranhão

São Luís - Maranhão
14 de dezembro de 2018

Este trabalho é dedicado à minha mãe Eliete
que fez dos meus sonhos os seus.

Agradecimentos

Os agradecimentos principais são direcionados à minha família, pelo amor e apoio incondicional, em especial a minha mãe Eliete, pelo suporte nas horas difíceis, de desânimo e cansaço.

A minha namorada Dayane que esteve comigo nessa jornada me apoiando, dando forças para continuar e me auxiliou neste trabalho durante as exaustivas transcrições das entrevistas.

A todos os meus professores, em especial, ao meu orientador Davi Viana, pela orientação, pela paciência, por suas correções, incentivo, confiança e seu imenso empenho em ajudar. Aos professores Sérgio Costa e Bruno Feres por serem minhas referências desde o início do curso.

Agradecimentos especiais são direcionados aos meus colegas de trabalho Filipe, Welyab, Otávio, Jefferson, Ramon, Lucas, Marcos Rogério e Gislaine que contribuíram com este trabalho me reservando uma parte do seu atarefado dia de trabalho e muito do seu conhecimento e experiência nas entrevistas.

E aos meus amigos Massetti, Larissa, Thamyla, Mariana, Micael e João Higo por estarem comigo no decorrer do curso.

"Sonhos determinam o que você quer. Ação determina o que você conquista." - Aldo Novak

Resumo

O mercado de tecnologia está a cada dia mais dinâmico e competitivo, o que exige uma rápida adaptação das as organizações de TI. Foi nesse cenário que surgiu o termo *DevOps*, um movimento que visa aumentar a colaboração entre os times de desenvolvimento e operações de forma alinhada ao negócio, não apenas na implantação do sistema, mas desde sua concepção e durante todo seu ciclo de vida. Este trabalho tem o objetivo de realizar um estudo de caso em uma empresa pública, a fim de analisar o seu ambiente de TI levando em conta os conceitos e as práticas *DevOps*, possibilitando uma posterior implantação desses conceitos e práticas. Foram realizadas entrevistas com os profissionais que trabalham nos setores de desenvolvimento e infraestrutura de TI. Para análise dos dados qualitativos, utilizou-se o método *Grounded Theory* para extrair informações relevantes para a pesquisa e relacionar os conceitos a fim de se ter uma análise coesa do ambiente de TI da instituição pública.

Palavras-chaves: princípios e práticas *DevOps*, estudo de caso, setor público, *grounded theory*, automatização.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O mercado de tecnologia está a cada dia mais dinâmico e competitivo, o que exige uma rápida adaptação das as organizações de Tecnologia da Informação (TI) para continuarem competitivas e atuantes no mercado. As mudanças nos softwares são praticamente diárias e o produto deve ser construído para executar numa quantidade considerável de dispositivos e sistemas operacionais. Os sistemas estão cada vez mais complexos, integrados a outros serviços e exigindo um alto grau de confiabilidade e disponibilidade dos mesmos (CORREA, 2017).

Neste contexto, muitas organizações de software tem introduzido práticas ágeis ao seu processo de desenvolvimento. O movimento ágil surgiu na década de 90 com o objetivo de que a TI respondesse de forma mais dinâmica as constantes mudanças do negócio. O resultado desse movimento foi o Manifesto Ágil (ÁGIL, 2011) e metodologias que ficaram conhecidas como metodologias ágeis. Vários problemas antes existentes entre o negócio e a equipe de desenvolvimento foram resolvidos com a adoção de práticas ágeis através de técnicas como a aproximação do cliente junto com a equipe de desenvolvimento, integração contínua, reuniões diárias, equipes pequenas e multifuncionais, dentre outras (BRAGA, 2015).

Em uma organização de TI, a equipe de operações é normalmente composto por administradores de sistemas, administradores de redes, administradores de banco de dados e analistas de segurança e eles são responsáveis por colocar os sistemas recebidos em produção, ou seja, colocar o sistema online para ser acessador pelos usuários. Também é tarefa da equipe de operações garantir que o sistema esteja sempre disponível aos usuários com o mínimo de falhas possível. Dessa forma, a equipe de operações visa manter a estabilidade dos sistemas. Já a equipe de desenvolvimento é comumente formado por programadores, analistas de requisitos, analistas de sistemas, analistas de testes e têm a tarefa de projetar, construir, testar e atualizar os sistemas (BASS; WEBER; ZHU, 2015).

A adoção de práticas ágeis de desenvolvimento contribuiu para reduzir o tempo de desenvolvimento de entregáveis de software e a periodicidade entre eles, assim como a liberação constante de novas versões de software para serem colocados em produção. Contudo, uma vez que uma nova versão de software fica pronta, a responsabilidade de colocá-la em produção é da equipe de operações de software. A redução do tempo de desenvolvimento e o aumento da periodicidade entre as entregas implica que agora a equipe de operações de software precisa lidar com uma quantidade considerável de mudanças constantes no ambiente de produção, aumentando seu acúmulo. Como resultado, profissionais de software relatam que a adoção de práticas ágeis expõe deficiências em seu processo de entrega e não melhoraram a confiabilidade, nem a comunicação entre a equipe de desenvolvimento de software e a equipe operações (FRANÇA; JUNIOR; TRAVASSOS, 2016).

1.1 Justificativa e Motivação

Trabalhando em equipes separadas e com objetivos distintos, as equipes de desenvolvimento e operações vez ou outra entram em conflito. Segundo Braga (2015), a qualidade do trabalho da equipe de desenvolvimento é medida pela quantidade de novos softwares e novas funcionalidades que são entregues, enquanto que a equipe de operações é medido pela estabilidade dos sistemas. Quanto maior o número de entregas por parte da equipe de desenvolvimento, maior é o risco de instabilidade nos sistemas. Estes sistemas podem vir com novos *bugs* e novas vulnerabilidades. Gera-se então um conflito de interesses entre essas duas equipes, onde deveria existir cooperação (JUNIOR; CARDOSO; ZALLA, 2016).

Foi nesse cenário que surgiu o termo *DevOps*, um movimento que visa aumentar a colaboração entre as equipes de desenvolvimento e operações de forma alinhada ao negócio, não apenas na implantação do sistema, mas desde sua concepção e durante todo seu ciclo de vida. Com o *DevOps*, a equipe de operações passa a ter mais voz e ser mais valorizado, trabalhando de conjunto com a equipe de desenvolvimento, podendo reagir de forma mais rápida as exigências do mercado. Um ponto muito importante no *DevOps* é a automação das diversas atividades necessárias para entregar código de qualidade em produção. O movimento *DevOps* também é um movimento cultural dentro das organização de T.I que busca incentivar o envolvimento de todos da equipe no projeto de um software como um todo, desenvolvedores, testadores e a equipe de operações. Esse envolvimento geral faz com que os objetivos fiquem alinhados, assim, produzir

software de qualidade e colocá-lo em produção de forma eficiente é responsabilidade de todos os envolvidos (BRAGA, 2015).

Dentre as práticas adotadas no *DevOps*, uma das principais é a Integração Contínua ou simplesmente CI (*Continuous Integration*). Essa prática consiste em um processo de integrar continuamente o trabalho desenvolvido em um repositório com o resto da equipe de desenvolvimento executando testes automatizados a cada novo trecho de código enviado para o repositório, permitindo assim que os erros sejam detectados, localizados e corrigidos de forma prematura o que agiliza o processo de entrega de software (RATO, 2018).

1.2 Objetivos

1.2.1 Objetivo Geral

Realizar um estudo de caso em uma empresa pública, a fim de analisar o seu ambiente de TI levando em conta os conceitos e as práticas *DevOps*, possibilitando uma posterior implantação desses conceitos e práticas.

1.2.2 Objetivos Específicos

- Caracterização do movimento *DevOps* e suas principais práticas.
- Análise das princípios *DevOps* através de Estudo de Caso.

1.3 Organização do trabalho

O presente trabalho apresenta 5 capítulos. O Capítulo 2 contém um referencial teórico sobre *DevOps* e suas práticas. O Capítulo 3, a metodologia para o desenvolvimento deste trabalho de conclusão de curso. O Capítulo 4 apresenta os Resultados do Estudo de Caso sobre a análise das práticas *DevOps* em uma organização de software. Por fim, o Capítulo 5 apresenta a conclusão e os trabalhos futuros.

2 Referencial Teórico

2.1 DevOps

O termo *DevOps* refere-se ao movimento de cooperações entre as equipes de desenvolvimento e operações. Formado pela junção das palavras em inglês *development* e *operations* (CORREA, 2017). O significado de *DevOps* é bastante amplo e ainda não prescritivo, ou seja, não existe uma definição precisa do termo, porém, sabe-se que o *DevOps* está relacionado a alguns princípios, tais como a cultura de colaboração, automação, medição e o compartilhamento de conhecimento entre as áreas (BRAGA, 2015).

Existem na literatura, pesquisas para encontrar uma definição compartilhada do termo *DevOps*. Segundo França, Junior e Travassos (2016), *DevOps* é um neologismo que representa um movimento de profissionais de TI que tem como foco a eficiência na entrega de software através da colaboração entre as equipes de desenvolvimento e operações. Esse movimento se baseia em um conjunto de princípios, como cultura, automação, medição e compartilhamento com o objetivo melhorar a qualidade dos processos de desenvolvimento e operações, bem como a qualidade dos produtos e serviços de software.

Sobre os princípios *DevOps*, a Cultura de Colaboração tem objetivo de eliminar as barreiras entre os times de desenvolvimento e operações de forma que eles trabalhem em conjunto (ROCHE, 2013). O princípio da Automatização busca reduzir as atividades manuais e repetitivas, deixando os indivíduos livres para se preocupar com as regras de negócio (VIRMANI, 2015). O princípio da Garantia de Qualidade busca garantir que as atividades sejam realizadas de maneira eficiente e confiável e que os produtos e serviços atendam aos padrões de qualidade estabelecidos. O princípio da Medição tem o objetivo de coletar métricas eficientes para apoiar a tomada de decisões no ciclo de vida de desenvolvimento e operações. Por fim, o princípio do Compartilhamento busca fornecer informações de forma transparente e em tempo hábil, além de promover o intercâmbio de aprendizado dentro da organização (ERICH; AMRIT; DANEVA, 2014).

O DevOps é complementar ao Desenvolvimento Ágil e veio para alcançar também a equipe de operações, inserindo-se no processo de entrega contínua de software, garantindo que o código esteja pronto para ser colocado em produção e gerando valor para o cliente, além de permitir um melhor fluxo de trabalho e integração entre as equipes de desenvolvimento e operações (CORREA, 2017).

É comum nas organizações que adotam o *DevOps* a utilização de princípios, tais como o Desenvolvimento e Testes em ambientes semelhantes ao da produção, processo de implantação repetível e confiável, monitoramento e validação da qualidade operacional e o aumento dos *feedbacks* dos clientes. A equipe de operações trabalha desde a concepção do projeto e nos primeiros ciclos de desenvolvimento fornecendo para a equipe de desenvolvimento e testes um ambiente similar ao ambiente de produção para que a aplicação seja desenvolvida e testada antes de estar pronta para a implantação, como é mostrado na Figura 2.1.

