UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

MALUANE PIZARRO RUBERT

ANÁLISE DE IMPACTO DA UTILIZAÇÃO DE ENTREGA CONTÍNUA DE SOFTWARE EM UMA ORGANIZAÇÃO:
Um Estudo Experimental

Porto Alegre
2017

MALUANE PIZARRO RUBERT

ANÁLISE DE IMPACTO DA UTILIZAÇÃO DE ENTREGA CONTÍNUA EM UMA ORGANIZAÇÃO:

Um Estudo Experimental

Artigo apresentado como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação, pelo Curso de Sistemas de Informação da Universidade do Vale do Rio dos Sinos - UNISINOS

Orientador: Prof. Dr. Kleinner Silva Farias de Oliveira

Porto Alegre

2017

ANÁLISE DE IMPACTO DA UTILIZAÇÃO DE ENTREGA CONTÍNUA EM UMA ORGANIZAÇÃO:

Um Estudo Experimental

Maluane Pizarro Rubert[1]*

Kleinner Silva Farias de Oliveira[2]**

Resumo: O processo de entrega contínua tem sido adotada por organizações para disponibilização do software para seus usuários a qualquer momento. A transição entre metodologias tradicionais de entrega de software para a entrega contínua pode causar impactos nos resultados gerados pelas organizações, como a qualidade dos produtos desenvolvidos. Porém pouco se sabe a respeito dos impactos da entrega contínua nas organizações que adotam. Por esta razão, este estudo experimental analisa a qualidade dos produtos desenvolvidos de uma organização antes e depois da adoção do processo, através das seguintes perspectivas: qualidade de código e qualidade de produtos entregues. Os resultados apresentados demonstram quais aspectos de qualidade são impactados pela transição para entrega contínua de software. Este trabalho contribui tanto para o meio acadêmico quanto para a prática, por meio dos conhecimentos empíricos adquiridos durante o estudo experimental.

Palavras-chave: Desenvolvimento de Software. Engenharia de Software. Entrega Contínua.

1 INTRODUÇÃO

Com a evolução da tecnologia nas últimas décadas, surgiu uma demanda progressiva por soluções computadorizadas em todos os setores do mercado. (VASCONCELOS; ROUILLER; MACHADO; MEDEIROS, 2006). Neste contexto, várias práticas e metodologias de desenvolvimento de software têm sido propostas, visando apoiar o processo de desenvolvimento com ciclos menores e entregas mais frequentes de partes do software. (MAXIM; PRESSMAN, 2016). Exemplos destas

[1]* Graduanda em Sistemas de Informação pela Unisinos E-mail: maluane.c@gmail.com.

[2]** Doutor em Informática pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Mestre em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS). Bacharel em Ciência da Computação pela Universidade Federal de Alagoas (UFAL). Tecnólogo em Tecnologia da Informação pelo Instituto Federal de Alagoas (IFAL). E-mail: kleinner@gmail.com

metodologias e práticas seriam *Scrum* [(SABBAGH, 2013)], *Extreme Programming* [(DAIRTON, 2014)], *Kanban* [(ANDERSON, 2011)], e Entrega Contínua [(HUMBLE; FARLEY, 2014)]. Tais metodologias têm como propósito apoiar o desenvolvimento de software, estabelecendo *workflows*, métodos, ferramentas e ambientes para tratar todos os aspectos relacionados ao ciclo de desenvolvimento.

Nas metodologias tradicionais, tais como modelo Clássico ou Sequencial [(PRESSMAN, 2001)], os sistemas de software são desenvolvidos e entregues ao cliente através de ciclos longos e de forma artesanal, exigindo o trabalho manual conjunto de diversas equipes para colocar o software em produção. Esta abordagem possui desvantagens, tais como: código já criado fica parado desnecessariamente no repositório aguardando pela formação de uma nova versão; ciclo longo de desenvolvimento, demora ao lidar com problemas e oportunidades, estresse das equipes para colocar software em produção, e distanciamento entre o time de desenvolvimento e os usuários finais. (HUMBLE; FARLEY, 2014).

Contudo, com o avanço de recursos tecnológicos e tecnologias propriamente ditas, práticas e abordagens ágeis começaram a ser adotadas, tendo seu uso intensificado, dentre elas a Entrega Contínua (EC). (HUMBLE; FARLEY, 2014). A EC vem do primeiro princípio do Manifesto Ágil: "Nossa maior prioridade é satisfazer o cliente por meio da entrega adiantada e contínua de software de valor". (MANIFESTO, 2001). Essa prática busca que a entrega de software ocorra automaticamente e com frequência, de forma que toda a modificação no projeto gere um sistema com potencial imediato de lançamento. (RIES, 2011).

Desta forma, este trabalho teve como objetivo analisar os impactos da utilização da EC em uma organização de desenvolvimento de software. Para isso, o impacto do uso da EC foi analisado através de duas perspectivas: a qualidade do código e a qualidade dos produtos entregues. Este trabalho se justifica pois, do ponto de vista acadêmico, observa-se que existem poucas publicações específicas, realizando uma análise comparativa entre antes e depois da utilização de entrega contínua em organizações. Além disso, por ser uma prática relativamente nova na engenharia de software, há poucas publicações sobre o uso de EC na prática. A

principal contribuição deste trabalho trata-se do conhecimento empírico gerado do uso de EC na indústria.

Este trabalho é organizado da seguinte forma: a Seção 2 apresenta a fundamentação teórica que onde estão principais os conceitos, necessários para entendimento deste estudo. Na Seção 3 a metodologia do estudo é apresentada. A Seção 4 descreve a análise dos resultados obtidos. Na Seção 5 é realizada uma análise comparativa de trabalhos relacionados. E por fim a Seção 6 apresenta as considerações finais deste estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentada a fundamentação teórica necessária ao entendimento deste trabalho. Para isso, serão apresentados os conceitos utilizados a respeito de processos como Práticas de Desenvolvimento de Software, Disponibilização de Software e Entrega Contínua de Software.

2.1 Práticas de Desenvolvimento de Software

Todos os aspectos do desenvolvimento de software, desde os primeiros estágios até a manutenção, incluem em especificar, desenvolver, gerenciar e entregar sistemas de software. Esse procedimento é trabalho da Engenharia de Software (ES), que surgiu para solucionar problemas de sistemas de software, com objetivo de dar suporte ao desenvolvimento de softwares utilizando processos, métodos, técnicas e ferramentas. (SOMMERVILLE, 2011).

