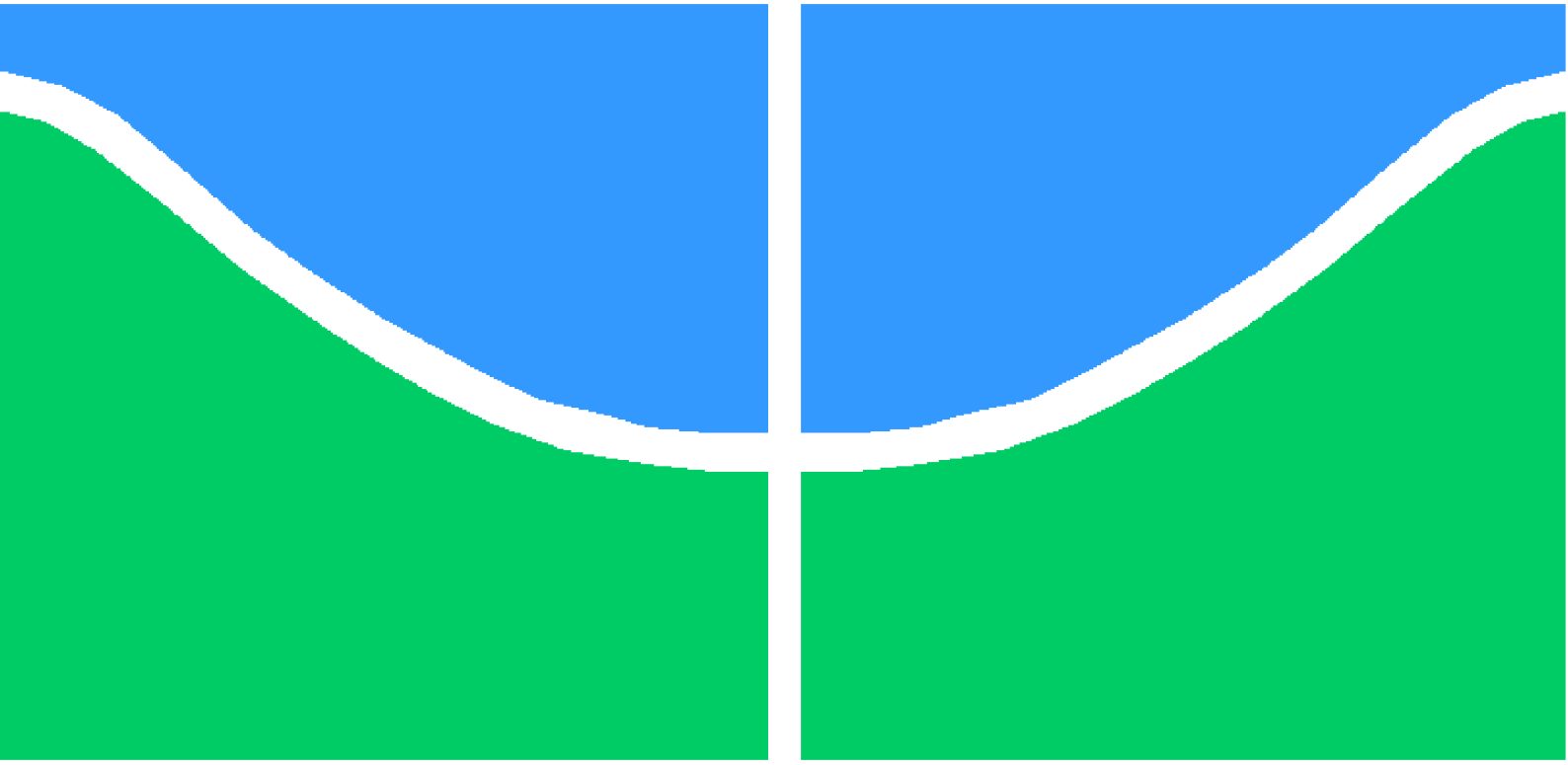

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de *Software*

Entrega Contínua de *Software*: Um estudo de caso

Autor: Sabryna de Sousa Pessôa
Orientador: Prof. Dra. Carla Silva Rocha Aguiar

Brasília, DF
2018

Sabryna de Sousa Pessôa

Entrega Contínua de *Software*: Um estudo de caso

Monografia submetida ao curso de graduação em Engenharia de *Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de *Software*.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dra. Carla Silva Rocha Aguiar

Brasília, DF

2018

Sabryna de Sousa Pessôa

Entrega Contínua de *Software*: Um estudo de caso/ Sabryna de Sousa Pessôa. – Brasília, DF, 2018-

55 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dra. Carla Silva Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Entrega contínua. 2. *DevOps*. I. Prof. Dra. Carla Silva Rocha Aguiar. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Entrega Contínua de *Software*: Um estudo de caso

CDU 02:141:005.6

Sabryna de Sousa Pessôa

Entrega Contínua de *Software*: Um estudo de caso

Monografia submetida ao curso de graduação em Engenharia de *Software* da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de *Software*.

Trabalho aprovado. Brasília, DF, 14 de dezembro de 2018 – Data da aprovação do trabalho:

Prof. Dra. Carla Silva Rocha Aguiar
Orientador

Prof. Me. Hilmer Nere
Convidado 1

Prof. Me. Ricardo Poppi
Convidado 2

Brasília, DF
2018

Resumo

Entrega contínua é um conjunto de práticas que buscam manter o código estável, com boa qualidade e frequência nas implantações do sistema em produção. O objetivo deste trabalho de conclusão é fazer uma análise qualitativa sobre os principais aspectos que impactam a entrega contínua de *software*. Para isto, será feito um estudo de caso em três órgãos do governo, tendo-se o objetivo de analisar estas práticas e avaliar a maturidade do processo adotado. Dentro dos resultados, pode-se observar que a visão dos gestores sobre a qualidade do processo de entrega, difere-se do que realmente acontece. E que na maioria dos órgãos não realizam de fato uma entrega contínua, já que faltam com qualidade, frequência ou automatização de partes do processo.

Palavras-chaves: entrega contínua. continuous delivery. pipeline. deploy. devops.

Abstract

Continuous delivery is a set of practices that seek to keep the code stable, with good quality and frequency in the deployments of the system in production. The purpose of this work of completion is to make a qualitative analysis on the main aspects that imply the continuous delivery of software. For this, a case study will be done in three government organs, with the objective of analyzing these practices and evaluating the maturity of the process adopted. Within the results, it can be observed that the managers' view of the quality of the delivery process differs from what actually happens. And most of the organs do not actually perform a continuous delivery, since they lack quality, frequency or automation of parts of the process.

Key-words: continuous delivery. pipeline. devops.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Contextualização

Produzir um *software* em pouco tempo e de boa qualidade, minimiza os custos e tempo para o cliente. Dentro deste cenário, surge então a entrega contínua, cujo o objetivo é fazer com que lançamentos de *software* ocorram com uma maior frequência no ambiente de produção. Entretanto, não basta aumentar a frequência de entregas, a aplicação produzida deve ter sempre um código estável e bem testado.

Para implementar de fato o uso desta metodologia é necessário a utilização de diversas práticas de desenvolvimento, qualidade, DevOps. Integração contínua e *deploy* contínuo são importantes práticas no processo de entrega.

1.2 Objetivo

O objetivo desta pesquisa é fazer uma análise qualitativa sobre os principais aspectos que impactam a entrega contínua de *software*. O estudo de caso será feito utilizando órgãos públicos.

1.2.1 Objetivos Específicos

- Identificar as práticas mais utilizadas que contribuem para a entrega contínua
- Analisar as etapas do processo, juntamente com a práticas de qualidade e o tempo de execução
- Avaliar a maturidade do processo

1.3 Estrutura do documento

O documento está divido em algumas seções. Na seção 2 encontra-se a fundamentação teórica que aborda os principais fatores que contribuem para uma entrega contíua.E na seção 3 é descrito o estudo de caso, onde é analisada a maturidade do fluxo de entrega contínua de *software* em três órgãos governamentais.

2 Fundamentação Teórica

2.1 Entrega Contínua

Entrega contínua aborda todo o fluxo do desenvolvimento de um *software*, desde o código, testes, até a implantação em produção. Ela visa simplificar o lançamento através de ciclos de *feedbacks* mais curtos entre desenvolvedores e *stakeholders* (KRUSCHE; ALPEROWITZ, 2014).

Empresas que adotam um modelo de entrega contínua, devem manter o *software* estável para que novas alterações possam ser lançadas em qualquer momento. Estes lançamentos regulares ajudam na melhoria do produto e satisfação do cliente. O controle do processo de entrega contribui para a minimização dos riscos, onde os erros podem ser detectados precocemente e corrigidos logo em seguida (KRUSCHE; BRUEGGE, 2017).

2.1.1 Processos

Pereira *et al.* define algumas etapas de um processo para realizar entrega contínua (SAMARAWICKRAMA; PERERA, 2017).

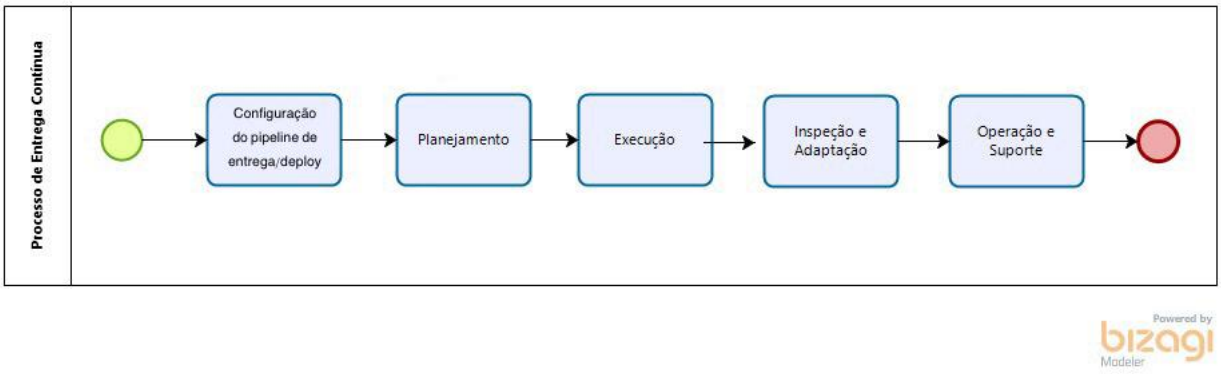

Figura 1 – Processo de entrega contínua

1. **Configuração do *pipeline* de entrega/*deploy*:** Nesta etapa são criados ambientes idênticos, sendo eles de teste, homologação e produção. Também são definidas as ferramentas de automação para serem utilizadas.

2. **Planejamento:** Esta atividade refere-se ao planejamento de uma *sprint*, onde são definidas as histórias para serem desenvolvidas. Um fator importante do ponto de

vista de entrega contínua, é a alta prioridade para o tempo dedicado a manter o *software* estável em produção.