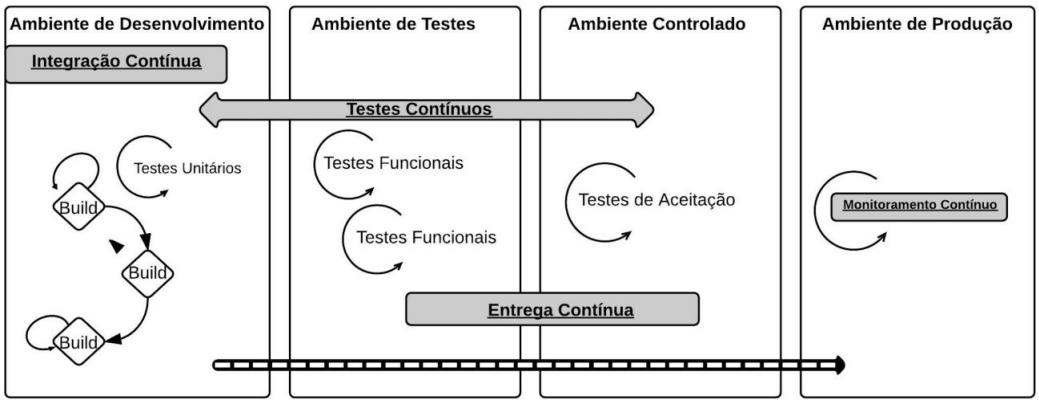
Um ambiente assim evita vários problemas entre as equipes de operações e desenvolvimento, pois é possível verificar o comportamento da aplicação o que facilita a comunicação entre as equipes e consequentemente o processo para a entrega contínua de software (BRAGA, 2015). Além incentivar a utilização de ambientes semelhantes ao da produção nas etapas de desenvolvimento e testes, as organizações que adotam o *DevOps* também seguem uma série de práticas, cujas principais são: Pipeline de Implantação, Integração Contínua (CI), Entrega Contínua (CD), Testes Contínuos e Automatizados e Infraestrutura como Código (ZHU; BASS; CHAMPLIN-SCHARFF, 2016). Na seção a seguir será apresentada a fundamentação teórica dessas práticas .

2.2 Práticas DevOps

2.2.1 Pipeline de Implantação

Para Braga (2015), o processo de implantação repetível e confiável é fundamental para se implantar DevOps e isso se dá através da automação criando assim um pipeline de entrega confiável. O pipeline

Figura 2.1 – Equipe de operações em ciclos iniciais de desenvolvimento



Fonte: (BRAGA, 2015)

de entrega consiste em um conjunto de estágios que uma aplicação passa desde o desenvolvimento até a produção. Um pipeline de implantação é, em essência, um processo automático de entrega de software. Isso não implica que não há interação humana com o sistema durante o processo de entrega; ela garante que passos complexos e passíveis de erro sejam automatizados (OUVERNEY; MARTINS, 2018).

O pipeline de implantação, assim como todo o processo, precisa ser monitorado em todo seu ciclo de vida através de automação e com visibilidade a todos os membros da equipe. Esse processo fornece uma base para ampliar o *feedback* dos clientes e consumidores fornecendo assim respostas mais rápidas através de um canal de comunicação eficiente. Cada organização pode implementar seu pipeline de acordo com suas necessidades e seus ambientes. Na Figura 2.2 há a representação de um pipeline típico de *DevOps* para fornecer agilidade à entrega contínua de software (CORREA, 2017).

Figura 2.2 – Pipeline de Implantação

Fonte: Adaptado de Humble e Farley (2014)

Em um pipeline típico o estágio de desenvolvimento é muito parecido na maioria das organizações. Desenvolvedores escrevem seus códigos colaborativamente e com ferramentas para dar apoio às atividades tais como gerenciamento de configuração, IDEs (integrated development environment), ferramentas de desenvolvimento, testes unitários, etc. Após o desenvolvimento a maioria das organizações possui um servidor de build centralizado onde o código é compilado (SATO, 2014). O pipeline de implantação é essencial para as práticas que serão apresentadas a seguir.

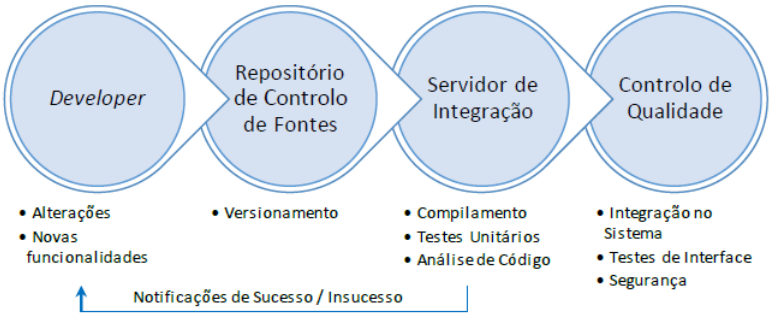
2.2.2 Integração Contínua

É uma prática de desenvolvimento de software que teve origem no *eXtreme Programming* (XP) e consiste que o desenvolvedor integre continuamente o código desenvolvido ao projeto principal, permitindo assim detectar e localizar, erros rapidamente, podendo haver múltiplas integrações por dia. A integração contínua pode trazer diversos benefícios para a organização e dentre eles pode-se citar um tempo menor de depuração, maior adição de características do software, redução de problemas e um menor tempo de integração, bem como o aumento de visibilidade e comunicação entre as equipes (FERNANDES et al.,

2017). Os testes de trabalho são integrados a cada *commit* no repositório, permitindo assim detectar e localizar erros rapidamente. Entre as atividades presentes nela estão o controle de versão com repositório único, *build* automatizado, *commits* diários no repositório principal, conjunto de testes automatizados e abrangentes, entre outros (CORREA, 2017).

Para o *DevOps*, a integração contínua é uma etapa extremamente importante. Através da integração contínua impede-se a criação de projetos em excesso evitando casos de conflito de código. Na prática, esta abordagem envolve um servidor centralizado que ininterruptamente importa alterações ao repositório central como é ilustrado na Figura 2.3. Caso existam falhas na compilação, é esperado que a equipe se reorganize para as resolver, com uma perspetiva evolutiva do tema, ou seja, que sejam cada vez menores as ocorrências de falhas na compilação automatizada que é um dos princípios do *DevOps* (RATO, 2018).

Figura 2.3 – Representação Esquemática do conceito Integração Contínua



Fonte: Rato (2018)

Neste caso, a integração contínua oferece uma visão em tempo real sobre o estado atual do sistema e métricas de certificação de qualidade, permitindo também a envolvência constante de todos os elementos da equipa responsável, sem exceção das Operações. Em suma, a grande vantagem que o conceito traz é a criação de uma forma transparente em que todos os interessados podem monitorizar, agir e contribuir positivamente para a evolução de determinado projeto sem a disrupção associada a reuniões de status ou duplicação de esforços (RATO, 2018).

2.2.3 Entrega Contínua

Entrega contínua é uma prática que garante a entrega de software da equipe de desenvolvedores para o ambiente de produção em um processo confiável, previsível, visível e o mais automatizado possível, com riscos quantificáveis e bem entendidos. Ao invés de planejar grandes entregas, a T.I. deve elaborar software em ciclos mais curtos, garantindo que o novo código possa ser implantado no ambiente de produção a qualquer momento de forma eficiente, sem comprometer a qualidade (FERNANDES et al., 2017).

As equipes que praticam entrega contínua fornecem pequenas entregas de software para a produção com uma frequência muito maior que entregas típicas, frequentemente várias vezes ao dia, diminuindo assim a diferença entre a ideia de desenvolver o software à disponibilização do software pelo usuário final, através da automação de todo o sistema de entrega. Entre as atividades que fazem parte dela estão automação de *builds*, testes automáticos, implantação automática, gerenciamento de infraestrutura, pipeline de implantação entre outros (CORREA, 2017). Entregas Contínuas não significam entregas literais, mas sim a filosofia e empenho em assegurar que o software desenvolvido está sempre pronto para ser entregue.

2.2.4 Testes Contínuos e Automatizados

Conforme Humble e Farley (2014), testar é uma atividade multifuncional que envolve todo o time e deve ser executada continuamente desde o início do projeto. No mundo *DevOps* e de entrega

contínua, testadores colaboram com os desenvolvedores e usuários para a escrita de testes automatizados desde o início do projeto, escrevendo testes antes que os desenvolvedores iniciem seus trabalhos nessas funcionalidades. O objetivo é fornecer uma boa abrangência do comportamento de modo a garantir assim que as funcionalidades sejam completamente implantadas e de forma correta.

Os testes contínuos aproximam as atividades de testes com código produzido, estendendo vários tipos de testes e os executando como partes do processo da integração contínua. Nos testes contínuos os erros podem ser corrigidos mais rapidamente, pois no contexto onde não se pode esperar que termine um ciclo de desenvolvimento para ocorrer os testes, os desenvolvedores podem encontrar e resolver os problemas mais rapidamente assim como podem automatizar tarefas mais repetitivas e priorizar as mais necessárias (BRAGA, 2015).

Os processos de testes contínuos e automatizados precisam abranger vários tipos de testes que devem ser usados para garantir a entrega de um software de qualidade. A entrega contínua pode abranger testes manuais ou exploratórios, mas os testes automáticos realmente são chaves para o processo de entrega contínua e a melhora da qualidade. Geralmente, o servidor de integração contínua é responsável por realizar a maioria dos testes automáticos e validar os *commits* dos desenvolvedores, entretanto, outros testes automatizados são executados quando o sistema é implantando em outros ambientes de testes, tais como testes de unidade, integração, aceitação, carga, performance, simulação, fumaça, etc... Todos esses testes podem ser executados durante o pipeline de implantação, podendo dar mais ênfase ao tipo de testes de acordo com o ambiente a ser testado (SATO, 2014).

2.2.5 Infraestrutura como Código

A infraestrutura como código (IaC) é um tipo de infraestrutura de T.I., na qual os times de infraestrutura e operação podem gerenciar configurações e automatizar o provisionamento da infraestrutura, além de implementações e realizar o fornecimento de serviços através de código escrito. Isso irá eliminar um processo manual seja para a configuração tanto para os servidores ou serviços, ou seja, gerenciar a infraestrutura como um sistema de software, com funcionalidades bem testadas, tarefas operacionais e rotineiras de gerenciamento, atualização e documentação da infraestrutura de forma segura e em larga escala (FERNANDES et al., 2017).

A infraestrutura como código pertence ao ecossistema de entrega contínua de software dando suporte ao pipeline de implantação e todo o processo que envolve a entrega de modo a permitir assim oferecer um controle dos ambientes gerenciados através de controle de gerenciamento de versões e ambientes de implantação. Enfim, é o processo de preparar os ambientes para implantação e a gestão destes ambientes após a implantação ajudando a criar e gerenciar o ambiente, instalar a versão correta da aplicação e a configuração da aplicação (HUMBLE; FARLEY, 2014). Este ecossistema tem por objetivo gerenciar todos os ambientes, incluindo os de integração contínua fornecendo uma abordagem holística em relação ao gerenciamento de toda a infraestrutura (BRAGA, 2015).

2.3 Grounded Theory

Descrito por Glaser e Strauss (1967) em seu livro seminal, a *Grounded Theory* é um método científico que utiliza um conjunto de procedimentos sistemáticos de coleta e análise dos dados para gerar, elaborar e validar teorias substantivas sobre fenômenos essencialmente sociais, ou processos sociais abrangentes. É uma abordagem de pesquisa qualitativa que busca explicar a realidade a partir dos significados atribuídos pelos envolvidos e suas experiências. Conforme Bianchi e Ikeda (2008), a *Grounded Theory* tem como objetivo desenvolver conceitos, teorias fundamentadas a partir das palavras e ações dos indivíduos em estudo onde pouco é conhecido. A coleta de dados pode combinar diversos métodos como estudos de caso, levantamentos, experimentos, entrevistas e/ou observações.

Para os criadores Glaser e Strauss (1967), existem dois tipos básicos de teorias: as formais e as substantivas. As teorias formais são compostas do que os autores chamam as "grandes" teorias, conceituais e abrangentes, enquanto que as teorias substantivas dizem respeito a explicações para situações cotidianas sendo, portanto, mais simples e acessíveis. O tipo de teoria a ser desenvolvido pela *Grounded Theory* é adequado ao tipo das teorias substantivas, ou a que foi desenvolvida por uma área de investigação empírica (BIANCHI; IKEDA, 2008).

