Pós-Graduação em Ciência da Computação

UM PANORAMA SOBRE O USO DE PRÁTICAS DEVOPS NAS INDÚSTRIAS DE SOFTWARE

Por

FILIPE ANTÔNIO MOTTA BRAGA

Dissertação de Mestrado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FILIPE ANTÔNIO MOTTA BRAGA

UM PANORAMA SOBRE O USO DE PRÁTICAS DEVOPS NAS INDÚSTRIAS DE SOFTWARE

ESTE TRABALHO FOI APRESENTADO À PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO REQUISITO PARCIAL PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

ORIENTADOR: *Dr. Alexandre Marcos Lins de Vasconcelos*

CO-ORIENTADOR: *Dr. Célio Andrade de Santana Júnior*

Catalogação na fonte
Bibliotecário Jefferson Luiz Alves Nazareno CRB4-1758

B813p Braga, Filipe Antônio Motta.
Um panorama sobre o uso de práticas DevOps nas indústrias de software / Filipe Antônio Motta Braga. – Recife: O Autor, 2015.
123 f.: fig., tab.

Orientador: Alexandre Marcos Lins de Vasconcelos.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIN, Ciência da Computação, 2015.

Inclui referências e apêndices.

1. Engenharia de software. 2. Avaliação de desempenho (Computadores) I. Vasconcelos, Alexandre Marcos Lins de. (Orientador). II. Titulo.

005.1 CDD (22. ed.) UFPE-MEI 2015-125

Dissertação de Mestrado apresentada por **Filipe Antônio Motta Braga** à Pós Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**UM PANORAMA SOBRE O USO DE PRÁTICAS DEVOPS NAS INDÚSTRIAS DE SOFTWARE**" orientada pelo **Prof. Alexandre Marcos Lins de Vasconcelos** e aprovada pela Banca Examinadora formada pelos professores:

_____
Profa. Simone Cristiane dos Santos Lima
Centro de Informática / UFPE

_____
Prof. Raoni Kulesza
Departamento de Sistemas e Computação/ UFPB

_____
Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 21 de agosto de 2015.

_____
Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

Agradecimentos

Primeiramente agradeço a Deus e ao Espírito Santo pelo projeto ao qual estou inserido com tanto afinco, concentração e foco. Agradeço a minha família o apoio incondicional a toda e qualquer decisão em minha vida e a minha esposa Andrea Motta, pelo incentivo na área acadêmica e também ao apoio a todas as incertezas durante este projeto.

Em especial, agradeço aos meus professores: orientador Alexandre Vasconcelos e co-orientador Célio Santana pelo apoio, disponibilidade e direcionamento durante todo o Mestrado.

Resumo

Hoje as organizações de T.I. estão enfrentando um sério desafio, por um lado temos mercados cada vez mais competitivos, com mudanças quase que rotineiras nos softwares e uma variedade imensa de dispositivos e sistemas operacionais; por outro, os sistemas ficam cada vez mais complexos, integrados a outros serviços e exigindo um alto grau de confiabilidade e disponibilidade dos mesmos, exigindo assim um processo de implantação cada vez mais eficiente e robusto. Nos últimos anos o termo *DevOps – colaboração entre desenvolvimento e operação* - e suas práticas foram amplamente usados e discutidos sob diferentes aspectos. Assim como no movimento ágil, *DevOps* também nasceu na indústria e a partir da necessidade da mesma. Por mais que várias organizações já adotem práticas que ficaram conhecidas como *DevOps*, seu uso ainda não é prescritivo, existindo assim uma variedade de diferentes manifestações de uso em termos de definição e padrão dentre as organizações. Diante disso, esta dissertação teve por objetivo realizar um mapeamento sistemático da literatura e um *survey* em busca do movimento *DevOps*, áreas de concentração dos estudos, os principais autores da área e as principais práticas e formas de uso de *DevOps* nas organizações. Como contribuição, esta pesquisa identificou que dentre as principais áreas de *DevOps* podem-se destacar (i) entrega, (ii) integração, (iii) e testes contínuos, além da (vi) automação da infraestrutura. Como principais práticas foram possíveis destacar: (a) implantações através de máquinas virtuais, (b) visibilidade do *pipeline* de implantação, (c) processos robustos de *roolbacks*, além de técnicas como (d) *canary release,* (e) *toogled features* e (f) *blue-green deployments*.

Palavras-chave: *DevOps*. Entrega Contínua. Integração Contínua. Testes Automáticos. Automação da Infraestrutura. Pipeline de Implantação.

Abstract

Today's IT organizations are facing a serious challenge, on the one hand we increasingly competitive markets with routine software's changes and a wide variety of devices and operating systems; on the other hand, the systems are increasingly complex, integrated with other services and requiring a high degree of reliability and availability requiring a deploy process increasingly efficient and robust. In recent years the DevOps term - a clipped compound of development and operations - and DevOps's practices were widely used and discussed under different aspects. As agile movement, the DevOps term was also born in the industry's need. Today many organizations already adopt practices that became known as DevOps its use is not prescriptive, so there is a variety of different manifestations of use in terms of definition and standard among organizations. Therefore, this work aimed to carry out a mapping study and a survey in search of the state of the art DevOps movement, concentration areas of study, the main authors of the area and the main practices and the use of DevOps in organizations. As a contribution, this research found that among the main DevOps area can highlight: (i) continuous delivery; (ii) continuous integration; (iii) continuous testing; and (vi) infrastructure's automation. Additionally, we can emphasize the main practices in the DevOps adoption: (a) deploys through virtual machines; (b) visibility's deployment pipeline; (c) robust processes roolbacks, and techniques such as (d) canary release, (e) toogled features and (f) blue-green deployments.

Keywords: DevOps. Continuous Delivery. Continuous Integrations. Automation Tests. Infrastructure Automations. Deployment Pipeline.

Lista de Figuras

Lista de Quadros

Lista de Tabelas

SUMÁRIO

1. INTRODUÇÃO

Hoje as organizações de T.I. estão enfrentando um sério desafio, por um lado mercados cada vez mais competitivos, com mudanças quase que rotineiras nos softwares executando numa variedade imensa de dispositivos e sistemas operacionais; por outro lado, os sistemas estão cada vez mais complexos, integrados a outros serviços e exigindo um alto grau de confiabilidade e disponibilidade dos mesmos (REBELLABS, 2013).

Durante vários anos a indústria desenvolveu software de forma prescritiva, ou seja, utilizou-se de práticas de desenvolvimento de software similarmente a outras atividades de áreas como a engenharia, por exemplo. Nessas atividades, exigia-se que fosse definido um conjunto específico de ações, atividades e tarefas definidas em fases de forma sequencial e linear. Alguns modelos prescritivos são citados em Pressman (2011), tais como o modelo em cascata, modelos incrementais, evolucionários e espirais. Todos estes modelos têm como característica principal o sequenciamento de fases e a prescrição de um conjunto de atividades, tarefas, produtos, mecanismos de garantia de qualidade e produto para cada projeto.

Durante muito tempo, as metodologias de desenvolvimento de software ditas como prescritiva atenderam de forma única e adequada às necessidades de negócio relativas àquela época, onde, na maioria dos casos, os softwares não exigiam muitas mudanças nos requisitos o que, inicialmente, eram suportados por esses modelos.

Esses modelos de desenvolvimento de software muitas vezes chamados como *tradicionais* são geralmente baseados em entrega de funcionalidades onde são especificados, desenvolvidos e testados em estágios sequenciais e lineares (HUTTERMANN, 2012). As metodologias tradicionais são também chamadas de *pesadas ou orientadas a documentação* (ROYCE, 1970). Um modelo em que existe uma sequência a ser seguida de uma etapa a outra, geralmente composta por definição de requisitos, projeto do software, implementação, testes, integração, operação e manutenção. Cada etapa tem associada ao seu término uma documentação padrão que deve ser aprovada para que se inicie a etapa imediatamente posterior. Nesses modelos, a equipe de desenvolvimento é composta apenas pelos desenvolvedores. Esses desenvolvem as funcionalidades desejadas e, após o desenvolvimento, a equipe de qualidade, formada também pelos testadores, verificam e validam as funcionalidades. Ainda nesse modelo, toda a responsabilidade

por detectar e encontrar *bugs* no software é da equipe de qualidade, mesmo quando esses são encontrados em produção. Métricas são estabelecidas e a equipe de qualidade é medida pela quantidade de erros encontradas no software. O software não passa para a fase seguinte até que a equipe de qualidade valide o mesmo e defina que o software possa ir à produção. Por fim, após a fase de testes, a equipe de operação, que é composta geralmente por administradores de redes, de sistemas, de banco de dados, dentre outros, prepara o ambiente - servidores, redes, banco de dados - para receber o software que foi previamente analisado, projetado, desenvolvido e testado em etapas anteriores. Geralmente, nesse modelo, a equipe de operação é a que possui a última responsabilidade nesse processo: preparar o sistema para a produção. Ainda nessa abordagem, pode-se dizer que a equipe de operação trabalha de forma completamente isolada das outras equipes, o que pode custar muito tempo para a equipe de operação de trabalhar o ambiente para se adequar ao software que foi desenvolvido e testado, visto que, nesse modelo, a equipe de operação não participa da concepção e do desenvolvimento do software. Sendo assim, a equipe de operação geralmente gasta muito tempo configurando sistemas operacionais, ambientes, infraestrutura de redes, configurações de hardware e, muitas vezes, um *middleware* para integração com a aplicação desenvolvida.

No contexto de negócio em si esse modelo de desenvolvimento pode nos proporcionar retrabalho e consequentemente estouro de custos estimados, pois os requisitos elicitados na concepção do software podem não mais serem necessários quando o software estiver apto a ir à produção. Nos modelos ditos tradicionais de desenvolvimento, quanto mais tarde detectar que os requisitos elencados não mais satisfazem as necessidades do negócio, mais caros ficam de serem desenvolvidos ou corrigidos (PRESSMAN, 2011).

Dados históricos de 2001 (STANDISH GROUP, 2001), utilizando como base 8380 projetos, mostraram que apenas 16,2% dos projetos foram entregues respeitando os prazos e os custos e com todas as funcionalidades especificadas. Aproximadamente 31% dos projetos eram cancelados antes de estarem completos e 52,7% eram entregues, porém com prazos maiores, custos maiores ou com menos funcionalidades do que especificado no início do projeto. Dentre os projetos que não eram finalizados de acordo com os prazos e custos especificados, a média de atrasos era de 222%, e a média de custo era de 189% a mais do que o previsto. Considerando todos os projetos que eram entregues além do prazo e com custo maior, na média, apenas 61% das funcionalidades originais eram incluídas. As principais razões destas falhas estavam

relacionadas com o modelo tradicional de desenvolvimento de software. Esses dados demonstraram que a maioria dos softwares desenvolvidos estouravam os prazos e orçamentos, entregavam software de baixa qualidade e que, muitas vezes, não atingiam os requisitos previamente estabelecidos gerando assim projetos não gerenciáveis e com códigos de difícil manutenção.

Diante dessas dificuldades no desenvolvimento de software nasceu, na década de 90, o movimento ágil, que teve como principal objetivo que a T.I. respondesse de forma mais dinâmica as constantes mudanças do negócio. Ficou estabelecido assim o Manifesto Ágil (AGILE MANIFESTO, 2004) que teve como conceitos privilegiar *indivíduos e interações* a processos e ferramentas, s*oftware executável* a documentação, *colaboração com o cliente* a negociação de contratos e *respostas rápidas a mudanças* a seguir planos. Nesse contexto as metodologias ágeis vieram como instrumento para recuperar a confiança do negócio com a equipe de desenvolvimento. 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, times pequenos e multifuncionais, dentre outras (RUBIN, 2012).

Diferentemente dos times tradicionais de desenvolvimento, onde a equipe de desenvolvimento trabalha de forma isolada do restante da equipe, no contexto das metodologias ágeis desenvolvedores, testadores, analista de negócios, gerente de projeto e produto trabalham juntos. Portanto, o time nesse aspecto compartilha o mesmo objetivo: desenvolver software de maneira eficiente alinhado aos requisitos de software, mesmo que em constantes mudanças. Com o surgimento e aceitação das metodologias ágeis de desenvolvimento, os desenvolvedores, testadores e demais membros da equipe do projeto se agruparam em prol do objetivo de desenvolver software ágil que responda as necessidades do negócio. Entretanto, ainda no contexto das metodologias ágeis, a equipe de operação continuou isolada, continuou trabalhando como um time a parte, recebendo os *builds*, integrando-os e os colocando em produção.

Com o uso cada vez maior de práticas ágeis as empresas - equipe de desenvolvimento e o negócio - se tornaram mais eficientes com entregas mais rápidas em um período de tempo menor. No entanto, toda essa mudança gerou, para a equipe de operação, uma demanda de novos sistemas e *builds* muito maior do que antes resultando, muitas vezes, em uma taxa muito elevada de erros, fazendo com que,

muitas vezes, o código desenvolvido e testado não fosse implantado com rapidez e a confiabilidade necessária (HUMBLE e FARLEY, 2010).

Assim surgiu outra lacuna ainda não contemplada por essas metodologias: Como entregar software ou uma nova funcionalidade efetivamente e continuamente para o cliente/produção com tantas mudanças nos requisitos?

Entregar um novo *build* ou novas funcionalidades para um sistema em produção se tornou um grande desafio para as empresas de T.I. Processos ágeis como *Scrum e XP*, por exemplo, propõem uma aproximação da equipe de negócio - cliente - com a equipe de desenvolvimento através de um modelo iterativo e incremental e com entregas oficiais entre 2 a 4 semanas. Times de desenvolvimento se tornaram ágeis conseguindo desenvolver rápidos produtos de um modo mais rápido e alinhados com a necessidade do mercado. Entretanto, esses produtos não conseguiam ser implantados com a mesma agilidade. Assim, a equipe de operação se tornou o gargalo do processo de desenvolvimento como um todo.

Trabalhando em times separados em um processo de entrega de software com constantes mudanças, um conflito iminente entre os dois times surge: o time de desenvolvimento é avaliado por desenvolver e entregar novas funcionalidades de maneira eficiente de modo a introduzir mudanças nos softwares, enquanto que o time de operação deseja manter e é avaliado pela estabilidade do software. Desenvolvedores querem que as mudanças desenvolvidas logo estejam em produção, enquanto que a equipe de operação muitas vezes reluta para colocá-las, pois a mudança de um software pode trazer risco à estabilidade no sistema ou na sua implantação. Um *bug* crítico quando o sistema é implantado é um problema de desenvolvimento ou um problema de configuração de ambiente, por exemplo? Problemas de performance na aplicação, por exemplo, tem como causa raiz problemas de desenvolvimento ou de ambiente? Problemas de difícil identificação da causa raiz ou de resolução quando uma equipe não conhece detalhes técnicos da outra nem trabalharam em conjunto na concepção do sistema. Gera-se então um conflito de interesses entre essas duas equipes. Essa divisão entre os times de desenvolvimento e operações se tornou um grave problema tanto na capacidade das empresas de obter novas funcionalidades mais rapidamente para o mercado e tanto na capacidade da T.I. de manter estável seus sistemas operando com alta disponibilidade e serviços de qualidade (DEBOIS, 2008).

Diante disso começaram a surgir alguns questionamentos: De que adianta usar processos e métodos ágeis se existe um grande gargalo na implantação? De que

adianta colocar novas funcionalidades adaptando a T.I. com o negócio de forma eficiente se os requisitos não funcionais não foram bem especificados? De que adianta descobrir que o software falha somente na implantação em produção? Foi a partir dessas necessidades que começaram a surgir movimentos para estreitar as relações entre a equipe de desenvolvimento e a equipe de operação de modo que possam colaborar entre si desde a concepção do projeto até a entrega final do software durante todo seu ciclo de vida. Conforme visto em Humble e Molesky (2011), a divisão tradicional *dev/ops* - desenvolvimento e operação – retarda a entrega de alta qualidade do software prejudicando a maneira da T.I. entregar valor ao negócio do cliente.

Liberar a versão de um software é um procedimento arriscado, não confiável, que custa às empresas tempo e alto custo (HUMBLE e FARLEY, 2010). Transformar esse processo de liberação em um sistema automatizado, confiável, de alta qualidade e de entrega contínua que "simplesmente funciona" é fundamental para competir no mercado de hoje permitindo a adição de novas funcionalidades em questão de horas em vez de dias de uma forma mais eficiente.

Assim surgiu o conceito *DevOps*, um movimento que tem por objetivo remover essas “barreiras” tradicionais do desenvolvimento e operação incentivando o uso de colaboração constante entre os times de desenvolvimento e operação de forma alinhada ao negócio, não apenas na implantação do sistema, mas desde sua concepção e durante todo seu ciclo de vida (DEBOIS, 2008). Além disso, os requisitos funcionais e não funcionais são avaliados de igual prioridade para os requisitos de negócio do cliente. O time de operação no *DevOps* passa a ser mais valorizado e ter voz ativa, trabalhando em conjunto com a equipe de desenvolvimento, dentro de um processo ágil, possuindo uma resposta mais rápida as exigências do mercado reforçando o conceito de infraestrutura como código (HUTTERMANN, 2012).

Práticas *DevOps* pretendem fornecer meios para que o código produzido no desenvolvimento seja implantando em produção, obtendo resultados mais rápidos e previsíveis, e fornecendo assim um fluxo do trabalho da concepção do projeto a entregas contínuas do software. Um dos objetivos do movimento é entregar software em tempos mais curtos (algumas vezes ao dia) aumentando a confiabilidade, estabilidade, resistência e segurança, até algumas vezes ao dia, ao mesmo tempo em que essas versões precisam ser testadas e liberadas em diversos ambientes. Para isso a equipe de operação trabalha de forma ágil lidando com diversos ambientes e mudanças de configuração para suportar a diversidade de ambientes com controle e gerenciando as versões de software. O movimento *DevOps* foca bastante nas práticas

de automação das diversas atividades necessárias para entregar código de qualidade em produção, como: compilação do código, testes automatizados, empacotamento, criação de ambientes para teste ou produção, configuração da infraestrutura, migração de dados, monitoramento, métricas, auditoria, segurança, desempenho, implantação, entre outros. Além do aspecto técnico, *DevOps* incentiva o envolvimento de todos da equipe do projeto de um software como um todo, desenvolvedores, testadores e a equipe de operação. A responsabilidade de especificação/desenvolvimento não é apenas dos analistas/desenvolvedores, a responsabilidade de manter a estabilidade do sistema não é apenas da equipe de operação, todos estão envolvidos, igualitariamente, em termos de responsabilidades no projeto a partir de seu nascimento. As duas equipes trabalhando em conjunto permitem uma melhor distribuição de conhecimento de todas as partes do projeto. Organizações que aplicam *DevOps* não enxergam mais a equipe de operação como um gargalo no processo, mas sim como um agente de capacitação do negócio.

A adoção de práticas *DevOps* proporciona: *rápido "time to market"* obtido através da redução de ciclos e ao aumento das taxas de implantação*, aumento da qualidade* obtido através do aumento da disponibilidade, da taxa de sucesso e de um menor número de falhas e o *aumento da efetividade da organização* obtido através da taxa de valor que agrega ao cliente (KIM, 2013). Além desses fatores, a adoção de práticas *DevOps* proporciona melhorias no desenvolvimento de código e do ambiente de infraestrutura, melhorando assim a performance de T.I. (PUPPET LABS et al., 2014). O processo de desenvolvimento ágil fornece uma versão de software entregável ao final de cada *sprint* e, no ambiente *DevOps*, a equipe de operação constrói o ambiente paralelamente, gerando um ambiente automatizado junto com o desenvolvimento do software (KIM, 2013).

1.1 Definição do problema

Nos últimos anos o termo *DevOps* e suas práticas foram amplamente usados e discutidos sob diferentes aspectos, pois, assim como movimento ágil, *DevOps* também nasceu na indústria e a partir da necessidade da mesma. Por mais que hoje várias organizações já adotem práticas que ficaram conhecidas como *DevOps* em comparação a modelos tradicionais de desenvolvimento (REBELLABS, 2013), seu uso ainda não é prescritivo, existindo assim uma variedade de diferentes manifestações de uso em termos de definição e padrão dentre as organizações.

Embora pesquisas recentes, como visto em Rebellabs (2013), revelem que o

investimento em melhoria de infraestrutura, tarefas de automação, melhoria de testes e monitoramento contra falhas gaste 33% mais tempo que times tradicionais que não fazem esse tipo de investimento*,* esses gastam, em média, 60% a mais no gerenciamento de suporte da aplicação e 26% a mais com problemas em produção. Isso significa dizer que times *DevOps* os quais investem em infraestrutura e colaboração e gastam menos tempo em implantação da aplicação e no gerenciamento de infraestrutura. A pesquisa ainda mostrou que times orientado a práticas *DevOps* liberam versão em até metade do tempo que times tradicionais. De acordo com esses resultados, membros das equipes que adotam a cultura *DevOps* investe mais tempo em automação e melhoria de infraestrutura do que no tratamento e resolução de incidentes. A pesquisa aponta que equipes orientadas a *DevOps* gastam mais tempo fazendo tarefas positivas ou neutras (ou seja, automatizar tarefas repetitivas, melhorias de infraestrutura e a autoeducação), enquanto que equipes tradicionais de TI são consumidas mais por tarefas negativas ou neutras (ou seja, resolução de incidentes, apoio, comunicação). Um dado que a referida pesquisa ainda mostrou que somente 13% das organizações pesquisadas possuem sua *pipeline* completamente automatizada e 58% semi-automatizada, ou seja, automatizada em algum ponto do processo.

Uma outra pesquisa (PUPPET et al., 2014) com mais de 4000 desenvolvedores e com equipe de operação, mostrou que ainda existe muita dificuldade em institucionalizar práticas *DevOps*. Dentre algumas dessas dificuldades foram citadas as que segue:

- 48% das pessoas pesquisadas acreditam que valor de *DevOps* não é entendido fora do grupo que utiliza;
- 43% acreditam que um dos problemas para a implementação de práticas *DevOps* decorre da falta de uma estrutura de gestão comum para as equipes de desenvolvimento e de operações;
- 43% acreditam que a falta de ferramentas específicas de *DevOps* influenciam para o não uso de práticas;
- 33% acreditam ser perda de tempo implementar práticas *DevOps*;
- 19% acreditam não haver suporte para a implementação dessas práticas;

Ainda em (PUPPET et al., 2013) e (PUPPET et al., 2014) os seguintes motivos em organizações que ainda não tinha planos para implementar *DevOps*:

- 43% apontaram que a falta de gerente com os *skills* apropriados influenciou para falta de planejamento;
- 38% apontaram para a falta de time com *skills* apropriados influenciou para a falta de planejamento;
- 19% apontaram como a causa do não planejamento tenha sido restrição orçamentária;
- 14% acharam que não haveria necessidade;

Das empresas pesquisadas que são capazes de aumentar o desenvolvimento e entrega de software para obter vantagem competitiva, quase 70% superam os seus concorrentes com eficiência que não possuem processos de entregas (PUPPET, 2013). Mas, enquanto a maioria das empresas acham que o desenvolvimento e a entrega de software são críticos, apenas 25% acreditam que utilizam com eficiência (REDDY, 2013). Este *gap* de execução mostra a diferença entre a necessidade de entregar software de forma eficaz e a capacidade de fazê-lo.

Essas pesquisas demonstraram, em partes, um desconhecimento da necessidade do uso de *DevOps*, da falta de informações sobre as práticas e de como utilizá-las. A pesquisa demonstrou também a falta de apoio por parte da alta gerência sobre a necessidade de implementar práticas *DevOps*. Extraiu-se também da pesquisa, a falta de um conhecimento agregado das práticas e de um estudo mais profundo a respeito do assunto.

Pesquisas feitas na área têm focado em comparar a respeito times tradicionais versus times orientados a práticas *DevOps* (números de implantação, automação, gerenciamento e melhoria de infraestrutura, comunicação, suporte, treinamento, *overhead*), procuram também verificar como os times reagem a falhas das aplicações, da comparação do tempo de recuperação pós falha, dos tipos de infraestrutura, da automação, do monitoramento e das ferramentas (PUPPET et al., 2014). Enquanto que outras pesquisas verificaram a performance pós implantação de algumas práticas e métricas a respeito de adoção, expertises necessárias, melhorias, métricas sobre o número de implantações em produção, dentre outras. (DAVIS, 2014).

Como se pode notar, a maioria das organizações percebe a importância de melhorar o desempenho da T.I. através de práticas *DevOps* com objetivo de obter uma vantagem competitiva real, porém como se chegar a esses resultados ainda não está claro. O uso de práticas *DevOps* ainda não é prescritivo e nem está estruturado em

formas de processos. O que se tem hoje são várias organizações que se utilizam de práticas que faz com que a equipe de operação trabalhe de forma ágil junto da equipe de desenvolvimento e a equipe de negócio consiga obter vantagens competitivas de mercado através de entregas contínuas tão rápido a funcionalidade esteja desenvolvida e testada, mas quais as práticas realmente as empresas estão utilizando e obtendo vantagens reais ainda não está claro.

Esta pesquisa tem como objetivo fornecer um retrato mais amplo e um mapeamento sobre quais as práticas de *DevOps* são mais empregadas pelas organizações.

1.2 Objetivos

Este estudo pretende realizar uma pesquisa da literatura a respeito do uso de práticas *DevOps* com o objetivo de identificar e entender o movimento de modo a estruturar assim estudos científicos por meio de um mapeamento sistemático. O estudo visa compilar práticas da indústria de software relacionadas à área bem como as pesquisas da literatura a respeito desse movimento. Com os resultados desta pesquisa serão obtidos:

- O movimento *DevOps*;
- As áreas de concentração dos estudos;
- Os principais autores da área e onde se concentram as maiores publicações;
- A extração e análise das principais práticas utilizadas no *DevOps*;
- A identificação das oportunidades de pesquisas na área;

Além do mapeamento, foi realizado uma pesquisa do tipo *survey* afim de extrair informações sobre as práticas que as empresas de fato estão empregando quando dizem que utilizam *DevOps*. A ideia é catalogar o que estas organizações têm em comum, fornecendo assim um guia de melhores práticas envolvendo cultura, processos e procedimentos por trás dessas organizações. O estudo envolve áreas como o aspecto cultural de *DevOps* e de envolvimento de times, padrões de desenvolvimento como a entrega e integração contínua (do inglês, “*continuous integration and delivery”*) e o gerenciamento da operação de T.I., objetivando assim encontrar padrões nas organizações.

1.3 Justificativa

A área de *DevOps* é muito ampla e hoje é mais considerada um movimento do que uma coleção de práticas relacionadas e descritiva, não possuindo assim processos nem metodologias relacionadas. Como uma área nova, em uma pesquisa *adhoc,* poucos estudos acadêmicos foram encontrados na área, a maioria dos artigos foram encontrados na indústria. Um mapeamento sistemático da literatura foi proposto que, segundo (KITCHENHAM, 2007), são mais amplos e podem ser classificados como de natureza exploratória e descritiva. Ainda, segundo Bayley et al. (2007), a extração de dados tem por característica ser mais abrangente focando na categorização e classificação dos resultados. Diante disso, optou-se por realizar o mapeamento sistemático em vez de uma revisão sistemática visto que é uma área nova não existindo assim pontos específicos de áreas de estudo. O objetivo é possuir uma visão geral da área respondendo as seguintes questões de pesquisas.

Q.1 Em quais áreas estão concentrados estudos/pesquisas na área de *DevOps*?

Q.2 Onde se concentram as maiores publicações na área de *DevOps* (periódicos e conferências)?

Q.3 Quem são os principais autores da área?

Q.4 Com quais metodologias ágeis o uso de práticas *DevOps* está(ão) relacionada(s)?

Q.5 Quais são as práticas e técnicas em uso nas organizações que utilizam *DevOps*?