3. **Execução:** Esta atividade do processo tem como foco a qualidade do produto que está sendo construído, portanto o código desenvolvido deve ter bons testes para não gerar problemas futuros. O desenvolvedor é responsável pelos *commits* e deve se preocupar com a construção da *build*. Nesta da atividade são utilizadas as ferramentas de verificação de qualidade.

4. **Inspeção e Adaptação:** Nesta etapa são executados os testes automatizados, integração contínua. Todos os testes devem estar passando e a *build* deve estar sendo construída corretamente para ir para a próxima etapa.

5. **Operação e Suporte:** Nesta atividade são realizadas as implantações automatizadas, visto que primeiro são feitas em um ambiente de homologação para testar. Caso esteja tudo funcionando corretamente, a implantação se estende para o ambiente de produção. Caso contrário, deve-se definir um tempo para liberar em produção. Erros no desempenho da aplicação devem ser resolvidos em uma próxima *sprint*. Já os erros críticos possuem uma prioridade maior e devem ser resolvidos na *sprint atual*. E os erros médios e baixos podem ser adicionados ao *backlog* para serem corrigidos em *sprints* futuras.

2.1.2 Indicadores

Rahman *et al.*, analisaram as características de entrega contínua em 19 empresas de *software*, empresas como *Facebook, Netflix, GitHub, Pinterest*, entre outras. A partir desta análise, elencou-se 11 práticas que evidenciam tal uso. Estas práticas estão listadas na tabela 1 (RAHMAN et al., 2015).

Tabela 1 – Indicadores de Entrega Contínua

Uso do repositório	Com uso de *branches* no Git, ferramenta de versionamento, é possível organizar de tal maneira que só determinada branch seja colocada em produção. Isto facilita o processo de mudanças de código e mesclagem de branches, possibilitando manter uma branch estável, com todos os testes passando, indicadores de qualidade bons e pronta para ser colocada em produção.
Intercomunicação	Refere-se a comunicação entre toda a equipe responsável pela produção do software. Através do compartilhamento de conhecimento, informações de implantação, é possível alcançar uma boa eficiência da equipe, contribuindo desta maneira para o crescimento de todos.
Monitoramento	Consiste em monitorar todo o processo de entrega contínua através dos servidores de integração, da coleta de métricas, contribuindo para a identificação de possíveis erros de implementação, bem como as possíveis causas de falhas, *bugs*.
Shepherding Changes	Tem o objetivo de manter a responsabilidade dos desenvolvedores em todas as etapas de entrega contínua, ou seja, desde o desenvolvimento, testes, implantação, correção de erros e possíveis mudanças após a implantação.
Testes automatizados	O código desenvolvido é testado automaticamente sempre que um novo *commit* é feito. Essa é uma prática auxilia os desenvolvedores na detecção de erros, falhas, *bugs* no software.
Revisão de código	É feita geralmente ao final de uma *sprint*, onde os desenvolvedores compartilham o código feito com os outros desenvolvedores da equipe. Isto ajuda a minimizar os erros cometidos na programação, além de contribuir com o conhecimento coletivo.
Implantação automatizada	Com o uso de *scripts*, o *software* desenvolvido é implantado automaticamente para os usuários finais, sem o esforço manual.
Feature flag	Consiste em uma técnica que executa o código através de ramificações. Cada ramificação é executada após uma condição ser aceita. Ou seja, se uma parte do *software* está com defeito, apenas essa parte necessita ser trocada.
Lançamento no escuro	É uma prática para implantar mudanças no *software*, mantendo ele funcional e com alterações ocultas para o usuário. O que motiva a utilização desta prática é o *feedback* sobre qualidade e desempenho, sem deixar que o usuário final perceba.
Encenação	Duas técnicas podem ser utilizadas (Dog-fooding e Implantação gradual). Savor *et al.* cita que o Facebook utiliza dessa técnica *Dog-fooding* com um grupo de usuários internos antes de lançar as modificações do site, aplicativo com os usuários externos. Além do uso da outra técnica também, em que ele lança para seus usuários gradualmente, de grupos em grupos. Uma das vantagens de utilizar essas técnicas é a obtenção do *feedback* dos usuários aos se depararem com as mudanças de *software* antes que sejam implementadas em totalidade (SAVOR et al., 2016).
Comunicação com o usuário final	Utilizam-se de redes sociais, fóruns, pesquisas, entre outros meios para obter um *feedback* do usuário sobre o *software*.

Além destas práticas, Samarawickrama *et al.* cita outras práticas que contribuem para um bom desempenho no processo de entrega contínua. Estas práticas encontram-se na tabela 2 (SAMARAWICKRAMA; PERERA, 2017).

Tabela 2 – Indicadores

Automatização da construção	Através dos servidores de integração contínua, a cada *commit* a *build* é construída automaticamente. Esta prática vale para os *commits* feitos em qualquer *branch* do repositório.
Construções quebradas possuem prioridade	Erros de compilação, falhas na *build* são resolvidos com prioridade para que não prejudiquem *commits* posteriores.

2.1.3 Vantagens

Segundo Savor *et al.*, existem algumas vantagens ao se utilizar entrega contínua de *software*, como mais mudanças incrementais, *feedback* mais rápido e respostas mais rápidas para ameaças de vulnerabilidade (SAVOR et al., 2016).

Para Rahman *et al.* as vantagens são: a melhor satisfação do cliente, melhor qualidade de *software* e menor esforço de desenvolvimento, já que partes do processo que seriam feitas de maneira manual, são realizadas de maneira automatizada (RAHMAN et al., 2015).

Soni acredita que a alta disponibilidade de recursos e o baixo tempo de inatividade são importantes vantagens (SONI, 2015).

Já para Siqueira *et al.* o tempo acelerado de comercialização, construção do produto certo, produtividade, melhorias de eficiência, lançamentos estáveis e maior satisfação do cliente são importantes vantagens (SIQUEIRA et al., 2018).

2.1.4 Desafios

Um dos desafios é o que acontece na transição de um processo de desenvolvimento tradicional para um processo de entrega contínua. As maiores barreiras são no que dizem respeito a realizar mudanças, já que é preciso que as pessoas se adaptem a nova metodologia. Na pesquisa de Rahman *et al.* pôde-se evidenciar em algumas empresas a falta de clareza no processo, problemas de rede, atualização (RAHMAN et al., 2015).

Falhas de compilação é outro desafio que pode ser encontrado ao lidar com entrega contínua. Podendo acarretar na baixa produtividade dos desenvolvedores, já que os mesmos deverão resolver estas falhas antes de dar continuidade ao processo (VASSALLO; ZAMPETTI; ROMANO, 2016).

2.2 DevOps

O termo DevOps vem da junção de "Dev"e "Ops"que vem de "*Developers*", desenvolvedores e "Ops"de "*Operation*", operação. Ou seja, o termo refere-se as práticas em que a equipe de desenvolvimento e operações se utilizam para entregar um *software* de qualidade, de maneira rápida e confiável. DevOps incentiva a automação de mudança, configuração, liberação de um *sofwtare* e consequentemente a sua entrega contínua (PERERA; SILVA; PERERA, 2017).

Segundo (FARROHA; FARROHA, 2014), o objetivo estratégico do DevOps é maximizar o resultado do investimento feito para a construção do *software* e garantir que os clientes recebam continuamente um serviço com maior qualidade e que satisfaça suas necessidades. Para ele os objetivos gerais do DevOps são:

- Entregar valor de negócio mensurável através da prestação de serviços contínuos e de alta qualidade;
- Enfatizar a simplicidade e agilidade nas aréas de tecnologia, processos e fatores humanos;
- Quebrar barreiras de divisão entre desenvolvimento e operações;

2.3 Ferramentas

Boa parte do processo de entrega contínua de um *software* utiliza-se de automação. Soni cita algumas ferramentas que contribuem para esse processo de desenvolvimento (SONI, 2015).

- Ambiente de desenvolvimento integrado: ambiente para desenvolvimento de *softwares*. Exemplo: Eclipse.
- Repositório de código fonte: utilizado para manter o código fonte em um único lugar. Exemplos: SVN, Git.
- Ferramentas de construção: ferramentas para automação do processo de compilação. Exemplos: Avn, Maven.
- Servidor de integração contínua: utilizado para realizar a construção automatizada e notificar sobre o *status* da construção. Exemplos: Jenkins, Travis.
- Testes automatizados: ferramenta para execução dos testes automaticamente. Exemplo: Junit.

- Ferramenta de análise de código estático: verifica a qualidade do código produzido, identifica possíveis *bugs*, complexidade, duplicação. Exemplos: Sonar, FindBug.
- Repositório binário: utilizado no versionamento dos arquivos binários. Exemplo: Artifactory.
- Infraestrutura: ambiente em nuvem. Exemplos: Amazon Web Service, VMware vSphere.
- Configuração do ambiente em tempo de execução: ferramenta que possui várias receitas para configurar o ambiente em tempo de execução. Exemplo: Chef.
- Ferramenta de implantação: utilizada para a implantação em diferentes ambientes. Exemplos: *Plugins* de implantação, *Shell Scripts*.
- Monitoramento: utilizado para monitorar vários ambientes e receber notificação de falhas. Exemplo: Jenkins Plugins.

2.4 Maturidade

Virtanen *et al.* fez um estudo de processos de entrega contínua de empresas do mercado e encontrou algumas que estabeleceram níveis de maturidade para o processo. Tomando como base o CMMI, a *Forrester Consulting* definiu características de maturidade para cada nível (Figura: 2), onde aplicou pra classificar os projetos produzidos pela empresa (VIRTANEN et al., 2014).

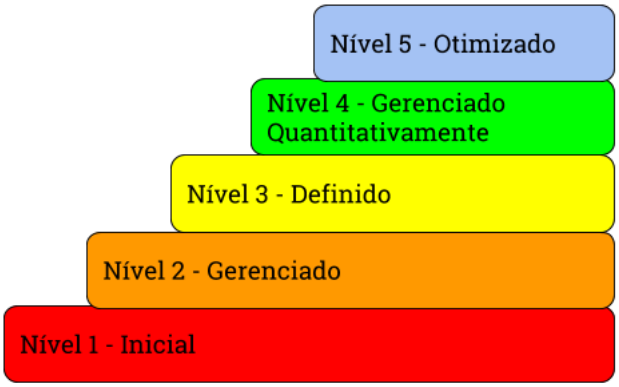

Figura 2 – Níveis de maturidade CMMI

No primeiro nível é necessário apenas uma entrega de lançamento. Já para ser considerado um projeto com nível de maturidade Gerenciado, são necessários lançamentos planejados e um processo básico. No nível Definido, deve-se ter um ritmo regular de entregas. Para o quarto nível o *software* deve estar sempre pronto para ser utilizado, ou

seja, são feitos lançamentos com frequência maior e de ótima qualidade. O quinto nível conta com um processo de lançamentos otimizados.