Os criadores da *Grounded Theory*, Glaser e Struss, discordavam em alguns aspectos do método e houve uma divisão em duas vertentes. A vertente defendida por Glaser dá destaque aos aspectos emergentes do método e aos processos indutivos desenvolvidos pioneiramente pelo Departamento de Sociologia da Universidade de Columbia nos anos 50 e 60. A vertente desenvolvida por Strauss e consolidada em Strauss e Corbin (1997), tem o objetivo de sistematizar o método de coleta e análise de dados. Segundo Mello e Cunha (2003), na vertente desenvolvida por Strauss a *Grounded Theory* é baseada na ideia de codificação, que se trata do processo de analisar os dados. Durante a codificação são identificados conceitos (ou códigos) e categorias. O código é um conceito que dá nome a um fenômeno de interesse para o pesquisador; abstrai um evento, objeto, ação, ou interação que tem um significado para o pesquisador. Categorias são agrupamentos de códigos unidos em um grau de abstração mais alto.

2.3.1 Codificação

Na *Grounded Theory*, a forma de coletar dados sugerida é um apanhado de várias outras técnicas qualitativas como entrevistas, discursos, estudos de caso, memorandos e outros documentos. Os dados coletados são repartidos, analisados e comparados, sucessivamente. A comparação de semelhanças e diferenças entre incidentes observados nos dados coletados é que promovem a diretriz para a busca de novos dados (BIANCHI; IKEDA, 2008). Para Mello e Cunha (2003), a codificação é a principal etapa da análise dos dados e pode ser divida em três fases: codificação aberta, axial e seletiva. Não sendo obrigatório executar as três fases. O pesquisador tem a liberdade de executar as fases adequadas à sua pesquisa.

2.3.1.1 Codificação aberta

A codificação aberta é a primeira fase do processo de análise de dados. Todo o material coletado é transcrito, as frases analisadas e são selecionados os códigos. Essa fase envolve a quebra, a análise, a comparação, a conceituação e a categorização dos dados. Nas fases iniciais da codificação aberta, o pesquisador explora os dados examinando minuciosamente aquilo que lhe parece relevante devido à leitura intensiva dos textos e os incidentes são agrupados em códigos através da comparação incidente–incidente (MELLO; CUNHA, 2003). A codificação pode ser feita linha por linha ou por frase ou parágrafo, ou até mesmo todo o documento (STOL; RALPH; FITZGERALD, 2016).

2.3.1.2 Codificação axial

Após a identificação de categorias conceituais pela codificação aberta, a codificação axial é a fase seguinte do processo. Ela é necessária devido à quantidade massiva de conceitos originados na fase anterior. Trata-se agora de analisar os conceitos selecionados, fazer uma reorganização e, destes extrair uma ideia central e suas subordinações (BIANCHI; IKEDA, 2008). Na codificação axial examina-se as relações entre categorias e subcategorias. Explicitam-se causas e efeitos, condições intervenientes e estratégias de ação, em proposições que devem ser testadas novamente nos dados (MELLO; CUNHA, 2003).

2.3.1.3 Codificação seletiva

Nesta fase, o processo chega ao seu final, quando ocorre a saturação teórica, isto é, nenhum novo dado acrescenta novas nuances ao processo de análise e categorização. Refina todo o processo identificando a categoria central da teoria, com a qual todas as outras estão relacionadas. A categoria central deve ser capaz de integrar todas as outras categorias e expressar a essência do processo social que ocorre entre os envolvidos. Ainda na codificação seletiva, categorias mal formuladas são revistas, e falhas na lógica da teoria são resolvidas (MELLO; CUNHA, 2003).

2.4 Secretaria de Estado da Fazenda do Maranhão

Nesta seção será apresentada a empresa do setor público que terá seu ambiente de TI estudado. A Secretaria de Estado da Fazenda do Maranhão (SEFAZ/MA) é uma secretaria de grande importância dentro do governo estadual, devido a sua finalidade de arrecadar tributos e estes serem convertidos em benefícios para a população. Grande parte do esforço realizado na arrecadação e fiscalização tributária

é tecnológico e por isso a SEFAZ/MA conta, em sua área de TI, com cerca de 50 pessoas trabalhando, divididos entre servidores públicos e funcionários terceirizados.

Existem duas empresas terceirizadas que prestam serviço na tecnologia. Os setores de Desenvolvimento e Infraestrutura de TI contam cada um com uma dessas empresas terceirizadas. Além dos terceirizados existem os servidores públicos da SEFAZ que fazem a supervisão dos serviços. Além dos setores de Desenvolvimento e Infraestrutura de TI, existem também os setores de Banco de dados, Escritório de projetos, Central de Serviços, Suporte e Manutenção.

A SEFAZ/MA conta com diversos sistemas para as mais variadas finalidades, conta com aplicações de institucional dentro da própria organização, como sistemas de folha de pagamento, frequência, gerenciamento de projetos e etc. Conta também com aplicações públicas destinadas aos contribuintes. O principal sistema sistema é o SEFAZNET, que é um sistema integrado composto por vários módulos, entre eles IPVA, ITCD, ICMS, Notas fiscais e variedade de outros módulos.

3 Metodologia

A fim de alcançar os objetivos apresentados no Capítulo 1, foram seguidas uma série de etapas conforme é mostrado na Figura 3.1. A primeira etapa consistiu em realizar um levantamento bibliográfico, a fim de compreender o movimento *DevOps*, seu conceitos, sua abrangência e principais características. Após compreender o *DevOps* de maneira geral, foi realizado um estudo na literatura focado em suas principais práticas.

Com a compreensão geral do movimento *DevOps* e um entendimento aprofundado de suas principais práticas, foi realizado um Estudo de Caso onde fez-se uma análise das características do ambiente de TI em uma empresa pública levando em conta os conceitos e práticas *DevOps*. A coleta de dados foi feita a partir entrevistas com uma amostra relevante dos profissionais de TI da empresa. Para a análise dos dados coletados, foi utilizado o método *Grounded Theory* (GT) ou Teoria Fundamentada nos Dados.

Figura 3.1 – Metodologia de Pesquisa

Fonte: O Autor

As fases de Levantamento bibliográfico e Estudo sobre as principais práticas já foram bem exploradas no Capítulo 2. A seguir será detalhado a fase de Coleta de dados e no Capítulo 4 será explorada a análise de dados e os resultados advindos desta análise.

3.1 Coleta de dados

A Coleta de dados pode ser dividida em outras duas etapas, o Planejamento e a Execução. A seguir serão detalhadas cada uma dessas etapas.

3.1.1 Planejamento

Na etapa de Planejamento, foi definido o tipo de coleta de dados baseado em entrevistas e a realização da análise dos dados coletados com o método *Grounded Theory*. Para as entrevistas foi elaborado um questionário baseado nos conceitos obtidos na fase do Levantamento bibliográfico e da fase de Estudo das principais práticas *DevOps*. As perguntas do questionário visam identificar aspectos da Cultura de Colaboração, Comunicação, Compartilhamento, Automatização e Medição. No entanto, é importante ressaltar que o questionário foi elaborado para ser um material de apoio e um roteiro de perguntas, porém novas perguntas surgem das respostas e dessa forma a entrevista não se limita ao conteúdo do questionário. As tabelas a seguir apresentam algumas das perguntas do questionário.

Tabela 3.1 – Perguntas de contexto geral

Pergunta
Quanto tempo você trabalha na empresa?
Quanto tempo você trabalha neste setor?
Poderia me contar um pouco sobre sua experiências anteriores em TI?
Descreva como é seu trabalho aqui no setor. Um dia típico e um dia atípico.
No seu ponto de vista, a TI sofre pressões externas para produzir mais e em menos tempo?

Tabela 3.2 – Perguntas de cultura de colaboração

Pergunta
Você trabalha diretamente em conjunto (para alcançar o mesmo fim) com alguém?
Há diretrizes/regras/processos de trabalho conjunto aqui na empresa?
Como você contacta alguém da equipe quando você precisa de ajuda? (ferramenta)?
Cita para mim, as pessoas que você consulta quando há algum problema (em ordem de importância para você)
Você é incentivado a colaborar com alguém aqui dentro da empresa?

Tabela 3.3 – Perguntas de compartilhamento

Pergunta
Como é o seu contato com os outros setores/times de TI?
É clara para você a função de cada setor/time de TI?
O seu setor/time depende do trabalho/resultados de outro ou vice-versa?
Existe alguma iniciativa de compartilhamento de conhecimento dentro da organização?

O público alvo das entrevistas são os integrantes do setor de desenvolvimento de sistemas e de infraestrutura de TI que para efeito de correlação com a nomenclatura *DevOps* utilizada na literatura, equivale ao time de Operações. A quantidade de entrevistas planejadas foram de no mínimo 6, buscando ter a mesma quantidade do setor de desenvolvimento e do setor de infraestrutura. As entrevistas foram planejadas para serem realizadas nos horários de menor fluxo de trabalho, sendo o início ou final do expediente de trabalho dos entrevistados. Para uma boa fluidez das entrevistas, os relatos dos entrevistados são gravados e posteriormente transcritos para a análise. Por fim, a análise é feita utilizando o método *Grounded Theory* com apoio do software *ATLAS.ti*[1] que se trata de um software de pesquisa e análise de dados qualitativos que implementa os conceitos da *Grounded Theory*.

3.1.2 Execução

As atividades foram executadas em conformidade com o que foi planejado. Por se tratar de uma instituição com um fluxo de trabalho muito grande e que se intensifica no período eleitoral (período

[1] Disponível em: https://atlasti.com/

Tabela 3.4 – Perguntas de automatização

Pergunta
Como é o ciclo de vida de desenvolvimento até chegar ao Deploy? Alguma parte desse ciclo é automatizada?
Como é o processo de implantação dos sistemas? Do *commit* no git até o *deploy* em produção (pipeline de implantação). Quais ferramentas, tecnologias e equipamentos auxiliam nesse processo? Alguma parte desse processo é automatizada?
Como é o processo de Disponibilização e Monitoramento dos sistemas? Quais medidas são tomadas para que o sistema esteja sempre no ar. Quais ferramentas, tecnologias e equipamentos auxiliam nesse processo? Alguma parte desse processo é automatizada?
(Além das mencionadas) Existem tarefas que poderiam ser automatizadas? (tarefas manuais e repetitivas)

Tabela 3.5 – Perguntas de garantia de qualidade, medição e monitoramento

Tipo de Pergunta	**Pergunta**
Garantia de Qualidade / Medição	Existem métricas de qualidade de serviço aqui?
Monitoramento	As etapas do ciclo de vida de desenvolvimento são monitoradas? Como ocorre esse monitoramento? Que métricas/dados são coletados?
Garantia de Qualidade	Como o processo de Deploy, Disponibilização e Monitoramento poderiam ser melhorados? Cite um a um.

em que as entrevistas ocorreram), foi difícil agendar entrevistas com os colaboradores. Mesmo com as dificuldades, foram entrevistadas 8 pessoas, sendo 4 do setor de desenvolvimento de sistemas e 4 do setor de infraestrutura de TI com os perfis descritos na Tabela 3.6.