Como visto nas pesquisas realizadas (PUPPET et al., 2013), (PUPPET et al., 2014, (DAVIS, 2014) e (REBELLABS, 2013) existem inúmeras formas de uso *DevOps* nas empresas. As práticas assim denominadas ainda são pouco utilizadas necessitando assim de uma pesquisa/estudo para catalogar informações do real uso em empresas de software afim de se extrair um guia com as melhores práticas de *DevOps*. Para esse fim se justificou realizar um *survey* nas empresas que usam métodos ágeis que estão adotando a utilização de práticas *DevOps.*

1.4 Estrutura do Trabalho

Além deste capítulo introdutório, este trabalho é composto pelos seguintes capítulos:

- **Capítulo 2 – Revisão da literatura:** apresenta o referencial teórico sobre conceitos de *DevOps* e suas práticas relacionadas, além de trabalhos relacionados com esta pesquisa.
- **Capítulo 3 – Metodologia da Pesquisa:** expõe o quadro metodológico, o ciclo da pesquisa, assim como, os procedimentos da coleta de dados;
- **Capítulo 4 – Resultados:** apresenta os resultados obtidos do mapeamento sistemático e da aplicação dos questionários, além de uma compilação das práticas identificadas no estudo;
- **Capítulo 5 – Considerações finais**: descrevem as ameaças à validade da pesquisa, assim como suas contribuições para a academia e para a indústria, recomendações para trabalhos futuros e conclusões finais do trabalho;
- **Apêndices:** apresentam o protocolo do mapeamento sistemático, os questionários de pesquisa e a distribuição de qualidade dos estudos.

2. REVISÃO DA LITERATURA

Este capítulo apresenta uma revisão não sistemática (assistemática) da literatura, utilizada como fundamentação para as teorias que foram usadas durante a pesquisa. Foram pesquisadas as definições de *DevOps* com enfoque em suas principais práticas e modelos, bem como os principais fatores que envolvem sua implantação.

O capítulo está organizado nas seguintes seções:

Referencial teórico: apresenta os problemas da divisão tradicional entre desenvolvimento e operação, a história do nascimento de *DevOps*, seus conceitos e suas práticas - integração e entrega contínua, testes contínuos e automatizados, *pipeline* de implantação, infraestrutura como código e os riscos relacionados à entrega contínua de software.

Trabalhos relacionados: apresenta trabalhos correlacionados a esta pesquisa que analisam os fatores de sucesso envolvidos na implantação, avaliação e continuação da melhoria do processo de software por meio da implantação de modelos nas organizações.

Relação da pesquisa com os trabalhos relacionados: apresenta nesta seção uma relação da pesquisa com os trabalhos relacionados identificando assim o objetivo principal de nossa pesquisa.

Considerações finais do capítulo: apresenta os principais conceitos para esta pesquisa. Esta revisão envolve os conceitos sobre a divisão tradicional sobre desenvolvimento e operação, sobre como nasceu o movimento *DevOps*, seus principais conceitos e práticas relacionadas, além de um mapeamento sistemático.

2.1 Referencial Teórico

Esta seção apresenta a base teórica utilizada para condução dessa pesquisa, resultante de uma pesquisa das publicações mais relevantes sobre entrega contínua de software e *DevOps*. Nas subseções apresentadas adiante serão abordados a divisão tradicional entre os times de desenvolvimento e operação, como surgiu o movimento *DevOps* bem como seus principais conceitos, técnicas e práticas.

2.1.1 Divisão Tradicional Dev/Ops

Em um modelo dito como tradicional em desenvolvimento de software, um grupo de desenvolvimento é chamado de desenvolvedores. Os testadores e a equipe de qualidade por sua vez trabalham à parte aguardando a finalização do trabalho dos desenvolvedores para começarem a iniciar a etapa de testes (verificação e validação) no software. Por fim, o software é entregue à equipe de operação que é geralmente composta por administradores de sistemas, de redes, de banco de dados e outros. A equipe de operação geralmente é a que possui a última responsabilidade nesse processo: colocar o sistema em produção. Geralmente, nesse tipo modelo, a equipe de operação trabalha de forma isolada das outras equipes.

Conforme descrito em Huttermann (2012), esse tipo de abordagem gera uma barreira cultural e organizacional ocasionando:

- Times separados: faz-se com que cada time defenda seus interesses individuais;
- Ausência de uma padronização nos artefatos de software (documentação, modelos, código-fonte etc.): Times especializados, trabalhando separadamente, desenvolvem idioma próprio trabalhando em seus problemas individualmente, ao invés de desenvolverem um idioma comum trabalhando em prol de um único objetivo.
- Temor: Equipes não compartilham conhecimento com receio de perder poder e de suas atividades serem confrontadas.

Nesse contexto, também não se pode deixar de citar os modelos de processo de desenvolvimento que se baseiam em metodologias ágeis. Nessa metodologia, testadores e equipe de qualidade agora fazem parte do time de desenvolvimento, junto com os desenvolvedores, formando assim times multifuncionais (do inglês *cross-functional*), onde cada pessoa, dentro da equipe, assume um papel em uma fase do desenvolvimento, não existindo mais assim funções pré-definidas. Muitas vezes, toda a equipe é chamada apenas de equipe de desenvolvimento. Portanto, o time nesse aspecto compartilha o mesmo objetivo: desenvolver software de maneira eficiente de acordo com as constantes mudanças nos requisitos de software. Em outras palavras, com o surgimento e aceitação das metodologias ágeis de desenvolvimento, os desenvolvedores, testadores e equipe de qualidade se agruparam em prol do objetivo de desenvolver software de modo ágil que responda as necessidades do negócio,

entretanto a equipe de operação continuou isolada, trabalhando como um time a parte.

Com o sucesso das metodologias ágeis em aceitar de forma mais dinâmica as mudanças do negócio, mais e mais empresas começaram a utilizá-las cada vez mais em seus projetos se tornando assim mais eficientes, com resultados mais rápidos em um período de tempo menor. Toda essa mudança gerou, para a equipe de operação, uma demanda muito maior do que antes esperada e suportada, de modo a colocar o sistema em produção e gerenciar sua estabilidade. Trabalhando em times separados em um processo de entrega de software com constantes mudanças, um conflito iminente entre os dois times surgia: o time de desenvolvimento é avaliado por desenvolver e entregar novas características de maneira eficiente, enquanto que o time de operação deseja manter o software estável. Ou seja, desenvolvedores querem que as mudanças implementadas logo estejam em produção, enquanto que a equipe de operação muitas vezes reluta para colocá-las para não gerar novas instabilidades no software na mudança ou em sua implantação.

Os conflitos entre a equipe de desenvolvimento e a equipe de operação podem surgir de diversas formas. Existem conflitos geralmente durante a implantação do software, onde, depois do software desenvolvido, a equipe de desenvolvimento necessita e precisa colocar novas funcionalidades rapidamente para produção. O mercado exige que as mudanças feitas em desenvolvimento sejam implantadas em produção o mais rápido possível, faz parte do negócio que a agilidade das mudanças no software sejam reflitas o mais rapidamente para produção. Mas, e se um *bug* crítico aparecer justamente quando o sistema é colocado em produção? Qual a origem do problema? Os problemas se originaram durante o desenvolvimento ou é um problema específico do ambiente de produção?

Desta forma inicia-se o jogo da culpa (do inglês, “*The blame game”*). Afinal, quem é responsável por resolver esse problema? Já que nesse modelo de desenvolvimento, as equipes trabalham sem cooperação durante o desenvolvimento e em silos separados, uma não conhece detalhes técnicos da outra. A equipe de desenvolvimento, por exemplo, afirma que todos os testes em desenvolvimento passaram e que esse *bug* não existia no ambiente de desenvolvimento, já a equipe de operação também afirma que o mesmo ambiente e infraestrutura estão livres de erros. Afinal de quem é culpa? Onde estão os erros? Alguns desses problemas também podem surgir pós-implantação onde, após algum tempo com o software estável, começa a apresentar problemas de *performance* geralmente originados por capacidade não planejada ou tempo de resposta insuficiente. Os mesmos problemas e

questionamentos em busca dos responsáveis citados anteriormente recorrerão novamente. Isso gera um ciclo de conflitos e interesses dentro da própria equipe.

2.1.2 A História de DevOps

O termo *DevOps* teve origem em 2009, mas teve suas raízes ainda em 2008 quando *Patrick Debois* publicou um artigo intitulado "*Agile and Operations Infrastrucuture: How Infra-gile Are You?*" (DEBOIS, 2008). Nessa pesquisa, *Debois* demonstrou como a infraestrutura poderia responder de forma ágil às mudanças do negócio similarmente como as metodologias ágeis de desenvolvimento respondiam à constante adaptação do negócio ao mercado. Ainda em 2008, em uma conferência sobre práticas ágeis, *Debois* e *Andrew Shafer* apresentaram o trabalho "*Agile Infrastructure*", e ao mesmo tempo, uma lista chamada *agile-sysadmin* foi criada na Europa para discutir sobre metodologia ágeis na infraestrutura, ou seja, como a equipe de operações poderia trabalhar de forma ágil acompanhando assim a equipe de desenvolvimento. A partir daí, uma série de iniciativas, estudos e conferências a respeito do assunto começaram a aparecer e se tornarem populares. Em 2009, durante a conferência *Velocity da O'Reilly*, foi apresentando o trabalho *10+ Deploys Per Day: Dev and Ops Cooperation at Flickr* por *John Allspaw* e *Paul Hammond* (ALLSPAW e HAMMOND, 2009), que teve como ênfase demonstrar um estudo de caso sobre a capacidade de implantação numa empresa depois que a mesma colocou em prática a colaboração entre os desenvolvedores e a equipe de operação priorizando entregar software de qualidade respondendo assim com eficiência à dinâmica do mercado. Essa apresentação foi um divisor de águas para impulsionar o crescimento do movimento. Nesse mesmo evento surgiu a ideia de criar os *DevOpsDays*, evento realizado em diversos países com iniciativas locais com objetivo de disseminar a cultura *DevOps*.

2.1.3 DevOps

As empresas estão sob enorme pressão para criar valor para os seus clientes alinhado à inovação empresarial através de software. Entregar software em produção é um processo que tem que se tornado cada vez mais difícil nas empresas de T.I., equipes ágeis encontram barreiras com o processo de entrega, mesmo o software estando "pronto" ao final de cada iteração. A tecnologia por si só não oferece vantagem competitiva; As organizações estão descobrindo que os modelos tradicionais de desenvolvimento de software e de entrega não são suficientes. Os processos manuais são propensos a erros, quebras, desperdício e atraso de resposta às necessidades de

negócios (WOOTTON, 2013). No entanto, a entrega de inovação de base tecnológica pode ser um diferencial competitivo e, quando sustentada ao longo do tempo, torna-se uma competência essencial. Inovação sustentada significa desenvolver continuamente novas ideias em software inovador, que por sua vez melhora continuamente o valor entregue aos usuários.

Hoje o uso do termo *DevOps* é muito amplo e ainda não prescritivo, porém, sabe-se que o termo *DevOps* envolve inúmeras atividades e aspectos, tais como a cultura, automação, medição e o compartilhamento de conhecimentos (HUTTERMANN, 2012). *DevOps* é, em muitos aspectos, um conceito abrangente que se refere a qualquer coisa que suaviza a interação entre desenvolvimento e operações. No entanto, as ideias por trás *DevOps* são muito mais profundas. *DevOps* é uma abordagem baseada em princípios *lean* e ágil - consideradas práticas principais em *DevOps* - em que as organizações e as equipes de desenvolvimento, operações e departamentos de controle de qualidade colaboraram para entregar software de forma contínua, permitindo a empresa aproveitar mais rapidamente as oportunidades de mercado e reduzir o tempo para obter o *feedback* do cliente. Estendendo os princípios *lean,* centrado na preservação do valor**,** *DevOps* aumenta a competitividade das empresas das empresas através de inovação, entrega e aprendizado contínuo fornecendo as organizações o aumento das oportunidades de mercado e a redução no tempo do *feedback* do cliente (REDDY, 2013).

Apesar de técnicas, processos e ferramentas serem essenciais, *DevOps,* na sua essência, é um movimento cultural, da colaboração e do compartilhamento de conhecimentos entre as equipes. Uma organização pode possuir processos e ferramentas automatizadas mais eficientes possível, porém isso se torna inútil se as pessoas não as usarem de maneira correta. Construir uma cultura *DevOps* então é um passo principal para adoção de *DevOps* focados no objetivo da organização ao invés dos objetivos de equipes separadas (SHARMA, 2014).

Em *DevOps* os times e as organizações adotam uma variedade de princípios de acordo com seu tamanho ou natureza ou metodologias utilizadas, porém, alguns desses princípios, são comuns entre elas, 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 consumidores/clientes* (SHARMA, 2014). Conforme visto na **Figura 1** o

time de operação em *DevOps* 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.

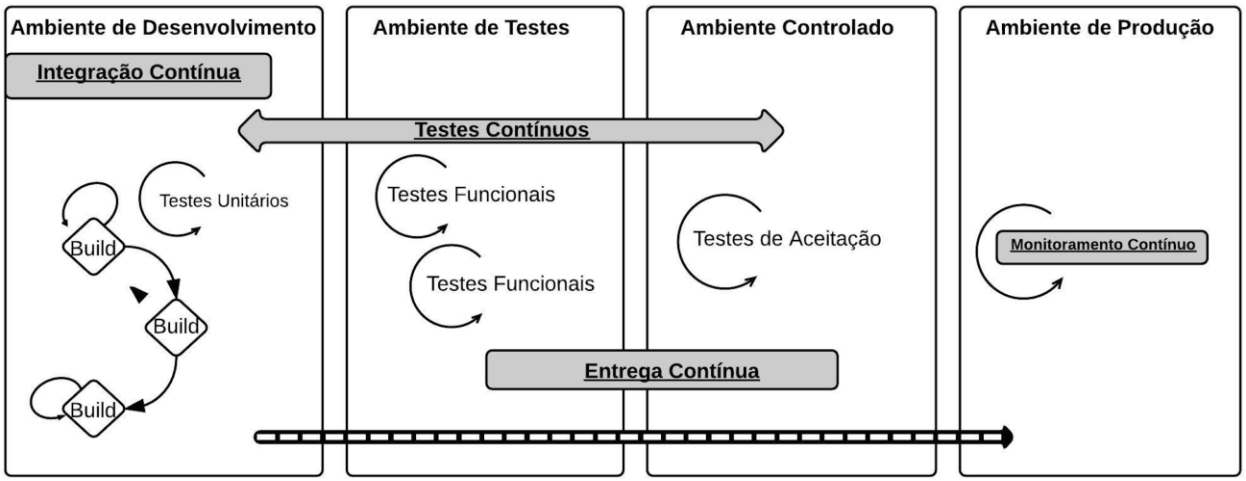

Figura 1: Equipe de Operação em ciclos iniciais de desenvolvimento

Este ambiente similar evita vários problemas a equipe de operação e desenvolvimento afinando a comunicação entre as equipes, verificando o comportamento da aplicação e facilitando assim o processo para a entrega contínua de software. 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* de entrega consiste em um conjunto de estágios que uma aplicação passa desde o desenvolvimento até a produção. 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. Todo 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, mas a **Figura 2** representa um *pipeline* típico de *DevOps* para fornecer agilidade à entrega contínua de software.

Figura 2: *Pipeline* de entrega *DevOps*
Fonte: (HUMBLE E FARLEY 2010)

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. É a partir do repositório de builds que o processo se diferencia das metodologias ágeis tradicionais, pois, a partir desses estágios, múltiplos *builds* são armazenados no repositório para facilitarem suas implantações para fins distintos, ou seja, implantação para o provisionamento de ambientes de testes dos mais variados tipos incluindo testes de volume, funcional, performance e ambiente de produção. Nas abordagens tradicionais, um *release candidate* é atrasado ao máximo para garantir que o software tenha qualidade e esteja funcionalmente completo, mesmo que demorando e se gastando mais tempo e custos, ao passo que em um *pipeline* típico *DevOps* cada *checkin* é um potencial *release candidate* obedecendo o ambiente automatizado dos estágios do *pipeline* e economizando assim tempo e custos em testes prolongados. No *pipeline* de ambiente *DevOps* são feitas várias implantações para vários ambientes similarmente às feitas em produção seguindo uma estratégia de entrega de software, tais como ambiente de testes, de produção, nuvem ou até ambientes compartilhados por terceiros. O objetivo é ter sempre em mente a visualização de versão para ser implantada de acordo com a necessidade de cada ambiente facilitando assim o processo de implantação na escolha da versão a ser implantada. Nesse processo é essencial o uso de virtualização, gerência de configuração e ativos em ambiente de infraestrutura e ambientes em nuvem. Assim, *DevOps* oferece um ambiente com possibilidade e eficiência de implantação em minutos para o ambiente desejado, além de um ambiente com toda visibilidade do *pipeline*.

O *pipeline* de *DevOps* envolve diversas práticas que hoje são implantadas por

organizações de diferentes maneiras, cada uma de acordo com sua necessidade. Conforme visto em (HUMBLE e FARLEY, 2010) e (SHARMA, 2014) algumas práticas são utilizadas no *pipeline* de entrega e também as práticas e técnicas que suportam esse *pipeline.* Algumas práticas que são utilizadas na implantação do *pipeline* são o planejamento do *release*, integração contínua, testes contínuos, implantação e entrega contínua, monitoramento e *feedback* contínuo. Como práticas que suportam esse *pipeline* se pode citar a infraestrutura de ambientes, a gerência de dados e o controle de versão avançado. A seguir, são fornecidas a fundamentação teórica dessas práticas.

2.1.4 Práticas DevOps

DevOps tem como foco principal a comunicação e colaboração entre as equipes de desenvolvimento e operação ajudando essas equipes a trabalharem juntas e de forma mais efetiva. Para isso, hoje, um conjunto de práticas se tornou conhecidas entre os profissionais e que fornecem de base para gerar confiabilidade no processo de institucionalização dessas práticas. Veremos a seguir algumas das práticas mais difundidas de *DevOps*.

2.1.4.1 Integração Contínua

Integração contínua é uma prática explicitamente identificada na metodologia de *Extreme Programming* (XP). É uma prática de desenvolvimento que consiste em um processo de integrar continuamente o trabalho desenvolvido em um repositório com o resto da equipe de desenvolvimento executando os testes do trabalho integrado a cada *checkin* no repositório, permitindo assim detectar e localizar erros rapidamente. O princípio por trás da integração da contínua é: se a integração do código é uma boa prática, por que não fazer isso o tempo todo? (BECK, 2000). Diferentemente de métodos tradicionais de integração onde projetos adiam a integração até o final do desenvolvimento para unificação de *branches* e testes de aceitação, a integração contínua exige que a aplicação seja compilada a cada mudança e que um conjunto de testes automatizados seja executado. Na integração comum, o software será considerado como não funcional até que a fase de testes prove que o mesmo funcione, já na integração contínua, o software é considerado funcional e, a cada mudança no software, um conjunto de testes automatizados é executado para garantir seu funcionamento. Caso haja problemas em alguns dos testes, assume-se que o software

falhou e detecta-se o erro de forma mais eficiente permitindo assim que os defeitos possam ser descobertos mais cedo gerando assim menos prejuízo. A integração contínua permite a entrega mais eficaz do software e com menos defeitos (HUMBLE e FARLEY, 2010).

A chave por trás da integração da contínua é resolver os problemas comuns de desenvolvimento mais rapidamente pois, integrando o software continuamente, ocorre a diminuição do tempo de integração permitindo assim maior fluidez no processo rastreabilidade do problema. Organizações que não usam integração contínua possuem longos períodos entre as integrações fazendo com que seu processo de rastreabilidade de localizar e resolver os problemas aumentem consideravelmente. A integração contínua pode trazer diversos benefícios para a organização, dentre eles podemos citar um tempo menor de depuração, maior adição de características do software, redução de problemas e um menor tempo de integração, e o aumento de visibilidade e comunicação entre as equipes (DUVALL et al., 2007).

Integração contínua é uma prática bastante utilizada nas organizações, entretanto existe uma variação de como ela é definida e usada nas organizações (STAHL e BOSCH, 2013). Para usar integração contínua, existem alguns princípios e práticas chaves que são usadas por algumas organizações. Essas práticas são importantes e trás benefícios a organização, dentre elas, pode-se citar:

- *Usar controle de versão com repositório único* - Todos os itens de configuração do projeto são armazenados em um único repositório (scripts, códigos, testes, scripts de compilação, banco de dados, etc.);
- *Emprego de build automatizado* - O processo de compilação do build deve ser automatizado, permitindo qualquer pessoa compilar, testar e instalar a partir da linha de comando, incluindo banco de dados
- *Executar testes próprios* - Automatizar testes locais antes de integrar o código desenvolvido.
- *Executar commits diários no repositório principal* - A integração contínua melhora a comunicação permitindo que desenvolvedores tomem conhecimento do código desenvolvido por outros para uma maior resolução e rastreabilidade de problemas.
- *Usar servidor de integração* - A cada *commit* no repositório o servidor de integração contínua verifica automaticamente o código fonte na máquina de

integração.

- *Executar um conjunto de testes automatizados e abrangentes* - Garantir que a aplicação foi sucedida na compilação não garante que ela está funcional, é necessário um conjunto de testes automatizados.

Como pode ser visto, integração contínua é o primeiro passo para se implantar *DevOps* e é o primeiro laço entre a equipe de desenvolvimento e equipe de operação. Desse modo a integração contínua se torna pré-requisito para a implantação e entrega contínua (HUMBLE, 2010).

2.1.4.2 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 (HUMBLE e FARLEY, 2010). Ao invés de planejar grandes releases e tudo que envolve esse evento, times que praticam entrega contínua fornecem pequenas fatias de software para a produção com uma frequência muito maior que entregas típicas, frequentemente várias vezes ao dia. Com a entrega contínua é possível entregar novas versões de software para produção, diminuindo assim o *gap* entre a ideia de desenvolver o software à disponibilização do software pelo usuário final, através da automação de todo o sistema de entrega (*build*, implantação, teste e liberação) (DUVALL, 2011). As técnicas e práticas de entrega contínua reduzem o tempo e o risco associados à entrega de novas versões do software para os usuários, permitindo assim o aumento de *feedback* e a colaboração entre desenvolvedores, testadores e o pessoal de operação responsáveis pela entrega (HUMBLE e FARLEY, 2010).

A repetitividade e confiabilidade são fundamentais para a entrega contínua de software. Eles são obtidos através da automatização de todo o processo necessário para compilar e usar o sistema de controle de versão para configurar, implantar e testar a aplicação (HUMBLE e FARLEY, 2010).

A entrega contínua de software tem por objetivo (i) entregar software de uma maneira mais rápida e frequente, adicionando valor ao negócio e obtendo feedback o mais rápido possível, (ii) aumentar a qualidade, *uptime* e estabilidade do software, (iii) reduzir o risco de um *release* testando seus produtos em ambientes de testes e ambientes semelhantes ao de produção, (iv) reduzir o desperdício e aumentar a

eficiência no processo de desenvolvimento e (v) entrega e manter o software em um estado de pronto onde se pode implantar o mesmo de acordo com sua necessidade (WOOTTON, 2013). Para tanto, é necessária automação e uma gerência sobre os artefatos de controle de versão, permitindo assim um processo repetível de *builds*, testes e implantação em um *pipeline* de implantação.

O processo de automação é essencial para entrega contínua, não se alcança entrega contínua sem automação de suas áreas. Wootton Benjamin (2013) descreve áreas e práticas essenciais a serem automatizadas na entrega contínua:

- *automação da compilação e do empacotamento* - automatizar o processo de compilação transformando código fonte em artefatos de implantação.
- *automação de builds e integração contínua* - a integração contínua é o processo central dos esforços da entrega contínua. É a partir da integração contínua que dispara outros processos do *pipeline*, tais como testes, implantação e gerenciamento do *release*.
- *testes automáticos* - processo essencial para a entrega contínua de software, geralmente executado pelo servidor de integração a cada *checkin* dos desenvolvedores, de modo a aumentar a velocidade do ciclo de entrega com aumento da qualidade. Os testes devem cobrir os mais variados níveis de abstração (unidade, integração, aceitação, carga, performance, fumaça). Os testes são distribuídos de acordo com a necessidade dos estágios do *pipeline* de implantação com objetivo de identificar problemas o mais rápido possível, fornecendo assim rápido *feedback*.
- *implantação automática* - Implantar software de forma automática de modo que os times possam ter a capacidade usá-lo de forma *self-service* implantando assim os sistemas em diferentes ambientes sem depender de outros times e/ou da equipe de operação. A implantação automática é uma das áreas chaves para a entrega contínua.
- g*erenciamento de infraestrutura e nuvem* - Incentiva o uso de técnicas de virtualização e implantação na nuvem afim de gerar flexibilidade, rapidez e elasticidade na criação de novos ambientes assim como o uso racional de recursos, usando somente o que for consumir.
- *infraestrutura como código* - Ferramentas de gerenciamento de configuração para infraestrutura permitem definir e controlar a infraestrutura de forma controlada, permitindo construir ambientes consistentes, confiáveis e repetíveis

de forma automática gerando assim a base para se formar um *pipeline* de implantação confiável.

- *pipeline de implantação* - implementar um *pipeline* de implantação é um processo chave para se chegar à entrega contínua. Ele fornece uma visibilidade de andamento de todo o processo de entrega nos estágios e etapas que os *releases candidates* se movem no processo, do código fonte a produção. Fornece também critérios para *build* se mover entre os estágios do *pipeline* e conhecimento para a tomada de decisão de acordo com os resultados.

O processo de entrega contínua preza pela constante melhora do seu ciclo de entrega de modo a buscar como um incremento crescente a estabilidade do processo e da entrega, para isso, a entrega contínua estabelece uma lista de melhores práticas a ser usada durante o processo.

O *monitoramento* é um passo importante na entrega contínua onde permite que alertas sejam disparados para a equipe sobre eventuais problemas que possam ocorrer em produção ou no processo de implantação fornecendo um sinal para que o problema seja investigado e resolvido.

O processo *roolback* também é uma prática fundamental, pois com ele se torna possível voltar para último estado funcional do software caso algum problema seja encontrado em produção, de modo a fornecer segurança suficiente para que o processo se torne confiável a encorajar ainda mais a automação no processo de entrega. Se o *pipeline* de implantação estiver confiável, o processo de *roolback* se torna natural.

Ainda no contexto das melhores práticas, *extrair uma versão correta e específica* para um estágio no *pipeline* de implantação é essencial para a correta validade do processo de testes de implantação. É fundamental que a aplicação possua o mesmo binário do produto de software em ambientes diferentes do *pipeline* para a posterior implantação em produção. Esse processo necessita estar sobre o controle de versão.

Uma outra boa prática para a entrega contínua consiste em implantação canário (do inglês, *canary release*) que consiste em liberar uma nova versão de software em produção somente para um pequeno grupo de usuários. Essa técnica tem por objetivo reduzir o impacto de um possível *bug* ser encontrado em um ambiente de produção isolando usuários não afetados. Além disso, aumenta-se a base de usuários a cada

nova entrega da implantação canário. Com esta nova versão, se consegue, aos poucos, verificar o comportamento normal do sistema. Esta prática também é usada quando se tem versões sutilmente diferentes da aplicação em produção ao mesmo tempo para efeitos de comparação e eficácia, coletando estatísticas de como a novas funcionalidades estão sendo usadas e eliminando as que não estiverem entregando valor suficiente. Isso oferece uma abordagem evolutiva muito eficaz para a adoção de novas funcionalidades. A implantação canário é bastante usada quando se tem uma base heterogênea de usuários onde os testes automáticos do *pipeline* não conseguem ter uma boa abrangência com relação a vários ambientes diferentes.

A implementação de *toogled features* também consiste das melhores práticas de entrega contínua com essa prática pode-se, gradualmente, implantar novas mudanças no sistema em produção habilitando e desabilitando as mudanças aos poucos de forma a adicionar controle e estabilidade às mudanças implantadas em produção. *Toogled features* permite testar uma combinação de mudanças para planejar seu uso em produção além de estar apto a testar também características de testes de desempenho para uma grande base de usuários.