A entrega de um incremento do software é uma etapa importante no seu projeto, uma vez que os usuários podem dar feedback a respeito do que foi entregue. (MAXIM; PRESSMAN, 2016). A qualidade do software é fortemente dependente da qualidade do processo utilizado na sua preparação. (PRIKLADNICKI; AUDY; EVARISTO, 2014). Fuggeta (2000) reflete e afirma que pesquisadores e praticantes têm percebido que o desenvolvimento de software é um esforço coletivo, complexo e criativo. Sendo assim, a qualidade do produto de software depende

muito das pessoas, organizações e processos utilizados para desenvolvê-los e entregá-los. (FUGGETTA, 2000).

Como parte integrante do desenvolvimento de software, a disponibilização do mesmo trata-se da entrega do software ao usuário [(MAXIM; PRESSMAN, 2016)], onde o usuário deverá receber suporte para sua utilização e irá prover feedback a respeito do produto recebido.

2.2 Disponibilização de Software

Na Engenharia de Software (ES), existe uma etapa caracterizada como a disponibilização do software. Esta acontece sempre que um incremento do software é entregue ao cliente, o que englobado a entrega, o suporte ao usuário e o feedback. Para que este processo ocorra existem alguns passos a serem observados como, (1) o gerenciamento das expectativas do cliente quanto à entrega, (2) criar uma estrutura de suporte aos usuários, (3) disponibilizar instruções para o uso do software, (4) corrigir os erros antes da disponibilização do software e por fim, (5) deve existir um pacote de entrega completo e testado. O pacote de entrega completo consiste em software executável, documentação, e os demais arquivos necessários completamente montados e testados. (MAXIM; PRESSMAN, 2016).

Uma vez que os 5 passos descritos acima foram realizados é possível disponibilizar para o cliente o software, realizando assim a sua entrega. (MAXIM; PRESSMAN, 2016). Esta entrega pode ser realizadas de diversas formas, de acordo com a metodologia de desenvolvimento utilizada. Em uma abordagem ágil, onde é seguido o princípio do manifesto de que a “maior prioridade é satisfazer o cliente, através de entrega adiantada e contínua de software de valor” [(MANIFESTO, 2001)] estas entregas ocorrem de forma contínua e automatizada.

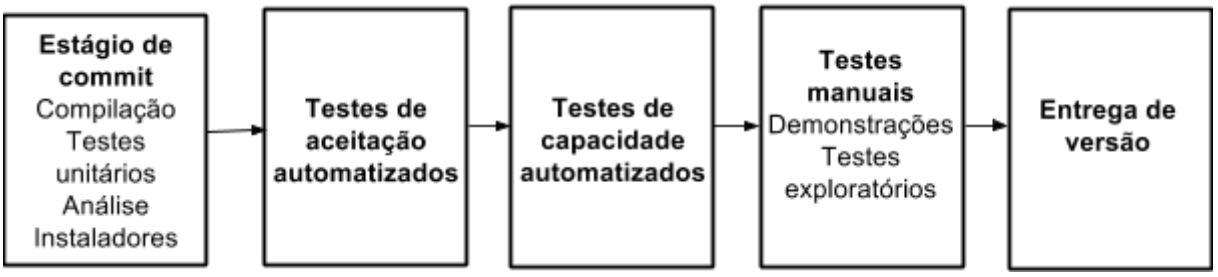
2.3 Entrega Contínua de Software

A entrega contínua trata-se de uma disciplina de desenvolvimento de software, onde o software é construído de forma que possa ser disponibilizado ao

cliente a qualquer momento. (FOWLER, 2017). Para realizar entrega contínua é necessário integrar constantemente o software pronto, buildings executáveis e rodar testes automatizados nos executáveis para detectar possíveis problemas. Desta forma o cliente, ou pessoa responsável pelo negócio, pode solicitar o deploy em produção da versão atual do software. (FOWLER, 2017).

O processo de entrega contínua possui um padrão, que pode variar de acordo com a organização e sua cadeia de valor para entrega de software. (HUMBLE; FARLEY, 2014). Este padrão é chamado de pipeline de implantação, que é “uma implementação automatizada do processo de compilar todas as partes de uma aplicação, implantá-la em um ambiente qualquer - seja de homologação ou produção - testá-la e efetuar sua entrega final”. Este está demonstrado na Figura 1.

Figura 1 - O *pipeline* de implantação



Fonte: Adaptado de Humble (2014, p. 4).

Desta forma se têm a automação dos testes e das instalações necessárias para colocar o software em produção, bem como torna possível entregar ou implantar qualquer versão do software para qualquer ambiente a qualquer momento que se desejar. (HUMBLE; FARLEY, 2014).

Martin Fowler afirma existir 3 principais vantagens na utilização de entrega contínua de software: redução de riscos de desenvolvimento, progresso crível e feedback do usuário o mais rápido e frequente. (FOWLER, 2017). Para Jez Humble e David Farley (2014) o principal objetivo é criar um processo de entrega que é possível de repetir, é previsível e é confiável. Por ser passível de repetição este reduz o tempo de ciclo, que consequentemente entrega mais rapidamente aos usuários novos projetos e correções de erros. (HUMBLE; FARLEY, 2014)

3 METODOLOGIA

Esta Seção apresenta as principais decisões subjacentes ao projeto experimental deste estudo. A Seção 3.1 apresenta os objetivos e questões de pesquisa. Seção 3.2 apresenta sistematicamente as hipóteses baseadas nas questões de pesquisa. Seção 3.3 descreve as variáveis e os métodos de quantificação. Seção 3.4 explica o contexto e a seleção das amostras. Seção 3.5 introduz o projeto experimental. E, por fim, a seção 3.6 descreve os procedimentos de análise.

3.1 Objetivo e Questões de Pesquisa

Este estudo tem como objetivo avaliar os impactos em uma organização da utilização de entrega contínua no desenvolvimento de software, utilizando quatro variáveis: satisfação dos usuários, número de defeitos, quantidade de demandas entregues e qualidade do código. Estes impactos são investigados em cenários reais de uma organização de forma a gerar conhecimento prático. Sendo assim o objetivo deste estudo é apresentado baseado no paradigma Objetivo Questão Métrica (do inglês, Goal Question Metric, GQM) [(BASILI, 1994)] que segue:

Analisar *processo de entrega contínua*
para o propósito de *investigar seus efeitos*
com respeito a *qualidade de código e qualidade dos produtos entregues*
a partir da perspectiva da *organização*
no contexto de *evoluir processos de desenvolvimento de software.*

Após apresentar o objetivo do estudo, duas questões de pesquisa foram formatadas para refinar a investigação:

- **QP01:** Qual o impacto da entrega contínua na qualidade do código produzido?