Outra empresa citada por Virtanen *et al.* definiu a maturidade da entrega contínua nos projetos feitos pela sua empresa conforme na tabela 3 (VIRTANEN et al., 2014).

Tabela 3 – Níveis de maturidade no processo de entrega contínua

Nível	Testes automatizados	Qualidade de Implantação	Deploy	Tempo
1	Testes unitários	Teste de aceitação antes de ir para produção	Controle de versão	>1 mês
2	Testes de integração	Ambiente correspondente ao em produção Processo documentado	Integração contínua a cada novo commit	2 - 4 semanas
3	Testes End-to-end	Métodos de testes de exploração Verificação de projeto baseada em protótipo	Scripts de implantação para o ambiente de qualidade Configuração é documentada Controle das migrações do banco de dados	1 - 2 semanas
4	Relatório de testes Testes de performance Testes de segurança	Política de defeito zero Análise da causa raiz	Controle de versão para ambientes separados Configuração de ambiente em um repositório separado	1 - 5 dias
5	Testes rodando em paralelo	Medidas de qualidade diretas pelos usuários	Entrega contínua em produção Procedimento para promover builds	<1 dia

2.5 Trabalhos Relacionados

Dentro do contexto de entrega contínua de *software*, existem estudos que foram feitos e serviram de base para este trabalho de conclusão. Dentre eles, três contribuiram para a condução desta pesquisa, um feito no Brasil e os outros dois no exterior.

O publicado nos EUA, mais precisamente na Universidade Estadual da Carolina do Norte, analisou as práticas de *software* em 19 empresas grandes como Facebook, Netflix, Pinterest, Github, entre outras. Rahman *et al.* utilizaram da pesquisa do Google para procurarem artefatos que falassem sobre o processo de desenvolvimento destas empresas. Após um estudo, definiram as 11 práticas mais comuns e verificaram se as empresas adotavam ao menos uma destas. Além de usarem a pesquisa do Google, entraram em

contato com alguns dos autores dos artefatos para extraírem mais informações (RAHMAN et al., 2015).

Um outro artigo de base fez uma análise qualitativa a cerca das principais práticas utilizadas no DevOps, e para isto utilizou-se de uma metodologia inspirada na *Grounded Theory*, cujo o objetivo era registrar os conhecimentos adquiridos durante a entrevista e organizá-los em palavras-chaves (A. . . , 2017).

Outro artigo que guiou este trabalho foi publicado por pessoas que fizeram parte do projeto de modernização do Portal do Software Público Brasileiro. Sendo este fruto de uma parceria entre a Universidade de Brasília e a Universidade de São Paulo, onde Siqueira *et al.* relatam a experiência ao se adotar entrega contínua e de como a adoção desta prática contribuiu para que o cliente tivesse uma maior confiança do trabalho que estava sendo desenvolvido (SIQUEIRA et al., 2018).

3 Estudo de caso

Este estudo de caso engloba três órgãos do governo que possuem um amplo setor de TI e que portanto terão nomes fictícios na apresentação deste trabalho, visto que o objetivo é analisar como determinadas práticas de *software* contribuem para uma entrega contínua.

Em um destes órgãos foram analisadas duas equipes: uma que realiza entrega contínua e outra que não realiza, para que ao final pudesse obter um comparativo tanto externo dos três órgãos, quanto interno do próprio órgão que possui equipes com abordagens diferentes.

3.1 Metodologia

A metodologia de pesquisa utilizada neste estudo é do tipo qualitativa e quantitativa. Para isto, a coleta de dados difunde-se em três abordagens que serão descritas em conseguinte: questionário, entrevista exploratória e extração de dados.

3.1.1 Entrevista exploratória

A metodologia utilizada para a entrevista foi baseada na adotada pelo Erich, onde ele utiliza o método descrito por Kvale para identificar práticas de DevOps (A. . . , 2017). Aplicando para o contexto deste trabalho, a entrevista foi feita sob a ótica de três níveis de abordagem.

No primeiro nível, o objetivo é conhecer os projetos feitos pelo setor entrevistado e a metodologia adotada na execução, a fim de que pudesse obter uma visão geral de como funciona a organização como um todo, tendo em vista que aspectos organizacionais influenciam na forma como uma equipe de *software* realiza seu trabalho.

O segundo nível adentra em perguntas mais específicas, onde busca-se entender o *pipeline* de entrega contínua, o principal objeto de estudo desta pesquisa. São perguntas relacionadas as etapas do processo, as ferramentas utilizadas, a qualidade, ao detalhamento de como tudo acontece.

No terceiro nível, tem-se o objetivo de entender características que diferenciam todo o processo, sendo elas o monitoramento de implantação, qualidade e a frequência com que realizam a entrega contínua.

A partir da definição do que esperava ser encontrado em cada nível da entrevista, definiu-se um roteiro apenas com os objetivos descritos acima, sem perguntas pré-

definidas. Isto deve-se ao fato de que cada organização se comporta de um jeito. Entretanto, as perguntas poderiam ser alteradas, mas de forma que não atrapalhasse os objetivos de cada nível.

3.1.2 Questionário

A construção do questionário teve sua base nos principais pontos abordados no referencial teórico deste trabalho, pontos no que se referem ao processo de entrega, as práticas e as experiências ao se adotar um método assim. As perguntas realizadas nele foram elaboradas com o objetivo de que os desenvolvedores pudessem responder sob o seu ponto de vista do funcionamento da entrega em seu setor de atuação. As perguntas abordaram o tanto conhecimento individual, quanto em equipe. O questionário foi feito através do *Google Forms* e encontra-se no Apêndice B.

3.1.3 Extração de dados

Para conseguir os dados referentes as entregas, utilizou-se da extração dos dados a partir do git ou do fornecimento em formato de planilha pelo setor entrevistado. Buscou-se extrair as datas dos dias em que as *releases* foram implantadas em produção no intervalo dos últimos seis meses aproximadamente, ocorrendo do período de 01/05 a 14/10 no órgão 1, 10/04 a 25/10 no órgão 2 e 10/04 a 05/11 no órgão 3, isto através das *tags* de implantação. Outro fator importante, é que para este estudo, os dados foram extraídos do sistema principal desenvolvido pelo setor entrevistado, ou seja, mesmo no setor que mantém muitos sistemas, a análise e resultados foram retirados de um único sistema.

3.2 Resultados

Serão apresentados nos tópicos seguintes os resultados da pequisa encontrados pela técnica do questionário e entrevista exploratória. O gráfico referente ao tempo de entrega será mostrado na seção de análise.

Para melhor entendimento, o resumo sobre as entrevistas realizadas foca nos principais pontos abordados: medição, monitoramento, DevOps, frequência, automação, garantia de qualidade e colaboração. Para definir as palavras-chaves, foi utilizado uma das técnicas utilizadas na *Grounded Theory*. Estas palavras foram escolhidas por serem os termos mais encontrados ao se falar de entrega contínua e a entrevista foi conduzida de forma a englobar estes termos, já que seriam os principais aspectos a serem analisados neste estudo.

O pipeline da figura 3 representa um fluxo comum de organizações que utilizam entrega contínua de *software*. Ele servirá como base para realizar o comparativo das

práticas adotadas na seção de análise.

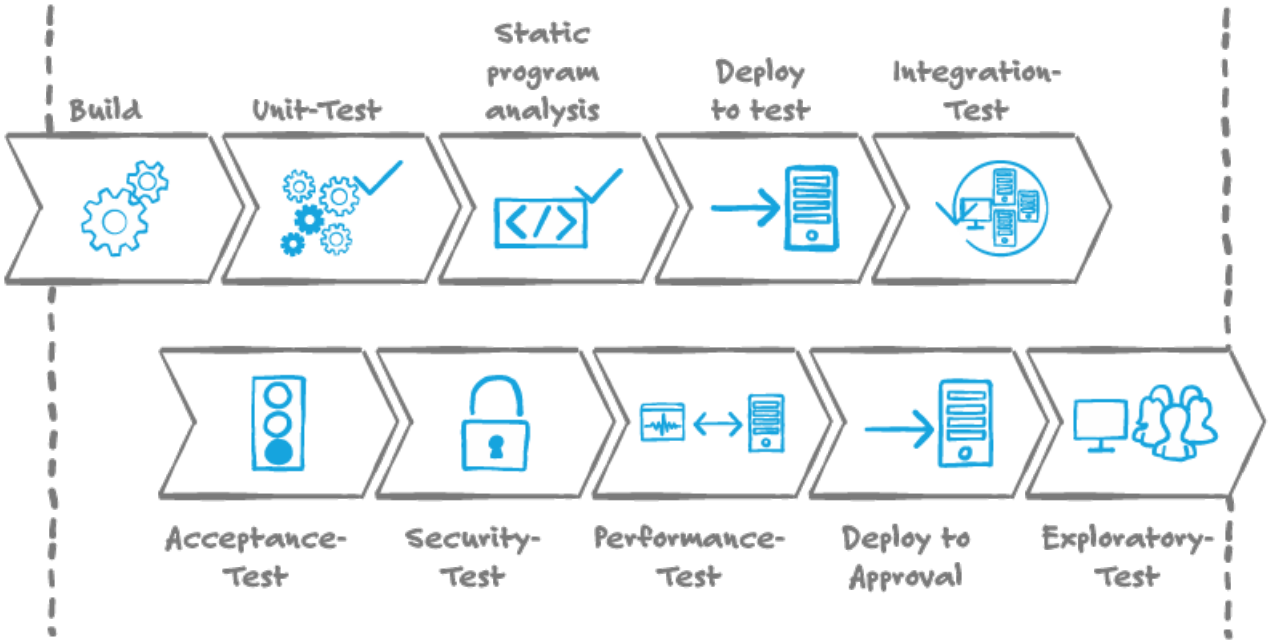

Figura 3 – Pipeline de entrega contínua

3.2.1 Órgão 1

Este órgão possui um setor amplo de tecnologia da informação, nele são desenvolvidos e mantidos cerca de 50 sistemas que auxiliam no funcionamento do órgão como um todo. Portanto, são inúmeros sistemas com funcionalidades diversas, sendo que apenas os projetos mais recentes se aproximam de um fluxo de entrega contínua. Os ditos sistemas legados executam um fluxo de entrega diferente, devido a algumas limitações. A coleta de dados sobre este órgão foi feita em um setor que cuida dos sistemas administrativos, sendo em sua maioria sistemas recentes e com a execução de boas práticas de engenharia de *software*. A tabela 3.2.1 contém os dados extraídos da entrevista feita com o gestor no que se refere as sete características citadas anteriormente.