Tabela 3.6 – Perfil dos entrevistados

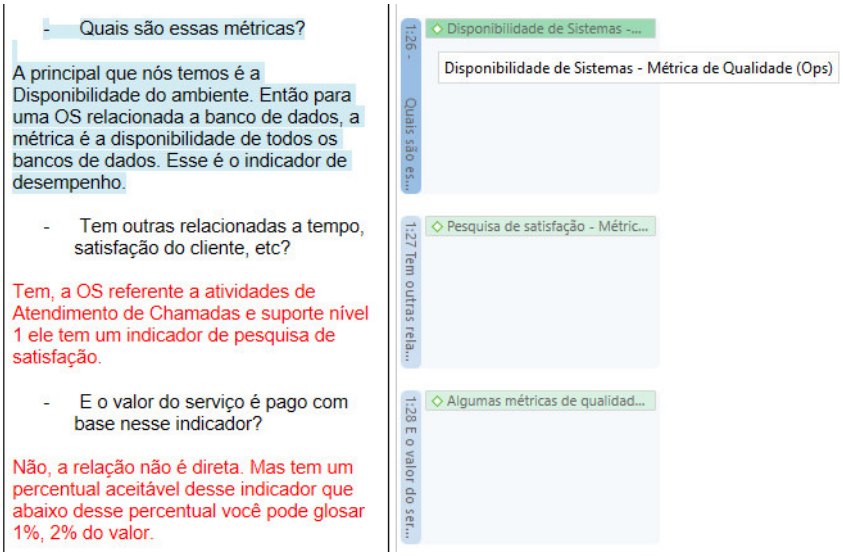
ID	**Tempo de Empresa**	**Setor**	**Função**
P1	1 ano e 8 meses	Desenvolvimento	Programador
P2	2 anos e 2 meses	Desenvolvimento	Arquiteto de Sistemas
P3	1 ano e 6 meses	Infraestrutura de TI	Supervisor de Redes e Datacenter
P4	3 anos	Desenvolvimento	Analista de Sistemas
P5	10 meses	Desenvolvimento	Programador
P6	3 anos	Infraestrutura de TI	Analista de Datacenter
P7	16 anos	Infraestrutura de TI	Analista de Segurança
P8	1 ano e 6 meses	Infraestrutura de TI	Supervisor de Segurança

As entrevistas tiveram durações que variaram de 26 minutos e 40 segundos, na entrevista mais curta, a 49 minutos e 31 segundos, na entrevista mais longa. Por consequência da longa duração das entrevistas, grande parte do esforço de trabalho na execução do Estudo de Caso foi aplicado nas transcrições dos áudios das entrevistas. As transcrições demoraram de 4 a 5 vezes a duração dos áudios, devido às pausas para a digitação e para compreender o que foi dito, além de formatar e estruturar os textos, sendo, dessa forma, um trabalho exaustivo.

Após já se ter uma boa quantidade de entrevistas, iniciou-se codificação aberta dos dados em paralelo com as entrevistas e transcrições restantes. Na codificação aberta, foram selecionadas citações importantes dos entrevistados e atribuído um código a cada uma dessas citações, o código em questão é uma frase com a informação que pode ser extraída daquela citação e que está relacionada a algum conceito

DevOps. A Figura 3.2 mostra um trecho da transcrição da entrevista do Participante 3 e como as citações (lado esquerdo) foram associadas aos códigos (lado direito).

Figura 3.2 – Codificação aberta

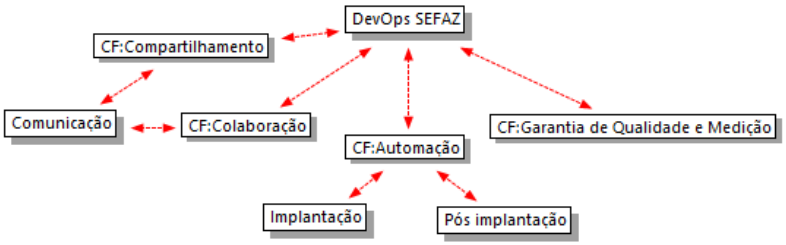


A partir da codificação aberta nos arquivos transcritos dos 8 entrevistados, foram criados 130 códigos. Então se iniciou a segunda parte da codificação, a codificação axial, que se trata de refinar e agrupar os códigos da fase de codificação aberta em famílias de códigos relacionados a um conceito mais amplo.

4 Resultados

Nesta seção detalhamos os resultados da análise qualitativa dos dados. Como foi dito anteriormente, a fase de codificação aberta gerou 130 códigos, na fase de codificação axial esses códigos foram agrupados em famílias de códigos seguindo os conceitos *DevOps*. As famílias de códigos foram Compartilhamento, Medição, Automação, Cultura de Colaboração, Garantia de Qualidade e também se julgou viável incluir a família Comunicação que está diretamente relacionada com a cultura de colaboração e compartilhamento. A Figura 4.1 mostra a rede dessas famílias. Cada uma dessas famílias de códigos tem sua própria rede de relacionamentos que serão detalhados a seguir.

Figura 4.1 – Rede Geral



4.1 Cultura de Colaboração

Ao analisar a cultura de colaboração na SEFAZ/MA, foi possível fazer uma rede de associações entre os códigos referentes a este assunto. Esta rede de associações pode ser vista na Figura 4.2. A partir da rede apresentada podemos notar que o código *Existe uma cultura de colaboração na SEFAZ* é nó que contém mais ligações na rede. Existem 7 citações associadas a esse código, onde os entrevistados passam a informação de que o ambiente de trabalho é colaborativo. Seguem duas dessas citações:

> *Você é incentivado a colaborar com alguém aqui dentro da empresa? "Não diria incentivo, mas sim um costume nosso de ajudar uns aos outros. Se a gente vê que uma pessoa está tentando resolver algo e não consegue, nós vamos lá e ajudamos. Incentivo em si da SEFAZ não tem."* [Participante 6]

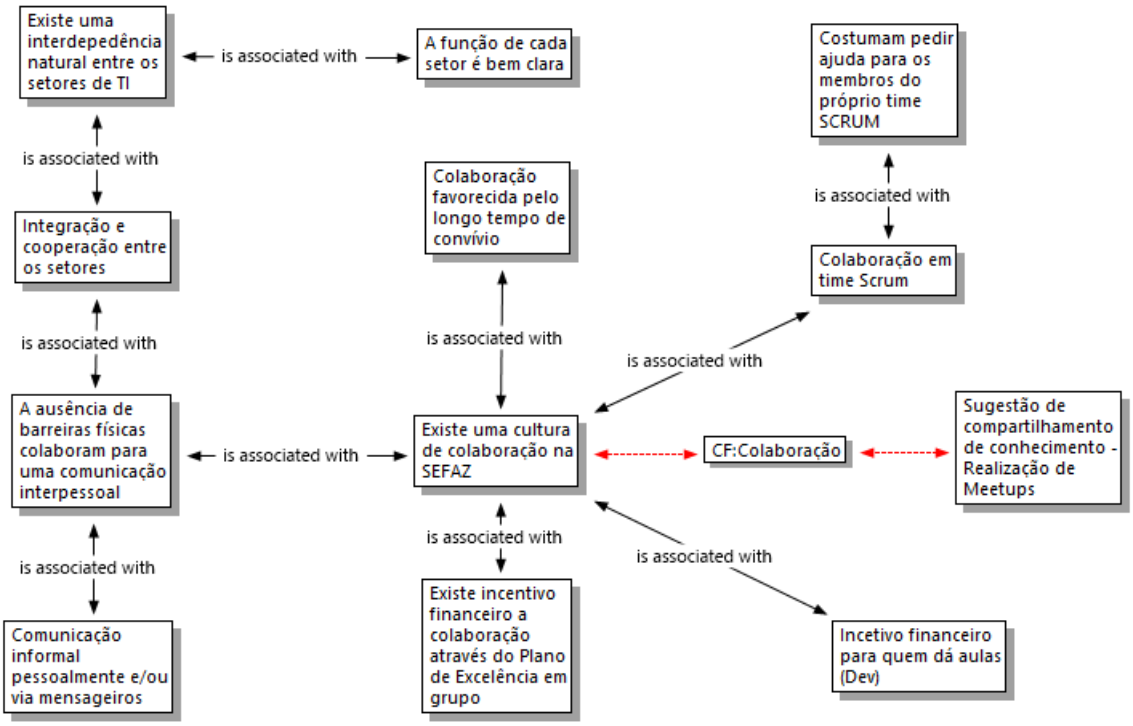
> *Como você contata alguém da equipe quando você precisa de ajuda? "Meu contato com eles é bem tranquilo, pois eles são bem solícitos. O pessoal aqui, seja do sistema, da segurança ou de redes, sempre que preciso de ajuda ou possuo dúvidas de como algo funciona, não existe nenhum problema, pois eles param o que estão fazendo e buscam ajudar."* [Participante 7]

Continuando com a análise da rede é possível perceber outras informações importantes sobre o ambiente de trabalho. É citado quatro vezes que "*A ausência de barreiras físicas colaboram para uma comunicação interpessoal*", a comunicação é presencial, cara a cara, os setores são próximos e as divisórias entre eles são de vidro o que facilita a comunicação. É dito que longo tempo de convívio favorece a colaboração, que a função de cada setor é bem clara e que existe uma interdependência natural entre os setores. Abaixo algumas dessas citações:

> *Você é incentivado a colaborar com alguém aqui dentro da empresa? "Aqui a colaboração e a comunicação são facilitadas, por não existirem barreiras físicas e também por ter ferramentas como o e-mail, o Citsmart, que é a nossa ferramenta de chamados, e o Redmine."* [Participante 2]

> *Como é o seu contato com os outros setores/times de TI? "É um contato harmônico e tranquilo, porque cerca de 80% da equipe já se conhece. Nós éramos uma empresa só e com*

Figura 4.2 – Rede do Grupo Cultura de Colaboração



Fonte: O Autor

> *a mudança de edital nos dividimos para duas empresas, uma de desenvolvimento e outra de infraestrutura, apesar de serem empresas distintas, o ambiente continua como se fosse um só."* [Participante 7]

> *O seu setor depende do trabalho/resultados de outro ou vice-versa? "Os quatro setores são bem interdependentes: desenvolvimento, banco de dados, infraestrutura e escritório de projetos."* [Participante 5]

No setor de desenvolvimento é adotado a metodologia *Scrum*, onde cada projeto tem um time que trabalha colaborativamente. Também são citados dois tipos de incentivos financeiros muito específicos relacionados a colaboração. Estes incentivos são para quem domina um assunto no desenvolvimento e pode ministrar aulas e o outro incentivo é relativo ao plano de excelência da empresa terceirizada de infraestrutura que pode ser individual ou em grupo, quando o grupo alcança a meta do plano, eles recebem um bônus em dinheiro, dessa forma, um incentiva e ajuda o outro a alcançar as metas do plano.

A partir da análise do ambiente de trabalho e de suas características relacionadas a colaboração, pode-se validar a afirmação de que existe uma cultura de colaboração na SEFAZ/MA.

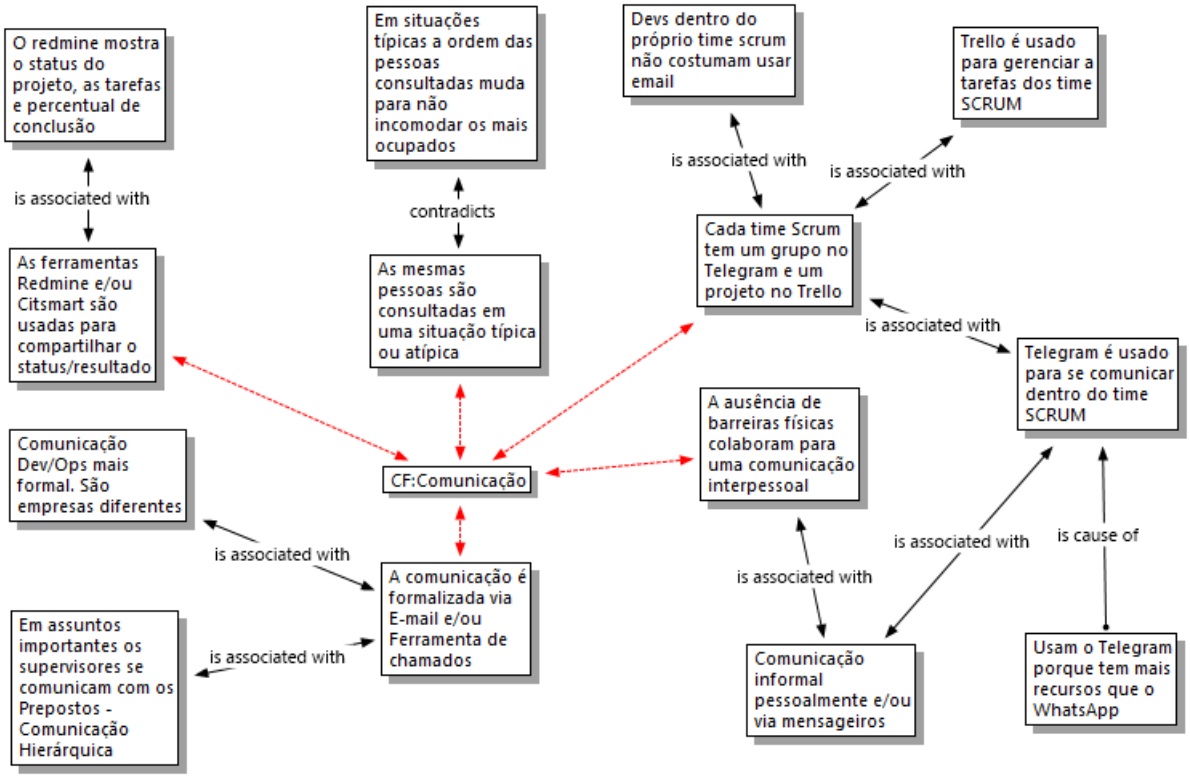
4.2 Comunicação

Para complementar os achados tanto relativo ao princípio de cultura de colaboração quanto ao de compartilhamento, se analisou de forma aprofundada o processo de comunicação dentro da área de TI da SEFAZ/MA. Na Figura 4.3, é apresentada a rede de relacionamentos identificada na instituição.