- **QP02:** Qual o impacto da entrega contínua na qualidade dos produtos entregues?

3.2 Hipóteses

3.2.1 Primeira hipótese: impacto da entrega contínua na qualidade do código

Na primeira hipótese, especula-se que trabalhar com processos automatizados pode levar a maior qualidade do código disponibilizado; uma vez que, é possível identificar em escala, e de forma mais rápida, os problemas de qualidade de código e corrigi-los antes da conclusão do desenvolvimento. Segundo Martin Fowler (2017), para que os processos de atualização de software sejam realizados rapidamente e sem erros é necessário um alto grau de automação [(FOWLER, 2017)], processos estes como testes automatizados e análise estática de código. Portanto, é possível identificar e corrigir erros de codificação para entregar código com mais qualidade. Esta hipótese avalia, portanto, se a qualidade do código com o processo de entrega contínua, é maior comparada a utilização de metodologias tradicionais não automatizadas.

3.2.2 Segunda hipótese: impacto da entrega contínua na qualidade dos produtos entregues

Na segunda hipótese, especula-se que reduzir o tamanho do lote entregue pode levar a maior qualidade dos produtos; uma vez que, é possível obter mais rapidamente *feedback* do usuário. Segundo Eric Ries, a lição fundamental da entrega contínua não é realizar diversas entregas por dia, mas que ao reduzir o tamanho do lote entregue, seja possível atravessar o ciclo de feedback construir-medir-aprender com mais rapidez que os concorrentes. (RIES, 2011). Sendo assim, é possível aprender mais rápido com os clientes e entregar produtos com mais qualidade. Esta hipótese avalia, portanto, se a qualidade dos produtos com o processo de entrega contínua, é maior comparada a utilização de metodologias tradicionais que trabalham com grandes lotes.

3.3 Variáveis do Estudo e Método de Quantificação

Primeira variável dependente. A variável dependente da hipótese 1 é a qualidade do código. Que quantifica número de vulnerabilidades, número de *bugs*, número de *code smells*, número de complexidades e percentual de duplicações. (ver Tabela 1). Entendemos como aumento da qualidade a diminuição dessas métricas.

Tabela 1 - Métricas usadas para analisar a QP1.

Métrica	**Descrição**
Número de vulnerabilidades	O número de vulnerabilidades de segurança
Número de *bugs*	O número de bugs que representam algo de errado no código
Número de *code smells*	O número de problemas de manutenção
Número de complexidades	O número de caminhos através do código
Percentual de duplicações	O número de densidade de duplicações no código

Fonte: elaborado pela autora.

Segunda variável dependente. A variável dependente da hipótese 2 é a qualidade dos produtos. Ela quantifica número de *feedbacks*, número de demandas entregues e número de defeitos (ver Tabela 2). Entendemos como aumento da qualidade a diminuição do número de defeitos e o aumento no número de feedbacks e de demandas entregues.

Tabela 2 - Métricas usadas para analisar a QP2.

Métrica	**Descrição**
Número de feedbacks	O número de feedbacks dados pelo usuário
Número de demandas entregues	O número de demandas entregues ao usuário
Número de defeitos	O número de defeitos reportados pelo usuário

Fonte: elaborado pela autora.

Variável independente. A variável independente das hipóteses 1 e 2 é o processo de entrega contínua que contém, diferentemente do processo tradicional, testes automatizados, análise estática de código e execução de apenas uma demanda por vez até a disponibilização aos usuários.

3.4 Contexto

Para preservar a identidade da organização, devido às informações fornecidas para este estudo, será utilizado o codinome “Alpha” para referenciar a mesma e o codinome "Omega" para referenciar o principal software desenvolvido pela organização, que será o produto analisado neste estudo.

A organização Alpha realiza desenvolvimento e implantação de software para informatização dos processos de diversos órgãos públicos, tais como Prefeituras, Câmaras de Vereadores, Tribunais de Contas, Ministérios Públicos e etc. Os produtos da Alpha são aplicações voltadas para internet (*web*) e de código aberto (*Open Source*) que possuem como objetivo apoiar a tomada de decisão e a gestão das entidades da administração pública.

Antes de realizar Entrega Contínua de Software, a metodologia utilizada para o desenvolvimento era o Scrum, um *framework* de desenvolvimento iterativo incremental baseado em metodologias ágeis. Os ciclos de desenvolvimento que formam os *releases* eram de 27 dias, composto por duas *sprints*. Uma *release* é composta por diversas melhorias, nas diversas áreas de conhecimento. Sendo assim cada atualização de *release* nos clientes significava, por ser um ERP, mudanças nas diversas áreas de uma entidade pública, como área Financeira, Tributária, Patrimonial e etc., o que impacta a vida profissional de diversos usuários do sistema. É comum que existam erros em uma *release*, e quando estes erros ocorrem somados com a resistência do usuário às mudanças disponibilizadas, resulta em um aumento dos *feedbacks* dados.

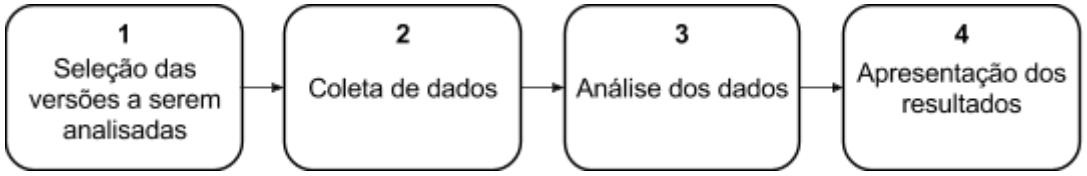
Tendo em vista este cenário a empresa percebeu a necessidade de buscar alternativas para realizar entrega de software aos seus clientes, de forma que fosse

possível diminuir a resistência a mudanças, bem como diminuir os defeitos em produção para garantir a confiabilidade das entregas. Desta forma em junho de 2016 iniciou-se, gradualmente, a implementação do processo de entrega contínua em todos os clientes da empresa. O objetivo era realizar entregas mais frequentes e diminuir a resistência a atualização de versão nos clientes, bem como automatizar processos manuais.

3.5 Estudo Experimental

A Figura 2 apresenta os quatro passos deste estudo experimental, após são descritos detalhadamente cada um deles.

Figura 2 - Passos do estudo



Fonte: elaborado pela autora.

Passo 1 teve como objetivo realizar a seleção das versões a serem analisadas. Conforme explicado, na Seção 3.4, o processo de transição para entrega contínua iniciou-se na Alpha em junho de 2016, onde implantou em um primeiro cliente para utilização e observação. Nos demais clientes, o processo foi implantado gradualmente até ser integralmente implantado em todos os clientes em dezembro de 2016. Portanto, o critério para escolha das versões selecionadas, para realizar o estudo experimental, foi selecionar as versões onde os processos estavam integralmente implantados, tanto o processo anterior à entrega contínua quanto o com a entrega contínua (ver Tabela 3).