Tabela 4 – Dados extraídos da entrevista - Órgão 1

DevOps	Um setor diferente do entrevistado é responsável pela infraestrutura dos sistemas em desenvolvimento. Mas dentro da própria equipe um integrante realiza configuração do jenkins, docker, e este inclusive faz parte de um comitê da organização responsável pelo DevOps de todos os sistemas.
Colaboração	Os integrantes da equipe desenvolvem, testam e cuidam da qualidade do projeto. Existe uma comunicação entre a equipe de desenvolvimento e a infraestrutura, mas quando se trata de algo mais próximo do desenvolvedor como integração contínua, por exemplo, a colaboração se dá pelo responsável pelo DevOps na própria equipe.
Automação	A integração contínua faz a construção, realiza os testes e publica no ambiente de homologação.
Frequência	As sprints tem em média a duração de uma ou duas semanas, ao final de cada sprint o *software* é implantado em produção após ter seguido as etapas anteriores. Mas em média ocorre a cada uma semana a entrega.
Medição	A análise de código é medida através do Sonar, onde é exigida uma cobertura de 70%, notas boas nos quesitos de análise e não possuir *issues* que são criadas pela própria ferramenta do tipo *critical*.
Monitoramento	O *software* é monitorado a cada *commit*, onde é realizada a integração contínua, e a análise do Sonar. Vale ressaltar que a análise feita pelo Sonar pode ser realizada também em um horário agendado.
Garantia de qualidade	A garantia de qualidade ocorre através do monitoramento de entrega.

As figuras apresentadas posteriormente são gráficos que foram gerados a partir das respostas obtidas no formulário respondido pelas pessoas que atuam na seção. Estes gráficos representam o nível conhecimento e a utilização no que se refere as ferramentas e práticas que contribuem para uma entrega contínua.

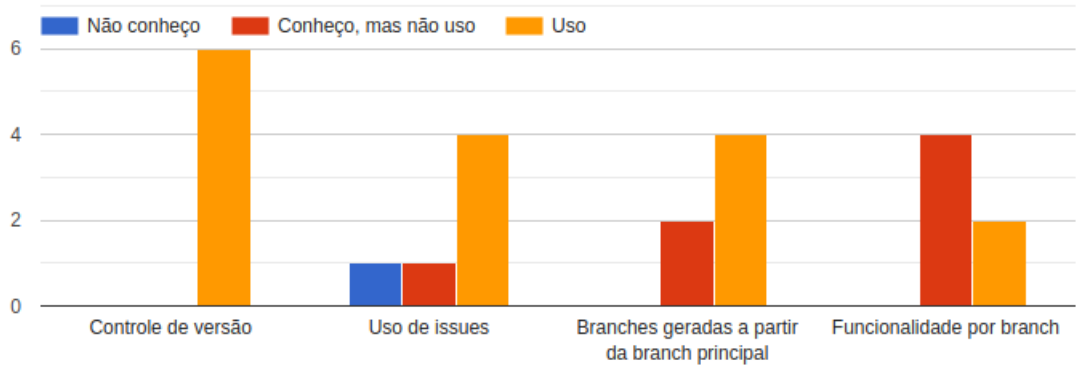

Figura 4 – Práticas de versionamento - Órgão 1

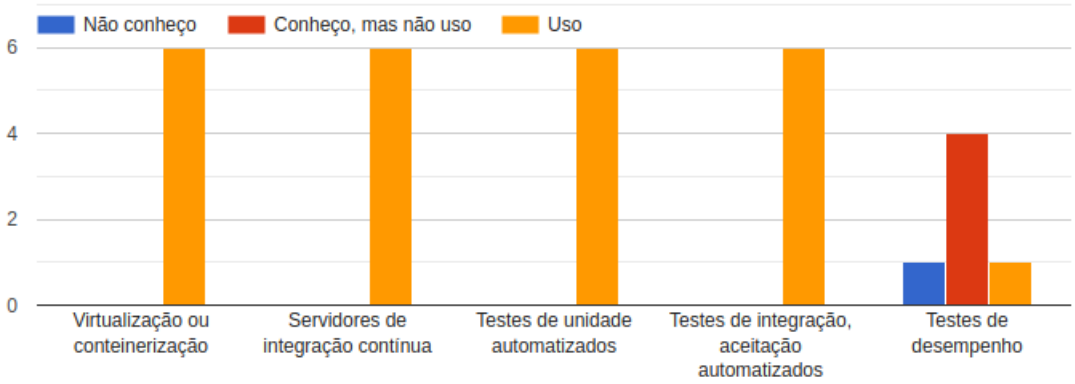

Figura 5 – Práticas relacionadas a Integração Contínua - Órgão 1

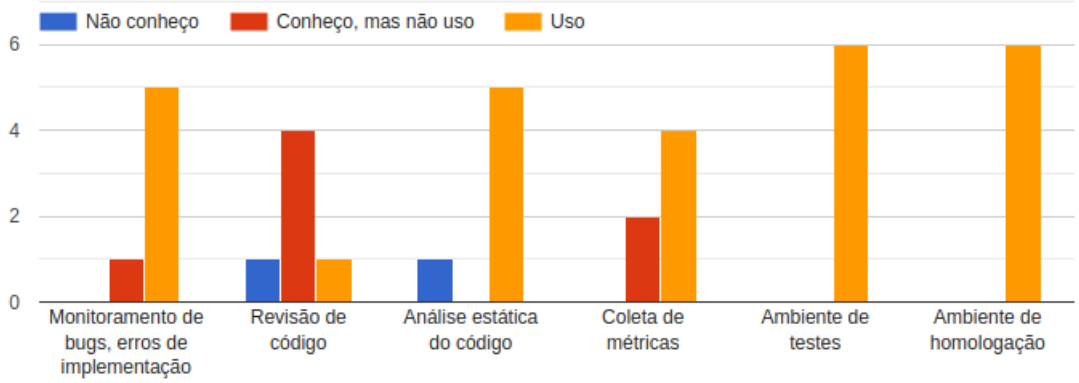

Figura 6 – Práticas de Qualidade - Órgão 1

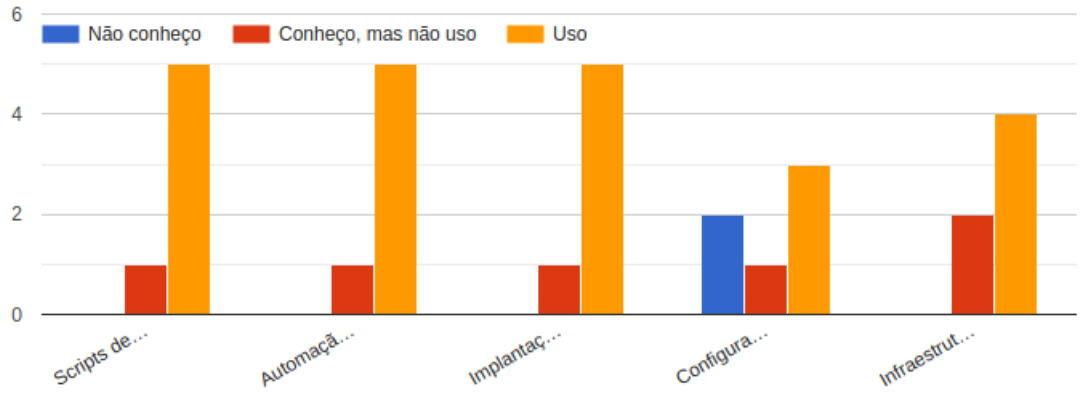

Figura 7 – Práticas de Automatização - Órgão 1

3.2.2 Órgão 2

Neste órgão são desenvolvidos diversos sistemas, mas dentre eles, existe um de maior relevância e que possui atualizações com mais frequência. Na tabela 3.2.2 estão os dados extraídos da entrevista a cerca do processo de desenvolvimento adotado pela equipe.

Tabela 5 – Dados extraídos da entrevista - Órgão 2

DevOps	Existe a equipe de desenvolvimento e a de infraestrutura, ambas se comunicam e possuem proximidade. Através do uso do Jenkins com alguns *pluggins*, o DevOps ocorre de maneira automatizada.
Colaboração	O time possui autonomia e tenta aplicar algumas práticas como o TDD. Existe uma proximidade entre a infraesturutra e a equipe de desenvolvimento, de forma que o DevOps acontece naturalmente.
Automação	Existem *scripts* que automatizam partes do processo. Para realizar o *deploy*, eles clicam em um botão no git e o *deploy* ocorre para o ambiente de homologação, produção.
Frequência	A frequência para a realização de entrega acontece geralmente a cada duas semanas no fechamento de *sprint*, podendo acontecer também durante a sprint por conta de algum *hotfix*.
Medição	A medição se dá apenas pela cobertura de código. Alguns dos projetos não possuem coberturas de código boas devido aos clientes, demanda e curto prazo para entrega.
Monitoramento	O monitoramento é feito pela integração contínua, entretanto não são coletadas métricas e nem indicadores de qualidade, exceto a cobertura.
Garantia de qualidade	Através de parcerias feitas com alunos de universidades, foi feita uma análise de vários pontos do código pelo Code Climate e aplicadas melhorias em alguns dos repositórios de desenvolvimento. Além disto, a equipe realiza refatorações de sistemas legados em vista de melhorar a manutenibilidade.

As figuras 8, 9, 10, 11 são referentes ao conhecimento sobre as práticas de versionamento, integração contínua, qualidade e automatização.