A partir da rede de relacionamentos pode se observar que no setor de desenvolvimento cada time *Scrum* tem um grupo no aplicativo de mensagens *Telegram*[1] e um projeto no *Trello*[2] que é um aplicativo de gerenciamento de projeto. Para se comunicar dentro do próprio time Scrum, costuma-se usar

[1] Disponível em: https://web.telegram.org/
[2] Disponível em: https://trello.com/

Figura 4.3 – Rede do Grupo Comunicação



Fonte: O Autor

essas ferramentas ou comunica-se pessoalmente com o membro do time. Quando perguntado o motivo para utilizar o *Telegram*, afirmou-se que ele apresenta mais recursos que o rival *WhatsApp*[3]. No setor de infraestrutura existem grupos de *WhatsApp* também para comunicação dentro do setor. Essas bolhas de comunicação dentro de cada setor, demonstram que a necessidade de comunicação é maior dentro do próprio setor que entre os setores, porém sem diminuir a importância de cada tipo de comunicação. Abaixo algumas dessas citações.

> *Como você contacta alguém da equipe quando você precisa de ajuda? "Sim. Não existem muitas restrições para isso. eu vou lá e pergunto. Não existem barreiras físicas. Existem algumas ferramentas como o Telegram, cada time tem um grupo e Trello cada time tem um projeto lá. Lá compartilhamos dúvidas, problemas e soluções."* [Participante 1]

> *Por que vocês decidiram usar o Telegram e não o WhatsApp? "O telegram tem algumas ferramentas mais interessantes do que o WhatsApp. No Whatsapp gente perde muito o time da conversa quando fica muito extenso. No Telegram gente consegue marcar alguns pontos, consegue marcar uma pessoa em específico e isso é mais interessante."* [Participante 5]

> *"Na SEFAZ tem a comunicação boca a boca, e-mail, o Citsmart e ainda tem os grupos do WhatsApp. Temos os grupo de incidentes no WhatsApp."* [Participante 3]

Como já foi dito, a comunicação informal é bastante presente na instituição, principalmente dentro do próprio setor, mas dependendo do tipo de solicitação, essa comunicação deve ser formalizada via e-mail ou ferramenta de chamados que atualmente é o *Citsmart*. Como as empresas de desenvolvimento e infraestrutura são diferentes, as solicitações entre eles deve sempre ser formalizada. Mesmo depois de solicitar via ferramenta é comum ir pessoalmente até o destinatário da solicitação explicar melhor do que se trata. Quando se trata dos supervisores, a comunicação é mais burocrática, existe uma hierarquia de

[3] Disponível em: https://web.whatsapp.com/

comunicação, onde o supervisor da SEFAZ solicita algum serviço para o preposto da empresa terceirizada e o preposto é quem aloca os recursos e conversa com a sua equipe. A seguir as citações relacionadas:

> *"Aqui na SEFAZ, por se tratar de um órgão público existem algumas questões burocráticas e a gente usa muito e-mail e abre chamados. Pode chegar e falar, mas tem que formalizar. Se o cara usar 30 min do expediente dele para resolver a algum problema tem que formalizar porque a SEFAZ vai ter que pagar depois. Mas a aqui nós temos muita abertura para chegar ir falar, não temos barreiras físicas para isso."* [Participante 2]

> *Cita para mim, as pessoas que você consulta quando há algum problema. "O preposto da empresa terceirizada responsável. Sempre que você quer formalizar. Uma coisa é você estar precisando de uma informação, estar precisando rápido de uma pessoa da equipe aí você direto com o técnico e pede pra ele realizar o serviço. Mas na grande maioria das vezes, quando são atividades mais macro, a gente tem que repassar para o preposto."* [Participante 3]

4.3 Compartilhamento

O compartilhamento de conhecimento e informação é importante para o intercâmbio de aprendizado dentro da organização. A Figura 4.4 apresenta a rede do grupo Compartilhamento que mostra que na SEFAZ/MA existem algumas ferramentas de compartilhamento de conhecimento, como uma Wiki e uma base de conhecimento em uma ferramenta recém implantada na instituição, o *Citsmart*[4]. Os os dados da Wiki estão sendo migrados para o *Citsmart*. A ferramenta *ownCloud*[5] já foi usada na instituição para este fim, mas foi substituída pelas ferramentas mais recentes. A base de conhecimento do *Citsmart* é bastante elogiada na instituição e existe um incentivo financeiro para quem a alimenta, isso faz parte do plano de excelência da empresa terceirizada de infraestrutura de TI. No setor de desenvolvimento, que tem uma outra empresa terceirizada, a questão de compartilhamento de conhecimento não é tão enraizada. Houve quem dissesse que não conhece nenhuma iniciativa de compartilhamento de conhecimento na instituição. Seguem algumas citações relacionadas:

> *Existe alguma iniciativa de compartilhamento de conhecimento dentro da organização? "Nós temos a Wiki que está ficando obsoleta porque a nova ferramenta Citsmart trás uma base de conhecimento."* [Participante 6]

> *Mas essa Wiki ainda é utilizada? Vocês alimentam ela? "Sim, é utilizada, alimentamos sim. Ela vai ficar obsoleta por causa do do Citsmart que a gente ainda está alimentando. Então tem muita coisa da Wiki para trazer para a base de conhecimento do Citsmart. Antes da Wiki tinha também a OwnCloud que o pessoal armazenava todos os documentos do setor e na Wiki e OwnCloud a gente tem tudo documentado."* [Participante 6]

> *E o desenvolvimento, utiliza a Wiki? "O desenvolvimento não. Eles nunca usaram."* [Participante 6]

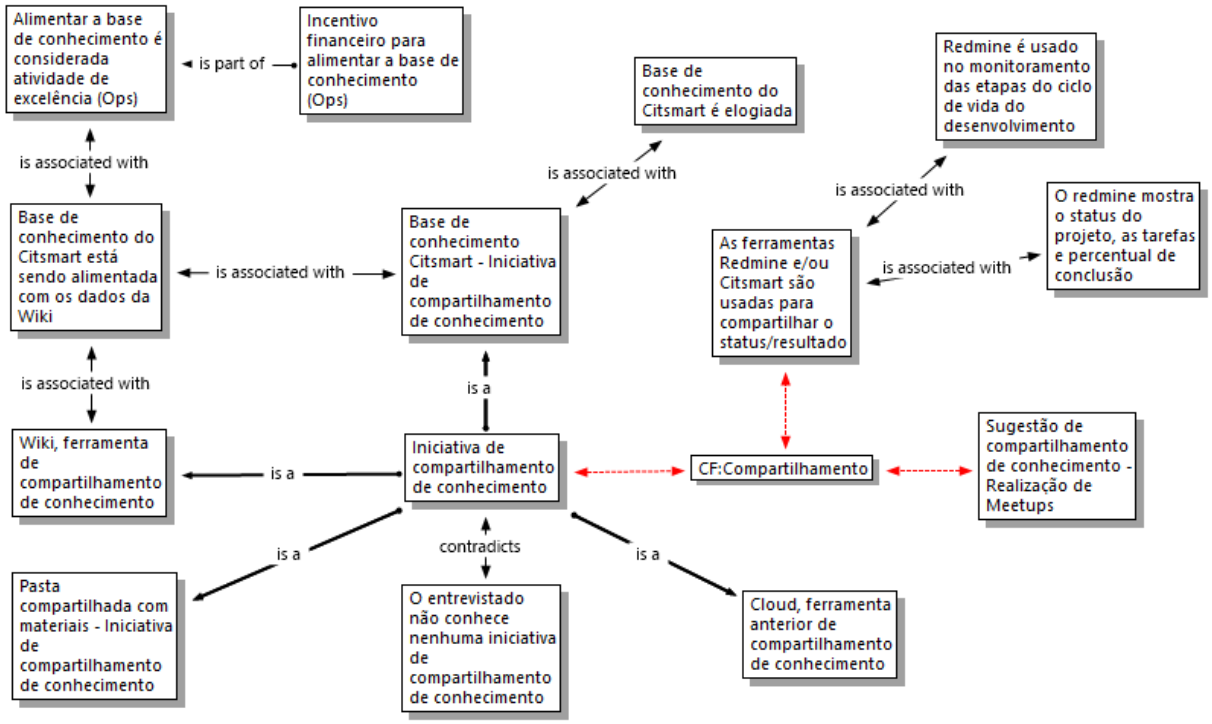
> *Existe alguma iniciativa de compartilhamento de conhecimento dentro da organização? "Não. No OTRS a gente coloca a solução aplicada no atendimento do chamado, mas é basicamente isso."* [Participante 4]

> *Existe alguma iniciativa de compartilhamento de conhecimento dentro da organização? "Não. Acho que só o Git, a pessoa vai e estuda o código. A gente comenta tudo. Mas não existe uma Wiki."* [Participante 5]

Além do compartilhamento de conhecimento, existe também o compartilhamento de resultados e status das atividades pela ferramenta de projetos *Redmine*[6] e pela ferramenta *Citsmart*. Por fim, foi

[4] Disponível em: http://www.citsmart.com/?lang=pt-br
[5] Disponível em: https://owncloud.org/
[6] Disponível em: https://www.redmine.org/

Figura 4.4 – Rede do Grupo Compartilhamento



Fonte: O Autor

sugerido o compartilhamento de conhecimento a partir da realização de *Meetups* que é um encontro informal onde é debatido um assunto de interesse geral. Abaixo duas dessas citações:

> *As etapas do ciclo de vida de desenvolvimento são monitoradas? "Sim. Pelo Redimine. Com ele podemos cadastrar as tarefas. O cliente pode acompanhar o status do desenvolvimento: Codificação, testes, homologação e produção. Pode acompanhar a porcentagem de conclusão."* [Participante 5]

> *"No Mateus nós tínhamos os Meetups, que é uma conversa sobre determinado assunto de interesse geral. Lá era a cada 15 dias, no sábado e a gente tinha de 10h a 12h para se realizar esse Meetup e a empresa dava uma verba pro lanche. Como era o Meetup? É uma reunião e quem domina determinado assunto vai lá apresentar. Eu apresentei dois Meetups sobre REST e Mensageria. Eu falava 40 min e o resto do tempo era discussão. Era bem legal. Eu pretendo fazer isso aqui na SEFAZ também."* [Participante 2]

Do ponto de vista *DevOps* o compartilhamento de conhecimento e resultados influenciam na cultura de colaboração e comunicação entre os setores de desenvolvimento e operações, além de agilizar o processo de tomada de decisão. Na SEFAZ/MA o compartilhamento de conhecimento está em amadurecimento com a base de conhecimento *Citsmart* e compartilhamento de resultados é razoável.