Tabela 3 - Versões selecionadas para o estudo

Antes da Entrega Contínua		Após a Entrega Contínua	
Versão 1	Janeiro 2016	Versão 7	Janeiro 2017
Versão 2	Fevereiro 2016	Versão 8	Fevereiro 2017
Versão 3	Março 2016	Versão 9	Março 2017
Versão 4	Abril 2016	Versão 10	Abril 2017
Versão 5	Maio 2016	Versão 11	Maio 2017
Versão 6	Junho 2016	Versão 12	Junho 2017

Fonte: elaborado pela autora.

Passo 2 teve como objetivo coletar os dados das métricas. Para coletar os dados das métricas detalhadas na Seção 3.3 foram utilizados três softwares, todos eles são Open Source:

1. Omega: software desenvolvido na linguagem PHP pela empresa Alpha, onde são disponibilizadas todas novas demandas. É através do Omega também que os usuários dão *feedbacks* para as demandas utilizadas por eles. Utilizado neste estudo para coletar os dados dos *feedbacks* dos usuários.
2. *RedMine*[3]: software que realiza gestão de projetos e de defeitos. É através do *Redmine* que a Alpha faz gestão de todas as demandas e defeitos disponibilizados no Omega. Utilizado neste estudo para coleta dos dados de quantidade de defeitos reportados e quantidade de demandas entregues.
3. *SonarQube*[4]: software que realiza análise estática de código. Utilizado neste estudo para varrer o código fonte do Omega e coletar métricas de qualidade de código.

Passo 3 teve como objetivo construir as tabelas das estatísticas descritivas das métricas descritas anteriormente, bem como seus gráficos para realização dos procedimentos de análise. A análise consistiu em observar os dados coletados antes

[3] "Redmine." https://www.redmine.org/. Acessado em 24 nov. 2017.
[4] "SonarQube." https://www.sonarqube.org/. Acessado em 24 nov. 2017.

e depois do processo de entrega contínua, constatando seu impacto, seja aumento ou diminuição, das variáveis dependentes. Afim de verificar a se as hipóteses, levantadas se confirmam.

Passo 4 teve como objetivo a apresentação dos resultados das observações dos impactos da utilização da entrega contínua, de acordo com a perspectiva das variáveis dependentes.

4 ANÁLISE DOS RESULTADOS

Esta seção analisa os dados obtidos dos procedimentos experimentais apresentados na Seção 3.5. Para isso, a Seção 4.1 apresenta os dados coletados para testar a primeira hipótese deste estudo. A Seção 4.2 introduz os dados coletados para testar a segunda hipótese.

4.1 QP01: Qual impacto da entrega contínua na qualidade do código gerado?

Tabela 4 demonstra as estatísticas descritivas computadas das métricas de qualidade de código (descritas na Seção 3.3), onde a primeira linha demonstra os dados das versões antes da utilização do processo de entrega contínua e a segunda linha demonstra os dados das versões após a utilização. Para análise das métricas de código foram coletadas e analisadas 5 versões antes e 5 versões depois da adoção da entrega contínua.

Tabela 4 - Estatísticas descritivas

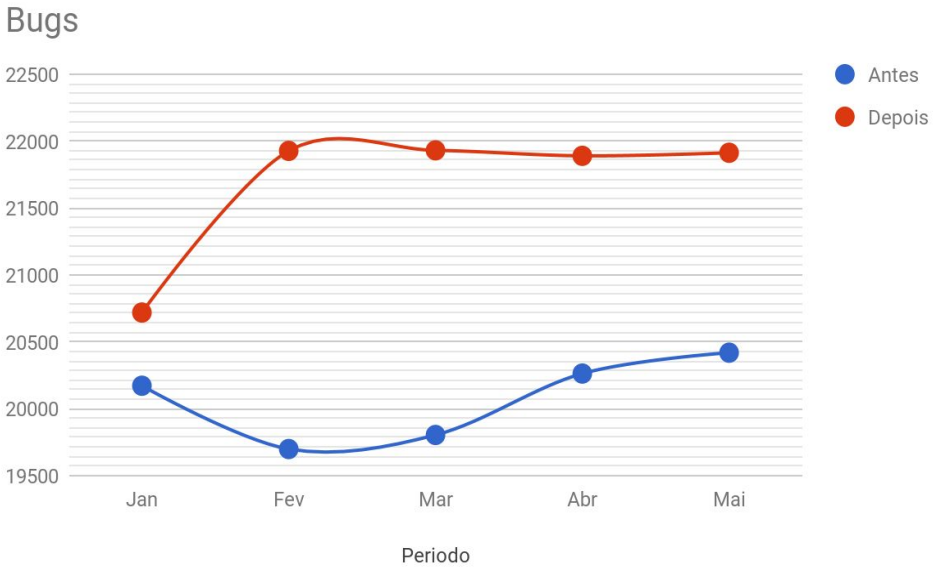
Métricas	mínimo	25th	mediana	75th	máximo	Desvio Padrão	média	% diff
Bug (Antes)	19.699	19804	20.172	20263	20.419	307,3439441	20.071	7,40%
Bug (Depois)	20.719	21890	21.913	21927	21.931	535,2195811	21.676	
Vulnerabilidades (Antes)	1.149	1164	1.171	1172	1.415	112,6263735	1.214	8,02%
Vulnerabilidades (Depois)	1.220	1344	1.344	1345	1.347	55,91511424	1.320	
Code Smells (Antes)	269.554	270504	271.325	273169	281.520	4829,588005	273.214	2,31%
Code Smells (Depois)	277.059	278117	280.763	280881	281.493	1950,215065	279.663	
% Duplicações (Antes)	18,2	18,3	18,7	19	20,1	0,7635443668	18,86	7,74%
% Duplicações (Depois)	11,6	18,7	18,8	18,9	19	3,24422564	17,4	
Linhas de código (Antes)	3.528.759	3546769	3.569.263	3584887	3.633.496	40186,11869	3.572.635	7,80%
Linhas de código (Depois)	3.777.236	3882247	3.901.752	3906317	3.907.449	55587,04952	3.875.000	
Complexidade (Antes)	654.728	658298	662.356	665402	676.830	8464,948742	663.523	6,16%
Complexidade (Depois)	695.434	708171	708.731	711370	711.670	6689,870754	707.075	

Fonte: elaborado pela autora.