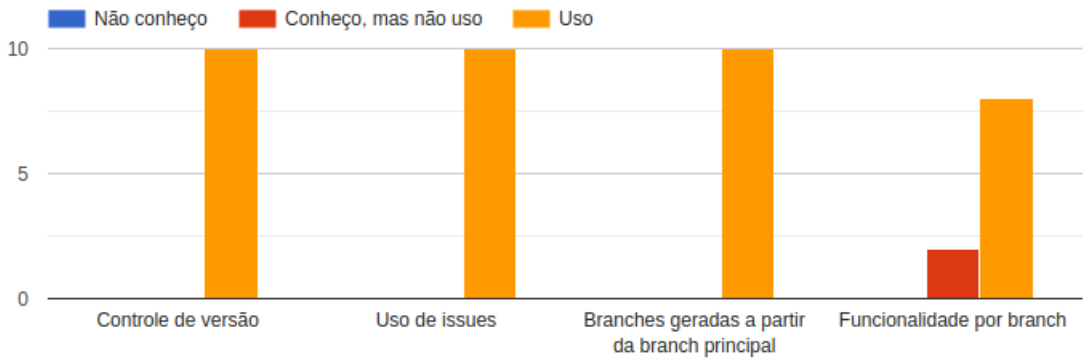

Figura 8 – Práticas de versionamento - Órgão 2

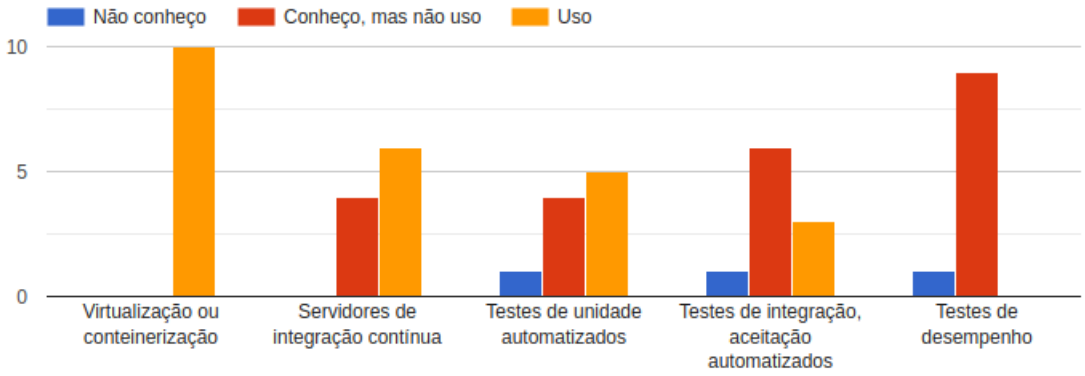

Figura 9 – Práticas relacionadas a Integração Contínua - Órgão 2

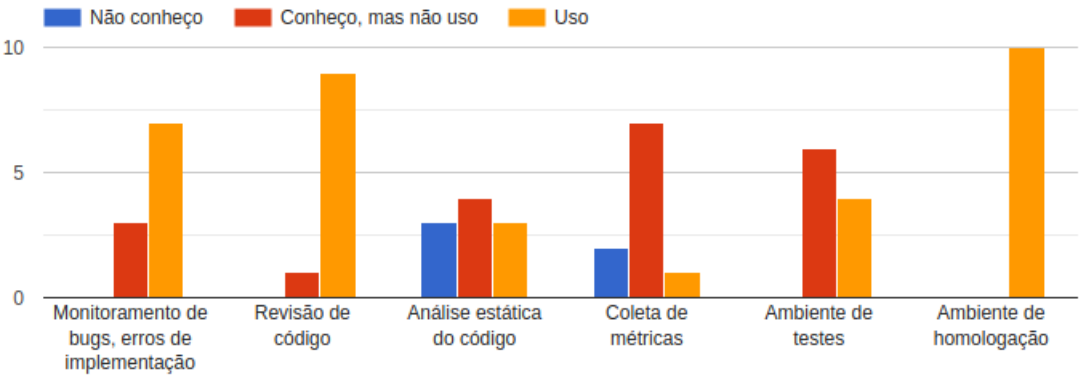

Figura 10 – Práticas de Qualidade - Órgão 2

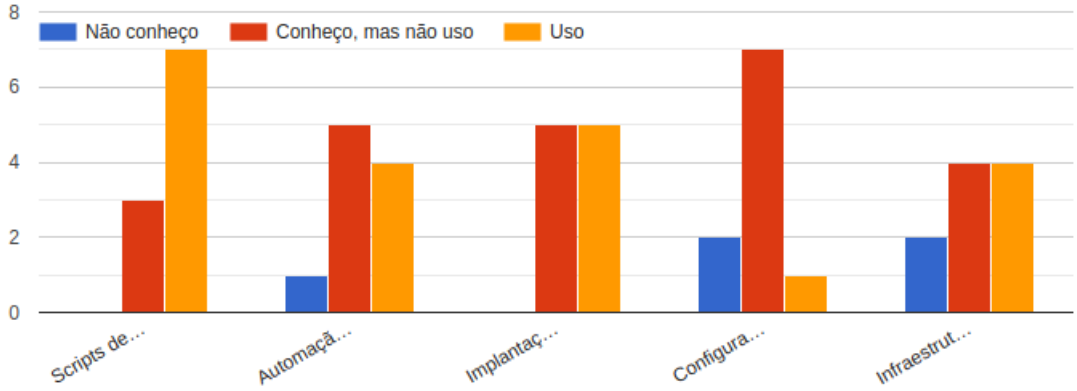

Figura 11 – Práticas de Automatização - Órgão 2

3.2.3 Órgão 3

O setor de tecnologia de informação presente neste órgão presta serviços para a manutenção, evolução tanto para *softwares* legados, quanto para sistemas novos. Dentro deste contexto, foram coletados dados de dois setores que possuem abordagens distintas no processo de desenvolvimento.

3.2.3.1 Setor 1

O setor 1 é responsável pela plataforma mais atual resultante de um trabalho gradual de modernização e unificação dos sistemas. A estratégia adotada por eles para a construção deste sistema foi de reorganizar e remodelar as soluções segundo um modelo organizacional mais aderente às necessidades. Na tabela 3.2.3.1 encontram-se os dados referentes a este setor citados durante a entrevista.

Tabela 6 – Dados extraídos da entrevista - Órgão 3

DevOps	A equipe tem tentado trabalhar baseada nos valores do DevOps: cultura, automação, medição e compartilhamento, tendo o foco em automatizar tudo. O *software* desenvolvido por este órgão possui uma arquitetura distribuída, onde várias equipes participam do desenvolvimento. Estas equipes são multidisciplinares e possuem várias pessoas responsavéis pelo DevOps, sendo que a equipe de Plataforma dá o apoio operacional sobre as ferramentas para as outras equipes.
Colaboração	Existe uma colaboração mútua entre ambas as equipes, principalmente no apoio ao uso de ferramentas, correções de bugs.
Automação	No *pipeline* de entrega possui processos automatizados que vão além a integração contínua e *deploy*. São processos feitos com o auxílio da ferramenta Spinnaker, como realizar a implantação em horário agendado, criar *issues* de implantação, atualizar versões do git, realizar *merge* para a *branch master*, gerar versão estável do docker e mandar e-mail alertando os usuários do sistema das novas atualizações.
Frequência	Durante um período acumularam vários meses de release e depois começaram a implantar com frequência. A frequência varia, ocorrendo sempre sob por demanda.
Medição	São coletadas métricas de código a cerca da qualidade, cobertura de testes e feita uma análise de segurança.
Monitoramento	O monitoramento é feito pelo git utilizando-se da integração contínua a cada *commit*. É feito também o monitoramento dos sistemas a cerca das versões implantadas em cada ambiente: qualidade, homologação e produção.
Garantia de qualidade	A garantia de qualidade vem das melhorias constantes no código a fim de se obter melhores métricas e um sistema com boa funcionalidade.

As figuras seguintes retratam o conhecimento dos envolvidos no estudo.

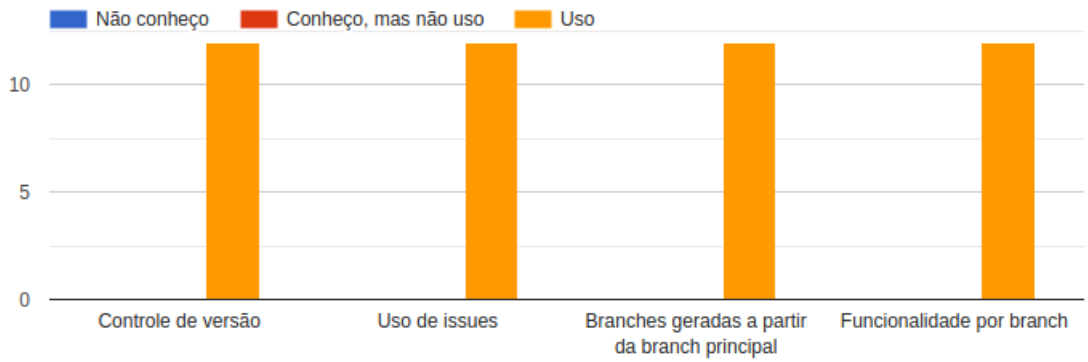

Figura 12 – Práticas de versionamento - Órgão 3 - Setor 1

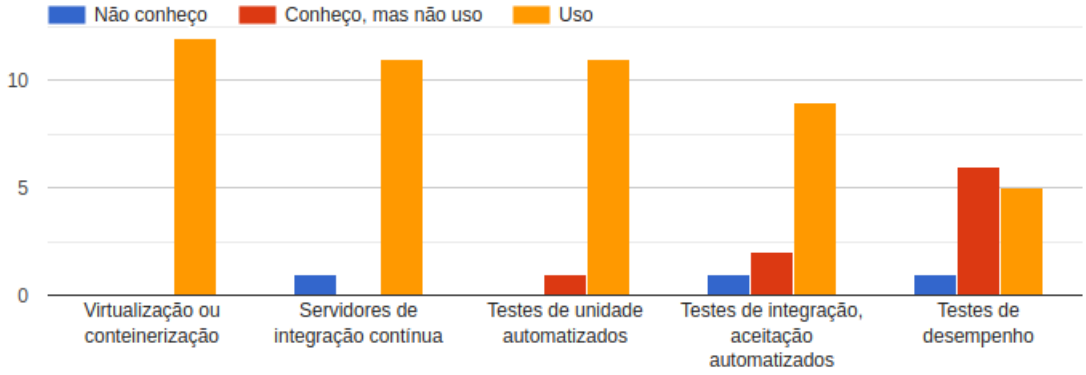

Figura 13 – Práticas relacionadas a Integração Contínua - Órgão 3 - Setor 1

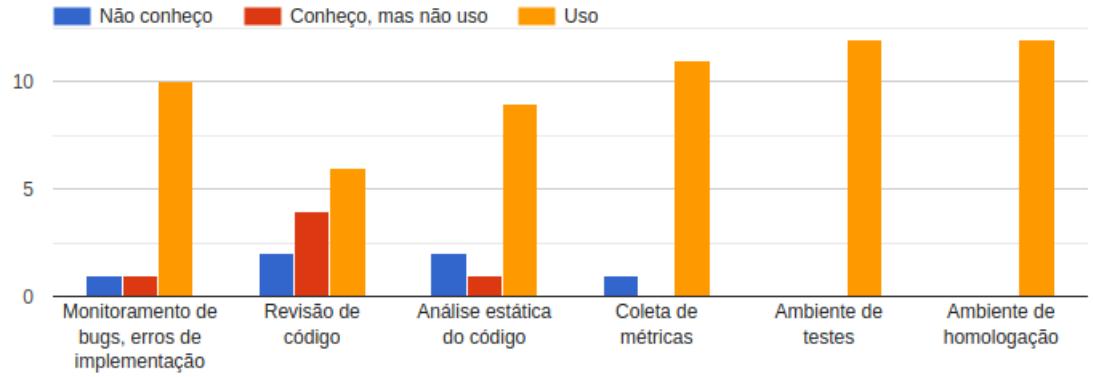

Figura 14 – Práticas de Qualidade - Órgão 3 - Setor 1

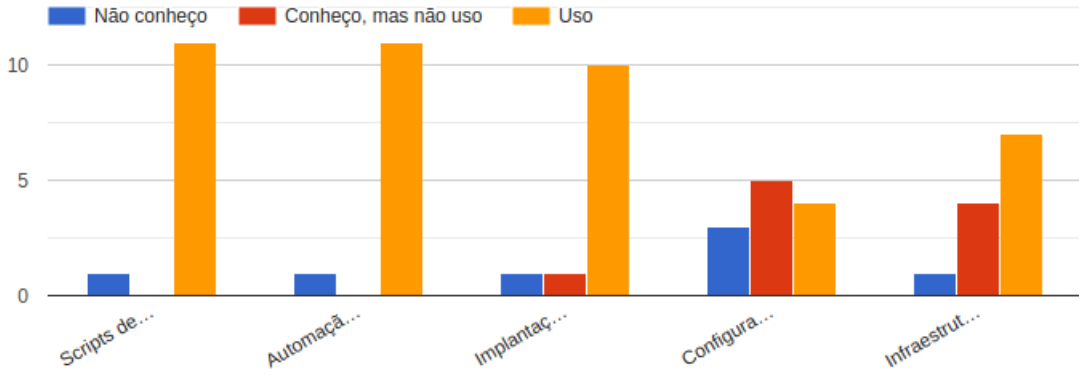

Figura 15 – Práticas de Automatização - Órgão 3 - Setor 1

3.2.3.2 Setor 2

Diferentemente do outro setor, este mantém *softwares* mais antigos com limitações de linguagem, tecnologia, infraestrutura e com um processo diferente, e por isto, surgiu o

incentivo para analisá-los separadamente. As figuras abaixo demonstram o conhecimento deles sobre os aspectos avaliados.