4.4 Automatização

A automatização é um dos princípios centrais do *DevOps* e é onde está concentrada a suas principais práticas. Ao investigar esse aspecto na SEFAZ/MA, além de buscar compreender o ambiente de TI, buscou-se obter sugestões de ferramentas atuais e como elas poderiam ser implantadas. Devido à rede de Automatização ter ficado muito extensa, para facilitar a visualização, o grupo Automatização foi dividido em dois subgrupos, Implantação e Pós implantação.

4.4.1 Implantação

Antes de iniciar a análise das características de automatização na etapa de implantação de sistemas, é importante ressaltar os ambientes e o Pipeline de implantação identificados. Os ambientes foram três: (1) Desenvolvimento, ambiente em que o código é produzido e testado. (2) Homologação, ambiente que simula o ambiente final. É onde o cliente testa e homologa a aplicação. (3) Produção, ambiente final. É onde aplicação vai funcionar de fato para todos os usuários finais. O Pipeline de implantação identificado é apresentado na Figura 4.5 e abaixo algumas citações que fornecem essas informações.

Figura 4.5 – Pipeline de implantação SEFAZ/MA

Fonte: O Autor

> *Como é o ciclo de vida de desenvolvimento até chegar ao Deploy? "Teoricamente as fases são: Solicitação de Serviço (SS), Análise de viabilidade, alocação de recursos, análise de requisitos, elaboração do plano de trabalho, codificação e testes, reunião de apresentação da solução, deploy em homologação, testes em homologação, deploy em produção."* [Participante 2]

> *Como é o processo de implantação dos sistemas? "O setor de desenvolvimento gera a aplicação e manda para o ambiente de desenvolvimento, de lá eles commitam no Git, pegamos do Git fazendo o pull e configuramos constantes de banco de dados, que é onde a aplicação vai se comunicar com o banco de dados, usuário de banco. Montamos a aplicação com um mvn (Maven) e botamos no servidor (Deploy). Ou em Homologação, ou em Produção."* [Participante 6]

Na Figura 4.6 é apresentada a rede de relações do subgrupo Implantação. A análise deste subgrupo pode ser iniciada com a seguinte citação quando perguntado se alguma parte do ciclo de vida de desenvolvimento é automatizada:

> *"Eu diria que na área do desenvolvimento de software, nada é automatizado. Nem os testes. Nenhuma parte do processo é automatizado. Nem o deploy que poderia ser automatizado com o Jenkins, de fato é."* [Participante 2]

O ciclo de vida de desenvolvimento se inicia com a solicitação do cliente para a construção de um software e vai até a disponibilização do software pelo *deploy* em produção. Apesar de não existir automatização no ciclo de vida, os entrevistados têm ciência de sua importância e de como isso melhoraria o ciclo. O nó com mais relações na rede é de que os testes, o *build* da aplicação e o *deploy* poderiam ser automatizados e são sugeridas diversas ferramentas que poderiam auxiliar nesse processo. Abaixo duas citações sobre o que poderia ser automatizado e qual ferramenta poderia ser utilizada:

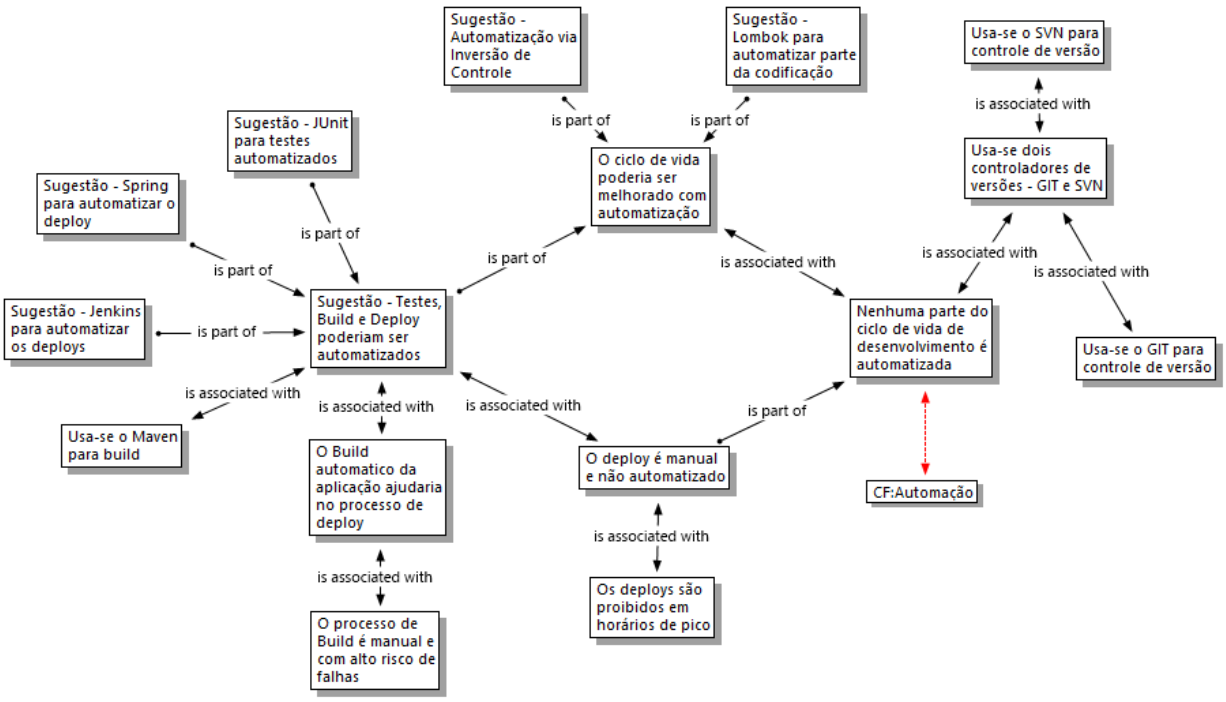
> *"A parte de deploy poderia ser automatizada. Por exemplo, a gente usa o Git e de lá mesmo poderia ser automatizada para fazer o deploy."* [Participante 4]

> *"Também tem ferramentas para automatizar os testes. JUnit, para teste. Jenkins para o deploy. "* [Participante 4]

Práticas *DevOps* como Integração Contínua (CI), Entrega Contínua (CD) e Testes Contínuos e Automatizados fornecem uma solução para os problemas identificados. O fato de já se utilizar o *Git*[7]

[7] Disponível em: https://git-scm.com/

Figura 4.6 – Rede do Subgrupo Implantação



Fonte: O Autor

para o controle de versão e o *Maven*[8] para o *Build* da aplicação é uma vantagem, pois estas ferramentas se integram ao *Jenkins*[9] que é a ferramenta sugerida para automatização dos *deploy* e é atualmente a ferramenta mais popular de Integração Contínua.

4.4.2 Pós Implantação

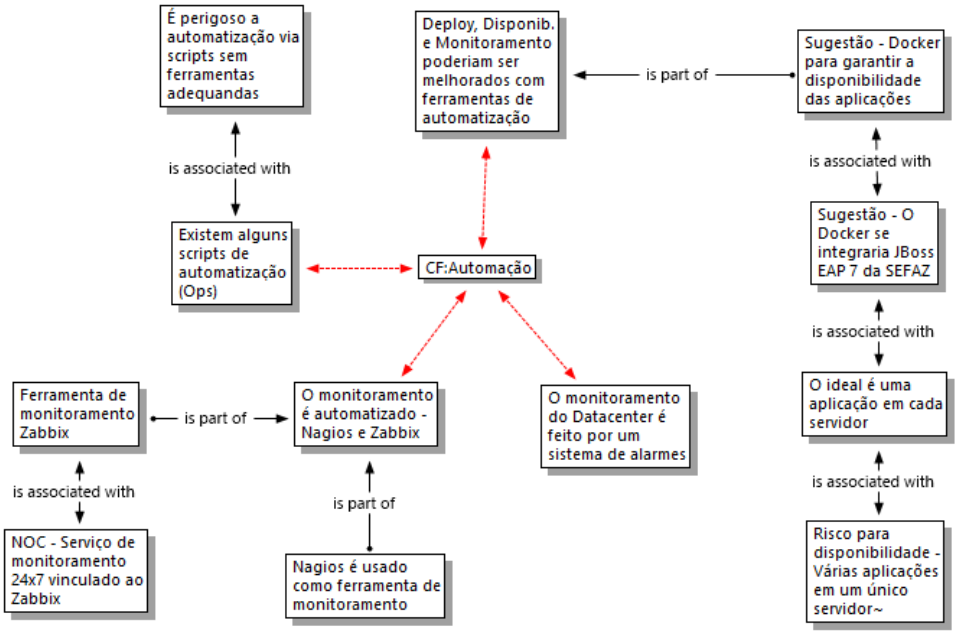
Na Figura 4.7 é apresentada a rede de relações do subgrupo Pós Implantação. Diferente do subgrupo Implantação que não possui nada automatizado, o subgrupo Pós Implantação apresenta alguns resquícios de práticas DevOps como a Infraestrutura como Código (IaC). É citado que existem alguns *scripts* para automatizar algumas tarefas do setor de infraestrutura como alteração de senhas, cargas de arquivos e até deploys, mas que esses *scripts* não são totalmente confiáveis e que podem não funcionar, necessitando-se de ferramentas adequadas para este fim. Segue uma dessas citações:

> *"Deploy poderia ser automatizado, eu tenho scripts prontos para automatizar deploy. Só que a gente não faz por motivo de segurança. Porque os scripts são falhos. pode perder uma conexão, ele cria uma flag lá e não resultar em nada. Temos scripts hoje de arrecadação de arquivos que se perdem facilmente, então tem que ficar olho para saber se ele está funcionando, se está recebendo arquivo, se está mandando arquivo para o banco, porque se ele copiar alguma coisa errada ele entra em loop e dá erro. E por isso, às vezes, a gente prefere não fazer os scripts por segurança."* [Participante 6]

O monitoramento dos sistemas na instituição é feito a partir de duas ferramentas, o *Nagios*[10], o *Zabbix*[11] que está substituindo o primeiro. O *Zabbix* estava em processo de implantação no período das entrevistas, ele faz a automatização do processo de monitorar se os sistemas estão disponíveis. Por outro lado, são citados problemas quanto a disponibilidade dos sistemas. Várias aplicações são colocadas

[8] Disponível em: https://maven.apache.org/
[9] Disponível em: https://jenkins.io/
[10] Disponível em: https://www.nagios.org/
[11] Disponível em: https://www.zabbix.com/

Figura 4.7 – Rede do Subgrupo Pós Implantação



Fonte: O Autor

em um mesmo servidor de aplicação, o que não é uma boa prática. Qualquer problema ou *deploy* de uma aplicação naquele servidor deixa todas as outras aplicações indisponíveis até que o processo seja finalizado ou corrigido. Para ajudar nesse processo de disponibilização, foi sugerido o uso do *Docker*[12] que é uma ferramenta que fornece contêineres. Essa ferramenta é muito associada ao movimento *DevOps*. Cada contêiner guarda uma aplicação. Os contêineres ficam isolados entre si e qualquer alteração em uma aplicação não afeta nas outras. Algumas dessas citações abaixo:

> *"Nós estamos implantando agora o Zabbix. Já colocamos os agentes nos servidores e queremos monitor tudo: aplicação, servidores, tudo. Têm ferramentas de monitoramento que estão sendo mostradas para o pessoal da casa que mostra exatamente onde está o erro da aplicação. Vai lá no Java, no JSF e mostra certinho para o desenvolvedor."* [Participante 6]

> *"Hoje nós temos um servidor, um Jboss para 30 aplicações. O JBoss EAP da Redhat*[13] *pode rodar várias instâncias em um único servidor. Então nós poderíamos colocar uma aplicação para cada instância e se precisássemos fazer um deploy, todas as outras aplicações não seriam derrubadas como é hoje, para poder subir uma aplicação só no servidor. Então eu acho que o Docker ajudaria bastante neste caso."* [Participante 6]