Bug. O número de *Bugs* foi calculado com base no número de erros encontrados no código, que estão ligados a confiabilidade do código.

A Figura 3 apresenta um gráfico com os dados coletados referentes ao número de *Bugs*. O eixo-X representa as versões analisadas, enquanto o eixo-Y consiste nos valores do número de *Bugs*. Não utilizando a entrega contínua (Antes) observa-se que o número era menor que o número de Bugs utilizando entrega contínua (Depois). O principal resultado é que o número de *Bugs* aumentou com a adoção da entrega contínua.

Figura 3 - Gráfico de *Bugs*



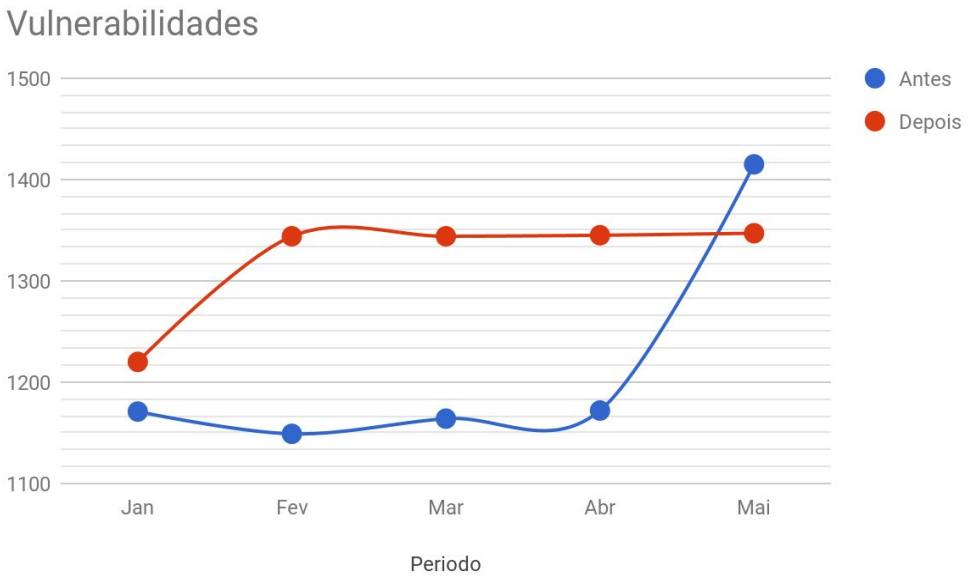
Fonte: elaborado pela autora.

Vulnerabilidade. O número de Vulnerabilidades foi calculado com base no número de vulnerabilidades de segurança encontrados no código.

A Figura 4 apresenta um gráfico com os dados coletados referentes ao número de Vulnerabilidades. O eixo-X representa as versões analisadas, enquanto o eixo-Y consiste nos valores dos números de Vulnerabilidades. Não utilizando a entrega contínua (Antes) observa-se que o número era menor que o número de Vulnerabilidades utilizando entrega contínua (Depois). Além disso é possível observar, que passado o primeiro mês da utilização da entrega contínua houve uma estabilidade no número de Vulnerabilidades disponibilizadas no código. Conforme apresentado na Tabela 4 o número máximo de vulnerabilidades foi menor na utilização da entrega contínua (Depois).

O principal resultado é que o número de Vulnerabilidades também aumentou com a adoção da entrega contínua.

Figura 4 - Gráfico de Vulnerabilidades

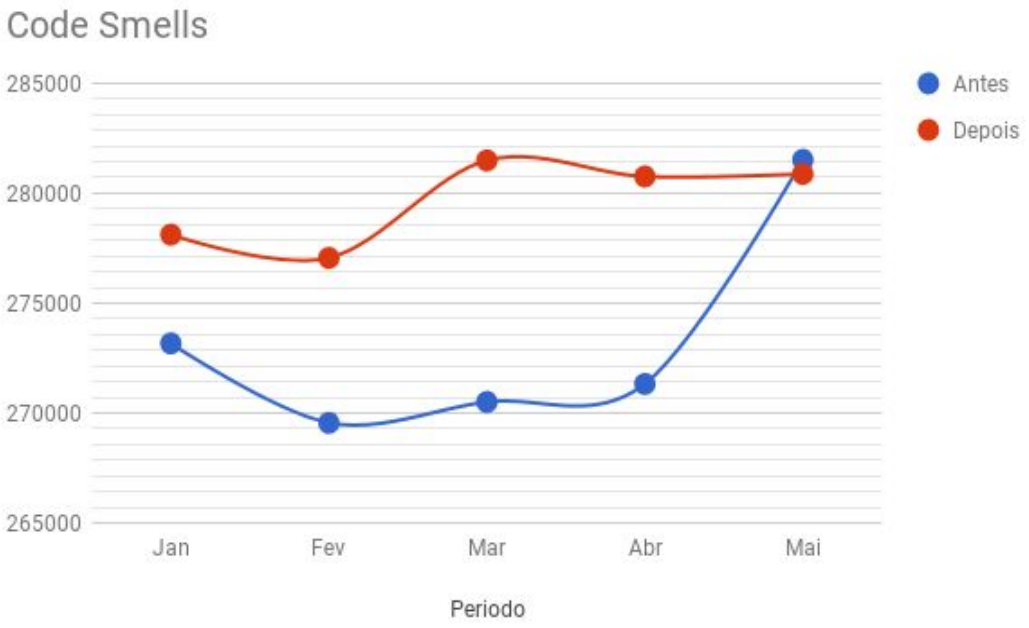


Fonte: elaborado pela autora.

Code Smells. O número de *Code Smells* foi calculado com base no número de problemas para manutenções posteriores no códigos.

A Figura 5 apresenta um gráfico com os dados coletados referentes ao número de *Code Smells*. O eixo-X representa as versões analisadas, enquanto o eixo-Y consiste nos valores dos números de Code Smells. Não utilizando a entrega contínua (Antes) observa-se que o número era menor que o número de *Code Smells* utilizando entrega contínua (Depois). Além disso é possível observar que apenas no último mês o número depois da utilização da entrega contínua foi menor. O principal resultado é que o número de *Code Smells* aumentou com a adoção da entrega contínua.