Uma outra técnica importante com objetivo de reduzir riscos de implantação é chamada de implantação azul-verde (do inglês, *blue-green deployments)*. Nessa prática o ambiente possui dois ambientes de produção, um ambiente em uso, podendo ser chamado de *blue*, por exemplo, e outro ambiente, o *green* por exemplo, será o release final após ter passado pelo *pipeline* de implantação estando também no estado de pronto contendo as novas mudanças. Uma vez que o *blue* está em produção, pode ser roteado para o ambiente *green*. Essa técnica de entrega contínua fornece um rápido caminho para o processo *roolback*, fornecendo um fácil caminho de troca caso aconteça algum problema na implantação de uma nova versão.

Para que a entrega contínua de software se torne de fato completa, é necessário a integração de um conjunto de práticas que se movem entre o gerenciamento da configuração, testes automáticos, integração e implantação contínua, gerenciamento de dados, de ambiente e de *release* (DUVALL, 2011).

2.1.4.3 Testes Contínuos e Automatizados

Como visto, os processos *DevOps* produzem código para a produção assim que o desenvolvimento é finalizado. Isso requer que o código seja continuamente integrado

sendo necessário, em paralelo, um conjunto de testes trabalhando junto com a integração dessas funcionalidades. Com implantações sendo feitas cada vez mais rápidas e frequentes, fica impossível executar testes manuais de todas as funcionalidades a cada *release*. Testar é uma atividade multifuncional que envolve todo o time e deve ser executada continuamente desde o início do projeto (HUMBLE e FARLEY, 2010). 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 (HUMBLE e FARLEY, 2010).

Os testes contínuos representam a automação de testes em cada fase fornecendo, portanto, a rastreabilidade de dessas fases e a rastreabilidade desses resultados (RASAD et al., 2014). 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 (MSULU et al., 2013). 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. Em Salf e Ernst (2003) foi feito um experimento que provou que testes contínuos reduzem o tempo de desenvolvimento em mais de 15%. Os testes automáticos e contínuos representam a fase final do *pipeline* de implantação. Em Humble et al. (2006) definem três princípios que devem ser seguidos para implantação de testes automáticos: o primeiro é que diferentes tipos de testes devem ser independentes uns dos outros e que devem receber uma informação se o teste passou ou falhou de acordo com o resultado deles. O segundo é automatizar toda a implantação dos ambientes de testes, estágios e até mesmo de produção. E, finalmente, o terceiro é que se possa testar todos os ambientes que deseje implantar.

Em testes contínuos e automatizados, a cada mudança feita na aplicação, configuração ou no ambiente são executados vários tipos de testes em paralelo como parte do *pipeline* de implantação - unidade, funcional, integração, desempenho - porém nem todas os *builds* precisam se submeter a todos os tipos de testes. Alguns tipos de testes para serem executados podem precisar somente de testes funcionais para

avaliar funcionalidades em vez de se submeter a todo processo de testes passando por testes de performance, por exemplo, como seria o caso de testes exploratórios. Vários testes automatizados são executados pelo sistema de integração contínua a cada mudança feita na aplicação testando aspectos funcionais, mas também aspectos não funcionais, tais como capacidade segurança e outros requisitos. Quanto mais cedo identificar problemas tantos de requisitos tanto funcionais quanto não-funcionais, menor custo de corrigi-los (HUMBLE e FARLEY, 2010). Quando o processo de integração contínua informa para os testadores que os testes automatizados estão completos, os testadores podem começar os testes manuais. Os testes manuais - exploratórios, usabilidade etc.- também são essenciais para se adicionar qualidade continuamente durante todo o projeto. Quando os testes manuais forem concluídos, eles podem aprovar ou não essa versão do software. Se o software passa em todos os testes manuais e automatizados, então ele está pronto para implantação em teste e produção (HUMBLE et al. 2006).

Finalmente, 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 (WOOTTON, 2013). Geralmente o servidor de integração contínua é responsável por realizar a maioria dos testes automáticos e validar os *checkins* 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.

Humble e Farley (2010) detalham os caminhos que uma versão de software passa desde os testes até chegar a produção, conforme visto na **Figura 3**.

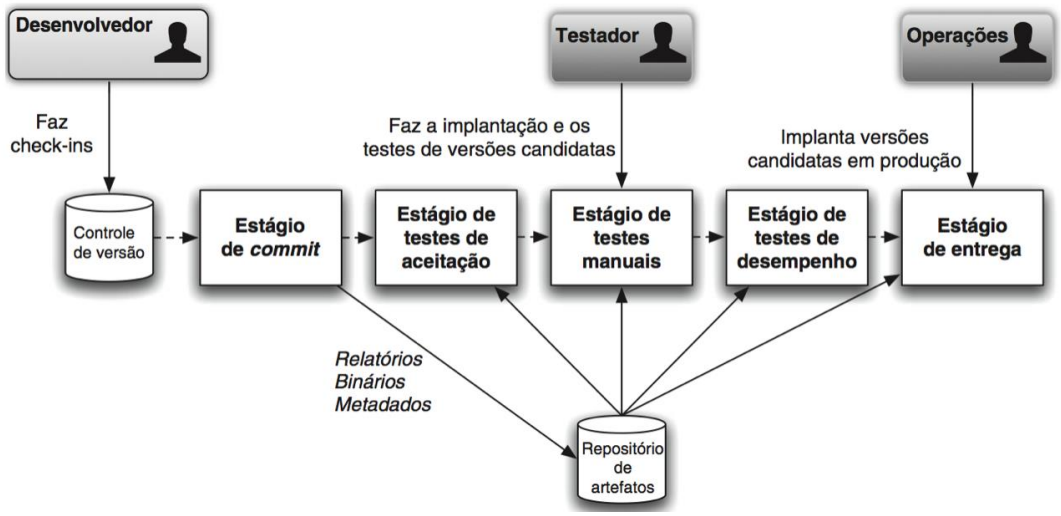

Figura 3: Estágios de testes em um *pipeline* de implantação
Fonte: (HUMBLE E FARLEY 2010)

Os passos são descritos a seguir:

1. A equipe de desenvolvimento executa *checkin* de alguma mudança;
2. O servidor de integração contínua executa commit;
3. Os binários gerados são salvos no repositório;
4. O servidor de integração faz a implantação de acordo com os binarios em um ambiente similiar ao de produção;
5. O servidor de integração executa testes de aceitação;
6. A versão é marcada como tendo passado pelo estágio de aceitação;
7. Os testadores obtêm uma lista de versões candidatas que passaram pelos testes de aceitação e as utilizam de forma *self-service*;
8. Os testadores utilizam testes manuais;
9. Os testadores atualizam a versão afirmando que forma executados testes manuais;
10. O servidor de integração contínua recupera a última versão candidata que passou pelos testes de aceitação, ou testes manuais, dependendo da configuração do *pipeline*, e a implanta em um servidor de testes de produção;
11. Os testes de capacidade são executados na versão candidata;

12. Se os testes de capacidade passarem nos testes de capacidade a versão é atualizada;

13. O padrão é executado para todos os estágios do *pipeline* existentes;

14. Após a versão passar por todos os estágios, a versão pode ser implantada em produção;

15. Finalmente a versão candidata é marcada como entregue ou implantada;

Por fim, algumas melhores práticas são consideradas para serem utilizadas nos testes automáticos e testes contínuos. Em Wootton (2013) considera as seguintes:

- Automatizar mais testes quanto possível;
- Prover uma boa cobertura de testes nos múltiplos níveis de abstração;
- Distribuir diferentes classes no seu pipeline de implantação aumentando os detalhes de testes em produtos similarmente ao de produção

Duvall (2011) considera a *automação de testes* incluindo verificação e validação nos testes de unidade componente, capacidade funcional e de implantação; o *isolamento de dados* de testes usando um subconjunto de dados para analisar o comportamento; os *testes paralelos* rodando através de várias instâncias de *hardware* para diminuir o tempo total dos testes e *o uso de stubs* para simular sistemas externos ou funcionalidades não terminadas/desenvolvida diminuindo assim a complexidade da implantação.

2.1.4.4 Pipeline de Implantação

A integração contínua é uma prática ágil que trouxe produtividade e qualidade para os projetos de software permitindo, que o código desenvolvido por uma equipe, funcione de forma integrada, de modo a fornecer assim um rápido *feedback* para a equipe de desenvolvimento. O principal objetivo da integração contínua é garantir que o código compile com sucesso efetuando alguns testes unitários e de aceitação. Porém, a integração contínua de software não entrega a funcionalidade ao cliente de fato, não implanta o software em produção, a integração contínua fornece *feedback* especificamente para a equipe de desenvolvimento.

Conforme visto em Humble e Farley (2010), é comum verificar equipes de operação aguardando versões de software, testadores à espera de versões testáveis,

equipes de desenvolvimento recebendo *feedback* tardio de funcionalidades já desenvolvidas há tempos. O ponto chave para a entrega contínua de software é a passagem do ambiente de integração contínua para o ambiente de produção, estreitando assim o gargalo entre a equipe de desenvolvimento e a equipe de operação e testes. Diante desse contexto é descrito o *pipeline* de implantação, uma manifestação automatizada do processo de levar o software de controle de versão até os usuários, ou seja, é a modelagem do processo de cada mudança de software que passa por um processo da compilação por vários estágios de testes e implantação.

O *pipeline* de implantação tem por finalidade fornecer *visibilidade* de todos os aspectos da entrega - compilação, implantação, testes, liberação - a todos envolvidos no projeto, fornecer *feedback* e fornecer a *implantação contínua* através de um processo automatizado onde se possa escolher a versão que se deseja implantar em qualquer ambiente (DUVALL, 2011). O *pipeline* também descreve e deixa explícito os estágios que o software passa do código fonte à produção, quais estágios possuem processos automatizados e quais possuem etapas manuais e também mostra os critérios para que o software se mova nos estágios do *pipeline*, fornecendo visibilidade e confiabilidade de todo o processo (WOOTTON, 2013).

A implementação do *pipeline* de implantação traz diversos benefícios, equipes de testes podem criar e implantar seus próprios ambientes sem necessidade da intervenção da equipe de operação e/ou espera de uma versão específica do desenvolvimento, a equipe de operação pode implantar uma aplicação em um ambiente desejado ou até mesmo em produção com um simples apertar de um botão; desenvolvedores podem criar um ambiente de desenvolvimento e podem ver quais versões passaram por quais estágios do processo de entrega e quais problemas foram encontrados fornecendo um *feedback* rápido, tanto sobre o código, quanto sobre o processo de implantação; os envolvidos no processo de entrega conseguem obter visibilidade sobre o processo de entrega de modo a identificar, otimizar e remover gargalos. Isso leva a um processo de entrega que não somente é mais rápido, como mais seguro (HUMBLE e FARLEY, 2010).

Cada mudança no código do *pipeline* de implantação passa por uma sequência de estágios de testes em que cada teste, de acordo com cada estágio, possui características diferentes passando dos testes de viabilidade à aceitação. À medida que o software passa em cada estágio ele vai se tornando mais confiável e mais

próximo de ir à produção. Já os testes que falhem em algum estágio a versão candidata é descartada e o processo de *feedback* a respeito da falha é obtido o mais rápido possível. Os testes automatizados em um *pipeline* de implantação devem provar que as versões candidatas atendem aos critérios de produção. Um *pipeline* básico de implantação pode ser visto na **Figura 4**.

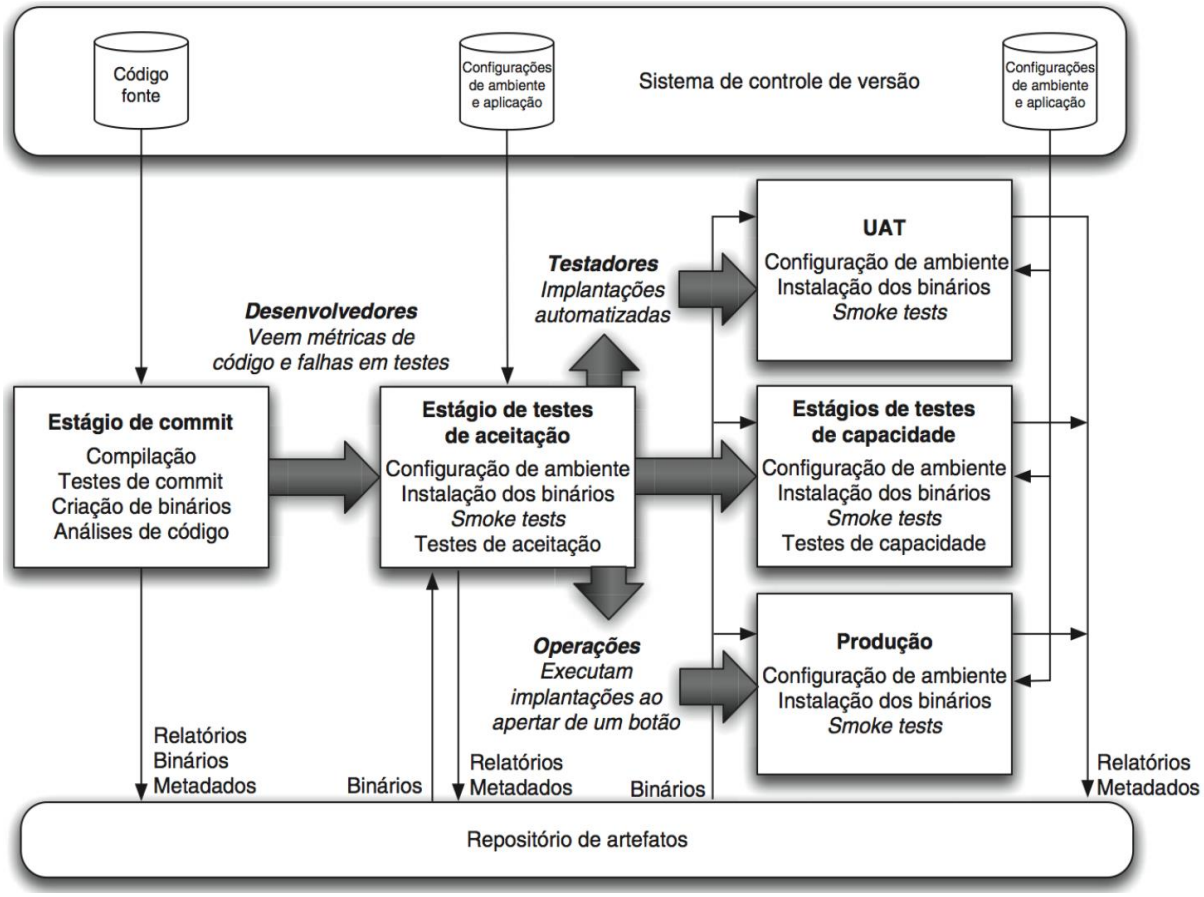

Figura 4: : Pipeline de Implantação Básico

Fonte: (HUMBLE e FARLEY, 2010)

A seguir, alguns passos extraídos de Humble Jez, Farley David (2010) para um *pipeline* de implantação básico conforme visto acima. O processo inicia quando:

1. O desenvolvedor faz checkin no sistema de controle de versão;

2. O sistema de integração contínua responde criando uma nova instância do *pipeline*;

3. O estágio de *commit* compila o código, executa testes unitários, realiza análise do código e cria instaladores e mais paralelamente executam-se outras tarefas como preparar sistemas de banco de dados e outros;

4. Os binários são armazenados em um repositório de artefatos caso os testes unitários sejam bem-sucedidos;

5. Testes de aceitação automatizados são executados se estágio de *commit* for bem-sucedido fornecendo rápido *feedback* à equipe;

6. Nesse ponto o *pipeline* de implantação é dividido em implantações de acordo com cada ambiente apropriado. A equipe deve ser capaz de identificar as versões candidatas, de acordo com cada tipo (testes de aceitação, de capacidade e de implantação) e utilizá-la de forma *self-service*.

7. Por fim, após validada, a equipe de operação aprova a implantação no ambiente de produção.

O *pipeline* fornece visibilidade durante todo o processo mostrando a compilação que está instalada em cada ambiente e por quais estágios ela passou, fornecendo assim *feedback* o mais rápido possível.

2.1.4.5 Infraestrutura como Código

O termo infraestrutura representa os ambientes da sua organização e os serviços que os apoiam, tais como servidores de DNS, *firewalls*, roteadores, repositórios de controle de versão, armazenamento, aplicações de monitoramento, servidores de e-mail e assim por diante. Com o avanço da cultura *DevOps* e o aumento da colaboração e da transparência entre administradores de sistema e desenvolvedores, surgiu a necessidade de gerenciar o conjunto da infraestrutura no mesmo caminho do código: usando controle de versões, realizando testes, empacotando e distribuindo módulos comuns e, obviamente, executando as mudanças de configuração no servidor (SATO, 2014). Ou seja, trazer uma abordagem ágil também à equipe de operação e infraestrutura. A infraestrutura como código surgiu com o objetivo de flexibilizar e agilizar a criação de ambientes em respostas às necessidades ágeis do projeto para a entrega contínua do software.

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 e FARLEY, 2010). 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.

Na entrega contínua de software os ambientes de testes precisam ser os mais próximos possíveis dos ambientes de produção com objetivo de identificar problemas no ambiente o mais cedo possível e testar atividades críticas ao processo como implantação e configuração antes de chegar à produção reduzindo assim o risco da entrega. Com isso, várias coisas precisam ser bem gerenciadas para fornecer confiabilidade à todo o processo de entrega, tais como as configurações e o versionamento dos sistemas operacionais, servidores, banco de dados e as ferramentas - softwares de controles de versão, diretório e monitoramento - e a infraestrutura de redes.

Padrões para usar a infraestrutura como código são vistos em Duvall (2011) para manter todo o processo entrega confiável. Algumas técnicas são estabelecidas:

- *Provisionamento automático*: Automatizar o processo de configuração de seu ambiente incluindo redes, serviços e infraestrutura;
- *Monitoramento dirigido por comportamento*: Execução contínua de testes automáticos para verificar o comportamento da infraestrutura;
- *Sistemas imunes*: Implantação de software enquanto conduz o monitoramento para reverter caso aconteça erros;
- *Ambientes bloqueados*: Bloquear ambientes compartilhados de acessos indevidos praticando todo versionamento através de automação.;
- *Ambientes similares ao de produção*: Ambientes de testes de destinos devem ser o mais próximo possível da produção;
- *Ambientes transitórios*: Os ambientes devem ser capazes de serem criados, gerenciados e terminados facilmente;

A infraestrutura precisa de uma estratégia de gerenciamento desde o início do projeto e que inclua todos os envolvidos das equipes de desenvolvimento e operação. (HUTTERMANN, 2012).

2.1.4.6 Identificação dos Riscos no Processo de Entrega

Alguns problemas comuns podem ocorrer durante o processo de entrega de uma

versão ou funcionalidade de um software. Para isso, é essencial um processo de gerenciamento de riscos eficiente com objetivo de identificá-los, rastreá-los e gerenciá-los de modo a mitigá-los com eficiência minimizando assim seu impacto. Como visto em Humble e Farley (2010) existem alguns problemas comuns no processo de entrega contínua que precisam ser observados, dentre eles, pode-se citar as implantações infrequentes e com erros, a baixa qualidade da aplicação, o processo de integração contínua mal gerenciado e o processo mal definido de gerência de configuração.

Com relação às implantações infrequentes e com erros é necessário sempre verificar sobre a automação do processo de automação, os recursos e o gerenciamento dos hardwares disponíveis, verificar a colaboração entre a equipe de operação com a equipe de desenvolvimento e verificar o comprometimento dos desenvolvedores com relação com relação a integração de mudanças curtas, incrementais e frequentes. A baixa qualidade da aplicação compromete uma qualidade estratégia de testes proporcionando colaboração ineficiente entre testadores e o restante da equipe de entrega e/ou testes implementados de maneira inadequada ou pouco automatizados. Já o processo de integração contínua mal gerenciado deve-se verificar os processos automáticos de testes e o processo de integração contínua. Finalmente a respeito do processo de gerência configuração deve-se verificar os ambientes de produção e homologação do produto, o processo de gerência de mudanças e os testes não funcionais.

2.2 Trabalhos Relacionados

Como visto anteriormente, a área de *DevOps* é muito recente e, inicialmente, em uma pesquisa *ad hoc*, poucos artigos na área acadêmica foram encontrados, o que justificaria uma pesquisa mais aprofundada sobre o assunto com o objetivo de analisar e estruturar a literatura através de um mapeamento sistemático. Assim como as metodologias ágeis, o termo *DevOps* também nasceu na indústria, possuindo assim suas maiores pesquisas nesse campo.

Como artigos relevantes na área, podemos citar o de *Humble* e *Molesky* (HUMBLE e MOLESKY, 2011) onde os autores elaboram o conceito de automação do *pipeline* de implantação (do inglês, "*automated deployment pipeline*") com o conceito de que, para integrar valor para os sistemas de TI e integrar TI ao negócio, seria necessário a criação de equipes multifuncionais que administrem os serviços em todo seu ciclo de vida, não apenas em *silos* isolados. A automatização deveria fazer parte de todo o processo de desenvolvimento, testes e operação. Os autores também

discutem como a adoção de práticas de entrega contínua proporciona benefícios para a governança de TI, inclusive através de uma melhor gestão de riscos.

Já *Eric Shamow* mostra em seu artigo (SHAMOW, 2011) um estudo de caso em uma empresa que estava num colapso de comunicação entre desenvolvimento e operação e que adotou diversas práticas para solucionar o problema que, posteriormente, foram rotuladas como *DevOps*. Defendeu ainda que *DevOps* é um caminho de descoberta, que pessoas e processos não mudam de uma hora para outra, que a adoção de práticas *DevOps* não se dá apenas pelo uso de ferramentas, mas, principalmente pelas mudanças culturais da organização que influenciam positivamente na maneira como as pessoas as usam.

Fitzpatrick e *Dillon* mostram como a automação é um instrumento eficaz na adoção de DevOps (FITZPATRICK e DILLON, 2011), reduzindo o ciclo de desenvolvimento e melhorando a repetibilidade. O autor mostra também que investindo tempo na automação de tarefas nos fornece tempo para nos dedicar a tarefas mais importantes de projetos e necessidades relevantes ao negócio, agregando mais valor à necessidade do cliente.

Já *Phifer* discute como os modelos CMMI para desenvolvimento e ITIL V3 podem ser integrados para permitirem uma melhor colaboração entre desenvolvimento e operações (BILL, 2011). *Next Generation Process Integration: CMMI and ITIL Do Devops*, afirma na pesquisa que o resultado da integração efetiva entre CMMI e ITIL, tal como aplicado ao desafio *DevOps,* é uma estrutura que garante alinhamento com a' necessidades do negócio e que deve ser aplicado a toda organização, não apenas ao desenvolvimento e a operação.

DeGrandis (DEGRANDIS, 2011) fala sobre as ações de liderança que são necessárias para permitir a revolução *DevOps* e descreve, estatisticamente, como o controle de processo pode ser aplicado para aumentar a previsibilidade e, portanto, a satisfação do cliente. A autora ainda descreveu sua experiência contando como práticas *DevOps*, como integração contínua, lançamentos menores e mais frequentes, aplicadas a uma empresa, contribuíram para melhorar a capacidade de organização de T.I. de entregar valor ao negócio do cliente.

O artigo *The Deployment Production Line* (HUMBLE et al., 2006) mostra como automatizar todo o ciclo de testes e implantação através de princípios e práticas que permitem que ambientes sejam criados, configurados e implantados de forma simples (toque em um botão) e com a segurança necessária de voltar a uma versão estável caso algum problema ocorra.

Pesquisas relevantes na área foram feitas conforme podemos destacar em Puppet et al. (2013) e Puppet et al. (2014). A primeira visou obter dados sobre o valor das práticas *DevOps* assim como os benefícios de quem implantou essas práticas. Já a segunda pesquisa teve por objetivo entender qual o impacto de práticas *DevOps* na performance de TI e nas organizações usando métricas de negócio e quais práticas influenciaram esse impacto. A primeira pesquisa procurou buscar os benefícios que as organizações que dizem implementar *DevOps* possuem, além de verificar o tempo que as organizações implantaram *DevOps*, a frequência de implantação, o número de falhas, tempo de recuperação após falha, ferramentas utilizadas, assim como as dificuldades de implementar *DevOps.* A segunda pesquisa também buscou medir a performance da organização fazendo uma relação entre práticas *DevOps* que influenciam a frequência de implantação, o *lead time for changes* e o tempo de recuperação, além de práticas de testes automáticos que tiveram impacto na performance de TI da organização. Essa pesquisa ainda procurou buscar quais fatores da cultura da organização que influenciaram a performance da organização. Uma outra pesquisa relevante (REBELLABS, 2013) buscou identificar uma comparação de performance entre os times tradicionais de operação de TI e as equipes orientadas à *DevOps* em termos de frequência de implantação, tarefas automatizadas, gerenciamento e melhoria de infraestrutura, suporte, comunicação e o uso *self-service* dos recursos de TI.

Gene Kim descreveu as principais 11 coisas que deveríamos saber sobre *DevOps*. O autor descreveu sobre o surgimento de *DevOps*, sobre como *DevOps* se difere de metodologias ágeis, de ITIL e de ISTM, quais os princípios e valores de *DevOps* e as principais áreas de destaque de *DevOps*. Já em *Roche James* o autor pesquisou sobre como as práticas *DevOps* influenciaram a qualidade de software no aspecto de times, cultura e práticas.

Finalmente foi levantado o problema das desconexões de várias áreas de desenvolvimento de software, tais como negócio, desenvolvimento, operação e qualidade ao longo do ciclo de vida do software, principalmente dada a necessidade de confiabilidade e resiliência dos sistemas complexos e intensivos de dados desenvolvidos através de uma série de atividades contínuas que são importantes para o desenvolvimento de software no contexto de hoje. Essas atividades incluem planejamento contínuo, integração contínua, implantação contínua, a entrega contínua, verificação contínua, testes contínuos, conformidade contínua, segurança contínua, uso contínuo, confiança contínua, monitoramento em tempo de execução contínuo,

melhoria contínua (processo e produto), todos sustentados por inovação contínua, defendendo assim que a ênfase nessas áreas de *DevOps* precisa ser contínua.

A **Tabela 1** mostra um resumo dos principais trabalhos relacionados com esta pesquisa:

Tabela 1: Resumo dos principais trabalhos relacionados

Estudo	**Objetivo**	**Escopo do Estudo**
HUMBLE E MOLESKY, 2011	*Automação do pipeline de implantação para integrar TI ao negócio através de automação*	*Indústria*
SHAMOW E., 2011	*Estudo de caso após adoção de práticas DevOps*	*Indústria e Academia*
Fitzpatrick e Dilon, 2011	*Demonstram a importância da automação para se institucionalizar DevOps*	*Academia*
Phifer B., 2011	*Discussão sobre como modelos CMMI e ITIL podem ser integrados para uma melhor colaboração entre equipe de desenvolvimento e operação*	*Academia*
Humble et al. 2006	*Mostra os passos necessários para a automação do ciclo de testes*	*Indústria*
PUPPET et al. 2013 e 2014	*Benefícios ao implantar DevOps, seus valores, comparação de frequência de implantação, numero de falhas, tempo de recuperação, ferramentas*	*Indústria*
REBELLABS, 2013	*Medição da frequência de implantação, performance, impacto de da automação de testes no processo de implantação*	*Indústria*
Este estudo	*Classificar e explorar áreas de DevOps através de um mapeamento sistemático e buscar, através de um survey, práticas do dia-a-dia que as organizações estão utilizando em DevOps.*	*Indústria e Academia*

2.3 Relação das Pesquisas com os Trabalhos Relacionados

Nossa pesquisa efetuou um mapeamento sistemático buscando classificar explorar e descrever sobre *DevOps* de modo a fornecer assim uma base para extração de dados da pesquisa. Neste mapeamento foram encontrados vários artigos na indústria e somente alguns poucos artigos acadêmicos. Na indústria várias pesquisas foram feitas na área de *DevOps* buscando obter informações sobre os benefícios de se implantar a cultura *DevOps* na organização, de medir a performance das organizações após a institucionalização do movimento e até fazendo comparações entre organizações atuais que utilizam *DevOps* e organizações tradicionais que mantém equipe de operação de TI como um silo separado numa organização. Nessas pesquisas foram estudados vários aspectos tais como a frequência de implantação, tempo de execução, monitoramento, verificação de testes automatizados até o aspecto do comportamento e a influência cultural. Durante a pesquisa, também não foi encontrado nenhum mapeamento ou revisão sistemática na área.