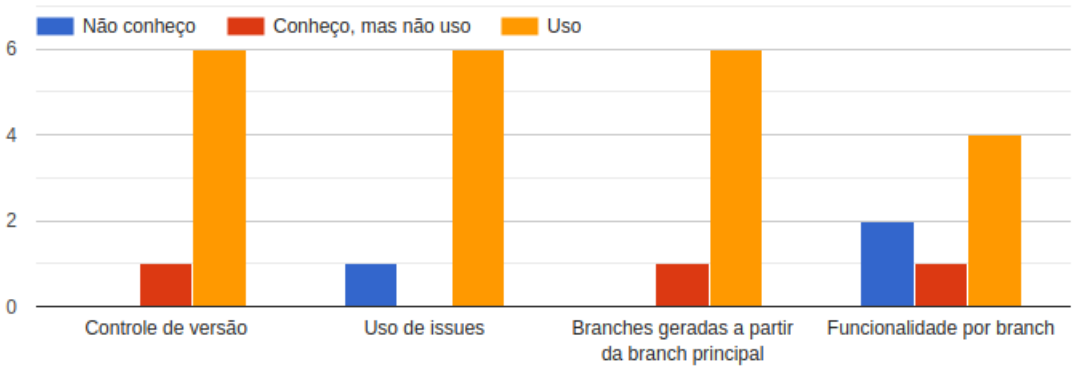

Figura 16 – Práticas de versionamento - Órgão 3 - Setor 2

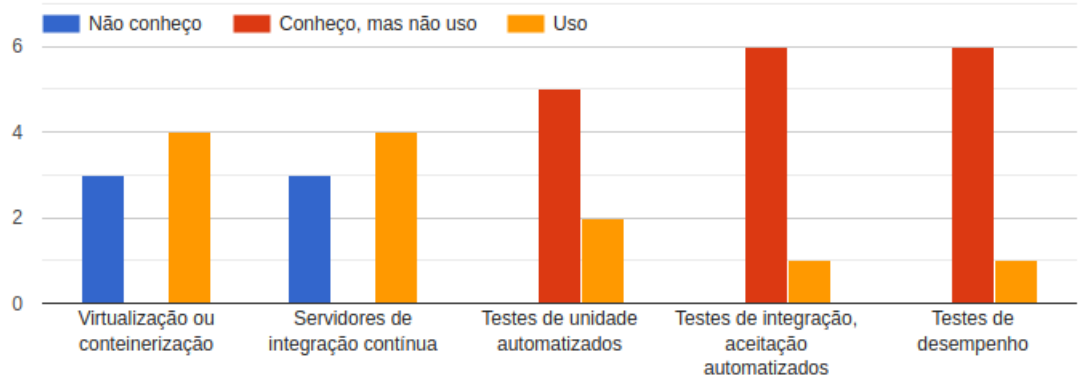

Figura 17 – Práticas relacionadas a Integração Contínua - Órgão 3 - Setor 2

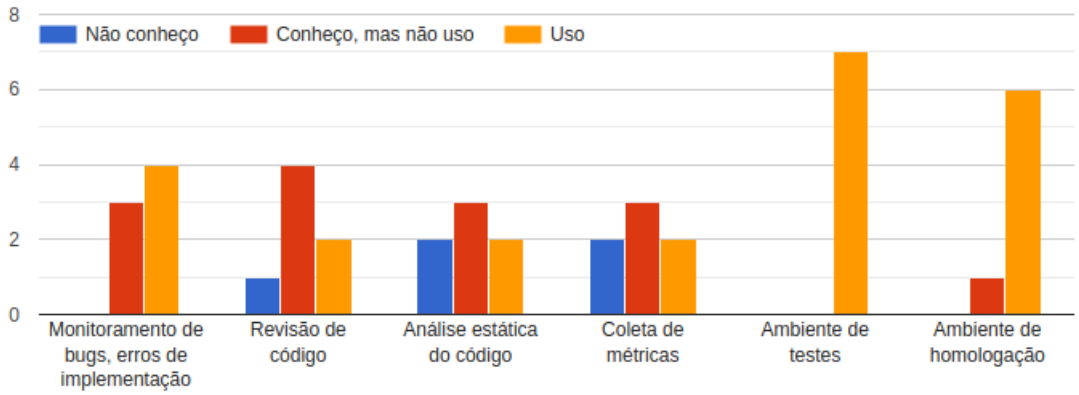

Figura 18 – Práticas de Qualidade - Órgão 3 - Setor 2

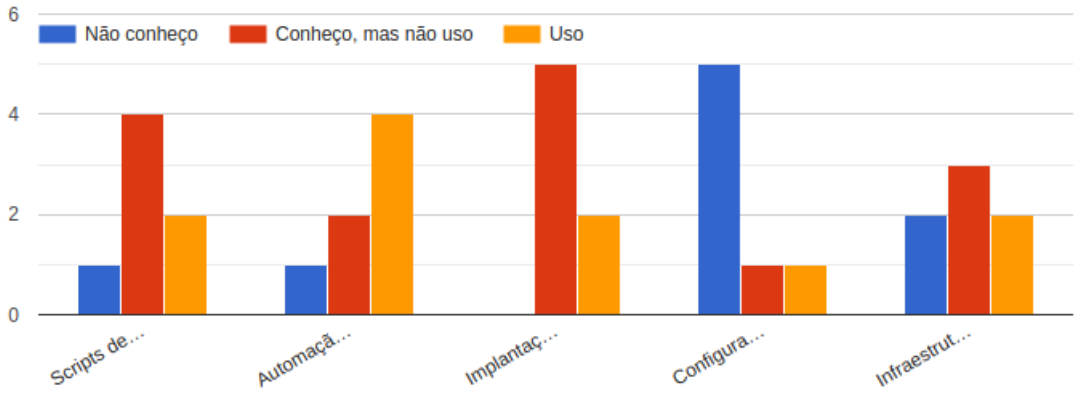

Figura 19 – Práticas de Automatização - Órgão 3 - Setor 2

3.3 Análise

Para realizar a análise dos dados obtidos pelo questionário, entrevista e pela extração de dados, elaborou-se a tabela 7 para guiar a classificação do nível de maturidade de entrega contínua. A definição das características de cada nível veio com base na tabela 3 citada na seção do Referencial Teórico.

Tabela 7 – Estudo de caso - Níveis de maturidade

Nível	Versionamento	Integração Contínua	Qualidade	Automatização
1	Controle de versão	Virtualização, servidores de integração contínua, testes de unidade	Ambiente de testes, homologação	Integração contínua, automação da construção,
2	Branches geradas a partir da principal, uso de issues	Testes de integração, aceitação	Monitoramento de bugs, erros de implementação	Implantação automatizada
3	Funcionalidade por branch	Testes de desempenho	Coleta de métricas, análise estática do código, revisão de código	Ambiente em nuvem, configuração do ambiente em tempo real

Para definir o nível em que o órgão se enxerga dentro deste contexto, foram utilizados os dados da entrevista com o gestor, onde a partir da observação e análise das respostas, foi possível notar a percepção do órgão sobre ele mesmo.

E para definir o nível em que a organização se encontra, optou-se por utilizar as respostas dos desenvolvedores no questionário, onde os gráficos foram expostos na seção de resultados. A partir disto, definiu-se que as respostas dos desenvolvedores que dizem que utilizam tal prática correspondente ao nível de maturidade, deve ser superior a 50% das respostas totais. Exemplificando, para identificar que o órgão 1 está no nível 2 nas práticas de versionamento, foram seguidos esses passos:

1. No nível 1, é necessário que tenha-se controle de versão. Seis pessoas responderam o questionário e todas afirmaram que existe o controle de versão. Por isto, a organização atinge a métrica acima de 50% e qualifica no nível 1.

2. Para o nível 2, é preciso que tenha *branches* geradas a partir da principal e uso de *issues*. Como é preciso ter resposta das duas práticas, o total de respostas foi 12 (6 para cada prática). 4 pessoas responderam que utilizam *issues* e 4 a prática de *branches*. Portanto, 8 das 12 respostas foi de que se utilizava as práticas, ou seja, 66,66% das pessoas conhecem as práticas, qualificando assim o nível 2.

3. No nível 3, é necessário desenvolver uma funcionalidade por *branch*. 2 das 6 pessoas que responderam, disseram que utilizam, totalizando 33%. Como 33% é menor que 50%, a órgão não atinge o nível 2 e fica qualificado como sendo parte do nível 2 de maturidade.

3.3.1 Análise comparativa

Neste tópico tem-se o objetivo de realizar um comparativo de como o órgão acredita que seja o seu nível de entrega contínua nos aspectos de versionamento, integração contínua, qualidade e automatização e de como ele realmente se encontra de acordo com os níveis definidos.

3.3.1.1 Órgão 1

A percepção que o gestor tem sobre este órgão é que eles possuem um bom processo de entrega contínua, visto que as entregas acontecem com frequência e com critérios de qualidade que devem ser cumpridos, como a cobertura de testes e a análise no Sonar.

Do ponto de vista da equipe, a entrega contínua contribui para a produtividade, *feedback* do usuário e para mitigar os erros que surgem no *software*. A diminuição destes erros, segundo um dos participantes da pesquisa, ocorre por serem pequenos incrementos a serem implantados, facilitando assim descobrir em um número menor de *commits* a origem da falha.

A partir dos dados fornecidos pelo órgão no que se refere as datas de entrega do período em que se passaram a anotar as *releases* implantadas até a data da entrevista

feita, pode-se obter um gráfico onde é possível notar a diferença de dias entre o lançamento de uma *release* em produção. A média de dias entre uma entrega em produção e outra foi de 6 dias, isto reflete no fato da equipe possuir o hábito de entregar algo durante o meio de uma *sprint*. Também é possível notar alguns intervalos maiores, provavelmente por terem encontrado dificuldades durante o desenvolvimento e terem conseguido entregar apenas na data próxima ao final da *sprint*. Os intervalos em que ocorreram entregas em um curto período são reflexo de demandas feitas a equipe com urgência, já que o sistema encontra-se em uso pelos setores do órgão.