Figura 5 - Gráfico de *Code Smells*

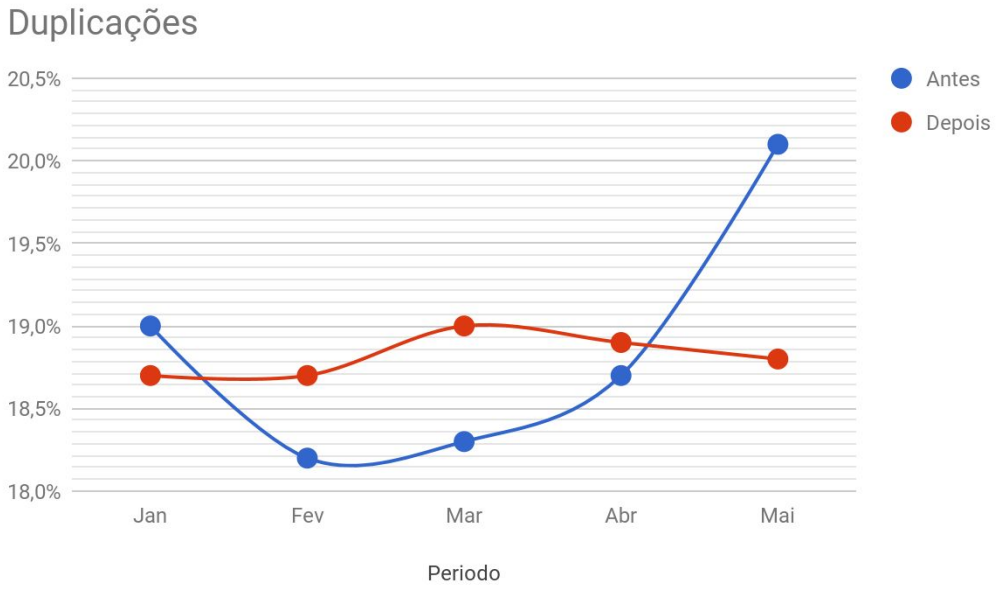


Fonte: elaborado pela autora.

Duplicações. O número de Duplicações foi calculado de acordo com o percentual de duplicações no código.

A Figura 6 apresenta um gráfico com os dados coletados referentes ao número de Duplicações. O eixo-X representa as versões analisadas, enquanto o eixo-Y consiste nos valores dos números de Duplicações. Não utilizando a entrega contínua (Antes) observa-se uma tendência de aumento do número de duplicações, e na utilização da entrega contínua (Depois) uma tendência de diminuição, em média as Duplicações diminuíram com a entrega contínua. Além disso é possível uma maior estabilidade no número com a utilização da entrega contínua (Depois), onde existe uma menor diferença de valores entre os períodos. O principal resultado observado é que o número de Duplicações diminuiu com a adoção da entrega contínua.

Figura 6 - Gráfico de Duplicações

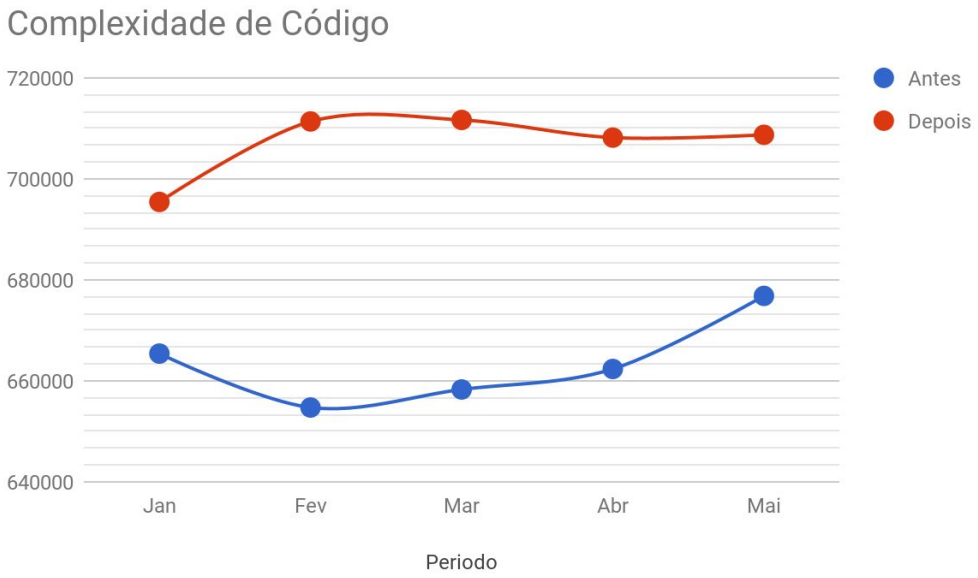


Fonte: elaborado pela autora.

Complexidade. O número de Complexidades foi calculado com base no número de caminhos através do código. Sempre que o fluxo de uma função se divide, o contador de complexidade é incrementado em 1.

A Figura 7 apresenta um gráfico com os dados coletados referentes ao número de Complexidades. O eixo-X representa as versões analisadas, enquanto o eixo-Y consiste nos valores dos números de Complexidades. Não utilizando a entrega contínua (Antes) observa-se que número de complexidades era menor que na utilização da entrega contínua (Depois). Dentre todas as variáveis analisadas de qualidade de código essa é a que possui o maior aumento após a entrega contínua.O principal resultado observado é que o número de Complexidade aumentou com a adoção da entrega contínua.

Figura 7 - Gráfico de Complexidade



Fonte: elaborado pela autora.

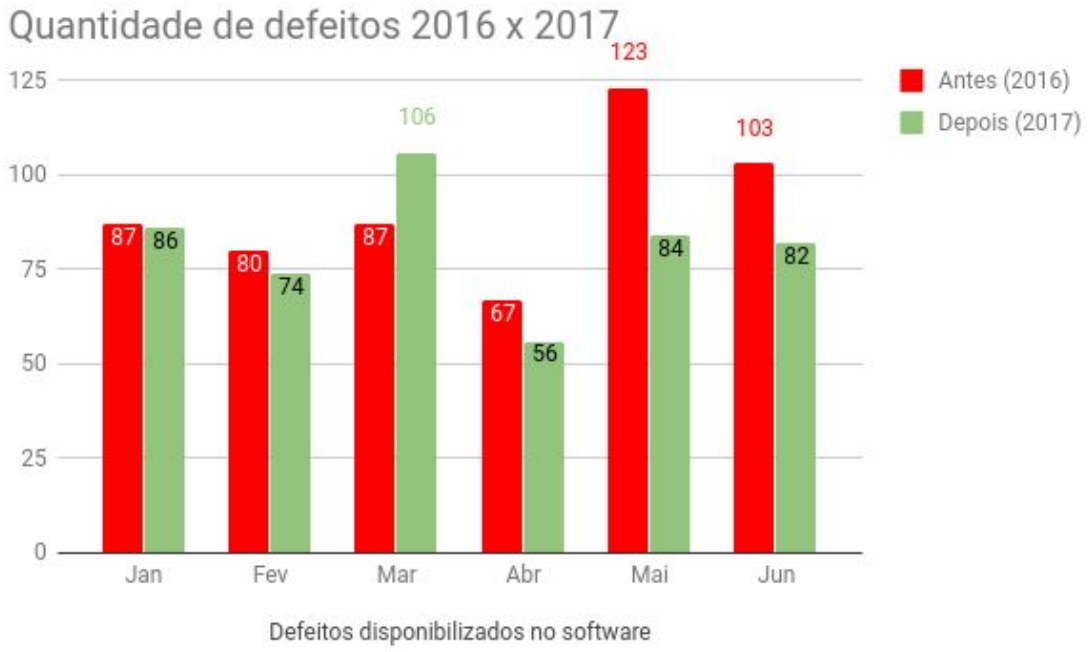
4.2 QP02: Qual o impacto da entrega contínua na qualidade dos produtos entregues?