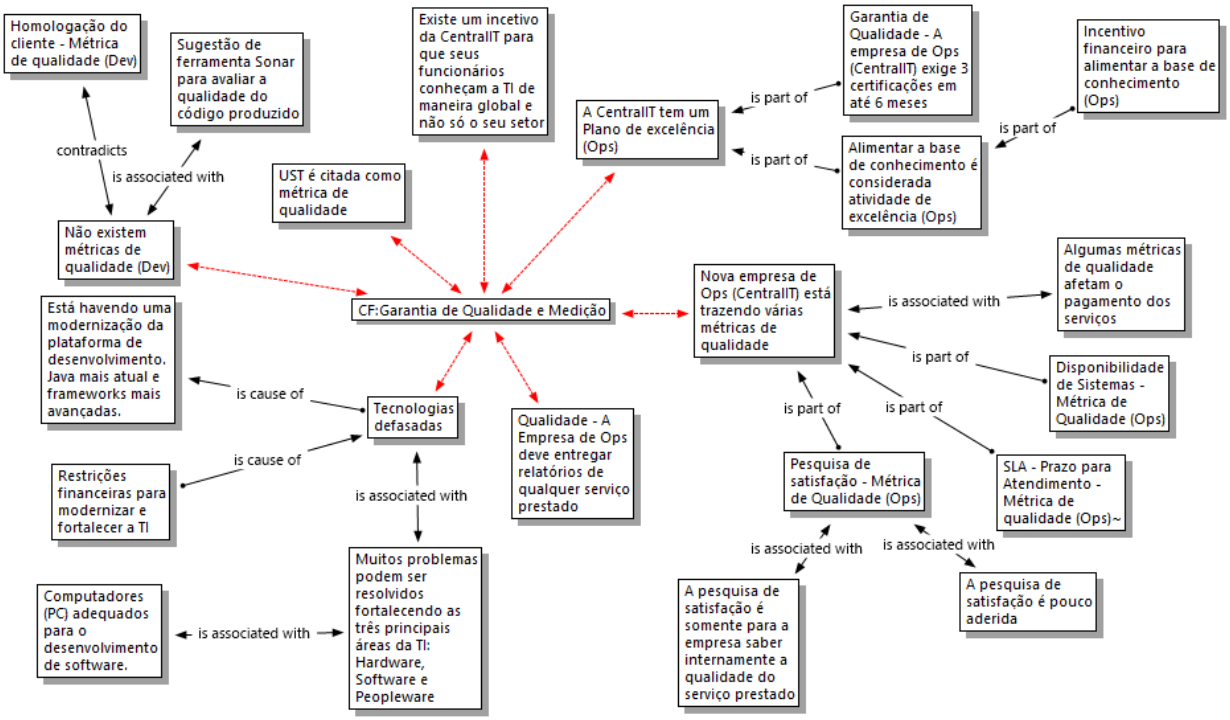
4.5 Garantia de Qualidade e Medição

Estes dois princípios tem uma relação muito próxima. Se por um lado, o princípio da Garantia de Qualidade tem o objetivo de garantir que as atividades sejam realizadas de maneira eficiente e confiável e que os produtos e serviços atendam aos padrões de qualidade estabelecidos. Por outro, a Medição tem o objetivo de coletar métricas eficientes para apoiar a tomada de decisões no ciclo de vida de desenvolvimento e operações. No estudo de caso, quando se analisou a Garantia de Qualidade foram encontradas métricas de qualidade, como podem ser observadas na Figura 4.8, e que atendem aos dois princípios. Então decidiu-se fazer a análise dos dois princípios em conjunto.

As métricas de qualidade do setor de infraestrutura foram levadas pela empresa terceirizada que presta serviço no setor. Fazem parte dessas métricas de qualidade a disponibilidade dos sistemas, o

[12] Disponível em: https://www.docker.com/
[13] Disponível em: https://developers.redhat.com/products/eap

Figura 4.8 – Rede do Grupo Garantia de Qualidade e Medição



Fonte: O Autor

prazo de atendimento de chamados e a pesquisa de satisfação. Onde a principal métrica de qualidade é a disponibilidade dos sistemas que afeta diretamente o pagamento dos serviços prestados, caso não alcance o percentual de disponibilidade estabelecido. A Pesquisa de Satisfação é uma medição interna da empresa terceirizada para saber a qualidade do serviço prestado por seus colaboradores, essa métrica não afeta o pagamento. Como são os clientes externos a tecnologia que fazem a avaliação da pesquisa de satisfação, muitos deixam de fazer. Seguem as citações relacionadas:

> *Quais são essas métricas? "A principal que nós temos é a Disponibilidade do ambiente. Então para uma OS (Ordem de Serviço) relacionada a banco de dados, a métrica é a disponibilidade de todos os bancos de dados. Esse é o indicador de desempenho."* [Participante 3]

> *Na questão contratual da empresa, existem metas que é preciso bater? Disponibilidade? "Sim, existe. O Zabbix vai gerar todos os relatórios para os gestores, porque tem que dá o mínimo de indisponibilidade que é 98,7% abaixo disso a empresa é penalizada, onde vai ser descontado todos os serviços dela, questões financeiras e a parte de glosar."* [Participante 7]

> *E além dessas, tem mais alguma de tempo, quantidade, avaliação do cliente? Dentro do Citsmart tem avaliação do Cliente. Tem a SLA que os chamados básicos você tem até 16 horas para concluir, os chamados médios são 8 horas e os chamados críticos tem até 2 horas para serem resolvidos."* [Participante 6]

Tanto desenvolvimento quanto infraestrutura relataram problemas referentes à defasagem de tecnologias ou equipamentos utilizados. No desenvolvimento, foi dito que possuem bons equipamentos para desenvolver os softwares, porém tecnologias defasadas como a linguagem de programação JAVA, bibliotecas e *Frameworks* em versões antigas. Devido a este problemas está havendo uma modernização da plataforma de desenvolvimento, cenário ideal para projetar uma nova arquitetura de desenvolvimento seguindo às práticas *DevOps*. No setor de infraestrutura, a defasagem relatada é referente aos equipamentos de *Datacenter* e foram apontadas restrições financeiras da instituição como impedimento da aquisição de novos equipamentos. Abaixo algumas dessas citações:

> *Você tem à disposição ferramentas, equipamentos e tecnologias atuais e competitivas? "Em questão de hardware temos sim, está bem evoluído, mas em questão de tecnologias está bem atrás, por exemplo, ainda usamos Struts, Java e JSF em versões antigas enquanto o mercado já tem tecnologias bem melhores. Hoje o mercado tem o React, Vue, Node e isso ainda não chegou aqui."* [Participante 1]

> *"Fui contrato para modernizar plataforma de desenvolvimento, então a gente tem uma plataforma que remete para o ano de 2006, que era quando a gente estava com início na questão da onda, Java na versão 6."* [Participante 2]

> *Você tem a disposição ferramentas, equipamentos e tecnologias atuais e competitivas? "Não. Hardware ultrapassados, máquinas IBM antigas, Blades antigas, Switches antigos. Muita coisa ultrapassada."* [Participante 3]

> *"Infelizmente nós temos restrições de aquisição de Hardware, aquisição de Software, restrições em relação à equipe, ou seja, Peopleware. Nós temos restrições em relação a nomeações de pessoal da casa, servidores da SEFAZ/MA, nós temos pouca de gente da SEFAZ/MA nas áreas."* [Participante 3]

No setor de desenvolvimento foi citada a homologação do sistema pelo cliente como forma de avaliar se o trabalho foi bem feito, mas a maioria dos entrevistados disse não haver métricas de qualidade no setor. Foi sugerida a ferramenta Sonar para avaliar a qualidade do código produzido. Ainda no setor de desenvolvimento, foi citada a UST (Unidade de Serviço Técnico) como métrica de qualidade, mas que deve ser considerada apenas como métrica, pois ela não diz respeito a qualidade do serviço. UST é a medida de esforço em uma determinada tarefa, como por exemplo a criação de um software, e é o critério usado para o pagamento da empresa terceirizada do setor de desenvolvimento.

> *Existem métricas de qualidade de serviço aqui? "Não. Normalmente a gente só percebe se tem qualidade ou não, depois que já está desenvolvido e se não tiver reclamação é porque atendeu o que pediram."* [Participante 4]

> *Existem métricas de qualidade de serviço aqui? "Não. Existe uma ferramenta chamada Sonar que avalia a qualidade do código produzido. Ela seria uma boa ferramenta para avaliar a qualidade do código produzido aqui."* [Participante 2]

> *Existem métricas de qualidade de serviço aqui? "A gente tem a métrica de tempo de trabalho e quantificação das tarefas, a UST."* [Participante 5]

4.6 Resumo dos resultados

A partir das citações dos entrevistados e da análise feita utilizando o método *Grounded Theory*, foi possível identificar as principais características do ambiente de TI da empresa pública estudada. As tabelas a seguir apresentam um apanhado dos achados deste estudo. A Tabela 4.6 apresenta uma comparação entre os princípios, seus achados e a literatura e a Tabela 4.2 faz o mesmo em relação às práticas.

Tabela 4.1 – Comparação entre os princípios, seus achados e a literatura

Princípios	**Achados**	**Literatura**
Cultura de Colaboração	Foi identificada uma cultura de colaboração entre os setores de desenvolvimento e infraestrutura, assim como entre os demais setores de TI da instituição.	O problema em que a separação física entre os setores de Desenvolvimento e Operações causa prejuízos para a colaboração, problema este identificado na literatura por Braga (2015) não é tão nítido aqui. Existe a separação física, mas os setores são próximos e acessíveis.

Além disso, neste estudo foram detectadas várias características e coletadas as sugestões de várias ferramentas dos especialistas de cada área. Esses dados serão úteis numa posterior implantação das práticas *DevOps*. A Tabela 4.3 apresenta uma lista das ferramentas identificadas na SEFAZ e sugeridas durante as entrevistas.

Estas ferramentas, sugeridas pelos próprios profissionais de TI da SEFAZ/MA serão as primeiras a serem avaliadas em uma posterior implantação. Estas ferramentas servirão de apoio para a pesquisa de outras ferramentas correlatas. Serão pesquisadas ferramentas que se integrem e se comuniquem bem umas com as outras levando em conta as ferramentas já existentes da instituição.

Princípios	**Achados**	**Literatura**
Compartilhamento	Existe o compartilhamento de informações e status do ciclo de vida de desenvolvimento através da ferramenta *Redmine*. O compartilhamento de conhecimento está em amadurecimento na instituição a partir de uma ferramenta de base de conhecimento e o incentivo à alimentação desta base, restando a institucionalização desta prática em toda a TI.	Para França, Junior e Travassos (2016) as informações sobre mudanças ou novas características do projeto ou produto devem ser disseminadas para os indivíduos envolvidos (equipes de desenvolvimento e operação de software), e isso promove a visibilidade do fluxo de trabalho. Além disso, para promover o intercâmbio de aprendizado pessoal, informações e conhecimento devem ser disseminados entre os indivíduos.
Automatização	Na etapa de implantação dos sistemas nada é automatizado. Na pós implantação foram identificadas algumas automatizações. Na Tabela 4.2 serão detalhadas a práticas envolvidas nas etapas de implantação e pós implantação dos sistemas.	Segundo, Erich, Amrit e Daneva (2014), há muitas oportunidades para automatizar etapas no processo de desenvolvimento de software. Isso de fato é constatado neste estudo de caso. Para os autores, ferramentas podem ser usadas para suportar automação. Isso melhora a escalabilidade e capacidade de teste, enquanto reduz o trabalho manual. Neste estudo de caso buscou-se identificar algumas essas ferramentas.
Medição	As principais medições identificadas foram a disponibilidade dos sistemas no setor de infraestrutura e a UST (Unidade de Serviço Técnico) que é uma medida de quantidade esforço aplicado em uma tarefa. Ambas influenciam no pagamento dos serviços prestados.	Aqui é observada uma situação mencionada por Bass, Weber e Zhu (2015) em que o time de operações visa manter a estabilidade dos sistemas e o time de desenvolvimento visa criar novos sistemas times, objetivos distintos e muitas vezes antagônicos, porém mesmo essa característica estando presente ela não é causadora de conflitos constantes. Os entrevistados disseram entender a função de cada setor e se acham parte de um todo.
Garantia de Qualidade e Medição	Existem algumas métricas de qualidade, mas precisam ser institucionalizadas em toda a TI.	Para Erich, Amrit e Daneva (2014), a garantia de qualidade tem sido afetada pela adoção de desenvolvimento ágil de software. Isso pode ser percebido na SEFAZ, pois é adotada a metodologia Scrum, mas pouco se importa com a qualidade, o importante é entregar o sistema funcionando. Os autores afirmam que o *DevOps* facilita a garantia de qualidade ao aproximar essas partes por meio de cooperação e ferramentas.