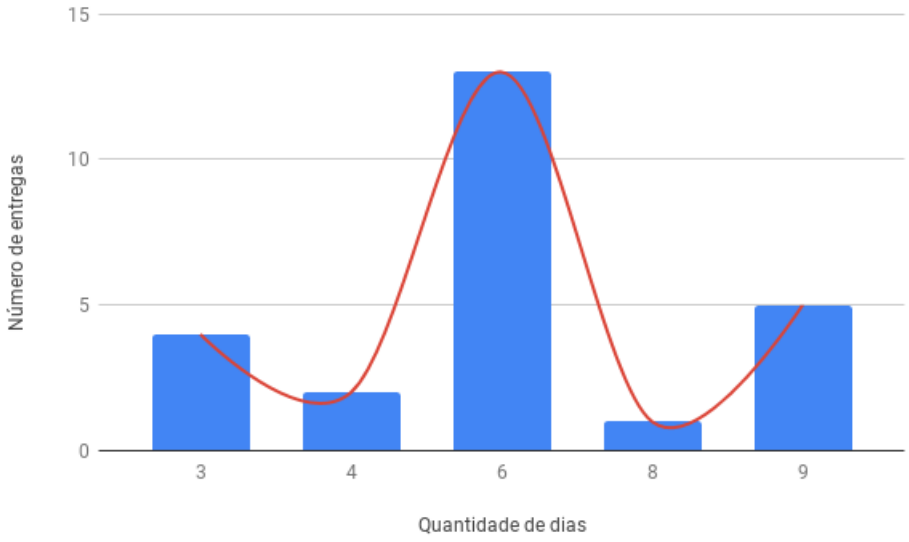

Figura 20 – Intervalo de dias entre as entregas

Na tabela 8 encontra-se um comparativo de como a organização se enxerga e de como ela realmente é.

Tabela 8 – Nível de maturidade - Órgão 1

	Nível em que a organização se enxerga	**Como a organização está**
Versionamento	**3**	**2**
Integração Contínua	**3**	**2**
Qualidade	**3**	**3**
Automatização	**3**	**3**

A partir dos dados do nível da organização e do intervalo de dias é possível concluir que dentro do aspecto de entrega contínua, eles tem realizado com eficiência. Isto vem de dois fatores importantes: o tempo e o nível da organização. Apesar dela não possuir nível 3 em todos os aspectos, a maioria das práticas estão sendo realizadas pela equipe e em um bom tempo, já que o intervalo de entrega está sendo entre 3 e 9 dias, sendo que a maioria ocorre em 6 dias.

Para esta organização atingir um nível melhor, seria preciso que começassem a adotar a prática de desenvolver uma funcionalidade pro *branch*, isto influenciaria também nos *merges* dados e poderia diminuir o tempo de entrega, já que seriam pequenos insumos de *software* a serem implantados. O outro fator é corresponde aos testes de desempenho.

3.3.1.2 Órgão 2

O gestor deste órgão acredita que a sua equipe possui um bom *pipeline* de entrega. A percepção dele é que tendo frequência, o processo de entrega contínua está bom. Apesar de entender a importância da qualidade no *software*, não percebe que a qualidade está ligada diretamente a todo o processo.

A percepção da equipe é que o processo adotado é um processo bom que ajuda no prazo, visto que uma boa parte da implantação é automatizada por meio dos *scripts*.

As informações presentes no gráfico da figura 21 retratam as entregas em feitas em um dos sistemas mais desenvolvido pela equipe. A maioria das entregas acontecem com um dia de diferença e com o intervalo de no máximo 11 dias até uma nova entrega.

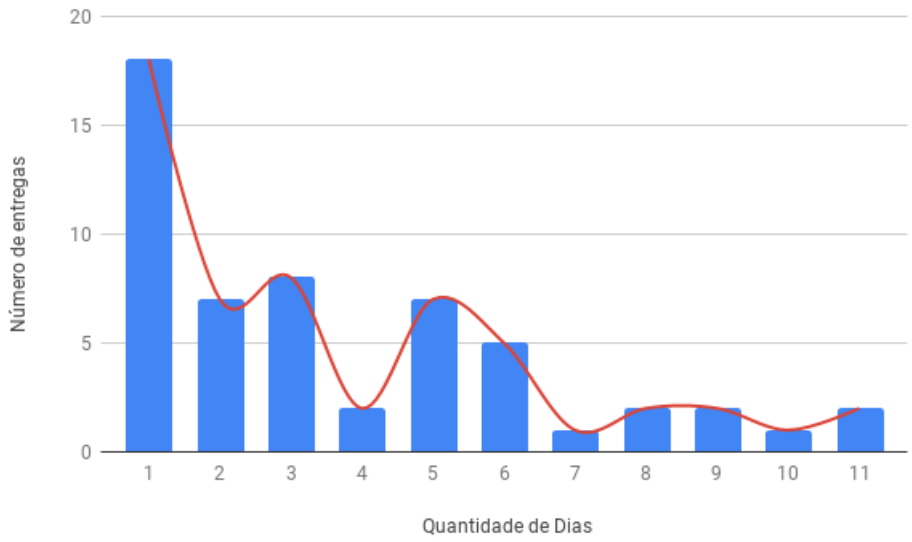

Figura 21 – Intervalo de dias entre as entregas

Analisando o gráfico do tempo, conclui-se que realizam muitas entregas com um intervalo curto de dias, o que é considerado um ponto positivo. Porém, ao verificar-se a qualidade do *software* produzido, é possível notar que não realizam testes para validar aquilo que foi desenvolvido e nem coletam métricas para analisar sua qualidade. A falta de uma boa integração contínua e de uma qualidade, afeta diretamente nestas entregas, porque uma boa parte delas foram entregas para corrigir aquilo que já tinha sido colocado para testar no ambiente de homologação. Ou seja, o fato de se ter entregas com frequên-

cia, não significa que a entrega contínua está sendo realizada corretamente. Na tabela 9 encontra-se os níveis da organização.

Tabela 9 – Nível de maturidade - Órgão 2

	Nível em que a organização se enxerga	**Como a organização está**
Versionamento	**3**	**3**
Integração Contínua	**3**	**1**
Qualidade	**2**	**2**
Automatização	**3**	**2**

3.3.1.3 Órgão 3

Sob o ponto de vista do gestor, o órgão adota uma série de práticas que caracterizaram o uso de entrega contínua e que portanto, acredita que esteja em um nível de maturidade alto. O uso destas práticas, ferramentas que ajudam na automatização, são os fatores que deram esta percepção. Falhas de compilação é um dos desafios que pode ser encontrado ao lidar com entrega contínua. Podendo acarretar na baixa produtividade dos desenvolvedores, já que os mesmos deverão resolver estas falhas antes de dar continuidade ao processo (VASSALLO; ZAMPETTI; ROMANO, 2016). Para a equipe, a entrega contínua é boa e contribui principalmente para acelerar a percepção de entrega de valor pelo cliente. Eles acreditam estar em um bom nível de maturidade.

Na figura 22 estão os intervalos de entregas nesta organização. O sistema desenvolvido por eles possui uma arquitetura distribuída, então cada grande funcionalidade do *software* faz parte de um contexto. Para gerar este gráfico, foi extraído dos repositórios dos principais contextos desenvolvidos as datas de entrega, e ao final juntou-se tudo em um gráfico só.

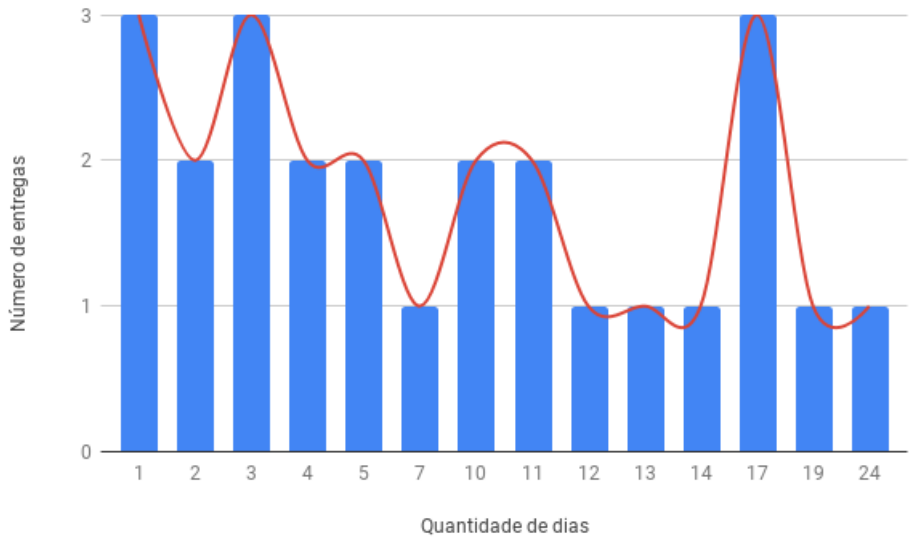

Figura 22 – Intervalo de dias entre as entregas

Na tabela 10 é possível verificar o nível de maturidade da organização.

Tabela 10 – Nível de maturidade - Órgão 3 - Setor 1

	Nível em que a organização se enxerga	Como a organização está
Versionamento	**3**	**3**
Integração Contínua	**3**	**2**
Qualidade	**3**	**3**
Automatização	**3**	**3**

Tendo como base os dados da tabela e figura acima é possível chegar em algumas conclusões. Assim como nas outras organizações analisadas, faltou a prática de realizar testes de desempenho para atingir o nível máximo de maturidade na integração contínua. Nesta organização foi encontrado o processo mais maduro das três analisadas. Todavia, mesmo no cenário de um processo de desenvolvimento onde aplicam a maioria das práticas de entrega contínua, o intervalo de dias encontrado não foi de um fluxo bom, já que chegou a demorar até 24 dias para realizar uma entrega. Este fluxo pode ser explicado por ser um sistema produzido por demandas, onde já não se tem tantas entregas e onde parte da equipe está responsável por aspectos gerenciais do setor.

3.3.1.4 Análise geral

Na tabela 11 encontram-se os níveis de todas as organizações.

Tabela 11 – Níveis de maturidade no processo de entrega contínua dos Órgãos

	Versionamento	**Integração Contínua**	**Qualidade**	**Automatização**
Órgão 1	**2**	**2**	**3**	**3**
Órgão 2	**3**	**1**	**2**	**2**
Órgão 3 - Setor 1	**3**	**2**	**3**	**3**
Órgão 3 - Setor 2	**3**	**1**	**2**	**1**

Analisando de maneira geral as três organizações, é possível notar que as práticas menos utilizadas são as relacionadas a integração contínua. As práticas de integração utilizadas pelos órgãos são aplicadas parcialmente e por isto o nível baixo de maturidade. Na maioria deles a integração se resume a construir e *builds* e realizar testes unitários, o que caracteriza o nível 1 de maturidade.