Como visto, várias pesquisas em *DevOps* foram feitas, mas notou-se que ainda

não ficou claro o que realmente as organizações estão praticando no aspecto cultural do movimento *DevOps* assim como quais as práticas do dia-a-dia de uma empresa que pratica *DevOps*. Diante disso, essa pesquisa visa também explorar quais práticas realmente estão sendo efetivamente usadas nas organizações *DevOps* para obterem vantagens competitivas no aspecto cultural e técnico.

2.4 Considerações Finais do Capítulo

Este capítulo apresentou os conceitos e fundamentos teóricos usados como base para esta pesquisa. Este referencial envolveu os conceitos sobre a divisão tradicional sobre desenvolvimento e operação, sobre como nasceu o movimento *DevOps*, seus principais conceitos e práticas relacionadas. Este referencial foi fruto da busca de conceitos nos principais estudos publicados em artigos, revistas científicas, livros e dissertações da área dessa pesquisa. No próximo capítulo será exposta a metodologia de pesquisa deste trabalho.

3. METODOLOGIA DE PESQUISA

Este capítulo descreve em detalhes os métodos de pesquisa utilizados para a condução desta pesquisa com objetivo de tornar os resultados mais confiáveis e possíveis de serem auditáveis e reproduzíveis.

A metodologia de pesquisa usada nesta dissertação foi um mapeamento sistemático da literatura devido à natureza das questões de pesquisas serem classificatória e exploratórias o que, segundo (KITCHENHAM et al., 2008), possibilita uma visão mais ampla dos estudos primários fazendo com que necessite da realização de mapeamentos para revelar as evidências das pesquisas. Como o objetivo de nossa pesquisa é explorar, encontrando e classificando o que já foi pesquisado na área, justifica-se a utilização de um mapeamento sistemático em vez de uma revisão sistemática da literatura, o que exigiria questões de pesquisas mais específicas e definidas focadas em um ponto específico de uma área de estudo. No mapeamento sistemático, a extração dos dados é mais abrangente e tem por objetivo classificar e categorizar os resultados. Também foi executado um *survey* com objetivo de complementar respostas às questões de pesquisas que não foram completamente respondidas.

Este capítulo se divide nas seguintes seções:

- Classificação da Pesquisa – apresenta a classificação da pesquisa com o quadro metodológico definido;
- Ciclo da Pesquisa – todas as etapas da pesquisa são detalhadas nesta seção: mapeamento sistemático e pesquisa de campo (*survey*);

3.1 Classificação da Pesquisa

Esta pesquisa é classificada de acordo com o quadro metodológico (**Quadro 1**) apresentado e detalhado a seguir. Como visto em Marconi e Lakatos (2008), o quadro metodológico proporciona mais rigor científico a um trabalho de pesquisa.

Método de Abordagem:	Indutivo

Objetivo da Pesquisa:	Exploratória
Quanto ao Escopo:	- Mapeamento Sistemático - Pesquisa em Campo (survey)
Natureza dos Dados:	Qualitativa
Método de Procedimento:	Análise e Síntese Temática
Variáveis Independentes (X)	- DevOps - Entrega Contínua
Variáveis Dependentes (Y)	- As principais práticas de DevOps e Entrega Contínua que são utilizadas nas organizações

Quadro 1: Quadro metodológico

Esta pesquisa baseia-se no método de abordagem indutivo, pois parte de dados particulares, sistematicamente constatados, o qual se infere uma verdade geral ou universal, não contidas nas partes examinadas. Assim, os argumentos indutivos objetivam conclusões muito mais amplas do que as das premissas nas quais se basearam (MARCONI e LAKATOS, 2008).

O escopo da pesquisa envolveu dois métodos de pesquisa: Um Mapeamento Sistemático da Literatura que, devido à natureza das questões de pesquisas serem classificatória e exploratórias, possibilita uma visão mais ampla dos estudos primários e a extração dos dados que tem por objetivo classificar e categorizar os resultados de forma mais abrangente (KITCHENHAM e CHARTERS, 2007), (KITCHENHAM et al., 2008). Também foi conduzida uma pesquisa de campo, conduzido pelo método *survey*, para identificar o conjunto de práticas *DevOps* e entregas contínuas que organizações *DevOps* utilizam na prática com objetivo de fornecer um guia com melhores práticas.

O método de procedimento constitui uma análise e síntese temática, pois infere seu valor informacional no conteúdo de suas pesquisas, compreendendo a mensagem do autor de acordo com suas ideias principais e secundárias.

A natureza das variáveis é qualitativa, portanto, apesar de que alguns resultados desta pesquisa apresentam dados quantitativos, essa pesquisa se caracteriza como uma abordagem qualitativa.

Segundo Marconi e Lakatos (2008), a abordagem qualitativa é apropriada para obter entendimento mais profundo e detalhado sobre as investigações, ambientes e comportamentos, o que de acordo com *Seaman* força o pesquisador a investigar a complexidade do problema, ao invés de abstraí-lo (MONTEIRO, 2010).

Segundo Marconi e Lakatos (2008), as variáveis de uma pesquisa podem ser consideradas independentes ou dependentes. As independentes são consideradas determinantes que fornecem condição ou causa para um resultado, influenciando, determinando ou afetando as variáveis dependentes. Com esta definição, este trabalho pretende investigar a área de *DevOps* e de Entrega Contínua de software como variáveis independentes (X) e as principais práticas de *DevOps* e Entrega Contínua que são utilizadas nas organizações como variáveis dependentes (Y).

3.2 Ciclo da Pesquisa

Esta seção apresenta as principais etapas que constituíram o ciclo desta pesquisa, com uma breve descrição e associação entre as etapas.

Fases do ciclo da pesquisa:

1. Revisão da literatura (*Adhoc*): Foi realizado uma revisão da literatura tradicional de modo *adhoc* em artigos, revistas científicas, livros e dissertações, com o objetivo de buscar os conceitos teóricos sobre o movimento de *DevOps* e sobre revisões e mapeamento sistemáticos da literatura na área. Identificou-se então uma carência de estudos científicos na área - primários e secundários - além de que, a maioria dos artigos e pesquisas são oriundas da indústria. Dessa forma, identificou-se então a necessidade na área de estudo através de um mapeamento sistemático da literatura com objetivo de identificar áreas em aberto e obter uma visão geral e abrangente dos estudos da área.
2. Planejamento da Pesquisa: Esta etapa iniciou-se com a definição das questões de pesquisa, o planejamento e a elaboração do protocolo do mapeamento sistemático da literatura, além da identificação da necessidade de um *survey* e da elaboração de suas questões.
3. Coleta de dados: etapa em que foi realizada a execução das fases do protocolo do

mapeamento sistemático e os questionários de pesquisa.
4. Análise dos dados: após a coleta dos dados iniciou-se o processo de categorização, análise e interpretação dos dados visando buscar respostas às questões de pesquisa.
5. Consolidação dos Dados: finalizando o processo, nesta etapa foram consolidados os dados, apresentadas as contribuições, limitações e trabalhos futuros, concluindo com a elaboração do relatório final da pesquisa.

3.2.1. Procedimento da coleta de dados

Na pesquisa foram utilizadas várias fontes de informações e coleta de dados no intuito de tornar os resultados mais confiáveis, auditáveis e reprodutíveis. Esta seção explica os procedimentos utilizados na coleta de dados e está dividida da seguinte forma:

- Mapeamento Sistemático – apresenta uma breve descrição sobre o mapeamento sistemático;
- Aplicação de questionários (*survey*) – apresenta os procedimentos adotados para a elaboração do questionário e os resultados obtidos de modo geral;

3.2.1.1 Mapeamento Sistemático

Segundo Kitchenham e Charters (2007), existem dois tipos de revisão da literatura: mapeamento sistemático e revisão sistemática da literatura.

O mapeamento sistemático deve ser realizado em estágio inicial de investigação de um determinado tópico, antes que seja realizada uma revisão sistemática ou quando o tema é muito amplo com objetivo de identificar possíveis áreas em que são necessárias realizações de estudos primários ou mesmo a exploração de uma área ainda desconhecida. Um mapeamento sistemático provê um objetivo e um procedimento sistemático para identificação dos dados dos estudos empíricos para responder a uma questão de pesquisa. (KITCHENHAM et al., 2008). De acordo com Kitchenham (2007), a diferença entre uma Revisão Sistemática e um Mapeamento Sistemático é que a revisão sistemática foca em questões de pesquisa mais específicas enquanto que mapeamento foca em questões mais amplas, com

o propósito de dar uma visão geral de uma área de pesquisa.

Esta pesquisa optou por efetuar um mapeamento sistemático visto o caráter exploratório da área ainda pouco explorada, tendo como objetivo identificar estudos já executados na área assim como uma visão ampla do objeto do estudo.

Uma visão resumida do protocolo do mapeamento sistemático se encontra no APÊNDICE A.

3.2.1.1.1 Protocolo da Pesquisa

O protocolo elaborado para descrever todos os procedimentos e métodos de como o mapeamento foi conduzido se encontra a seguir.

Definição da Questão de Pesquisa

Para delinear o escopo da pesquisa e identificar os elementos que fizeram parte do seu escopo, foi utilizado a estrutura PICOC (*Population, Intervention, Comparison, Outcome, Context*) como recomendada por Kintchenhan (KITCHENHAM e CHARTERS, 2007) e Petticrew (PETTICREW et al., 2005).

População: Pesquisas na área de *DevOps*.
Palavras Relacionadas: *DevOps, Dev and Ops, , Agile System Administration, IT Operations*
Intervenção: Identificar áreas de pesquisas científicas sobre práticas *DevOps*.
Palavras Relacionadas: *Continuous Delivery ,Continuous Integrations, Continuous Deployment, Continuous Release, Release Management, Deployment Automation, Infrastructure Management, Infrastructure as a code, automation, ,Virtualization, Cloud, Quality of Software Deployments, Frequent Software Releases, Agile Development, software development cycle, life cycle, Process, Maturity Model, Productivity, Agility, Reliability ,Report.*

O item comparação não foi utilizado pois a pesquisa não tem por objetivo estudos primários comparativos. Os outros itens (resultados e contextos) não foram utilizados pois são atrelados ao item comparação.

Foram definidas 5 questões do tipo classificatória e exploratórias a serem respondidas com o mapeamento sistemático, são elas:

Q.1 Em quais áreas estão concentrados estudos/pesquisas na área de *DevOps*?

Q.2 Onde se concentram as maiores publicações na área de *DevOps* (periódicos e conferências)?

Q.3 Quem são os principais autores da área?

Q.4 Com quais metodologias ágeis o uso de práticas *DevOps* está(ão) relacionada(s)?

Q.5 Quais são as práticas e técnicas em uso nas organizações que utilizam *DevOps*?

Definição da Estratégia de Busca

A construção da *string* de busca nas bibliotecas digitais e engenhos de busca iniciou-se a partir da extração de palavras-chave das questões de pesquisa e dos elementos das estruturas PICOC, no nosso caso, população e intervenção, identificando sinônimos para as palavras chave, usando assim o conector booleano OR para indicar as palavras alternativas e AND para a ligação das palavras chaves.

A realização de um piloto foi de fundamental importância para afinar e definir uma *string* de busca. Foi importante a realização de um piloto para efeito comparativo de resultados que deveriam aparecer de autores e pesquisas conhecidas para avaliar a string de busca, bem como para retirar strings desnecessárias que não afetam no resultado final.

Com base nestas informações descritas anteriormente, ficou definida a string de busca para procurar informações gerais do termo da pesquisa (*DevOps*) ligadas a informações sobre as áreas a serem pesquisadas, suas práticas e seus sinônimos. Assim, a *string* de busca ficou definida de acordo com a seguinte estrutura:

<tema da pesquisa> AND (<areas> <praticas>)

Os termos sinônimos para cada palavra chave da *string* são os que seguem:

Tema da pesquisa: *DevOps, Dev and Ops, Agile System Administration, IT Operations and Devs.*

Áreas/Práticas: *Continuous Delivery, Continuous Integrations, Continuous Deployment, Continuous Release, Baby Steps, Release Management, Deployment, Frequent Software Releases, Quality Assurance, Infrastructure Management, Infrastructure as a Code, Automation, Virtualization, Cloud, Agile Development, Software Development Cycle, Lifecycle.*

Assim a *string* ficou definida da seguinte forma:

("DevOps" OR "Dev and Ops" OR "Dev Ops" OR "Agile System Administration" OR "Dev and IT Operations") ***AND*** *("Practices" OR "Continuous Delivery" OR "Continuous Integration" OR "Continuous Deployment" OR "Continuous Release" OR "Baby Steps" OR "Release Management" OR "Deployment" OR "Frequent Software Releases" OR "Quality Assurance" OR "Infrastructure Management" OR "Infrastructure as a Code" OR "Automation" OR "Virtualization" OR "Cloud" OR "PaaS" OR "Agile Development" OR "Software Development Cycle" OR "Lifecycle" OR "Process" OR "Maturity Model" OR "Productivity" OR "Agility" OR "Metrics" OR "Report" OR "Skills" OR "Tools")*

Data Sources

O processo utilizado para buscar estudos primários na literatura incluiu buscas automatizadas através de bibliotecas digitais e engenhos de busca, aplicando a *string* de busca definida anteriormente. Os engenhos de buscas diferenciam-se das bibliotecas digitais porque esses possuem artigos científicos publicados de múltiplas fontes e aqueles somente de suas editoras.

Bibliotecas Digitais

- ACM Digital Library (http://portal.acm.org)
- IEEEXplore Digital Library (httt://ieeexplore.ieee.org/)
- Elsevier ScienceDirect (http:// www.sciencedirect.com)

Engenhos de Buscas

- Scopus (http://www.scopus.com/)
- Springer Link (http://www.springerlink.com/)
- Engineering Village (http://www.engineeringvillage2.org)

Também foram feitas buscas manuais em periódicos e conferências sobre engenharia de software com o objetivo de ampliar os resultados e diminuir possíveis viéses na construção das *strings* de buscas automáticas, problemas de indexação de artigos e de não inclusão de palavras chaves nos artigos. Livros e artigos, não incluídos nas buscas automáticas, também foram adicionados no mapeamento a partir do momento do reconhecimento de sua importância para a pesquisa científica e os autores sejam de renome na área.

Conferências

- Agile Development Conference;
- International Conference on Agile Software Development;

Journals

- ACM Transactions on Software Engineering and Methodology;
- IEEE Software;
- IEEE Internet Computing

- IEEE Transactions on Software Engineering;
- Journal of the ACM;
- International Journal of Agile and Extreme Software Development – IJAESD;
- Cutter IT Journal;

Definição dos Critérios de Seleção e Inclusão/Exclusão

Todos os estudos foram avaliados com base nos critérios de inclusão e exclusão. Inicialmente, a seleção dos estudos primários foi feita em 3 fases, são elas:

- Seleção dos estudos primários relevantes resultantes das buscas automáticas de acordo com o critério de inclusão;
- Avaliação dos estudos primários aplicando-se os critérios de inclusão/exclusão;
- Etapas para resolução dos conflitos e seleção final;

Na primeira rodada da seleção foram selecionados artigos com base no **Título** e no ***Abstract***. Para efeito de seleção na primeira rodada, foram selecionados estudos que tinham alguma relação com a área de pesquisa, seja no título ou *abstract* da pesquisa. Dessa forma, os pesquisadores envolvidos realizaram a seleção separadamente e, os conflitos surgidos, foram discutidos em reuniões posteriores. Na segunda rodada, os pesquisadores fizeram uma releitura independente dos estudos com base na **Introdução** e na **Conclusão** dos artigos e aplicaram os critérios assim os critérios de **Inclusão** e **Exclusão** (ver **Tabela 2**) de forma independente de cada pesquisador. Resumidamente, os critérios de inclusão foram aplicados como:

- O Artigo respondia a alguma questão de pesquisa?
- O Artigo pertencia a fontes aceitáveis?
- Estudos que focaram em práticas conhecidas de *DevOps;*
- Estudos que abrangiam alguma área de *DevOps;*
- Relatórios de uso em empresas e artigos da indústria também foram incluídos;

Após a seleção inicial aplicando os critérios de inclusão, foram aplicados os critérios de

exclusão, definidos como:

- Se o "artigo" era uma apresentação ou um resumo estendido;
- Se o artigo não tinha relação com *DevOps*;
- Se o artigo não tinha relação com as áreas e/ou práticas do *DevOps*;

Na etapa 3, caso detectado algum conflito entre os pesquisadores, foram definidos critérios para resolver esses conflitos e descrever a seleção final.

Avaliação da Qualidade

Após a aplicação dos critérios de seleção, inclusão e exclusão foram aplicados critérios de qualidade dos estudos com objetivo de selecionar os trabalhos mais relevantes de acordo com os critérios de qualidade definidos no protocolo. A classificação da qualidade dos estudos foi atribuída entre 0% e 100% de acordo com o somatório das notas atribuídas com os critérios de qualidade.

Extração dos Dados e Mapeamento dos Estudos

A extração dos dados foi realizada pelo autor da pesquisa e revisada pelo orientador e co-orientador desta pesquisa. Os estudos que passaram na fase de qualidade também foram analisados pelo autor desta pesquisa, realizando assim a extração estruturada dos dados da publicação, do contexto e das evidências para responder às questões de pesquisa com base no processo sugerido por Cruzes e Dybå (2011).

Basicamente o processo resumido consistiu em fazer uma leitura inicial dos estudos e, posteriormente, a extração dos dados e das evidências. Este estudo conduziu uma síntese e análise temática dos dados, conforme processo recomendado por Cruzes e Dybå (2011).

3.2.1.2 Aplicação de Questionários (Surveys)

Uma pesquisa de questionário (*survey*) é utilizada para identificar características

de uma ampla população de indivíduos com objetivo de extrair informações de uma população, geralmente baseada em questões (EASTERBROOK et al., 2008). A precondição para conduzir uma pesquisa baseada em questionário é possuir uma clara questão de pesquisa e uma população bem definida e representativa.

Diante do exposto, esse trabalho utilizou o método *survey* para coleta de dados através de uma aplicação de questionários online com objetivo de identificar quais são as práticas de *DevOps* e de Entrega Contínua de Software que as organizações realmente estão utilizando e que estão agregando valor ao seu negócio. As perguntas foram extraídas de pesquisas na literatura e a partir de guias da indústria de software sobre as principais práticas conhecidas.

A coleta de dados ocorreu através de um formulário online para organizações previamente acordadas e que se diziam utilizar práticas *DevOps* para o desenvolvimento de software. A mesma pesquisa também foi aplicada de forma presencial para as organizações presentes no evento *Agile Brazil 2014*, realizado de 05 a 07 de novembro em Santa Catarina - Florianópolis. O formulário da pesquisa encontra-se disponível APÊNDICE B.

3.3 Resumo

Para alcançar o objetivo desta pesquisa foi adotado o método indutivo de natureza qualitativa, fazendo assim o uso de métodos mistos.

A pesquisa utilizou dois métodos para coleta de dados: (1) mapeamento sistemático da literatura e (2) aplicação de questionários.

O mapeamento sistemático, por ser uma área de pesquisa ainda pouco explorada, teve por objetivo executar um procedimento sistemático para identificação dos dados dos estudos empíricos para responder quais áreas de estudos e onde estão as concentradas as maiores publicações, quem são os principais autores da área e quais são as principais práticas e técnicas utilizadas no contexto de *DevOps*.

A pesquisa de questionário (*survey*) foi utilizada para identificar características de ampla população de indivíduos com objetivo de selecionar exemplos representativos de

uma população geralmente baseada em questões (EASTERBROOK et al., 2008). Sendo assim, este trabalho utilizou o método *survey* para coleta de dados através de uma aplicação de questionários online com objetivo de identificar quais são as práticas de *DevOps* e de Entrega Contínua de Software que as organizações realmente estão utilizando e que estão agregando valor ao seu negócio.

4. RESULTADOS

Este capítulo apresenta os resultados obtidos a partir da execução do mapeamento sistemático da literatura e da aplicação dos questionários, realizados com o objetivo de responder às questões de pesquisa deste trabalho. Este capítulo está divido nas seguintes seções:

- **Síntese dos resultados do mapeamento sistemátic**o – são apresentados as estratégias e procedimentos para a obtenção dos resultados do mapeamento sistemático.
- **Respostas às questões de pesquisas** – são apresentados os resultados das questões de pesquisas e as evidências encontradas no mapeamento que correspondem a essas questões.
- **Síntese dos resultados da aplicação dos questionários** – são apresentados os resultados da aplicação dos questionários.

4.1 Síntese dos Resultados do Mapeamento Sistemático

Esta seção apresenta as estratégias e procedimentos para obtenção dos resultados deste mapeamento que permitem responder às questões de pesquisa deste trabalho. Os resultados do mapeamento serão apresentados a seguir.

4.1.1. Resultados da Busca e Seleção dos Estudos

Nesta subseção, são apresentadas as estratégias e procedimentos para a obtenção dos resultados desta revisão.

4.1.1.1. Resultados da Busca e Seleção

Para a condução desta pesquisa foram envolvidos três pesquisadores, um estudante de pós-graduação (mestrando) em Ciência da Computação, pelo Centro de Informática (CIn) da Universidade Federal de Pernambuco (UFPE), pesquisador autor, e dois professores da mesma universidade, sendo um orientador e outro co-orientador da pesquisa. Entre os pesquisadores citados, apenas o pesquisador autor não tinha

experiência prévia em revisões ou mapeamento sistemático. Durante todo o processo, os professores envolvidos orientaram e revisaram os resultados obtidos durante todo o processo. Conforme orientado por Dybå e Dingsøyr (2008), quase que todas as etapas da seleção foram feitas por mais de um pesquisador a fim de reduzir um possível viés envolvendo a seleção dos estudos, com exceção das fases de busca automática e manual, extração e sínteses dos resultados, que foram desenvolvidas pelo autor desta pesquisa.

4.1.1.2. Execução

O mapeamento sistemático foi executado de acordo com o protocolo descrito no capítulo 3. O mapeamento completo está disponível no APÊNDICE A. Os resultados de cada etapa da condução são descritos nas próximas subseções.

4.1.1.2.1. Busca Automática e Manual

No início desta etapa foram realizadas buscas *ad-hoc* nos principais engenhos de buscas e, posteriormente, depois de definido a *string* de busca, foram executados pilotos para que a mesma *string* pudesse ser executada em todas as fontes e que pudesse ser usada por todos os pesquisadores, podendo assim o estudo ser replicado.

O processo utilizado para buscar estudos primários na literatura incluiu buscas automatizadas através de bibliotecas digitais e engenhos de busca, aplicando a *string* de busca definida anteriormente. Diferenciamos os engenhos de buscas das bibliotecas digitais porque esses possuem artigos científicos publicados de múltiplas fontes e aqueles somente de artigos de suas editoras. Também foram feitas buscas manuais em periódicos e conferências em engenharia de software com o objetivo de ampliar os resultados e diminuir possíveis viés na construção das *strings* de buscas automáticas, problemas de indexação de artigos e de não inclusão de palavras chaves nos artigos. Livros e artigos, não incluídos nas buscas automáticas, também foram adicionados no mapeamento, quando tiveram importância para a pesquisa científica e os autores identificados de renome na área.

As buscas foram realizadas entre os meses de abril e maio de 2014 sendo selecionados 85 estudos entre 3 engenhos de buscas e 3 bibliotecas digitais. A

distribuição quantitativa dos estudos foi definido da seguinte forma: *Scopus*(39), *Elsevier ScienceDirect* (6), *ACM Digital Library* (19), *IEEEXplore Digital Library* (9), *Springer Link* (25) e *Engineering Village* (13). Essa distribuição pode ser vista de forma percentual na **Figura 5**.

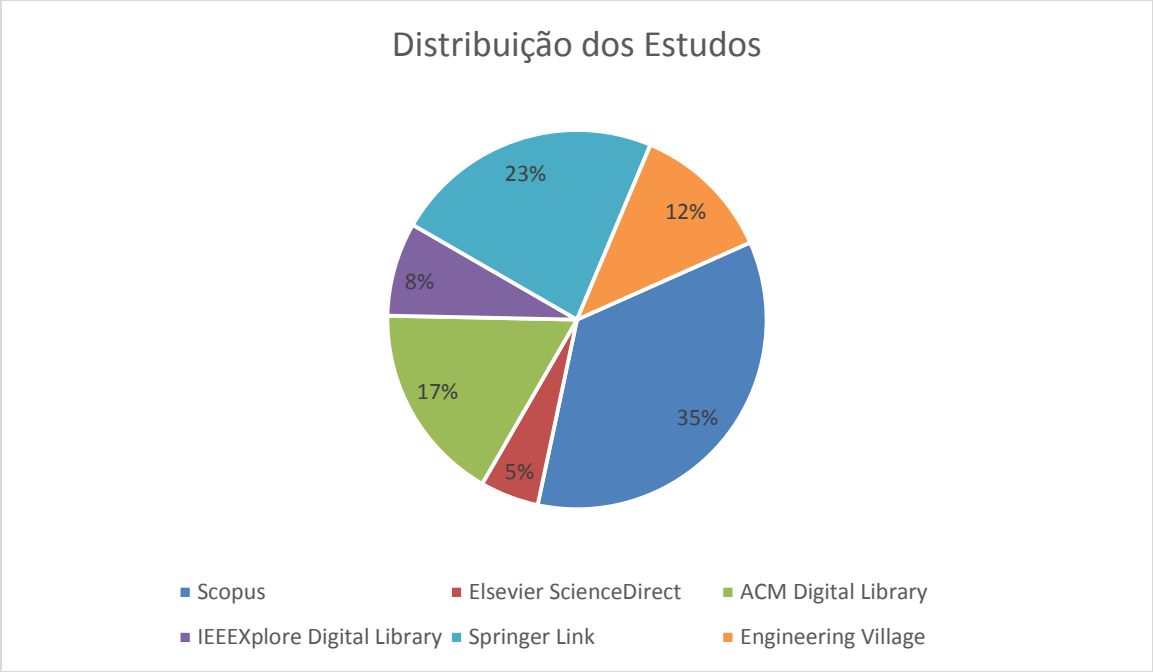

Figura 5: Distribuição da participação dos engenhos na busca automática

Todos os estudos encontrados foram identificados através de códigos únicos e foram inseridos em planilhas para controle dos pesquisadores.

4.1.1.2.2. Seleção dos Estudos

Na primeira rodada da seleção foram selecionados artigos com base no **Título** e no ***Abstract***. Para efeito de seleção na primeira rodada, foram selecionados estudos que tinham alguma relação com a área de pesquisa, seja no título ou *abstract* da pesquisa. Dessa forma, os pesquisadores envolvidos realizaram a seleção separadamente e, os conflitos surgidos, foram discutidos em reuniões posteriores. Assim, foram selecionados 85 estudos que tinham alguma relação com a área pesquisada.

Na segunda rodada, os pesquisadores fizeram uma releitura independente dos estudos com base na **Introdução** e na **Conclusão** dos artigos e aplicaram os critérios assim os critérios de **Inclusão** e **Exclusão** (ver **Tabela 2**) de forma independente de cada pesquisador. Novamente os conflitos foram resolvidos em reuniões posteriores.

Após a aplicação da *string* de busca, foram selecionados 85 estudos para a primeira seleção. Após a execução da primeira seleção, foram excluídos 47 estudos com base no título e no *abstract* antes da segunda seleção. Após a aplicação dos critérios para a segunda seleção, agora com base na introdução e conclusão, foram considerados 40 estudos relevantes para a fase de avaliação de qualidade dos estudos, sendo assim excluídos 7 estudos após a execução da segunda rodada de seleção. A quantidade de estudos selecionados foi extensa visto que ainda é uma área nova, principalmente na academia. Acredita-se que a quantidade de estudos selecionados para a aplicabilidade dos critérios de qualidade foi extensa devido às perguntas das questões de pesquisas terem por objetivo um escopo mais abrangente do mapeamento sistemático. Os critérios aplicados na segunda seleção dos artigos podem ser vistos de forma resumida na **Tabela 2**.