Para análise das métricas de qualidade dos produtos entregues foram coletadas e analisadas 6 versões antes e 6 versões depois da adoção da entrega contínua.

Defeitos. Com a utilização da entrega contínua foram implementados testes automatizados que são sempre realizados no processo de *deploy*. Dessa forma o software não é disponibilizado até que todos os defeitos encontrados pelos testes automatizados tenham sido corrigidos. Os resultados associados a este fator de influência são demonstrados na Figura 7, que apresenta um gráfico onde é possível observar que na maioria das versões após a entrega contínua contêm um número menor de defeitos. Com a utilização da entrega contínua foram em média

disponibilizados 10 defeitos a menos no software. Este resultado confirma o que diz Martin Fowler (2017) a respeito da automação dos processos de atualização de software, onde afirma que um alto grau de automação permite que atualização ocorra rapidamente e com menos erros. (FOWLER, 2017).

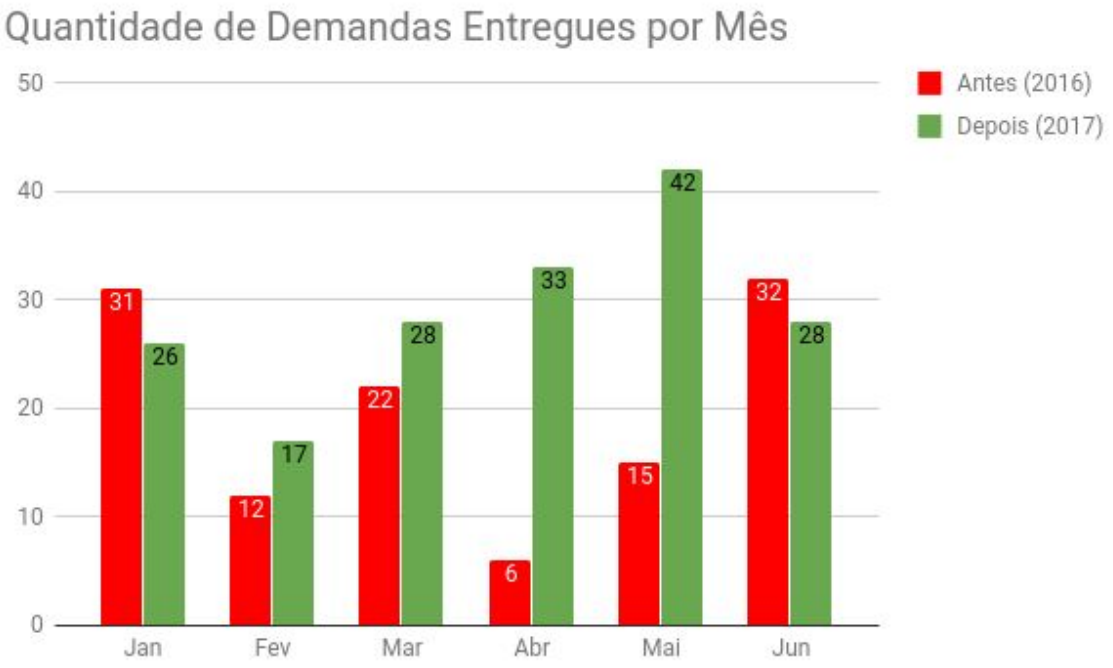
Figura 7 - Gráfico de Defeitos



Fonte: elaborado pela autora.

Demandas Entregues. Com a utilização da EC os times de desenvolvedores, analistas e testadores passaram a planejar entregas menores para desenvolver. Anterior ao processo eram planejados itens para um mês inteiro de trabalho, e após o processo o planejamento ocorre item a item. Os resultados associados a este fator de influência são demonstrados na Figura 8, que apresenta um gráfico onde pode ser visto o aumento de demandas entregues após a adoção da entrega contínua. Em média foram entregues 9 demandas a mais após a utilização do processo.

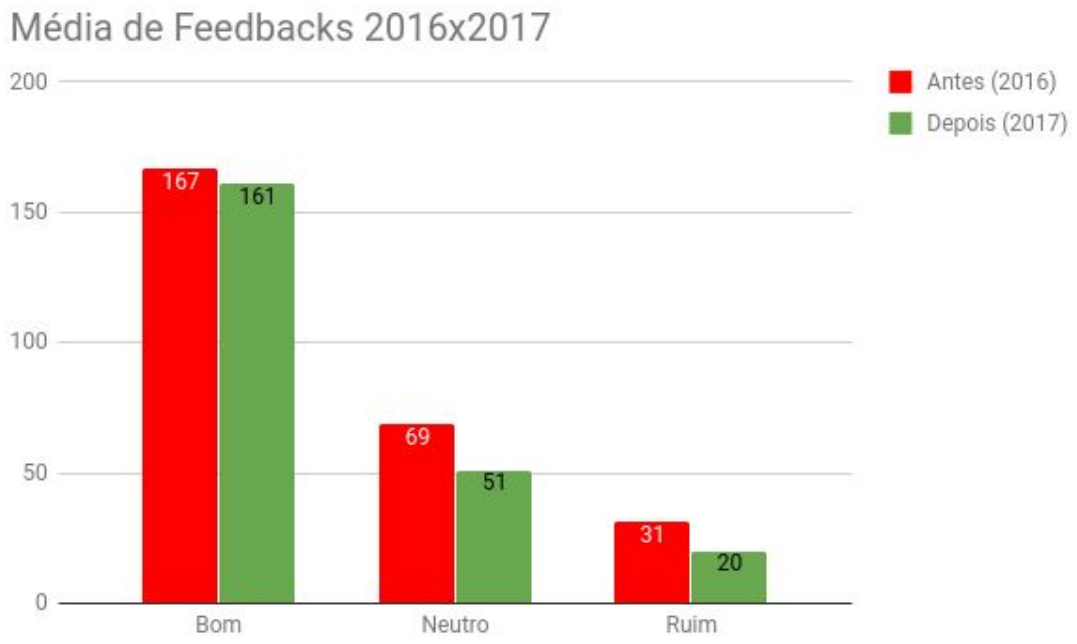
Figura 8 - Gráfico de Demandas Entregues



Fonte: elaborado pela autora.