A prática de versionamento possui um bom nível nos três órgãos, isto deve-se ao fato de ser uma das principais caracteristícas para ocorrer o processo de entrega contínua. Outra maturidade que se destacou foi a qualidade, mas esta em específico, seria melhor se todos tivessem alcançado o nível 3, já que o último nível é o que retrata a coleta de métricas. A qualidade foi atingida em um nível considerado bom até no setor que não realiza entrega contínua, isto pelo fato de monitorarem os erros e testarem o *software* em um ambiente de homologação, antes de ir para produção.

A automatização ainda falha na implantação automatizada e na configuração do ambiente em tempo real para alguns órgãos, mas encontrou o nível de maturidade máximo em dois destes.

O processo de entrega contínua adotado pelos órgãos apresentou bons resultados com o uso destas práticas, mas ainda assim poderiam ser implementadas algumas melhorias que faltaram para atingir o mais alto nível de maturidade em todo o processo.

4 Considerações Finais

Esta pequisa alcançou o objetivo de analisar o processo de entrega contínua. Foi possível notar que nem sempre a visão que se tem sobre a qualidade do processo implementado, é a que de fato acontece. Outro importante aspecto, foi que o entendimento do que se é entrega contínua está limitado para maioria dos desenvolvedores na frequência em entregar algo. Isto pode-se perceber ao analisar qualitativamente um órgão com um fluxo ideal de entregas no aspecto do tempo, mas que por conta de uma qualidade ruim no desenvolvimento não se qualificou como um processo de entrega contínua. Isto deve-se por uma parte do fluxo de entregas neste órgão ser pra corrigir erros que poderiam ser mitigados com práticas de qualidade, outra característca de um processo bom que não foi encontrada no mesmo.

Além disto, pode-se notar que mesmo um processo que não tem entrega contínua, é possível ter características de um processo bom de entregas, podendo apenas amadurecer com as práticas de automatização, integração contínua.

Para trabalhos futuros, poderia ser implantado do zero um processo de entrega contínua em um órgão que não realiza versionamento, integração contínua, qualidade, automatização, e comparar os resultados do antes da adoção do processo e depois.

Referências

A qualitative study of DevOps usage in practice. *Journal of software: Evolution and Process*, John Wiley and Sons Ltd, v. 29, n. 6, 6 2017. ISSN 2047-7481. Citado 2 vezes nas páginas 24 e 25.

FARROHA, B.; FARROHA, D. A framework for managing mission needs, compliance, and trust in the devops environment. *IEEE software*, IEEE Military Communications Conference, 2014. Citado na página 21.

KRUSCHE, S.; ALPEROWITZ, L. Introduction of continuous delivery in multi-customer project courses. *IEEE software*, ACM, 2014. Citado na página 17.

KRUSCHE, S.; BRUEGGE, B. Csepm - a continuous software engineering process metamodel. *IEEE software*, IEEE, 2017. Citado na página 17.

PERERA, P.; SILVA, R.; PERERA, I. Improve software quality through practicing devops. *IEEE software*, Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer), 2017. Citado na página 21.

RAHMAN, A. A. U. et al. Synthesizing continuous deployment practices used in software development. *IEEE software*, IEEE, 2015. Citado 3 vezes nas páginas 18, 20 e 24.

SAMARAWICKRAMA, S. S.; PERERA, I. Continuous scrum: A framework to enhance scrum with devops. *IEEE software*, IEEE, 2017. Citado 2 vezes nas páginas 17 e 20.

SAVOR, T. et al. Continuous deployment at facebook and oanda. *IEEE software*, IEEE, 2016. Citado 2 vezes nas páginas 19 e 20.

SIQUEIRA, R. et al. Continuous delivery: Building trust in a large-scale, complex government organization. *IEEE software*, IEEE, 2018. Citado 2 vezes nas páginas 20 e 24.

SONI, M. End to end automation on cloud with build pipeline: The case for devops in insurance industry, continuous integration, continuous testing, and continuous delivery. *IEEE software*, IEEE, 2015. Citado 2 vezes nas páginas 20 e 21.

VASSALLO, C.; ZAMPETTI, F.; ROMANO, D. Continuous delivery practices in a large financial organization. *IEEE software*, IEEE, 2016. Citado 2 vezes nas páginas 20 e 40.

VIRTANEN, A. et al. On continuous deployment maturity in customer projects. *IEEE software*, ACM, 2014. Citado 2 vezes nas páginas 22 e 23.

Apêndices

APÊNDICE A – Revisão de Literatura

A.1 Planejamento

Questões de pesquisa:

Q1. Quais são indicadores que evidenciam entregas contínuas de um *sofwtare*?

Q2. Quais as dificuldades que um time pode encontrar para realizar entregas com frequência?

Q3. Qual é o processo adotado por um time que realiza entregas contínuas?

Q4. Quais práticas são adotadas por um time de *DevOps*?

Tabela 12 – Estabelecimento do PICOC para o objetivo de pesquisa

Elemento do PICOC	**Descrição**
População	Times agéis (*agile*)
Intervenção	Processo de entrega contínua(*continuous delivery*)
Comparação	Não se aplica
Resultado	Indicadores de entregas(*indicators*)
Contexto	DevOps (*devops*)

String de busca:

"agile"AND "continuous delivery"AND ("indicators"OR "devops")

Critérios de aceitação:

C1. O conteúdo da publicação deve responder as questões de pesquisa.

C2. Deve conter algum estudo de caso ou análise de práticas.

Critérios de exclusão:

CE1. Texto escrito em idiomas diferentes do português ou inglês.

CE2. Publicação não disponível na integra.

A.2 Execução

Para executar a revisão sistemática foi feita uma busca nas bases escolhidas com o uso da *string* definida anteriormente. Os resultados dessa busca encontram-se na tabela 13.

Tabela 13 – Busca nas bases

Base de pesquisa	**Quantidade de resultados encontrados**
ACM Digital Library	9
IEE Xplore	109
Scopus	17
Web of Science	11

Com essa pesquisa encontrou-se um total de 146 artigos. Após a remoção dos artigos duplicados, chegou-se em 119 artigos para executar a segunda etapa da revisão. Para selecionar os artigos de estudo foram analisados os critérios de inclusão e exclusão.

APÊNDICE B – Questionário

Estudo sobre Entrega Contínua de Software

Se você permitir, usarei suas respostas anonimizadas em um trabalho de conclusão de curso cujo o tema é Entrega Contínua de Software em órgãos governamentais.

* Required

1. **Sobre respostas anônimas a este questionário: ***
   Check all that apply.

   - [ ] Permito utilizar os dados fornecidos a este questionário de forma anônima.

Perfil

2. **Idade ***

3. **Gênero ***
   Mark only one oval.

   - ( ) Feminino
   - ( ) Masculino
   - ( ) Prefiro não dizer

4. **Você possui quantos anos de experiência profissional? ***

5. **Qual o nome do lugar onde você trabalha? ***

6. **Há quanto tempo você trabalha nele? ***

7. **Qual é o seu cargo? (Ex: estagiário, gerente de software, desenvolvedor, etc.) ***

Atuação

8. **Qual ou quais projetos você participa? ***

9. **Em qual etapa do processo de desenvolvimento de software você trabalha atualmente? (Ex: Requisitos, Desenvolvimento, DevOps, Testes, etc)** *

10. **De quais atividades você participa?** *
Check all that apply.

- [ ] Planejamento de Sprint
- [ ] Revisão de Sprint
- [ ] Programação
- [ ] Realização de testes
- [ ] Revisão de código
- [ ] Definição dos requisitos
- [ ] Prototipagem da interface
- [ ] Análise de usabilidade
- [ ] Gerenciamento do pipeline de implantação

Entrega Contínua

11. **O projeto em que você trabalha realiza entrega contínua?** *
Mark only one oval.

- ◯ Sim
- ◯ Mais ou Menos
- ◯ Não

12. **O projeto em que você trabalha possui um pipeline de entrega?** *
Mark only one oval.

- ◯ Sim
- ◯ Não

13. **Marque as práticas que a sua equipe utiliza.** *

Check all that apply.

- [ ] Repositório (Controle de versão.)
- [ ] Intercomunicação (Comunicação entre toda a equipe responsável pela produção do software.)
- [ ] Monitoramento (Monitorar todo o processo de entrega contínua.)
- [ ] Shepherding Changes (Manter a responsabilidade dos desenvolvedores em todas as etapas de implantação contínua, ou seja, desde o desenvolvimento, testes, implantação, correção de erros e possíveis mudanças após a implantação.)
- [ ] Testes automatizados (Código é testado a cada commit.)
- [ ] Revisão de código (Desenvolvedor compartilha o código feito durante a sprint com toda a equipe.)
- [ ] Implantação automatizada.
- [ ] Feature flag (Técnica que executa o código através de ramificações. Cada ramificação é executada após uma condição ser aceita.)
- [ ] Lançamento no escuro (Implantar mudanças no software, mantendo ele funcional mas com alterações ocultas para o usuário.)
- [ ] Encenação (Lança modificações na aplicação para um grupo interno, antes de passar para todos. Ou lança gradualmente para os seus usuários.)
- [ ] Utiliza-se de redes sociais, fóruns, pesquisas para obter um feedback do usuário
- [ ] Automatização da construção (A cada commit, a build é construída automaticamente.)
- [ ] Construções quebradas possuem prioridade em relação ao desenvolvimento de novas funcionalidades

14. **Agora de forma mais detalhada, classifique seu conhecimento sobre as práticas de entrega contínua utilizadas no projeto em que você trabalha atualmente.** *

Mark only one oval per row.

	Não conheço	Conheço, mas não uso	Uso
Controle de versão (Ex: git)	◯	◯	◯
Uso de issues	◯	◯	◯
Branches geradas a partir da branch principal	◯	◯	◯
Funcionalidade por branch	◯	◯	◯
Virtualização ou conteinerização	◯	◯	◯
Servidores de integração contínua	◯	◯	◯
Testes de unidade automatizados	◯	◯	◯
Testes de integração, aceitação automatizados	◯	◯	◯
Testes de desempenho	◯	◯	◯
Monitoramento de bugs, erros de implementação	◯	◯	◯
Revisão de código	◯	◯	◯
Scripts de automação	◯	◯	◯
Automação da contrução	◯	◯	◯
Implantação automatizada	◯	◯	◯
Análise estática do código	◯	◯	◯
Coleta de métricas	◯	◯	◯
Ambiente de testes	◯	◯	◯
Ambiente de homologação	◯	◯	◯
Configuração do ambiente em tempo de execução (Ex: Chef)	◯	◯	◯
Infraestrutura (Ambiente em nuvem)	◯	◯	◯

15. **Qual é sua percepção em relação a entrega contínua? Acha que contribui positivamente ao projeto?**

16. **Existe algum impedimento para realizar a entrega contínua?** *