Tabela 2: Critérios de Inclusão e Exclusão da segunda seleção

ID do Pesquisador	Identificação do Pesquisador
ID do artigo	Identificação Sequencial do artigo
TAG	TAG destinada a descrever o contexto
Conclusão	Incluído/Excluído
Decisão baseado em	Título/Abstract/Keywords/Introdução/Conclusão/Metodologia/Resultados
Exclusão A	É um estudo abstrato ou um breve artigo ?
Exclusão B	Não é relacionado à DevOps?
Exclusão C	Não é relacionado a nenhuma prática ou área DevOps?

Exclusão D	O estudo não possui uma questão de pesquisa definida.
Inclusão A	O estudo é de uma fonte aceitável?
Inclusão B	O estudo é focado em alguma prática ou área DevOps?
Inclusão C	Se os critérios de inclusão A e B não forem satisfeitos, verificar se é um relatório da indústria ou algum relato de experiência.

4.1.1.2.3. Avaliação da Qualidade e Extração dos Dados

Após aplicado os critérios de inclusão e exclusão na segunda seleção, os estudos foram submetidos aos critérios de qualidade de acordo com o quadro da **Tabela 3**.

Tabela 3: Critérios de Qualidade do mapeamento sistemático

ID do Pesquisador	Identificação do Pesquisador
ID do artigo	Identificação Sequencial do artigo
Conflito	Sim/Não (aceito por um pesquisador e excluído por outro)
Resultados Claros	Possui resultados objetivos baseado em evidências e argumentos? (Sim=1/Não=0)
Estudo Empírico	É um estudo Empírico? (Sim=1/Não=0)
Representação	(Somente para estudos Empíricos). Representação Randômica=1;Não Randômica=0.5;Não Representativo=0
Replicado	(Somente para estudos Empíricos). O estudo pode ser replicado?
Participantes	(Somente para estudos Empíricos). Número de Participantes (N/A, 0, 0.5 & 1) ver protocolo
Questionário	O estudo é baseado em questionários? (Sim=1/Não=0)
Respostas	(Somente para estudos Empíricos). Média de respostas dos questionários (

	sem respostas=0;80%=1;abaixo de 20%=0;Entre 20/80%=0.5
Teórico	O estudo é teórico?
Referências	(Para estudos teóricos apenas). Possui referencias apropriadas? (Sim=1/Não=0)
Resultados	Resultado final entre 0 e 1

Um resumo das distribuições da qualidade dos estudos pode ser visto na **Figura 6**. No APÊNDICE C observa-se que, depois da segunda seleção, a maioria dos estudos apresentou mais de 50% de qualidade de acordo os critérios de qualidade aplicados.

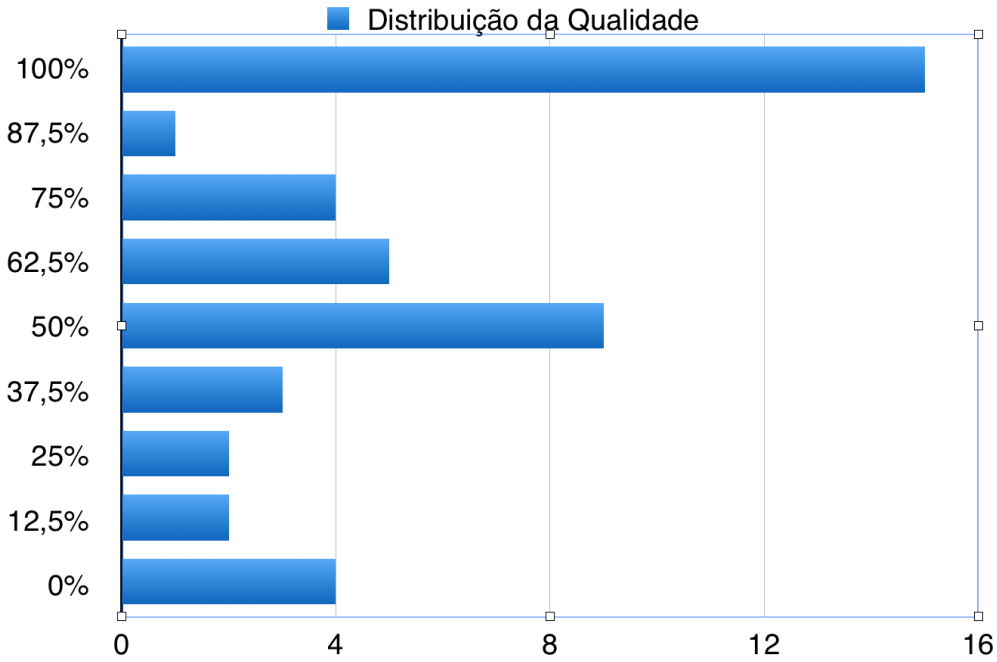

Figura 6: Distribuição dos estudos após a avaliação da qualidade

A extração dos dados foi realizada de forma estruturada, a partir dos 40 estudos resultantes da fase de avaliação de qualidade, utilizando dados de publicação, contexto e as evidências. A fase de Extração dos dados foi executada completamente pelo autor desta pesquisa, porém, foi auxiliada e revisada pelo co-orientador da pesquisa e pelo orientador. Essa limitação na quantidade de participantes nessa fase é prevista por

Kitchenham (2007).

Nesta etapa, foi feita uma relação entre os estudos e as questões de pesquisa relativas a cada estudo. O detalhamento desta etapa pode ser visto no APÊNDICE A.

4.2 Respostas às Questões de Pesquisas

Esta seção mostra o mapeamento das evidências encontradas nos estudos primários assim como as respostas aos questionamentos levantados nas questões pesquisa propostas.

4.2.1 (Q.1.) Em quais áreas estão concentrados estudos/pesquisas na área de *DevOps*?

Esta questão visa identificar em quais áreas estão concentrados os principais estudos na área de *DevOps*. Esta é uma questão abrangente que busca obter uma visão geral das principais áreas onde as pesquisas já foram concentradas. A relação pode ser vista na **Tabela 4**.

Tabela 4: Relação das pesquisas com as áreas relacionadas

Estudos	Área de Pesquisa
ID001, ID072	Orientados à comportamento
ID002,ID008, ID010, ID048,ID049,ID057, ID060	Integração, Entrega e Implantação contínua
ID008,ID056	Desenvolvimento Ágil
ID003	Requisitos
ID005	Gerência de Configuração
ID006,ID030,ID042,ID036	Ferramentas
ID007,ID018,ID021,ID030,ID038,ID042,ID056,ID061,ID071	Computação em nuvem
ID020	Dívida técnica
ID026,ID063, ID067	Qualidade e Testes
ID041,ID050	Modelos e Metodologias
ID013, ID043	Cultura DevOps
ID045,ID055, ID076	Métricas DevOps

ID034,ID062,ID071, ID074	Infraestrutura como código
ID020,ID063,ID072	Risco
ID008, ID046,ID054,ID058,ID059	Práticas DevOps
ID017	Logs

Pode-se constatar que a maioria dos estudos encontrados se concentra em integração, desenvolvimento, entrega e implantação contínua, em virtualização e nuvem, automação de infraestrutura, ferramentas e princípios *DevOps*.

Os resultados apresentados demonstram que existe uma área abrangente de pesquisa em torno de *DevOps*, o que se abstrai uma falta de padronização de definição das áreas de pesquisa. Ou seja, como resultado a maioria dos estudos não possui uma área definida de *DevOps,* portanto demonstra que é uma área ainda pouco explorada com muita abrangência em suas pesquisas.

4.2.2 (Q.2.) Onde se concentra as maiores publicações na área de DevOps (periódicos, conferências e bibliotecas digitais)?

Esta questão teve por objetivo buscar as fontes principais de pesquisa na área de *DevOps* afim de identificar alguma relação entre uma fonte específica com área de pesquisa de forma a facilitar possíveis buscas futuras para pesquisa.

Conforme pode ser visto na **Figura 5**, mais de 80% das publicações são encontradas nas bibliotecas digitais *Scopus, Springer Link, ACM Digital Library* e *Engineering Village*. Com relação a periódicos, foi identificado um periódico do *Cutter IT Journal* que mantém publicações periódicas relacionadas à área de pesquisa, conforme pôde ser visto em 18 estudos feitos na área de *DevOps* relacionados ao periódico.

Outros resultados relevantes onde os estudos foram mais encontrados foram *IEEE Computer Society, IEEE Eighth World Congress on Services, Advances in Recent Technologies in Communication and Computing, International Conference on Software Engineering, IEEE/ACM International Symposium on Cluster, Cloud And Grid Computing, Engineering and Computer Science, International Conference on Computing in High Energy and Nuclear Physics, International Journal of Computational*

Science and Engineering, Journal of Internet Services and Applications e *Architecture and Patters for IT Service Management.*

4.2.3 (Q.3.) Quem são os principais autores da área?

Esta questão do mapeamento, busca identificar os principais autores da área afim de obter informações sobre a origem de *DevOps* e onde se encontra as principais informações do movimento. Esta questão teve sua importância devido a identificação dos principais autores facilitar pesquisas *ad-hocs* na área em busca de estudos relevantes. Os critérios para identificação dos principais autores foram voltados à quantidade de publicações na área, seja artigos científicos ou livros, e até mesmo sua importância no movimento *DevOps*.

Tabela 5: Relação dos principais autores da área

Autor	Autor
Andreas Schaefer	Patrick Debois
Marc Reichenbach	Kevin Behr
Dietmar Fey	George Spafford
Jez Humble	John Wills
Gene Kim	Mike Orzen
Gary Gruver	Tommy Mouser
Joanne Molesky	Eric Shamow
Lawrence Fitzpatrick	Michael Dillon
Dominica DeGrandis	Marc Dupuis
Sowmya Karunakara	Matthew Sacks
Michael Huttermann	David Farley

Dentre os principais autores, pode-se destacar *Patrick Debois*, o qual iniciou o movimento *DevOps* através do *agile system administration*, que tinha por objetivo diminuir o gap entre o desenvolvimento e operação usando técnicas de desenvolvimento ágil no mundo de operação. Pode-se destacar também *Gene Kim, Kevin Behr e George Spafford* autores do livro The Phoenix Project que trata a respeito de como *DevOps* pode ajudar a organização a alinhar o negócio com desenvolvimento

e operação. Também pode-se destacar *Jez Humble* e *David Farley* com sua contribuição do livro sobre entrega contínua e suas contribuições, junto com *Patrick Debois* e outros sobre o livro em andamento de *DevOps Cookbook* com objetivo de catalogar práticas *DevOps* que as organizações estão tendo mais sucesso.

4.2.4 (Q.4.) Com quais metodologias ágeis o uso de práticas DevOps estão relacionadas?

Esta questão do mapeamento teve por objetivo identificar alguma relação das práticas *DevOps* com metodologias ágeis através de estudos na área. Sabendo que o *DevOps*, em seu fundamento, usa práticas ágeis de desenvolvimento na infraestrutura buscando assim o alinhamento entre as áreas, a questão buscou identificar quais metodologias já tinham algum estudo na área.

As metodologias ágeis que tiveram relações com estudos do mapeamento sistemático foram *Scrum* e *Lean*. Os estudos identificados como relacionados a essas áreas foram o ID001 (Towards Behavior Driven Operations (BDOps)), ID008 (Climbing the *"Stairway to Heaven"*), ID013 (*A Grounded Theory Analysis of Modern Web Applications - Knowledge, Skills, and Abilities for DevOps*), e ID020 (*From Assessment to Reduction: How Cutter Consortium Helps Rein in Millions of Dollars in Technical Debt*), respectivamente.

As evidências retiradas dos estudos relativo à essa questão de pesquisa são apresentadas a seguir:

- [Climbing the *"Stairway to Heaven"*]: *"towards agile development requires a careful introduction of agile practices into the organization, a shift to small development teams and a focus on features rather than components"; "towards continuous integration requires an automated test suite, a main branch to which code is continually delivered and modularized development"; "towards continuous deployment requires organizational units such as product management to be fully involved and a pro-active lead customer to work closely with when exploring the concept further"; "Agile software development is well-known for its focus on close customer collaboration and customer feedback. In emphasizing flexibility,*

efficiency and speed, agile practices have lead to a paradigm shift in how software is developed. However, while agile practices have succeeded in involving the customer in the development cycle, there is an urgent need to learn from customer usage of software also after delivering and deployment of the software product."

- [A Grounded Theory Analysis of Modern Web Applications - Knowledge, Skills, and Abilities for DevOps]: *"DevOps, the combination of development and operation, has been adopted within organizations in industry, such as Netflix, Flickr, and Fotopedia. Configuration management tools have been used to support DevOps. However, in this paper we investigate which Knowledge, Skills, and Abilities (KSA) have been employed in developing and deploying modern web applications and how these KSAs support DevOps"*
- [Towards Behavior Driven Operations (BDOps)]: *"BDOps is a key to achieving many related areas like DevOps, Continuous delivery, automating deployment and provisioning that cultivates the idea of treating infrastructure as code. It helps operations adopt agile techniques, which is a step Towards making the business more agile"*
- [From Assessment to Reduction: How Cutter Consortium Helps Rein in Millions of Dollars in Technical Debt]: *"To become actionable, a technical debt assessment plan must be followed by a technical debt reduction project. The project implemented in the Cutter client engagement reported herein builds on the following elements: SWAT team, Evangelism, Agile methods, Technical debt items in the backlog of every team that converts to Agile"*

4.2.5 (Q.5.) Quais são as práticas e técnicas em uso nas organizações que utilizam *DevOps*?

Esta questão do mapeamento tentou buscar informações na literatura sobre as práticas e técnicas mais usadas de *DevOps* nas organizações. Por se tratar um movimento relativamente novo, nota-se que ainda não se tem práticas ou áreas definidas onde se aplica o movimento de *DevOps*, mas sim algumas tentativas para definir práticas que são usadas em *DevOps*, geralmente denominadas *Continuous**,

relacionada a algumas práticas ou áreas onde se aplicam nas organizações uma estrutura automatizada de integração, entrega, desenvolvimento, teste e implantação contínuo encurtando ao máximo à distância entre equipe de desenvolvimento e equipe de operação e proporcionando que o código desenvolvido seja entregue o mais rapidamente possível na produção.

A **Tabela 6** mostrou uma relação das principais práticas identificadas com seus respectivos estudos. As principais práticas ou áreas identificadas no mapeamento como resposta à questão de pesquisa são mostradas na **Tabela 6**, a seguir:

Tabela 6: Principais práticas de *DevOps* identificadas no Mapeamento Sistemático

ID001,ID002,ID006,ID008,ID021,ID034,ID042,ID048,ID049,ID051,ID057,ID060,ID062,ID071,ID072	Integração Contínua Entrega Contínua Testes Contínuos e Automatizados Implantação Contínua Liberação Contínua Infraestrutura como Código

As evidências retiradas dos estudos relativo à essa questão de pesquisa são apresentadas a seguir:

- [Behavior Driven Operations - BDOps, Behavior Driven Monitoring – BDM]: *"BDOps is a key to achieving many related areas like DevOps, Continuous delivery, automating deployment and provisioning that cultivates the idea of treating infrastructure as code. It helps operations adopt agile techniques, which is a step Towards making the business more agile"*
- [Development and Deployment at Facebook]: *"the practices that Internet companies use are known as continuous deployment. This reflects the habit of deploying new code as a series of small changes as soon as it's ready"*
- [Fast Development Platforms and Methods for Cloud Applications]: *"The framework, which is underpinned by the concept of lifecycle management for application, fills in the gap between the development and production phases of cloud application development. The framework identifies roles in development and phases in the development process for cloud applications."*
- [Climbing the "Stairway to Heaven"]: *"we presented the transition process when*

moving towards continuous deployment of software"

- [DevOps Patterns to Scale Web Applications using Cloud Services]: *"something that can be transformed in a Pattern Language of DevOps practices to scale web applications using cloud services."*
- [Continuous integration and automation for DevOps]: "*an insight in how automation can to improve scalability and testability while simultaneously reducing the operators' work*"
- [System management and operation for cloud computing systems]: "*way to deploy applications that is best suited to the application's characteristics and SLAs, and their operations during dynamic configuration changes, as operations management technologies for a cloud environment*"
- [Why enterprises must adopt devops to enable continuous delivery]: "*Continuous delivery enables businesses to reduce cycle time so as to get faster feedback from users, reduce the risk and cost of deployments, get better visibility into the delivery process itself, and manage the risks of soft- ware delivery more effectively*"
- [Devops: So you say you want a revolution?]: "*Continuous integration (although a major cost) was implemented, allowing developers to immediately see the impact of their code changes and fix problems on the spot in the development environment*"
- [The business case for devops: A five-year retrospective]: "*this devops group has collected a number of concrete measurements of its incremental value"; "justify adopting devops practices and that, in short order, devops will become the norm within the industry*"
- [Automatic Releasing]: "*automatic releasing is a building block of DevOps and that it can be a powerful strategy to decouple deployment and release*"
- [Infrastructure as Code]: "*With configuration management and the infrastructure as code paradigm in general, the col- laboration of development and operations is no longer merely a possibility, but almost a necessary*"
- [Automating Infrastructure and Application Provisioning]: *"web infrastructure should be built with automation in mind, by leveraging virtualization and*

configuration management systems”

- [Specification by Example]: “*fundamental principles of DevOps is that any build that successfully passes the gauntlet of automated checks can potentially be delivered into production”; “intend to automatically deliver a product into production, every team member must also be convinced of the quality of these tests, in the sense of whether the tests truly capture what the customers desired in the product and whether the tests have any informative expressiveness to help developers develop the solution*”

O mapeamento sistemático não foi suficiente para o aprofundamento das principais práticas e técnicas usadas no dia a dia das organizações, ele apenas identificou algumas dessas práticas o que acarretou da necessidade de executar um *survey* com objetivo de identificá-las nas organizações.

4.3 Síntese dos resultados da aplicação dos questionários

Esta seção apresenta os resultados e a análise da aplicação dos questionários aplicados em congressos da área e pela Internet. Os resultados são apresentados da seguinte forma:

- **Análise e discussão dos resultados -** apresenta uma análise dos principais resultados obtidos na pesquisa e cataloga as principais formas de uso e práticas DevOps nas organizações.
- **Conclusão –** apresenta uma recomendação com guia de boas práticas para o uso de *DevOps* de acordo com pesquisas realizadas nas organizações objeto desta pesquisa.

4.3.1 Análise e discussão dos resultados

O questionário da pesquisa foi dividido em 3 subgrupos por questões de organização com relação ao escopo das questões. Os questionários foram assim distribuídos:

- **DevOps Survey** - Parte do questionário que tem por objetivo obter informações gerais sobre a área de *DevOps*, o comportamento da equipe e ferramentas, sem se ater a detalhes de técnicas específicas.
- **Dev - Continuous Delivery Survey** - Por se tratar de parte fundamental de *DevOps* voltada para técnicas na área de desenvolvimento, esta parte do questionário busca informações a respeito de técnicas e práticas de Entrega Contínua utilizadas pela equipe de desenvolvimento nas organizações.
- **Ops - DevOps Survey** - Esta parte do questionário é voltada a buscar informações a respeito de técnicas, práticas e comportamento utilizadas pela equipe de operação na organização

O questionário ficou disponível na Internet entre o dia 10/11/2014 ao dia 23/11/2014 e, ao mesmo tempo, foi rodado em congresso específico da área, o *Agile Brazil 2014,* (AGILE BRAZIL, 2014), com participantes das organizações que se mostraram interessados no tema corrente e se encaixavam no perfil da pesquisa. No total, 28 pessoas responsáveis pelas organizações responderam às pesquisas solicitadas.

Com relação ao público alvo da pesquisa, podemos verificar na **Figura 7** que 39% possui conhecimento e participa do processo de *DevOps* da instituição, bem como também outros 39% possui algum conhecimento na área e entende sobre o básico do assunto. Outros 18% possui muito conhecimento e são responsáveis pela área de *DevOps* na organização e somente 4% não possui conhecimento nem trabalha na com *DevOps*.

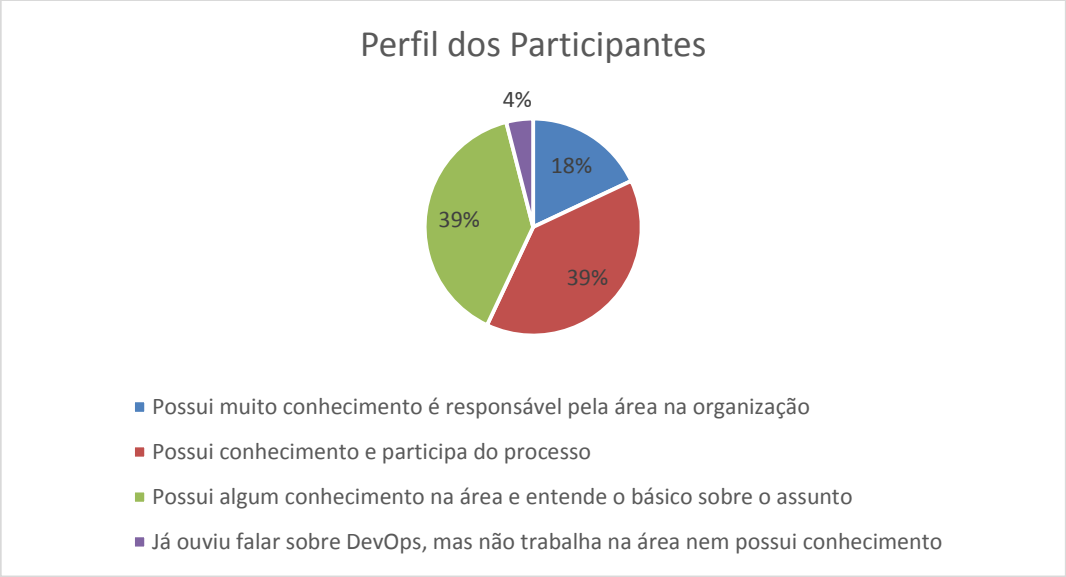

Figura 7: Distribuição dos participantes que possuem conhecimento na área de *DevOps*

De acordo com a **Figura 8**, quando perguntado sobre quais áreas de práticas *DevOps* sua organização estava trabalhando, podemos verificar que todos os entrevistados informaram que estão trabalhando com **Desenvolvimento e integração contínua**, assim como 24 de um total de 28 informaram que as organizações estão trabalhando com **Liberação, entrega e implantação contínua**, 9 responderam que as empresas estão investindo em **Automação da infraestrutura** e 12 responderam que as empresas estão trabalhando com ***Feedback* contínuo**.

Esses resultados mostram que as organizações estão em busca constante por formas de liberar cada vez mais rapidamente versões para os clientes através de práticas de integração e entrega contínua e de maior interação com o cliente. Ao mesmo tempo, curiosamente, apenas 32% afirmaram que as organizações estão trabalhando e investindo em automação da infraestrutura, enfoque esse pilar fundamental para implantação dessas práticas *DevOps* nas organizações.

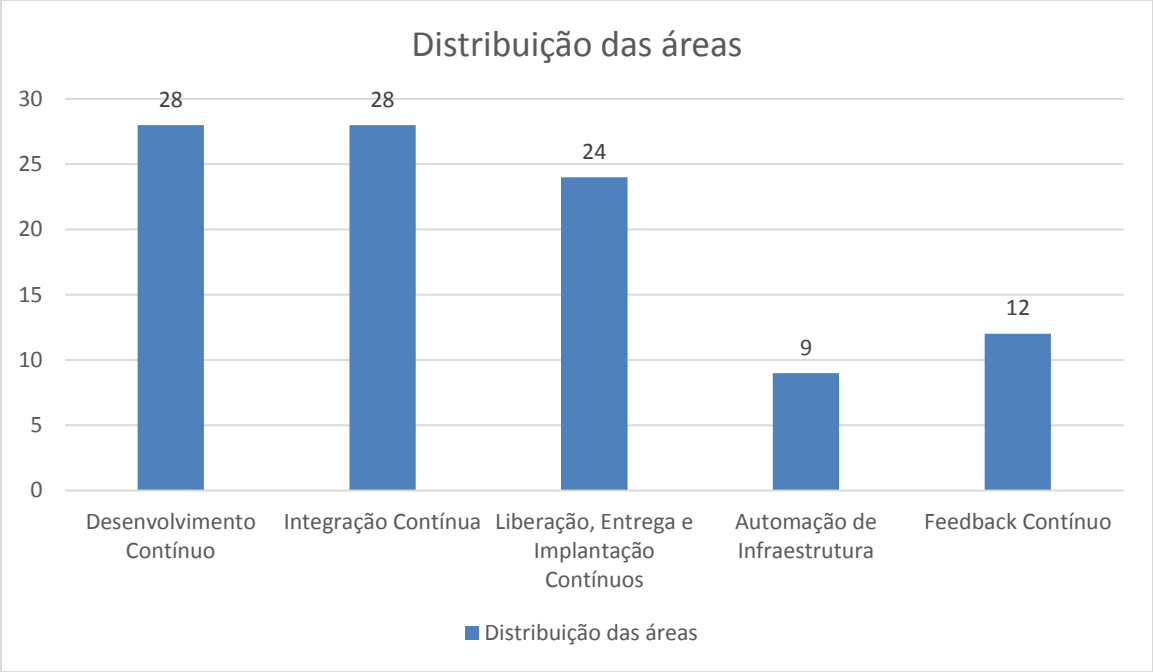

Figura 8: Distribuição das áreas que as organizações afirmam estar trabalhando

A **Figura 9** mostra como as equipes tratam os requisitos não funcionais nas organizações. Práticas *DevOps* mostram a importância dos requisitos não funcionais serem tratados por ambas as equipes, não apenas tratados pela equipe de operação (DEBOIS, 2008). Em nossa pesquisa foi perguntado se a equipe de desenvolvimento considera os requisitos não funcionais durante o desenvolvimento de aplicações.

De acordo com a **Figura 9**, pode-se notar que 36% das equipes consideram requisitos não funcionais no desenvolvimento de aplicações, enquanto que 18% não consideram requisitos não funcionais no desenvolvimento de aplicações, 39% considera responsabilidade de ambas as equipes, mas não desenvolve considerando esses requisitos e 7% considera total responsabilidade da equipe de operação.

Os resultados apresentados neste gráfico, **Figura 9**, mostram que a maioria dos entrevistados afirmaram que consideram requisitos não funcionais durante o desenvolvimento e que são responsabilidades de ambas as equipes, porém destes, 36% consideram efetivamente esses requisitos durante o desenvolvimento, enquanto outros, 39%, consideram responsabilidade de ambas as equipes mas não consideram durante a codificação. Os resultados mostraram uma mudança cultural voltada às práticas *DevOps*. Antes esses requisitos eram apenas considerados responsabilidades

da equipe da operação: 7% dos entrevistados ainda consideram total responsabilidade da equipe de operação.

Figura 9: Responsabilidades sobre requisitos não funcionais

Com relação às práticas que influenciaram positivamente o nível da entrega de software, podemos destacar que 27 dos entrevistados informaram que desenvolvimento e integração contínuo influenciaram positivamente; 19 responderam liberação, entrega e implantação contínua; 5 responderam automação de infraestrutura e feedback; e apenas um dos entrevistados respondeu que a melhora contínua influenciou positivamente na melhoria da entrega de software, conforme gráfico demonstrado na **Figura 10**.