Tabela 4.2 – Comparação entre as práticas, seus achados e a literatura

Práticas	**Achados**	**Literatura**
Integração contínua	Inexiste, mas a instituição já conta com ferramentas como *Git* (repositório), *Maven* (para o *build*), *JUnit* (para os testes) restando apenas uma ferramenta que automatize e e forneça o monitoramento das tarefas.	Na integração contínua, segundo CORREA (2017), os testes de trabalho são integrados a cada *commit* no repositório, permitindo assim detectar e localizar erros rapidamente. Entre as atividades presentes nela estão o controle de versão com repositório único, *build* automatizado, *commits* diários no repositório principal, conjunto de testes automatizados e abrangentes, entre outros.
Testes contínuos e automatizados	Inexiste. É uma etapa diretamente associada à Integração contínua que pode ser implantada na SEFAZ com ferramentas de testes como o *JUnit* e alocando um especialista em testes para esta prática.	Geralmente, o servidor de integração contínua é responsável por realizar a maioria dos testes automáticos e validar os *commits* dos desenvolvedores, entretanto, outros testes automatizados são executados quando o sistema é implantando em outros ambientes detestes, tais como testes de unidade, integração, aceitação, carga, performance, simulação,fumaça, etc.
Entrega contínua	Inexiste. É complementar à integração contínua e testes contínuos.	Entre as atividades que fazem parte dela estão automação de *builds*, testes automáticos, implantação automática, gerenciamento de infraestrutura, pipeline de implantação entre outros (CORREA, 2017).
Infraestrutura como Código	Existe de forma parcial. Existem alguns *scripts* de automatização de tarefas rotineiras, mas carecem de uma ferramenta adequada e segura.	Fernandes et al. (2017) escreve que a infraestrutura como código (IaC) é um tipo de infraestrutura na qual pode-se gerenciar configurações e automatizar através de código escrito. Isso irá eliminar um processo manual em tarefas operacionais e rotineiras de forma segura e em larga escala. Restando na SEFAZ uma ferramenta que garanta essa segurança e automatizar mais tarefas.

Tabela 4.3 – Lista de ferramentas identificadas e sugeridas

Princípio / Área	**Ferramenta**
Automatização / Repositório	Git - Sugerido
Automatização / Build	Maven - Sugerido
Automatização / Testes	JUnit - Sugerido
Automatização / Integração contínua	Jenkins - Sugerido
Automatização / Monitoramento	Nagios
Automatização / Monitoramento	Zabbix
Automatização / Disponibilização	Docker - Sugerido
Automatização / Servidor de Aplicação	Jboss EAP 7
Comunicação / Projetos	Trello
Comunicação / Mensageiro	WhatsApp
Comunicação / Mensageiro	Telegram
Compartilhamento / Chamados / Base de conhecimento	Citsmart
Compartilhamento	OunCloud
Compartilhamento / Projetos	Redmine

5 Conclusão

Este trabalho teve como objetivo a realização de um estudo de caso em uma empresa pública, a fim de analisar o seu ambiente de TI levando em conta os conceitos e as práticas *DevOps*, possibilitando uma posterior implantação desses conceitos e práticas. O estudo de caso foi realizado na Secretaria de Estado da Fazenda do Maranhão, onde foram realizadas entrevistas com os profissionais que trabalham nos setores de desenvolvimento e infraestrutura de TI. Esses profissionais formam questionados sobre seu dia a dia na empresa, suas atividades e sobre informações que poderiam vir a ser úteis em uma futura implantação das práticas *DevOps*.

Com as entrevistas, coleta de dados qualitativos, utilizou-se o método *Grounded Theory* para extrair informações relevantes para a pesquisa e relacionar os conceitos a fim de se ter uma análise coesa do ambiente de TI da instituição pública. Os resultados da análise usando o método *Grounded Theory* foram as citações dos entrevistados, códigos relacionados a estas citações, as famílias de códigos e a redes relacionando os códigos.

A partir dos resultados foi possível identificar os ambientes de desenvolvimento da instituição, o seu pipeline de implantação, além de várias características sobre os princípios *DevOps*. Pôde-se observar que já existe uma boa cultura de colaboração na instituição. Existem ferramentas que possibilitam o compartilhamento de informações e de conhecimento, restando institucionalizar este princípio. Existem algumas métricas de qualidade que também precisam ser institucionalizadas em toda a TI da SEFAZ. Foram observadas algumas automatizações via *scripts* típicas da prática Infraestrutura como Código, porém a maior parte do Pipeline de implantação não é automatizado, abrindo a possibilidade de uma série de automatizações através das práticas *DevOps* em trabalhos futuros. Por fim, foram analisadas as ferramentas existentes na instituição e coletadas uma série de sugestões de ferramentas que podem ser úteis numa futura implantação das práticas *DevOps*. Um dos diferenciais deste trabalho é que não existem empresas no Maranhão que fazem este tipo de análise qualitativa.

Durante a realização do estudo de caso, houveram algumas dificuldades encontradas que podem ser evitadas por quem desejar fazer trabalhos similares. A principal dificuldade foi em relação a transcrição das entrevistas. Durante a gravação das entrevistas poderia ter sido utilizada alguma ferramenta de transcrição para coletar a massa de palavras, o conteúdo precisaria ser tratado e formatado posteriormente, mas já pouparia um certo esforço de digitação. Tentou-se fazer esse processo depois que as entrevistas já tinham sido gravadas, porém sem sucesso. Essas ferramentas de transcrições não funcionam bem depois que as falas estão gravadas, pois elas são gravados com uma série de ruídos do ambiente.

As pesquisas do autor permanecem em andamento, alguns trabalhos futuros e oportunidades foram viabilizadas a partir deste estudo, tais como um estudo aprofundado sobre as ferramentas utilizadas por instituições que adotam as práticas *DevOps*, a implantação das práticas *DevOps* na SEFAZ/MA a partir de outro estudo de caso, a comparação do ambiente de TI antes e depois da implantação das práticas *DevOps*, suas melhorias, consequências e a identificação das dificuldades encontradas. Elaborar orientações de como implantar tais práticas e a criação de um catálogo de informações e dados sobre as práticas *DevOps*. Propor um modelo de diagnóstico de maturidade das práticas *DevOps* em instituições. Comparar a aceitação e adaptação dessas práticas por profissionais com diferentes níveis de experiência.

Referências

ÁGIL, M. Manifesto para o desenvolvimento ágil de software. *Disponıvel em: http://manifestoagil.com.br/. Acessado em: 08 jul. 2018*, v. 17, 2011. Citado na página 11.

BASS, L.; WEBER, I.; ZHU, L. *DevOps: A software architect's perspective*. [S.l.]: Addison-Wesley Professional, 2015. Citado 2 vezes nas páginas 11 e 34.

BIANCHI, E. M. P. G.; IKEDA, A. A. Usos e aplicações da grounded theory em administração. *GESTÃO. Org-Revista Eletrônica de Gestão Organizacional-ISSN: 1679-1827*, v. 6, n. 2, 2008. Citado 2 vezes nas páginas 16 e 17.

BRAGA, F. A. M. Um panorama sobre o uso de práticas devops nas indústrias de software. Universidade Federal de Pernambuco, 2015. Citado 6 vezes nas páginas 11, 12, 13, 14, 16 e 33.

CORREA, R. O. Os desafios na adoção de devops em empresa brasileira de tecnologia da informação no setor bancário. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, 2017. Citado 5 vezes nas páginas 11, 13, 14, 15 e 35.

ERICH, F.; AMRIT, C.; DANEVA, M. Report: Devops literature review. *University of Twente, Tech. Rep*, 2014. Citado 2 vezes nas páginas 13 e 34.

FERNANDES, T. C. M. et al. Influência das práticas do devops nos processos de gestão de ti conforme o modelo cobit 5. *Navus-Revista de Gestão e Tecnologia*, v. 8, n. 1, p. 20–31, 2017. Citado 3 vezes nas páginas 15, 16 e 35.

FRANÇA, B. B. N. de; JUNIOR, H. J.; TRAVASSOS, G. H. Characterizing devops by hearing multiple voices. In: ACM. *Proceedings of the 30th Brazilian Symposium on Software Engineering*. [S.l.], 2016. p. 53–62. Citado 3 vezes nas páginas 11, 13 e 34.

GLASER, B. G.; STRAUSS, A. L. The discovery of grounded theory: strategies for qualitative theory. *New Brunswick: Aldine Transaction*, 1967. Citado na página 16.

HUMBLE, J.; FARLEY, D. *Entrega contínua: como entregar software de forma rápida e confiável*. [S.l.]: Porto Alegre: Bookman, 2014. Citado 3 vezes nas páginas 14, 15 e 16.

JUNIOR, V. G. D. R.; CARDOSO, L. F.; ZALLA, R. S. Devops: Aproximando a área de desenvolvimento da operacional. Instituto Militar de Engenharia, 2016. Citado na página 11.

MELLO, R. Bandeira de; CUNHA, C. Operacionalizando o método da grounded theory nas pesquisas em estratégia: técnicas e procedimentos de análise com apoio do software atlas/ti. *Encontro de Estudos em Estratégia*, v. 1, p. 2003, 2003. Citado na página 17.

OUVERNEY, K.; MARTINS, D. M. S. Automação do processo de implantação de software usando docker e microserviços-um estudo de caso. *Seminários de Trabalho de Conclusão de Curso do Bacharelado em Sistemas de Informação*, v. 3, n. 1, 2018. Citado na página 14.

RATO, F. d. O. *DevOps em sistemas de informação: implementação em operações de tecnologias de informação*. Tese (Doutorado), 2018. Citado 2 vezes nas páginas 12 e 15.

ROCHE, J. Adopting devops practices in quality assurance. *Communications of the ACM*, ACM, v. 56, n. 11, p. 38–43, 2013. Citado na página 13.

SATO, D. *DevOps na prática: entrega de software confiável e automatizada*. [S.l.]: Editora Casa do Código, 2014. Citado 2 vezes nas páginas 14 e 16.

STOL, K.-J.; RALPH, P.; FITZGERALD, B. Grounded theory in software engineering research: a critical review and guidelines. In: IEEE. *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*. [S.l.], 2016. p. 120–131. Citado na página 17.

STRAUSS, A.; CORBIN, J. M. *Grounded theory in practice*. [S.l.]: Sage, 1997. Citado na página 17.

VIRMANI, M. Understanding devops & bridging the gap from continuous integration to continuous delivery. In: IEEE. *Innovative Computing Technology (INTECH), 2015 Fifth International Conference on*. [S.l.], 2015. p. 78–82. Citado na página 13.

ZHU, L.; BASS, L.; CHAMPLIN-SCHARFF, G. Devops and its practices. *IEEE Software*, IEEE, v. 33, n. 3, p. 32–34, 2016. Citado na página 13.