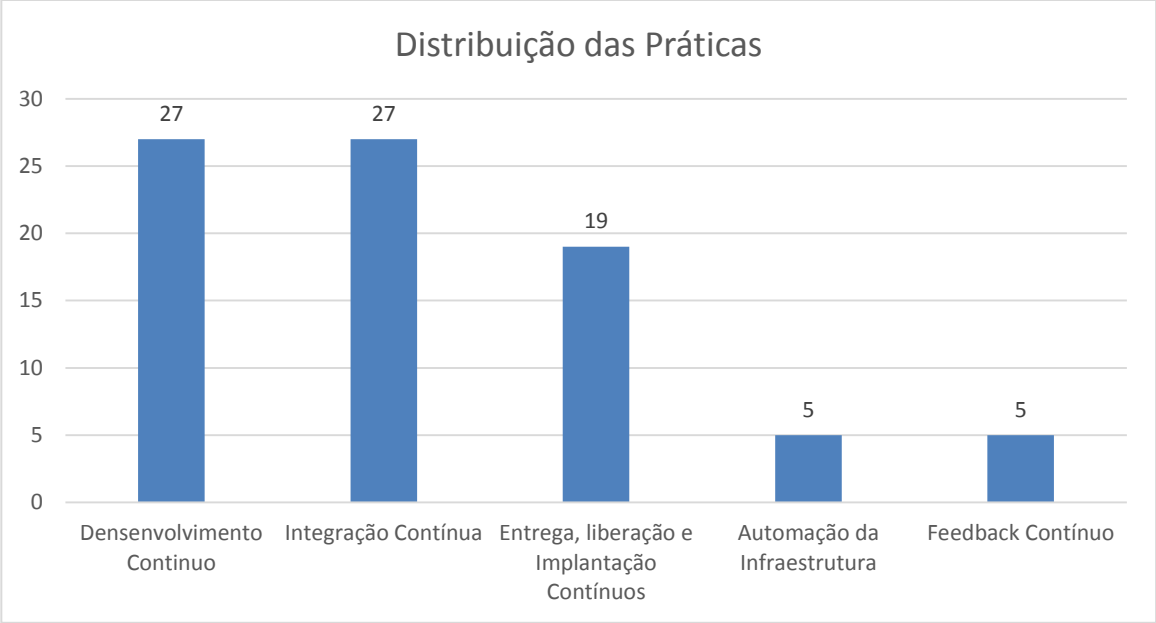

Figura 10: Distribuição das práticas que influenciaram positivamente a entrega de software

Um dos pilares de *DevOps* se refere a time *cross-functional*, o que significa que todos os envolvidos são responsáveis pelo processo de entrega, qualquer membro do time pode modificar qualquer parte do sistema, seja da equipe de desenvolvimento ou de operação. A **Figura 11** mostra que dentre os entrevistados, 50% afirma que a organização ao qual ele está envolvido usa time *cross-functional* em todos os seus projetos enquanto que 39% utilizam times *cross* em alguns projetos e apenas 11% não utilizam, ou seja, existe uma separação clara de responsabilidade entre as equipes. Essa relação mostra uma tendência entre as organizações que utilizam equipes multidisciplinares e a distribuição de responsabilidade entre os envolvidos aderindo assim às práticas *DevOps* nas organizações.

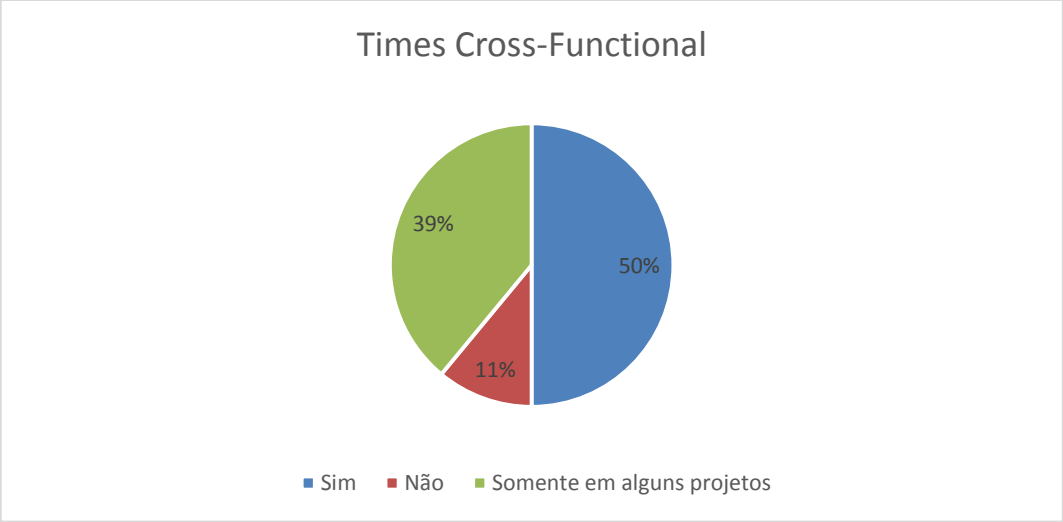

Figura 11: Distribuição de times *cross-functional* nas organizações

Outra pergunta do questionário foi relacionada às ferramentas necessárias para se obter sucesso na implantação de *DevOps*. A **Figura 12** reflete esse gráfico demonstrando as principais ferramentas de acordo com o questionário.

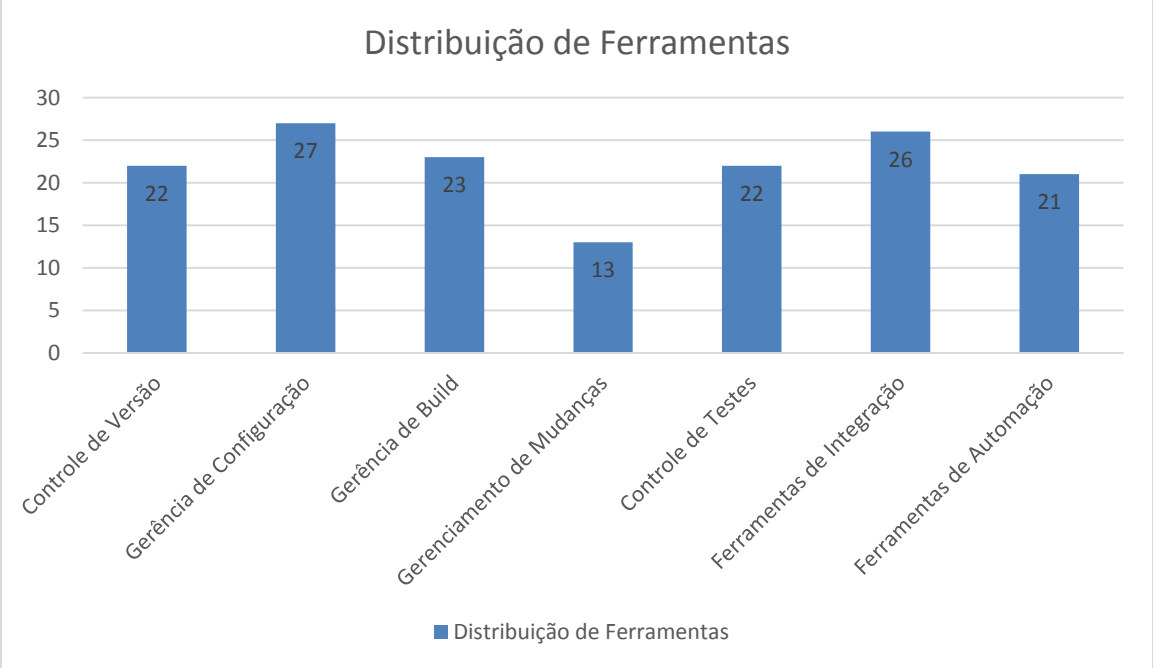

Figura 12: Distribuição de ferramentas para sucesso na implantação de *DevOps*

Devido a sua importância no âmbito de *DevOps*, foi feito um questionário específico a respeito de entrega contínua de software e suas práticas relacionadas. A

Figura 13 mostrou a relação sobre o uso de entrega contínua nas organizações. Um dado importante da pesquisa é que mais de 80% demonstram que usam práticas de entregas contínuas em todos ou alguns projetos da organização e que a maioria das organizações possui um estreitamento com *DevOps* através das práticas de entregas contínuas. As outras parcelas das organizações que foram entrevistadas afirmaram que estão estudando em qual projeto implantar ou pesquisando a respeito das práticas. Apenas 4% dos entrevistados afirmaram que não possuem nenhum projeto relacionado à entrega contínua de software.

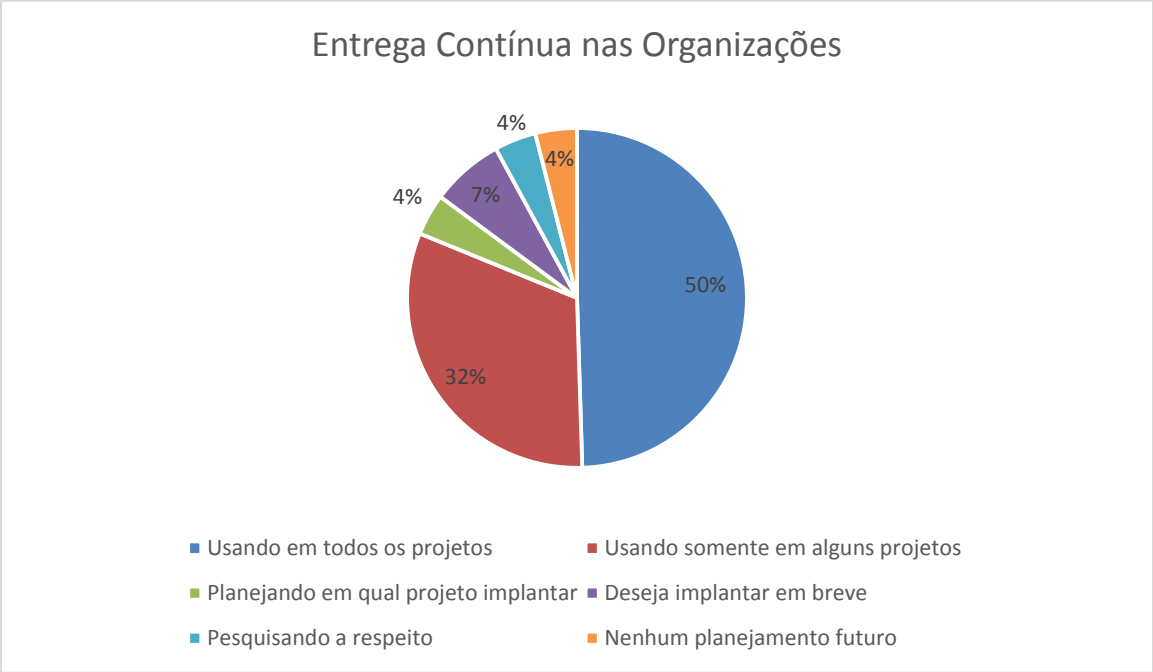

Figura 13: Relação sobre o uso de entrega contínua nas organizações

Como prática fundamental de implantação e entrega contínua, os testes automáticos fazem parte do coração do processo de implantação. A Figura 12 mostra informações a respeito do uso de testes automáticos no processo de implantação demonstrando essa relação em todas as fases do processo que utiliza testes automáticos. Os testes automáticos são práticas fundamentais no processo de implantação e entrega contínua e são um dos pilares fundamentais para o *pipeline* de implantação (HUMBLE e FARLEY, 2010). De acordo com o gráfico mostrado na **Figura 14** percebemos uma relação bem distribuída do uso de testes automáticos

durante o processo de implantação. Mais de 50% dos entrevistados informaram que entre 40% a 80% dos testes são executados de forma automática. O que representa uma boa parcela visto que, muitas vezes, a tarefa de automação de testes representa um grande esforço para a organização. Apenas 15% dos entrevistados possuem menos de 5% dos testes automáticos no processo de implantação.

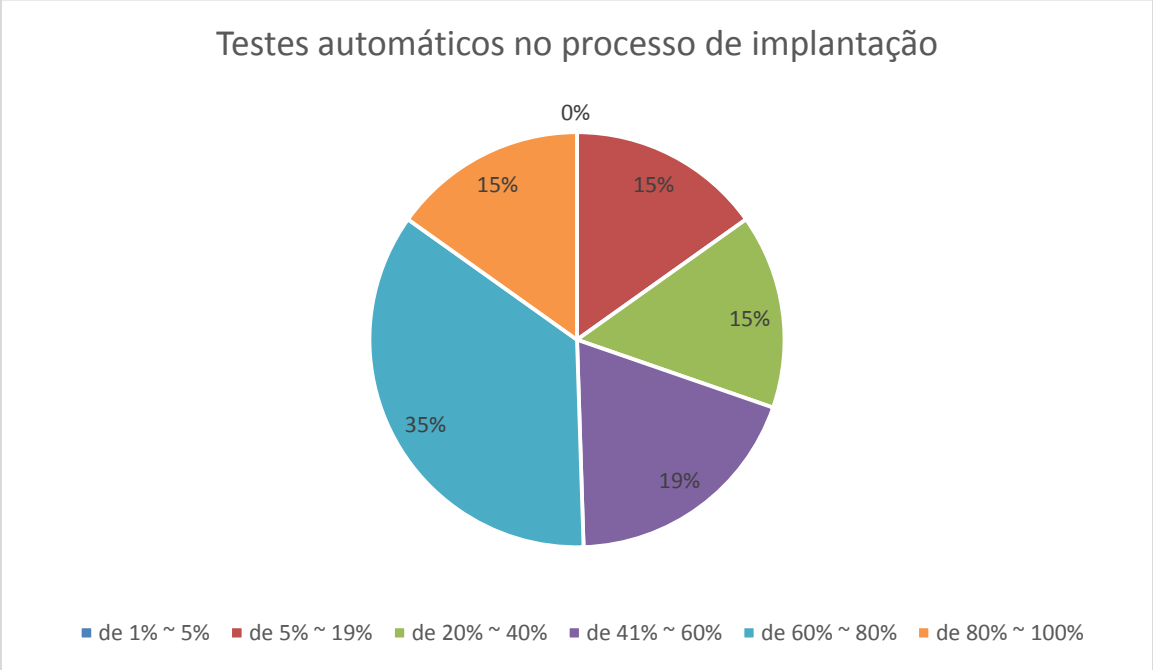

Figura 14: Distribuição dos testes automáticos no processo de implantação.

Executar implantação de software em máquinas virtuais permite simular vários ambientes em todo o processo de entrega de forma mais próxima possível do ambiente de produção proporcionando um processo mais dinâmico na entrega de software. Pensando nisso, esta questão abordou sobre o o uso de máquinas virtuais em ambientes controlados (sob controle de versão) com objetivo de reproduzir qualquer ambiente, a qualquer tempo, procurando assim entender como as organizações estão utilizando esta prática *DevOps*. Como pode ser visto em Duvall (2011), implantação de software em máquinas virtuais proporciona um processo mais dinâmico permitindo assim simular vários ambientes em todo o processo de entrega de forma mais próxima possível do ambiente de produção. A **Figura 15** mostrou a relação das organizações que fazem implantação de software em máquinas virtuais para ambientes diferentes

(testes, desenvolvimento) com esses ambientes controlados sob controle versão. O resultado mostrado da pesquisa mostrou que 90% utilizam recursos de máquinas virtuais para fazer a implantação da aplicação. Do total, 57% não necessitam da intervenção da equipe de operação durante esse processo, ou seja, existe automação da infraestrutura para esse fim e esse processo é iniciado pela equipe responsável pela implantação, enquanto que para 33% o processo de implantação precisa de total intervenção de uma equipe de operação para esse fim.

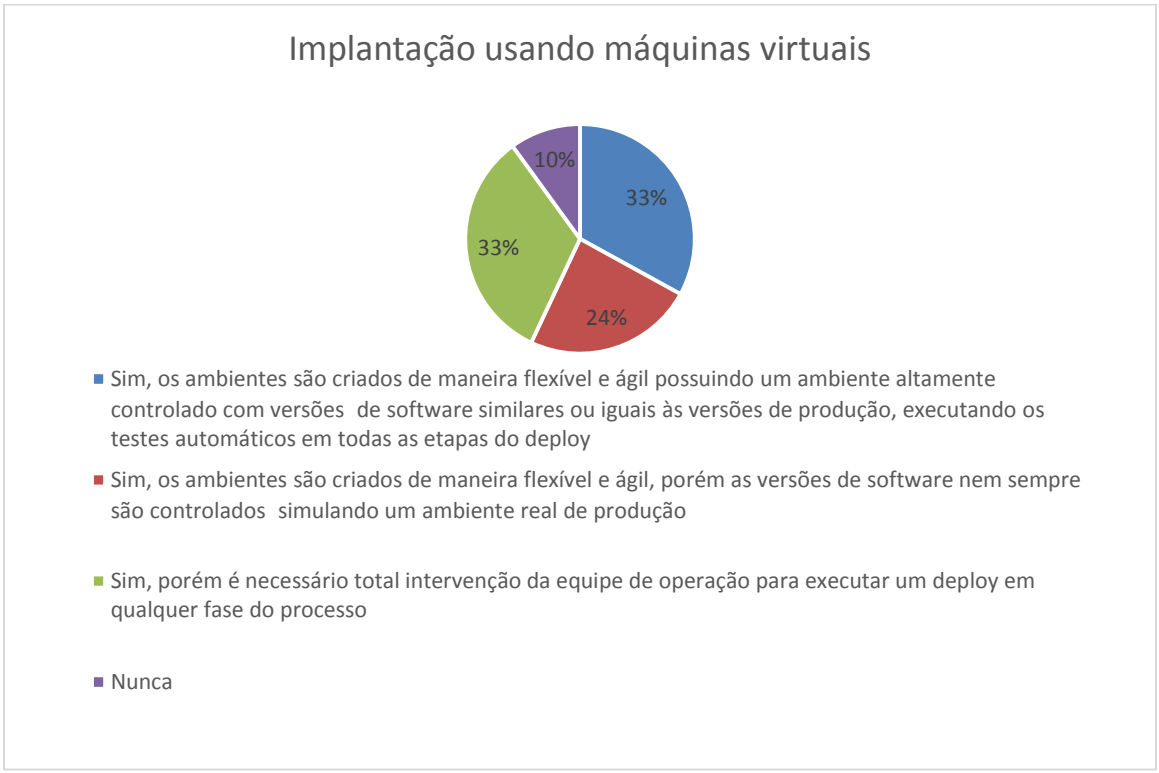

Figura 15: Uso de ambiente controlados para implantação de softwares usando maquinas virtuais

A **Figura 16** mostrou que as organizações objeto da pesquisa não estão fazendo uso de nuvem pública para fazer a implantação de suas aplicações. A maior parte das respostas inclusive afirma que não faz parte de nenhum tipo de nuvem para implantação da aplicação (64%) o que contrariou algumas pesquisas algumas pesquisas recentes relativas ao uso de nuvem no universo de *startups* e *DevOps* (PUPPET et al., 2013). Apenas 36% faz algum tipo de uso de nuvem privada.

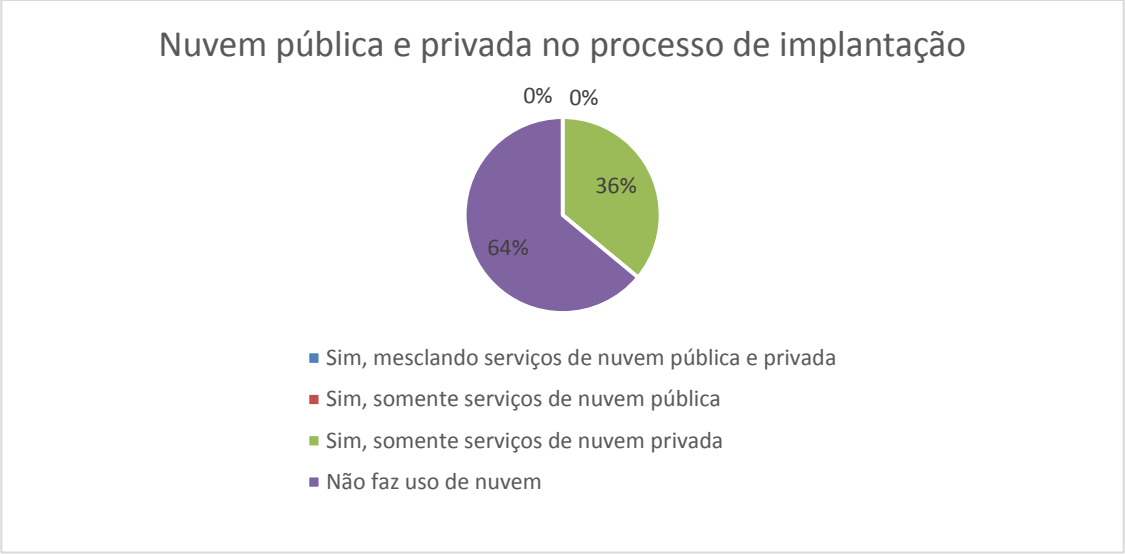

Figura 16: Relação de nuvem pública e privada no processo de implantação.

Uma das principais práticas de entrega contínua de software é fornecer visibilidade de todo o *pipeline* de implantação/entrega a todos os envolvidos no projeto para permitir aos membros da equipe um acompanhamento único da divulgação da informação durante todo o processo (*build,* implantação e testes). Com objetivo de obter informação se as organizações estão utilizando essa prática, a **Figura 17** mostrou a relação de uso dessa prática. Como se pode perceber, em torno de 90% das organizações pesquisadas utilizam essa prática em todos os projetos de suas organizações.

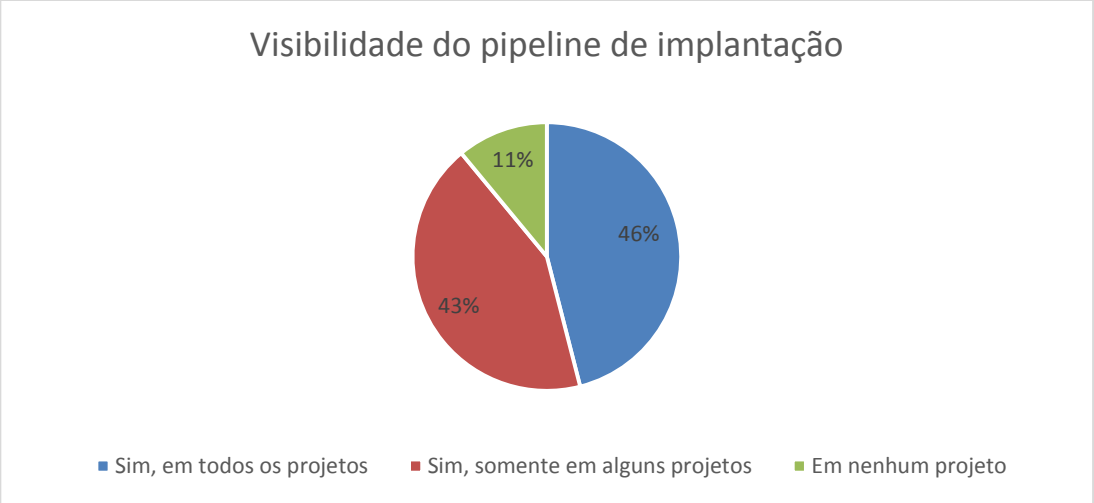

Figura 17: Visibilidade do pipeline de implantação nas organizações

À medida que se institucionaliza a entrega e implantação contínua na organização, exige-se também que a mesma possua um processo de *rollback* repetível e automático no caso de algum problema ser encontrado em produção após alguma entrega, fornecendo assim um fácil caminho de reversão caso aconteça algum problema na implantação de uma nova versão. Portanto, essa pergunta procurou entender se as empresas estão usando processo de *rollback* em seus processos por conta do uso cada vez maior de entrega e implantação contínua. A medida que se estabelece um processo de entrega contínua com o seu *pipeline* de implantação automatizado, o processo de *roolback* se torna ainda mais importante no caso de algum problema ser encontrado em produção após alguma entrega. Esse processo fornece um fácil caminho reversível caso aconteça algum problema na implantação de uma nova versão. **A Figura 18** mostra que 64% das organizações utilizam o processo de *roolback* receptível e automático em todos ou alguns projetos enquanto 34% não utilizam, não são automáticos e repetíveis ou desconhecem essa prática.

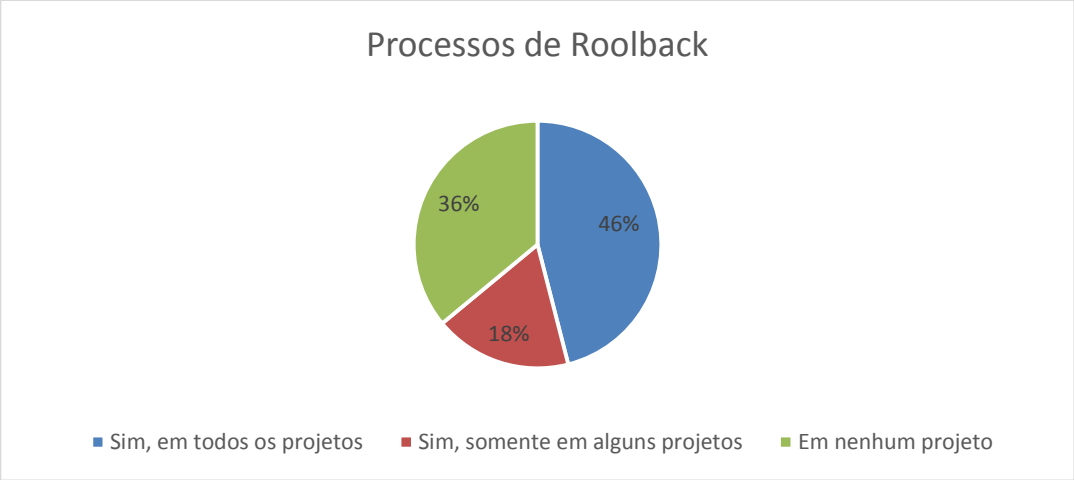

Figura 18: Relação das empresas que utilizam processo de *roolback*

A entrega e implantação contínua exige processos de maduros *roolback* em conjunto com técnicas como *canary release, toogled features* e *blue-green deployments.* O *canary release* consiste em liberar uma nova versão de software em produção somente para um pequeno grupo de usuários. Essa técnica tem por objetivo reduzir o impacto de um possível *bug* ser encontrado em um ambiente de produção isolando usuários não afetados implantando assim a nova versão aos poucos. A técnica de *Canary Release* também pode ser usada quando se tem versões sutilmente diferentes da aplicação em produção ao mesmo tempo com objetivo de coletar estatísticas de como novas funcionalidades estão sendo usadas e eliminando as que não estiverem entregando valor suficiente. Isso oferece uma abordagem evolutiva muito eficaz para a adoção de novas funcionalidades. O *canary release* é bastante usado quase se tem uma base heterogênea de usuários onde os testes automáticos do *pipeline* não conseguem ter uma boa abrangência com relação a vários ambientes diferentes.

Toogled features também consiste das melhores práticas de entrega contínua. Esta técnica consiste em implantar novas mudanças no sistema em produção habilitando e desabilitando as mudanças aos poucos adicionando assim controle e estabilidade às mudanças em produção. A técnica de *toogled features* permite testar uma combinação de mudanças no software com objetivo de planejar seu uso em produção além de poder testar também características de testes de performance para

uma grande base de usuários.

Outra técnica importante com o objetivo de reduzir riscos de implantação é chamada de *blue-green deployments*. Nesta prática o ambiente possui dois ambientes de produção, um ambiente que está sendo usado, podendo ser chamado de *blue*, e outro ambiente, o *green,* que será o *release* final após ter passado pelas etapas do *pipeline* de implantação estando apto para ir à produção com as novas mudanças. Uma vez que o *blue* está em produção, pode ser roteado para o ambiente *green* sem impacto aos usuários e principalmente fornecendo um rápido caminho para o processo *roolback*, facilitando assim a troca para o ambiente anterior caso aconteça algum problema na implantação de uma nova versão.

Estas práticas, assim como o processo de *roolback*, são formas de garantir mais estabilidade na implantação das mudanças em produção, principalmente mudanças em ambientes de larga escala com grande base de usuários. As Figuras **19** e **20** mostram a relação de uso dessas práticas nas organizações.

A **Figura 19** mostra que 68% dos entrevistados desconhecem essas técnicas e 18% utilizam ambas as práticas nas organizações. Apenas 4% e 7% utilizam somente as práticas de *canary release* e *toogled feature* respectivamente, o que pode ser talvez justificado por ambas serem mais utilizadas em ambientes de larga escala. Mesmo assim ainda pode ser considerado um número bastante alto que ainda desconhecem essas práticas visto a importância delas no âmbito que representa na entrega contínua e em *DevOps*.

O mesmo também ocorre com a prática de *blue-green deployment* que tem por objetivo de reduzir riscos de implantação possuindo dois ambientes de produção - *blue* e *green* - um servindo a produção propriamente dita e outro com o ambiente com infraestrutra semelhante para testar novas *builds* ou características, chaveando assim entre os ambientes quando requisitado ajudando a reduzir o risco da implantação. Na relação da **Figura 20**, 79% dos entrevistados desconhecem essa prática e 18% dos entrevistados informaram que utilizam essa prática.

Como visto, estas três práticas - *canary release, toogled feature* e *blue-green deployment* - são peças chaves para se fornecer uma base confiável para uma implantação em larga escala e em um ambiente com *pipeline* de implantação

automatizado, porém hoje, de acordo com a pesquisa, ainda poucas organizações as utilizam e/ou desconhecem essas práticas.

Mostrada a importância dessas práticas na entrega e na implantação contínua, as **Figuras 19** e **20** mostram a relação de uso destas práticas nas organizações.

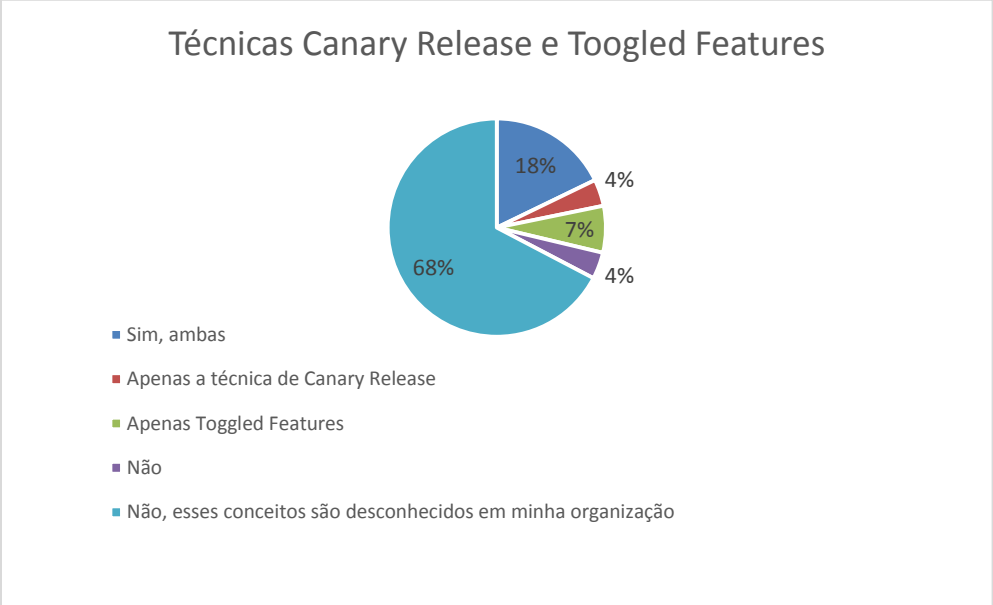

Figura 19: Uso de técnicas *Canary Release* e *Toogled Features* nas organizações

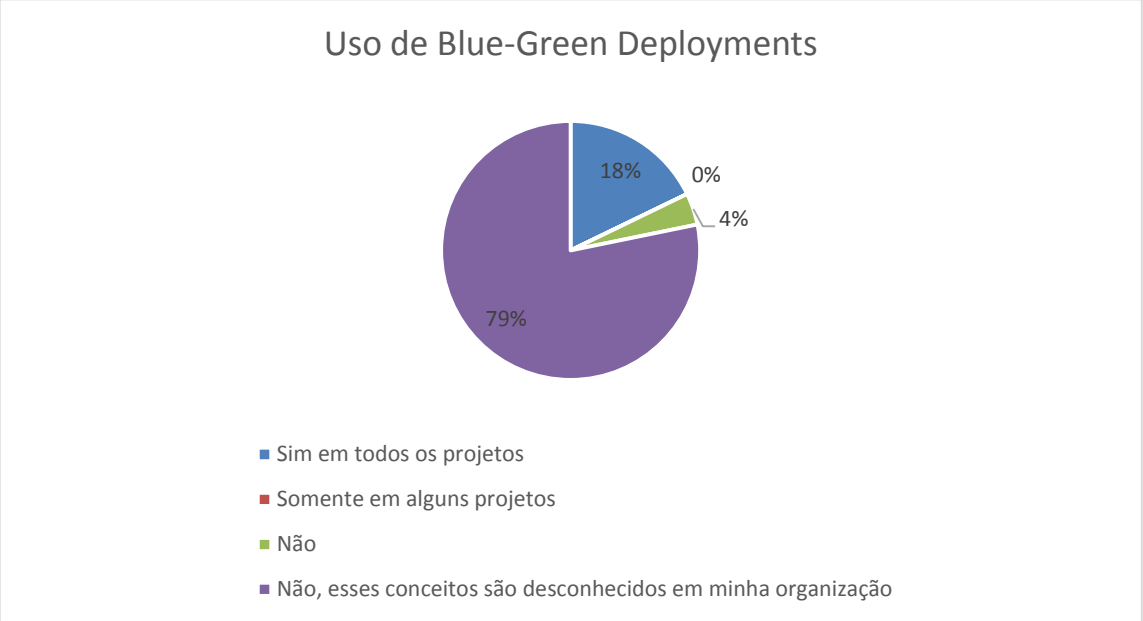

Figura 20: Demonstrativo do uso de *Blue-Green Deployments* nas organizações

Sistemas de tracking e gerenciamentos de mudanças são fundamentais para o controle de versões e a rastreabilidade de artefatos, porém, para uma maior agilidade com práticas *DevOps*, é recomendado que eles sejam atrelados ao servidor de integração da organização com objetivo de construir um processo mais automatizado de release notes fornecendo assim rastreabilidade e auditoria à organização. A **Figura 21** demonstra essa relação.

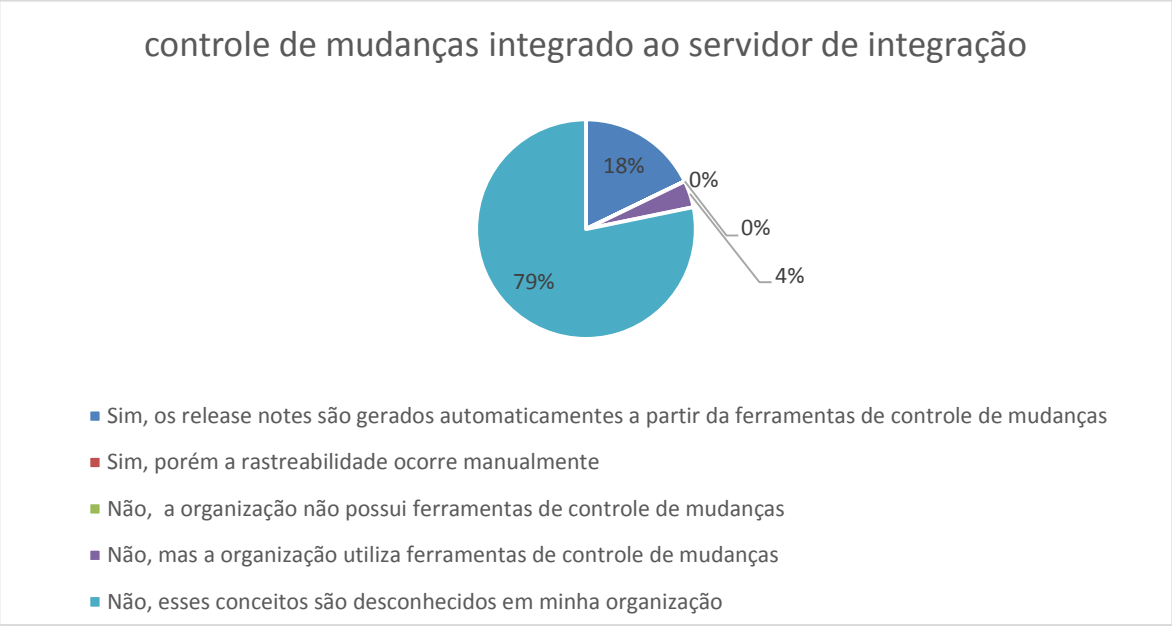

Figura 21: Demonstrativo da relação de uso do controle de mudanças integrado ao servidor de integração

Com relação à equipe de operação, foram formuladas perguntas relativas a essa equipe e suas tecnologias relacionadas para gerar *builds* do sistema de forma centralizada assim como a rastreabilidade e agilidade de informações. Primeiramente, foi perguntado como a organização faz a *build* da aplicação. A **Figura 22** mostra a relação.

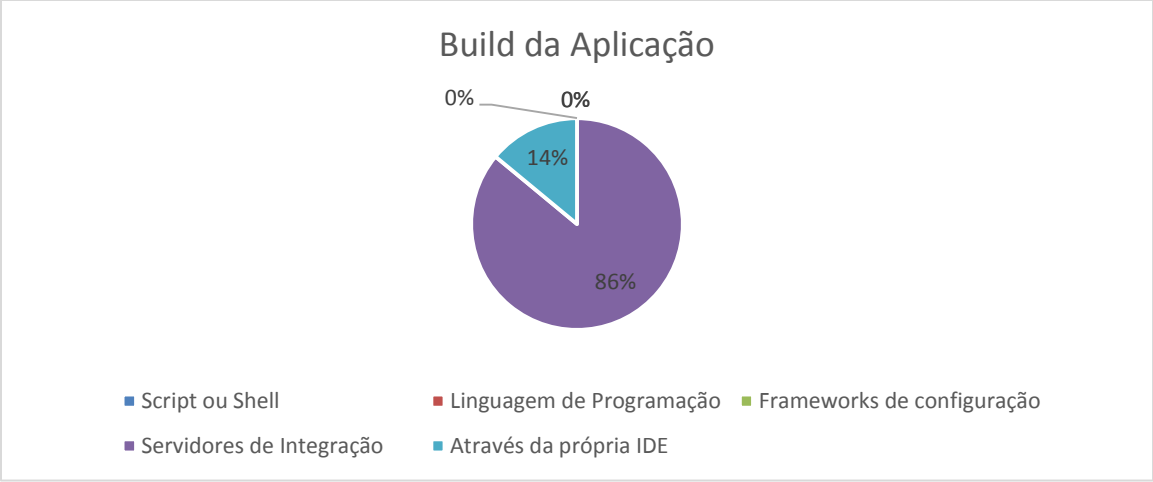

Figura 22: Maneiras de gerar build da aplicação

Controle de mudanças para rastreabilidade de artefatos também faz parte dia-a-dia de equipe da equipe de operação fundamentada no alicerce *DevOps* para gerenciar a infraestrutura controlando assim as versões de ambiente. Cada vez mais automatizada com uso de *DevOps*, a infraestrutura exige um maior gerenciamento para controlar as mudanças, rastrear os artefatos e controlar e gerar as versões de ambientes. Ferramentas para gerenciamento e automatização da infraestrutura - *puppet* e *chef* - estão cada vez mais sendo usadas em ambientes de infraestruturas atuais que exigem controle, eficácia e flexibilidade em suas instalações. A **Figura 23** mostra que 41% dos entrevistados tem a equipe de desenvolvimento e equipe de operação trabalhando em conjunto e construindo automaticamente um ambiente controlado, consistente e reproduzível em alguns projetos de projetos enquanto que para 48% esse trabalho é realizado somente pela equipe de operação.

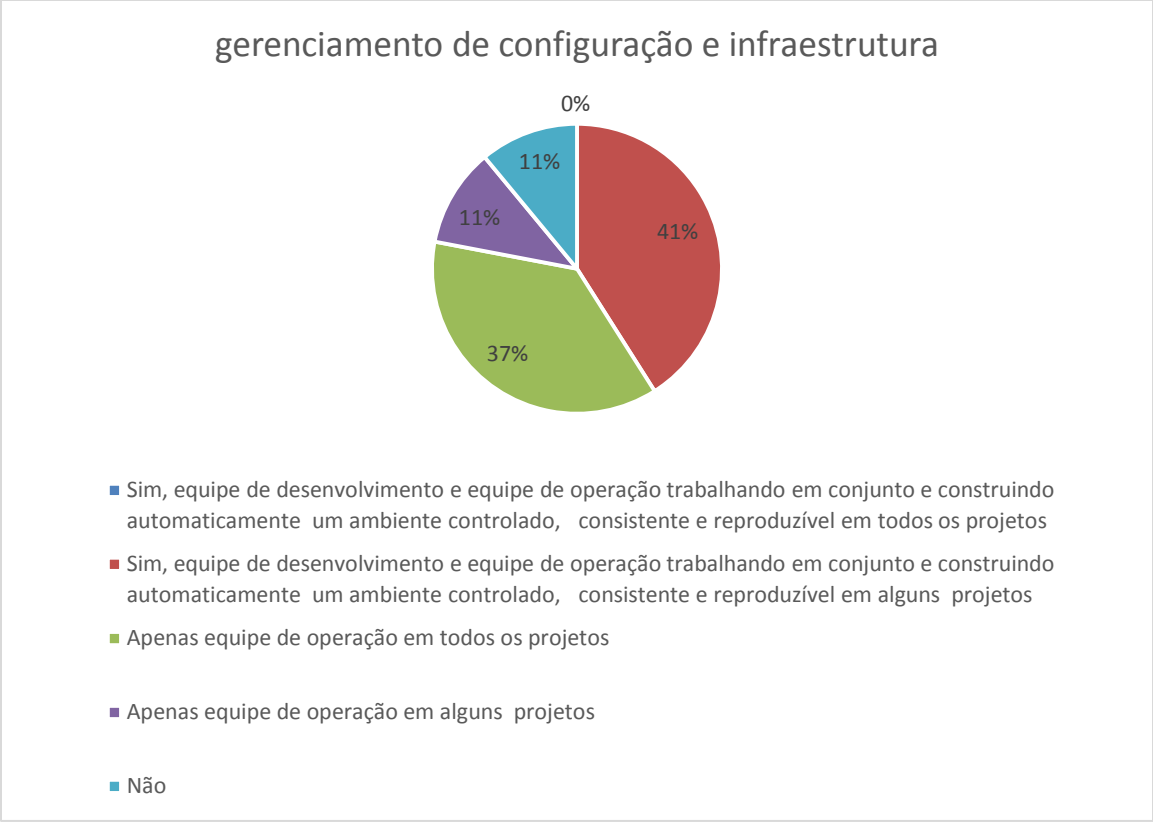

Figura 23: Uso de ferramentas para controle de gerenciamento de configuração e infraestrutura

Tão importante quanto o controle dos ambientes de configuração e infraestrutura é como é feita a montagem de um ambiente para uma implantação de uma aplicação, seja de desenvolvimento, testes ou produção. A **Figura 24** mostra o uso e quais as ferramentas para fazer a implantação da aplicação para ambientes de desenvolvimento, testes e produção. Aproximadamente 90% utilizam máquinas virtuais para fazerem a implantação da aplicação nos ambientes, destes, 38% as utilizam de forma *self-service* sem a intervenção da equipe de operação no processo e 12% desses solicita a equipe de operação para fazer a implantação da aplicação.

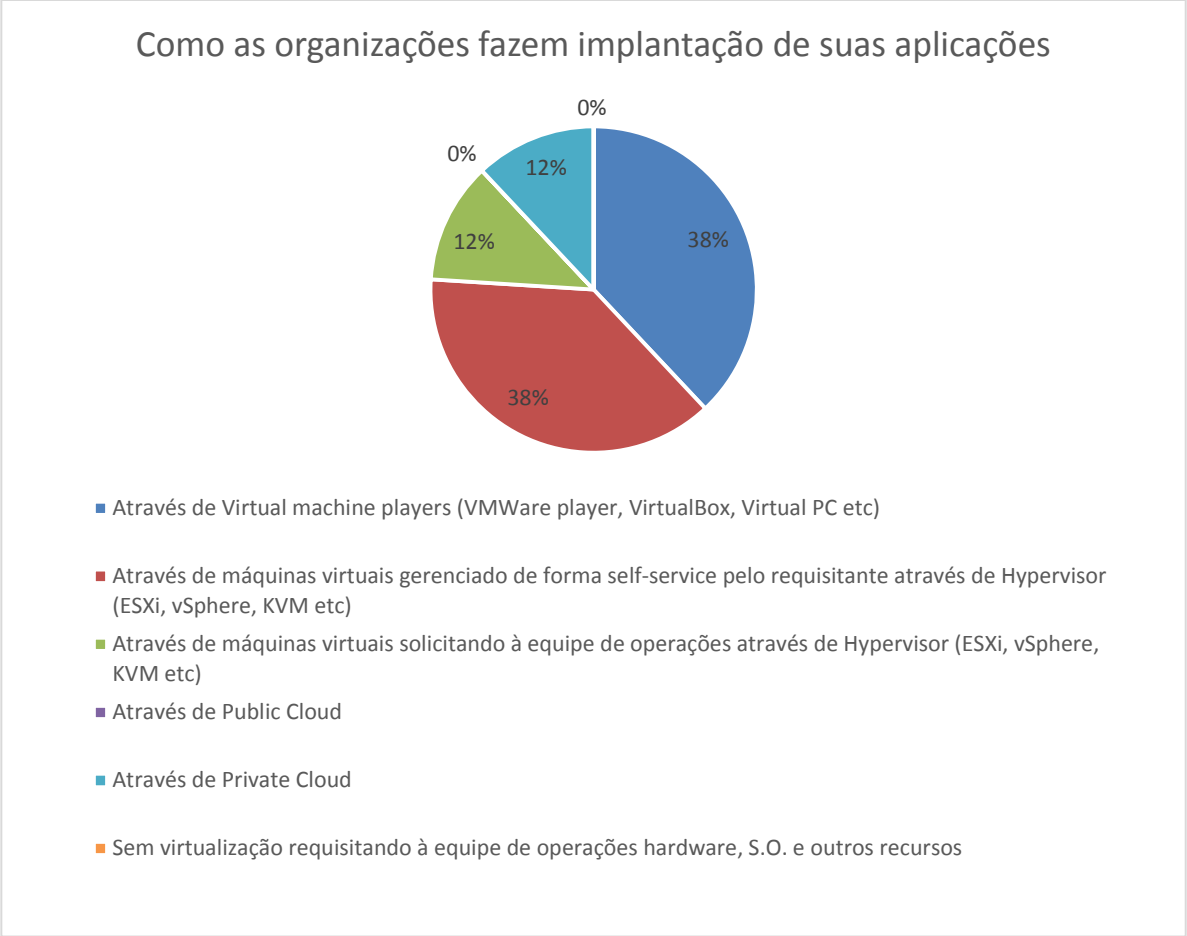

Figura 24: Relação de como as organizações fazem a implantação de suas aplicações

4.4 Conclusão

Atualmente, *DevOps* é considerado mais que um movimento filosófico, mas ainda não se pode descrevê-lo como uma metodologia, um *framework* ou algo descritivo tais como CMMI, ITIL, *Agile* ou *Lean*, por exemplo. Portanto, esta pesquisa também teve por objetivo fornecer um guia com as melhores práticas de *DevOps* que as organizações estão usando em comum para ajudar outras organizações que pretendem começar a utilizar *DevOps* compartilhando experiências como um guia mais prescritivo de seu uso. O catálogo com as melhores práticas pretende fornecer informações suficientes para as empresas criarem, valores, culturas, processos e práticas do dia a dia de *DevOps*.

Este guia não tem por objetivo detalhar as práticas e técnicas usadas em *DevOps*, mas sim fornecer como um passo inicial informações essenciais ao uso de *DevOps* nas organizações.

Com base nas pesquisas do mapeamento sistemático e principalmente da aplicação dos questionários, pode-se apresentar algumas orientações que podem servir como boas práticas:

Requisitos não funcionais – requisitos não funcionais são importantes serem considerados desde a concepção do projeto até a fase de desenvolvimento e testes. No âmbito *DevOps*, requisitos não funcionais deixaram de ser responsabilidade somente da equipe de operação, mas sim ser responsabilidade de ambas as equipes – desenvolvimento e operação – igualmente dividida.

Áreas de *DevOps* – A distribuição das áreas que mais influenciaram positivamente o sucesso de *DevOps* na pesquisa foram desenvolvimento, integração, entrega, liberação e feedback contínuo a automação da infraestrutura.

Times *Cross-Funcional* – O uso de times *cross-funcional* significa que todos os envolvidos são responsáveis pelo processo de entrega, qualquer membro do time pode modificar qualquer parte do sistema, seja da equipe de desenvolvimento ou de operação.

Principais Ferramentas *DevOps* – Como principais ferramentas para o sucesso de implantação *DevOps* na organização podemos citar ferramentas de gerência de versão, configuração e mudança, ferramentas para geração de *builds* e integração e ferramentas para automação infraestrutura e testes.

Entrega e Implantação Contínua de Software – Pode-se afirmar que a entrega e implantação contínua de software fazem parte do coração de *DevOps*. Essas práticas *DevOps* englobam várias outras técnicas e práticas para se institucionalizar em organização, tais como a automação da infraestrutura, testes e de todo *pipeline* de implantação.

Testes Automáticos - Os testes automáticos são práticas fundamentais no processo de implantação e entrega contínua e são um dos pilares fundamentais para o *pipeline* de implantação.

Implantação usando máquinas virtuais – Sabe-se que uso de máquinas virtuais melhorou muito a eficiência e dinâmica do uso dos sistemas operacionais nas organizações. Portanto, fazer uso da implantação em máquinas virtuais para ambientes diferentes de desenvolvimento, testes, homologação ou produção melhora a eficiência, produtividade e rastreabilidade do desenvolvimento, testes e da implantação no *pipeline* de implantação.

Uso de computação em nuvem – O uso de nuvem pública ou privada no processo de implantação melhora eficiência do desempenho e dos custos relativo a recursos computacionais, além de fornecer maior dinamismo em testes automáticos e na implantação e liberação de softwares.

Visibilidade do pipeline de implantação – A visibilidade de todo o *pipeline* de implantação/entrega a todos os envolvidos no projeto para permitir aos membros da equipe um acompanhamento único da divulgação da informação durante todo o processo (*build,* implantação e testes).

Processos de *roolback* - A medida que a entrega e implantação contínua se estabelece na organização, exige-se também um processo de *rollback* repetível, automático e confiável no caso de algum problema ser encontrado em produção após alguma entrega, fornecendo assim um fácil caminho de reversão caso aconteça algum problema na implantação de uma nova versão.

Canary release – A técnica de canary release consiste em liberar uma nova versão de software em produção somente para um pequeno grupo de usuários. A medida que se software se comporta da maneira esperada, mais usuários receberão a nova versão. Essa técnica pode ser usada com o objetivo de testar novas versões, diminuir o impacto de um possível bug ser encontrado em produção ou mesmo coletar estatística de novas características de uma versão.

Toogled Features - Esta técnica consiste em implantar novas mudanças no sistema em produção habilitando e/ou desabilitando as mudanças aos poucos adicionando assim controle e estabilidade às mudanças em produção. Usados geralmente para testar novas características em produção ou executar testes de performance para uma grande base de usuários.

Blue-green Deployments – O objetivo de blue-green deployments é possuir dois ambientes em produção com possibilidade de rápido roteamento entre eles, podendo assim fazer mudanças em uma de suas versões e também com possibilidade de *roolback* ao ambiente anterior caso necessário.

Uso de ferramentas para controle de gerenciamento de configuração e infraestrutura – o gerenciamento da configuração e a infraestrutura exigem um maior gerenciamento para controlar as mudanças, rastrear os artefatos e controlar e gerar as versões de ambientes, permitindo assim uma maior automatização do ambiente.

5. CONSIDERAÇÕES FINAIS

Este capítulo apresenta as limitações e ameaças à validade desta pesquisa bem como trabalhos futuros e conclusões acerca deste trabalho.

5.1 Limitações da pesquisa

Esta pesquisa seguiu padrões de acordo com o quadro metodológico descrito no capítulo 3, porém, ainda assim, esta pesquisa apresenta algumas limitações.

Uma limitação comum em mapeamentos sistemáticos, segundo Kitchenham (2007), é poder não encontrar todos os artigos relevantes existentes sobre o tema a ser pesquisado. Afim de minimizar essa limitação, esta pesquisa utilizou, além de buscas automáticas nos engenhos de buscas mais relevantes, buscas manuais nos periódicos e anais das principais conferências como forma de reduzir a possibilidade de algum estudo relevante não ter sido encontrado na execução das buscas automáticas.

Uma outra limitação com relação ao mapeamento sistemático, mais especificamente com relação aos critérios de inclusão e exclusão, é referente à data limite da pesquisa, visto que, em se tratando de uma área nova, onde pesquisas estão sendo publicadas com maior periodicidade, podem haver estudos relevantes que não entraram como objeto desta pesquisa. Outra limitação está relacionada à *string* de busca pois alguma palavra relevante pode não ter sido incluída neste termo. A *string* de busca procurou ser a mais abrangente possível para evitar este problema.

Com objetivo de evitar o viés na seleção dos estudos, pilotos foram realizados em cada etapa da pesquisa e essas foram realizadas por mais de um pesquisador. A fase de extração de dados foi realizada por um único pesquisador, o autor desta pesquisa e auxiliada pelo orientador e co-orientador deste trabalho.

O *survey* desta pesquisa teve objetivo buscar informações das práticas *DevOps* mais usadas na indústria de software, então uma limitação deste questionário é a amostra da quantidade de empresas que responderam ao formulário o que pode não representar todas as áreas identificadas no *survey*. Uma outra limitação relacionada ao *survey* é relacionado ao perfil profissional do entrevistado não corresponder diretamente ao desejado pelo entrevistador na elaboração dos questionários. Para

esse fim, foi determinada algumas questões relativas à identificação do perfil profissional do entrevistado.

5.2 Recomendações para trabalhos futuros

A partir desta pesquisa, foi evidenciado algumas áreas a serem exploradas como trabalhos futuros a partir desta pesquisa. Como uma área nova, *DevOps* se mostra como uma área muito promissora para fins de pesquisas e publicações, além de se mostrar muito eficiente aos resultados que propõe. *DevOps*, como uma área relativamente nova na indústria e na academia, não existe ainda um padrão de práticas e quais as áreas as organizações estão usando em seu dia a dia. Como visto, este trabalho procurou identificar as principais áreas e práticas que as indústrias de software estão utilizando atualmente, porém, como trabalho futuro, podem ser identificadas as seguintes posições:

- A partir das práticas identificadas, realizar um estudo de caso em organizações afim de avaliar o impacto no negócio dessa adoção;
- Elaborar um modelo de maturidade com objetivo de separar níveis de implementação das práticas *DevOps* de acordo com o nível de maturidade das empresas;
- Definição de um modelo ou metodologia para o processo de adoção *DevOps*;
- Identificação das principais ferramentas e tecnologias usadas em *DevOps*;
- Fazer um mapeamento com o uso de nuvem com as áreas e práticas em *DevOps*;
- Executar revisões sistemáticas com assuntos interessantes entre uma práticas ou área de *DevOps*;

5.3 Conclusões

Esta dissertação teve por objetivo executar um mapeamento sistemático procurando identificar as áreas de *DevOps*, seus principais autores, assim como suas principais práticas relacionadas à indústria.

Como o movimento de *DevOps* é relativamente novo, o resultado do mapeamento não foi suficiente para identificar as principais práticas de *DevOps*, sendo necessário

assim a realização de *survey* com as principais práticas identificadas no mapeamento para identificar os padrões da industrias relativo a essas práticas.

As principais áreas de *DevOps* identificadas através do mapeamento foram **Integração, Entrega, Liberação, Implantação, *Feedback* e Automação de Testes e Infraestrutura**. Com relação às principais práticas *DevOps* usadas nas organizações identificadas através do *survey* pode-se destacar **testes automáticos**, **implantação** usando **máquinas virtuais**, uso de **nuvem (pública e privada)** no processo de ***deploy*, visibilidade** do ***pipeline* de implantação,** processos de ***roolback*, *canary release*** e ***toogled features, blue-green deployments*** e **automação** do gerenciamento da **configuração** e da **infraestrutura.**

REFERÊNCIAS

AGILE BRAZIL. Disponível em http://www.agilebrazil.com/2014, acessado em 23 de fevereiro de 2015.

AGILE MANIFESTO. Disponível em http://agilemanifesto.org/, acessado em 23 de setembro de 2014.

ALLSPAW J.; HAMMOND P. 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr. 2009.

BECK K. Extreme Programming Explained. 2000.

BILL P. Next Generation Process Integration: CMMI and ITIL Do Devops. 2011.

CRUZES, D.S; DYBA, T. Recommended Steps for Thematic Synthesis in Software Engineering. ESEM, 2011.

DAVIS A. Research devops survey 2014. 2014.

DEBOIS P. Agile infrastructure and operations: how infra-gile are you?. 2008.

DEGRANDIS D. Devops: So You Say You Want a Revolution? 2011.

DUVALL M. P.; MATYAS S.; GLOVER A. Continuous Integration: Improving Software Quality and Reducing Risk. 2007.

DUVALL M. P. Continuous Delivery Patterns and AntiPatterns in the Software LifeCycle. 2011.

EASTERBROOK, S. et al. Selecting Empirical Methods for Software Engineering

Research. In: SHULL, F.; SINGER, J.; SJøBERG, D. I. K. Guide to advanced empirical software engineering. (S.l.): Springer-Verlag London, 2008.

FITZGERALD B. ; DILON M. The Business Case For Devops: A Five Year Retrospective. 2011.

FITZGERALD B. ; JAN S. K. Continuous Software Engineering and Beyond: Trends and Challenges. 2014.

HUMBLE J.; FARLEY D. Continuous Delivery: Reliable Software Release Through Build, Test and Deployment Automation. 2010.

HUMBLE J.; READ C ; NORTH D. The Deployment Production Line.2006.

HUMBLE J. “Continuous delivery vs continuous deployment” .2010. Disponível em http://continuousdelivery.com/2010/08/continuous-delivery-vs-continuous-deployment/, acessado em 20 de fevereiro de 2015.

HUMBLE J.; MOLESKY J. Why Enterprises Must Adopt Devops to Enable Continuous Delivery. Cutter IT Journal. 2011.

HUTTERMANN M. DevOps for Developers: Integrate Development and Operations, The Agile Way. 2012.

KIM G. Top 11 Things You Need to Know About DevOps. 2013.

KITHENNHAM B. et al. Using Mapping Studies in Software Engineering, in: Proceedings of the 20th Annual Meeting of the Psychology of Programming Interest Group (PPIG), 2008.

KITHENNHAM B. CHARTES S. Guidelines for performing Systematic Literature

Reviews in Software Engineering, Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report, 2007.

MUSLU; BRUN; MELIOU. Data debugging with continuous testing. 2013.

PETTICREW, M.; ROBERTS, H. Systematic Reviews in the Social Sciences: a practical guide. Oxford: Blackwell Publishing, 2006.

PRESSMAN, R. Engenharia de Software. McGraw-Hill, 2011.

PUPPET LABS; IT REVOLUTION PRESS. 2013 State of DevOps Report by Puppet Labs and IT Revolution Press. 2013.

PUPPET LABS; IT REVOLUTION PRESS; THOUGHTWORKS. 2014 State of DevOps Report by Puppet Labs, IT Revolution Press and ThoughtWorks.2014.

RASAD R.; ALISHA B. MATHANGI P. M. Demystifying DevOps Through a Tester's Perspective. 2014.

REBELLABS. IT Ops & DevOps Productivity Report 2013: Tools, Methodologies and People. 2013.

REDDY A. DevOps: The IBM Approach. 2013.

ROCHE J. Adopting DevOps Practices in Quality Assurance. 2012.

ROYCE W. Managing the development of large software systems: concepts and techniques. Proc. IEEE Westcon, Los Angeles, CA. 1970.

RUBIN K. Essential Scrum: A Practical Guide to the Most Popular Agile. 2012.

SALF; ERNST. Reducing wasted development time via continuous testing. 2003.

SATO D. DevOps na prática: entrega de software confiável e automatizada. 2014.

SHARMA S. DevOps for Dummies: A Wiley Brand. 2014.

SHAMOW E. Devops at Advance Internet: How We Got in the Door. 2011.

STANDISH GROUP. CHAOS report, 586 Olde Kings Highway. Dennis, MA 02638, USA. 1995.

STAHL D.; BOSCH J. Modeling Continuous Integration Practice Differences in Industry Software Development. 2013.

WOOTTON B. Preparing for Continuous Delivery. 2013.

APÊNDICE A – Protocolo do Mapeamento Sistemático da Literatura

O conteúdo deste apêndice refere-se ao protocolo do mapeamento sistemático da literatura conduzida para esta dissertação como parte do processo de coleta de dados.

UFPE - UNIVERSIDADE FEDERAL DE PERNAMBUCO
CIn - CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROTOCOLO DO MAPEAMENTO SISTEMÁTICO DA LITERATURA

Mapeamento Sistemático da Literatura

Filipe Antônio Motta Braga
Alexandre Marcos Lins de Vasconcelos
Célio Andrade de Santana Júnior

Agosto de 2015.
Recife-PE

Equipe

Nome	Afiliação	Papel
Filipe Antônio Motta Braga	CIn – Universidade Federal de Pernambuco	Autor
Alexandre Marcos Lins de Vasconcelos	CIn – Universidade Federal de Pernambuco	Coautor e Revisor
Célio Andrade de Santana Júnior	CIn – Universidade Federal de Pernambuco	Coautor e Revisor

1. Definição da Questão de Pesquisa

O protocolo elaborado para descrever todos os procedimentos e métodos de como o mapeamento foi conduzido se encontra a seguir.

Para delinear o escopo da pesquisa e identificar os elementos que fizeram parte do seu escopo, foi utilizado a estrutura PICOC (*Population, Intervention, Comparison, Outcome, Context*) como recomendada por Kintchenhan (KITCHENHAM e CHARTERS, 2007) e Petticrew (PETTICREW et al., 2005).

População: Pesquisas na área de *DevOps*.
Palavras Relacionadas: *DevOps, Dev and Ops, , Agile System Administration, IT Operations*
Intervenção: Identificar áreas de pesquisas científicas sobre práticas *DevOps*.
Palavras Relacionadas: *Continuous Delivery ,Continuous Integrations, Continuous Deployment, Continuous Release, Release Management, Deployment Automation, Infrastructure Management, Infrastructure as a code, automation, ,Virtualization, Cloud, Quality of Software Deployments, Frequent Software Releases, Agile Development, software development cycle, life cycle, Process, Maturity Model, Productivity, Agility, Reliability ,Report.*

O item comparação não foi utilizado pois a pesquisa não tem por objetivo estudos primários comparativos. Os outros itens (resultados e contextos) não foram utilizados pois são atrelados ao item comparação.

Foram definidas 5 questões do tipo classificatória e exploratórias a serem respondidas com o mapeamento sistemático, são elas:

Q.1 Em quais áreas estão concentrados estudos/pesquisas na área de *DevOps*?

Q.2 Onde se concentram as maiores publicações na área de *DevOps* (periódicos e conferências)?

Q.3 Quem são os principais autores da área?

Q.4 Com quais metodologias ágeis o uso de práticas *DevOps* está(ão) relacionada(s)?

Q.5 Quais são as práticas e técnicas em uso nas organizações que utilizam *DevOps*?

Definição da Estratégia de Busca

A construção da *string* de busca nas bibliotecas digitais e engenhos de busca iniciou-se a partir da extração de palavras-chave das questões de pesquisa e dos elementos das estruturas PICOC, no nosso caso, população e intervenção, identificando sinônimos para as palavras chave, usando assim o conector booleano OR para indicar as palavras alternativas e AND para a ligação das palavras chaves.

A realização de um piloto foi de fundamental importância para afinar e definir uma *string* de busca. Foi importante a realização de um piloto para efeito comparativo de resultados que deveriam aparecer de autores e pesquisas conhecidas para avaliar a *string* de busca, bem como para retirar *strings* desnecessárias que não afetam no resultado final.

Com base nestas informações descritas anteriormente, ficou definida a *string* de busca para procurar informações gerais do termo da pesquisa (*DevOps*) ligadas a informações sobre as áreas a serem pesquisadas, suas práticas e seus sinônimos. Assim, a *string* de busca ficou definida de acordo com a seguinte estrutura:

<tema da pesquisa> AND (<areas> <praticas>)

Os termos sinônimos para cada palavra chave da *string* são os que seguem:

Tema da pesquisa: *DevOps, Dev and Ops, Agile System Administration, IT Operations and Devs.*

Áreas/Práticas: *Continuous Delivery, Continuous Integrations, Continuous Deployment, Continuous Release, Baby Steps, Release Management, Deployment, Frequent*

Software Releases, Quality Assurance, Infrastructure Management, Infrastructure as a Code, Automation, Virtualization, Cloud, Agile Development, Software Development Cycle, Lifecycle.

Assim a *string* ficou definida da seguinte forma:

("DevOps" OR "Dev and Ops" OR "Dev Ops" OR "Agile System Administration" OR "Dev and IT Operations") ***AND*** *("Practices" OR "Continuous Delivery" OR "Continuous Integration" OR "Continuous Deployment" OR "Continuous Release" OR "Baby Steps" OR "Release Management" OR "Deployment" OR "Frequent Software Releases" OR "Quality Assurance" OR "Infrastructure Management" OR "Infrastructure as a Code" OR "Automation" OR "Virtualization" OR "Cloud" OR "PaaS" OR "Agile Development" OR "Software Development Cycle" OR "Lifecycle" OR "Process" OR "Maturity Model" OR "Productivity" OR "Agility" OR "Metrics" OR "Report" OR "Skills" OR "Tools")*

Data Sources

O processo utilizado para buscar estudos primários na literatura incluiu buscas automatizadas através de bibliotecas digitais e engenhos de busca, aplicando a *string* de busca definida anteriormente. Os engenhos de buscas diferenciam-se das bibliotecas digitais porque esses possuem artigos científicos publicados de múltiplas fontes e aqueles somente de suas editoras.

Bibliotecas Digitais

- ACM Digital Library (http://portal.acm.org)
- IEEEXplore Digital Library (httt://ieeexplore.ieee.org/)
- Elsevier ScienceDirect (http:// www.sciencedirect.com)

Engenhos de Buscas

- Scopus (http://www.scopus.com/)

- Springer Link (http://www.springerlink.com/)
- Engineering Village (http://www.engineeringvillage2.org)

Também foram feitas buscas manuais em periódicos e conferências sobre engenharia de software com o objetivo de ampliar os resultados e diminuir possíveis viéses na construção das *strings* de buscas automáticas, problemas de indexação de artigos e de não inclusão de palavras chaves nos artigos. Livros e artigos, não incluídos nas buscas automáticas, também foram adicionados no mapeamento a partir do momento do reconhecimento de sua importância para a pesquisa científica e os autores sejam de renome na área.

Conferências

- Agile Development Conference;
- International Conference on Agile Software Development;

Journals

- ACM Transactions on Software Engineering and Methodology;
- IEEE Software;
- IEEE Internet Computing
- IEEE Transactions on Software Engineering;
- Journal of the ACM;
- International Journal of Agile and Extreme Software Development – IJAESD;
- Cutter IT Journal;

Definição dos Critérios de Seleção e Inclusão/Exclusão

Todos os estudos foram avaliados com base nos critérios de inclusão e exclusão. Inicialmente, a seleção dos estudos primários foi feita em 3 fases, são elas:

- Seleção dos estudos primários relevantes resultantes das buscas automáticas de acordo com o critério de inclusão;
- Avaliação dos estudos primários aplicando-se os critérios de inclusão/exclusão;
- Etapas para resolução dos conflitos e seleção final;

Na primeira rodada da seleção foram selecionados artigos com base no **Título** e no ***Abstract***. Para efeito de seleção na primeira rodada, foram selecionados estudos que tinham alguma relação com a área de pesquisa, seja no título ou *abstract* da pesquisa. Dessa forma, os pesquisadores envolvidos realizaram a seleção separadamente e, os conflitos surgidos, foram discutidos em reuniões posteriores. Na segunda rodada, os pesquisadores fizeram uma releitura independente dos estudos com base na **Introdução** e na **Conclusão** dos artigos e aplicaram os critérios assim os critérios de **Inclusão** e **Exclusão** de forma independente de cada pesquisador. Resumidamente, os critérios de inclusão foram aplicados como:

- O Artigo respondia a alguma questão de pesquisa?
- O Artigo pertencia a fontes aceitáveis?
- Estudos que focaram em práticas conhecidas de *DevOps;*
- Estudos que abrangiam alguma área de *DevOps;*
- Relatórios de uso em empresas e artigos da indústria também foram incluídos;

Após a seleção inicial aplicando os critérios de inclusão, foram aplicados os critérios de exclusão, definidos como:

- Se o "artigo" era uma apresentação ou um resumo estendido;
- Se o artigo não tinha relação com *DevOps*;
- Se o artigo não tinha relação com as áreas e/ou práticas do *DevOps*;

Na etapa 3, caso detectado algum conflito entre os pesquisadores, foram definidos critérios para resolver esses conflitos e descrever a seleção final.

Avaliação da Qualidade

Após a aplicação dos critérios de seleção, inclusão e exclusão foram aplicados critérios de qualidade dos estudos com objetivo de selecionar os trabalhos mais relevantes de acordo com os critérios de qualidade definidos no protocolo. A classificação da qualidade dos estudos foi atribuída entre 0% e 100% de acordo com o somatório das notas atribuídas com os critérios de qualidade.

Extração dos Dados e Mapeamento dos Estudos

A extração dos dados foi realizada pelo autor da pesquisa e revisada pelo orientador e co-orientador desta pesquisa. Os estudos que passaram na fase de qualidade também foram analisados pelo autor desta pesquisa, realizando assim a extração estruturada dos dados da publicação, do contexto e das evidências para responder às questões de pesquisa com base no processo sugerido por Cruzes e Dybå (2011).

Basicamente o processo resumido consistiu em fazer uma leitura inicial dos estudos e, posteriormente, a extração dos dados e das evidências. Este estudo conduziu uma síntese e análise temática dos dados, conforme processo recomendado por Cruzes e Dybå (2011).

APÊNDICE B – Survey sobre DevOps e Entrega Contínua

Empresa ID * Digite um ID para sua empresa

Você possui conhecimento sobre os conceitos de DevOps? * *Marcar apenas uma opção.*

a) Possuo muito conhecimento e sou o responsável na organização
b) Possuo conhecimento e participo do processo
c) Possuo algum conhecimento, entendendo o básico
d) Já ouvi falar sobre DevOps, mas possuo quase nenhum conhecimento sobre o assunto
e) Outro:

Em qual área de práticas DevOps você está envolvido em sua organização (Marque todas que se aplicam) ? * *Marque todas que se aplicam.*

a) Continuous Development (metodologia ágil de desenvolvimento)
b) Continuous Integrations (Dev, QA e automação de Builds)
c) Continuous Release, Delivery, Deployment
d) Infrastructure automation
e) Continuous feedback, monitoramento de processos
f) Continuous Improvement (segurança, direcionamento da qualidade, processo….)
g) Outro:

Quais áreas de práticas DevOps sua organização está trabalhando (Marque todas que se aplicam)? * *Marque todas que se aplicam.*

a) Continuous Development (metodologia ágil de desenvolvimento)
b) Continuous Integrations (Dev, QA e automação de Builds)
c) Continuous Release, Delivery, Deployment
d) Infrastructure automation
e) Continuous feedback, monitoramento de processos
f) Continuous Improvement (segurança, direcionamento da qualidade, processo….)
g) Outro:

A equipe de desenvolvimento considera requisitos não funcionais no desenvolvimento de aplicação? * *Marcar apenas uma oval.*

a) Sim
b) Não
c) Considera responsabilidade apenas da equipe de operação
d) Considera responsabilidade de ambas as equipes, mas não considera no desenvolvimento

Na sua opinião, quais as principais razões para que sua organização adote práticas DevOps (escolha até 3)? * *Marque todas que se aplicam.*

a) Entrega contínua com objetivo de melhorar a eficiência do deploy
b) Melhorar a habilidade de monitorar, alertar e auditar mudanças no ambiente de produção
c) Obter transparencia entre a qualidade da aplicação e a performance da rede
d) Ajudar os desenvolvedores a entenderem o ambiente de produção
e) Qualidade
f) Melhorar a resiliencia e recuperação de desastre
g) Automação pervasiva
h) Ajudar a equipe de operação a entender os desafios do time de desenvolvimento

i) Outro:

Na sua opinião, a partir da adoção de práticas DevOps, quais dessas práticas influenciou positivamente o nível de eficiência na entrega do software ? * *Marque todas que se aplicam.*

a) Continuous Development (metodologia ágil de desenvolvimento)
b) Continuous Integrations (Dev, QA e automação de Builds)
c) Continuous Release, Delivery, Deployment
d) Infrastructure automation
e) Continuous feedback, monitoramento de processos
f) Continuous Improvement (segurança, direcionamento da qualidade, processo….)
g) Outro:

Na sua opinião, quais são os tipos de ferramentas necessárias para obter sucesso na implantação de DevOps (marque todas que se aplicam)? * *Marque todas que se aplicam.*

a) Ferramentas de controle de versão
b) Ferramentas de gerência de configuração
c) Ferramentas de gerência de build
d) Ferramentas de gerenciamento de mudanças
e) Ferramentas de testes
f) Ferramentas de Integração
g) Ferramentas de automação (puppet, chef)
h) Outro:

Sua organização trabalha com time Cross-Functional (Todos são responsáveis pelo processo de entrega, qualquer membro do time pode modificar qualquer parte do sistema)? * *Marcar apenas uma oval.*

a) Sim
b) Não
c) Somente em alguns projetos

Como está o uso da práticas de Continuous Delivery na sua organização?

***** *Marcar apenas uma oval.*

a) Usando a prática em todos os projetos
b) Usando a prática somente em alguns projetos
c) Planejando em qual projeto implantar
d) Não usando, mas desejo implantar em breve
e) Fazendo pesquisas a respeito
f) Nenhum planejamento sobre o uso
g) Outro:

Na sua opnião quais as áreas de sua organização que estão prontas para alcançar a adoção de continuous delivery ?

***** 1 Não está pronto para o uso e 5 Está totalmente pronta para o uso

Marcar apenas uma oval por linha.

	Pronto	Em implantação	Sem planejamento
Chefia/Direcao			
Desenvolvimento			
Operação			
Processo			
Aspecto Cultural			

Quantos porcento de testes automáticos, em média, sua equipe utiliza nas fases do processo de deploy?

* Engloba-se nessa pergunta todas as fases do processo de deploy da organização *Marcar apenas uma oval.*

de 1% ~ 5%
de 5% ~ 19%
de 20% ~ 40%
de 41% ~ 60%
de 60% ~ 80%
de 80% ~ 100%

Na escala de 1 a 5, em sua opinião, quais tipos de testes sua organização executa de forma manual?

* 1 para testes manuais e 5 para Testes completamente automatizados

Marcar apenas uma oval por linha.

	1	2	3	4	5
Testes de Aceitação					
Testes de Unidade					
Testes Exploratórios					
Testes de Perfomance					
Testes de Carga					
Testes de Simulação					
Testes de Revisão de Código					
Testes de Usabilidade					
Testes de Fumaça					
Testes de Integração					

Testes de Qualidade

Sua organização faz deploy de software em máquinas virtuais para ambientes diferentes (testes, desenvolvimento) com esses ambientes controlados sob controle versão? * *Marcar apenas uma oval.*

a) Sim, os ambientes são criados de maneira flexível e ágil respondendo assim às necessidades de negócio, além de possuir um ambiente altamente controlado com versões de software similares ou iguais às versões de produção, executando os testes automáticos em todas as etapas do deploy
b) Sim, os ambientes são criados de maneira flexível e ágil, porém as versões de software nem sempre são controlados ou simulando um ambiente real de produção
c) Sim, porém é necessário total intervenção da equipe de operação para executar um deploy em qualquer fase do processo
d) Sim, porém os ambientes criados não são sempre iguais ao ambiente de produção
e) Nunca
f) Outro:

Sua organização é capaz de entregar a mesma aplicação para testes a times diferentes em ambientes distintos e controlados, paralelamente ? * *Marcar apenas uma oval.*

a) Sim, em todos os projetos
b) Em nenhum projeto
c) Somente em alguns projetos

O time de desenvolvimento ou operação faz deploy de software em serviços de nuvem (PaaS ou IaaS)? * *Marcar apenas uma oval.*

a) Sim, mesclando serviços de nuvem pública e privada
b) Sim, somente serviços de nuvem pública
c) Sim, somente serviços de nuvem privada
d) Não faz uso de nuvem
e) Outro:

Sua organização fornece visibilidade de seu pipeline mostrando os estágios do mesmo (build, deployment, test e release) e a progressão das builds a todos os membros das equipes? * *Marcar apenas uma oval.*

a) Sim, em todos os projetos
b) Somente em alguns projetos
c) Em nenhum projeto

Marque as opções que incluem tarefas de seu processo de deploy: * *Marque todas que se aplicam.*

a) O processo de build é separado em estágios
b) Detecção de problemas de perfomance
c) Detecção de problemas de segurança
d) Detecção de problemas de usabilidade
e) Visibilidade do processo para qualquer pessoa da organização
f) Uma trilha de auditoria completa
g) Verificações automáticas
h) Verificações manuais
i) Nenhum das anteriores
j) Outro:

Sua organização implementa processos roolback rápido, repetível e automático (sem necessidade de codificação de desenvolvedores) caso algum bug seja

encontrado em produção? * *Marcar apenas uma oval.*

a) Sim, em todos os projetos
b) Sim, em alguns projetos
c) Em nenhum projeto

Sua organização utiliza técnicas como Canary Release ou Toggled Features em seu processo de deploy? * *Marcar apenas uma oval.*

a) Sim, ambas
b) Apenas a técnica de Canary Release
c) Apenas Toggled Features
d) Não
e) Não, esses conceitos são desconhecidos em minha organização

Sua organização utiliza técnicas de Blue-Green deployments ?

* *Marcar apenas uma oval.*

a) Sim em todos os projetos
b) Somente em alguns projetos
c) Não
d) Não, esses conceitos são desconhecidos em minha organização

Sua organização utiliza sistema de tracking ou gerenciamento de controle de mudanças integrado com seu servidor de integração?

* *Marcar apenas uma oval.*

a) Sim, os release notes são gerados automaticamente a partir da ferramenta de controle de mudanças permitindo a organização possuir rastreabilidade e auditoria
b) Sim, porém a rastreabilidade ocorre manualmente

c) Não, mas a organização utiliza ferramentas de controle de mudanças
d) Não, a organização não possui ferramentas de controle de mudanças

Como você faz a build de sua aplicação?

* *Marcar apenas uma oval.*

a) Script ou Shell
b) Linguagem de programação
c) Frameworks de Configuração
d) Servidores de Integração
e) Através da própria IDE
f) Outro:

Sua organização possui um servidor de Build centralizado?

* *Marcar apenas uma oval.*

a) Sim
b) Não
c) Em implantação

Sua equipe de operação possui knowhow para escrever e manter scripts de integração e automação de deployments?

* *Marcar apenas uma oval.*

a) Sim e desenvolvem seus scripts
b) Sim, mas são de responsabilidade dos desenvolvedores
c) A organização tem uma equipe separada para esse fim
d) Não

As mudanças no ambiente e na infraestrutura (scripts, database scripts, arquivos de configuração, deploy scripts, java properties, etc) estão sob controle

de versão?

* *Marcar apenas uma oval.*

a) Sim, todas as mundaças
b) Sim, em sua grande maioria
c) Sim, parcialmente
d) Não

Sua organização utiliza ferramentas de gerenciamento de configuração de infraestrutura para controlar as versões de ambiente (Chef, Puppet, CFEngine)?

* *Marcar apenas uma oval.*

a) Sim, equipe de desenvolvimento e equipe de operação trabalhando em conjunto e construindo automaticamente um ambiente controlado, consistente e reproduzível em todos os projetos
b) Sim, equipe de desenvolvimento e equipe de operação trabalhando em conjunto e construindo automaticamente um ambiente controlado, consistente e reproduzível em alguns projetos
c) Apenas equipe de operação em todos os projetos
d) Apenas equipe de operação em alguns projetos
e) Não

Caso sua organização deseje fazer um deploy em um novo ambiente para desenvolvimento, testes ou produção, como é feita a montagem desse ambiente (marque todas que se aplicam)? * *Marque todas que se aplicam.*

a) Através de Virtual machine players (VMWare player, VirtualBox, Virtual PC etc)
b) Através de máquinas virtuais gerenciado de forma self-service pelo requisitante através de Hypervisor (ESXi, vSphere, KVM etc)
c) Através de máquinas virtuais solicitando à equipe de operações através de Hypervisor (ESXi, vSphere, KVM etc)

d) Através de Public Cloud
e) Através de Private Cloud
f) Sem virtualização requisitando à equipe de operações hardware, S.O. e outros recursos

APÊNDICE C – Distribuição da Qualidade dos Estudos

Pesquisador	ID do Estudo	Resultados Claros	Estudo Empírico	Representação	Replicável	Participantes	Teórico	Referências	Resultado
ID2, ID3	ID001	1	1	0	1	0	0	N/A	50.00%
ID2, ID3	ID002	1	1	1	0	0	0	N/A	37.50%
ID2, ID3	ID003	1	1	0	0	0	0	N/A	25.00%
ID2, ID3	ID005	0	0	N/A	N/A	N/A	1	0	0.00%
ID2, ID3	ID006	1	1	1	1	1	0	N/A	75.00%
ID2, ID3	ID007	1	1	0	1	0	0	N/A	50.00%
ID2, ID3	ID008	1	1	1	1	1	0	N/A	87.50%
ID2, ID3	ID010	0	0	N/A	N/A	N/A	1	0	0.00%
ID2, ID3	ID013	0	1	1	0	1	0	N/A	25.00%
ID2, ID3	ID017	0	1	1	0	1	0	N/A	37.50%
ID2, ID3	ID018	0	1	0	0	0	0	N/A	0.00%
ID2, ID3	ID020	0	1	1	0	0	0	N/A	12.50%
ID2, ID3	ID021	1	1	1	1	0	0	N/A	62.50%
ID2, ID3	ID026	1	0	N/A	N/A	N/A	1	0	50.00%
ID2, ID3	ID030	1	1	1	1	0	0	N/A	62.50%
ID2, ID3	ID034	1	1	1	1	0	0	N/A	62.50%
ID2, ID3	ID036	1	1	1	0	0	0	N/A	37.50%
ID2, ID3	ID042	1	0	N/A	N/A	N/A	1	0	50.00%
ID2, ID3	ID048	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID049	1	1	1	1	0	0	N/A	75.00%
ID2, ID3	ID050	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID051	1	1	1	1	0	0	N/A	75.00%
ID2, ID3	ID052	0	1	1	0	0	0	N/A	12.50%
ID2, ID3	ID054	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID055	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID056	1	1	1	1	1	0	N/A	75.00%
ID2, ID3	ID057	1	1	1	1	0	0	N/A	62.50%
ID2, ID3	ID058	1	1	0	1	1	0	N/A	62.50%
ID2, ID3	ID059	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID060	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID061	1	1	0	1	0	0	N/A	50.00%
ID2, ID3	ID062	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID063	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID065	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID067	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID070	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID071	1	0	N/A	N/A	N/A	1	1	100.00%

ID2, ID3	ID072	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID074	1	0	N/A	N/A	N/A	1	1	100.00%
ID2, ID3	ID076	1	0	N/A	N/A	N/A	1	1	100.00%