Satisfação dos Usuários. Com a utilização da entrega contínua as demandas deixaram de ser disponibilizadas em lote aos usuários, que resultava no usuário receber muitas mudanças de uma única vez e passou a ocorrer de forma diluída. Os resultados associados a este fator de influência são demonstrados (na Figura 9 e na Tabela 5 abaixo), onde é possível observar que ao adotar o processo de entrega contínua o número médio de *feedbacks* realizados pelos clientes diminuiu. Este resultado confirma o que diz Eric Ries a respeito da lição fundamental da entrega contínua, que trata-se não de realizar diversas entregas por dia, mas que ao reduzir o tamanho do lote entregue se possa atravessar o ciclo de feedback construir-medir-aprender com mais rapidez que os concorrentes. Sendo assim, é possível aprender mais rápido com os clientes e entregar produtos mais assertivos. (RIES, 2011).

Figura 9 - Gráfico de média de *Feedbacks* dos usuários



Fonte: elaborado pela autora.

Tabela 5 - Quantidade de *Feedbacks* dos usuários

	Jan	Fev	Mar	Abr	Mai	Jun	Média
2016 - ruim	42	27	37	19	27	33	31
2016 - neutro	90	56	67	67	71	62	69
2016 - bom	275	167	142	144	161	113	167
2017 - ruim	18	15	12	33	27	16	20
2017 - neutro	50	41	45	44	70	55	51
2017 - bom	157	140	179	104	154	234	161

Fonte: elaborado pela autora.

5 Trabalhos Relacionados

A evolução da indústria de desenvolvimento de software tem evoluído e com isso, no meio acadêmico, estudos focados em princípios e práticas ágeis têm ganhado espaço. Porém, quando se trata especificamente, de entrega contínua de

software pouco se tem relato na literatura. Sendo assim, foram realizadas pesquisas com o objetivo de localizar trabalhos publicados em bibliotecas digitais, tais como IEEE Xplore[5].

5.1 Análise dos Trabalhos Selecionados

No total, 6 trabalhos foram selecionados, os quais são descritos e analisados a seguir.

Chen (2015) explica a razão de a empresa, Paddy Power, decidir adotar entrega contínua. A empresa atua com apostas onlines, e possuía em 2015, aproximadamente 4 mil funcionários, um grande volume de transações em seus softwares, cerca de 6 releases (entregas) ao ano nas atualizações de software, muitos defeitos e uma grande quantidade de trabalho manual para atualização. Tendo em vista esse cenário a empresa decide pela adoção da entrega contínua, sua prioridade é diminuir erros causados pela configuração manual nos processos de atualização. O artigo apresenta a análise da qualidade dos produtos entregues, onde o número de defeitos diminuiu após a adoção da entrega contínua. Além disso o artigo comenta que a satisfação dos clientes aumentou, porém não detalha como esse aumento foi medido. (CHEN, 2015).

Neely e Stolt (2013) reporta a experiência da adoção da entrega contínua na empresa Rally Software. Fundada em 2002 e atualmente provê soluções na nuvem (*cloud*) para gestão de desenvolvimento de software ágil. A transição para a Rally Software foi ir de uma cadência de *release* de 8 semanas para entrega contínua de software, onde foram observados impactos na perspectiva de negócio e de engenharia. A principal razão para adoção da entrega contínua foi entregar funcionalidades mais rapidamente aos clientes, ao invés de todas ficarem esperando o fechamento da *release*. A conclusão que se chegou é que a adoção da entrega contínua aumento a quantidade de demandas/projetos entregues, e que os defeitos

[5] "IEEE Xplore Digital Library." http://ieeexplore.ieee.org/. Acessado em 24 nov. 2017.

reportados estão estreitando e ficando mais fácil de prever, porém não é descrito como se chegou nestas conclusões. (NEELY; STOLT, 2013).

Krusche, Alperowitz, Bruegge e Wagner (2014) descrevem um modelo processo ágil chamado Rugby que inclui fluxos de trabalho para a entrega contínua. Nos anos de 2012 e 2013 eles avaliaram Rugby em duas grandes universidades de engenharia de software com 100 participantes trabalhando em 10 simulações de projetos. Objetivo principal era avaliar o *feedback* dos participantes a respeito do modelo, onde chegaram a conclusão que a adoção melhorar a comunicação do time de desenvolvimento como a comunicação com o cliente. Assim como a entrega passou a ser mais rápida, os *feedbacks* dos usuários também, o que permitiu seus usuários recebessem soluções mais rapidamente. Este estudo foi realizado simulando uma situação real com equipes de estudantes, não foi realizado em uma organização. (KRUSCHE; ALPEROWITZ; BRUEGGE, WAGNER, 2014).

Akerele, Ramachandran e Dixon (2014) investigam vários fatores, práticas ágeis em especial, sobre a qualidade do software desenvolvido no processo de entrega contínua. O estudo que eles realizaram tinha o objetivo de desenvolver uma dinâmica de sistema que permitisse a novos adotantes de entrega contínua controle sobre os fatores de riscos de entrega. Para isso investigaram todos os fatores pertinentes a entrega contínua e desenvolveram um modelo de uma dinâmica de sistema para atuar na tomada de decisões sobre melhorias de processos para times praticantes de entrega contínua. No estudo não fica claro o que eles definem como qualidade do software desenvolvido. (AKERELE; RAMACHANDRAN; DIXON, 2014)

5.2 Análise Comparativa dos Trabalhos

Nesta seção, uma comparação dos trabalhos relacionados é apresentada, considerando algumas características dos mesmos. O Quadro 1 organiza essas características em critérios de avaliação e apresenta uma comparação dos estudos descritos anteriormente. Os critérios de comparação são:

- Estudo Experimental: estudos que buscam estabelecer relação de causa e efeito.
- Indústria: estudos realizados em organizações da indústria.
- Defeitos em produção: estudos que avaliam quantidade de defeitos disponibilizados em produção.
- Quantidade entregue: estudos que avaliam quantidade de demandas/projetos entregues.
- Satisfação dos Usuários: estudos que avaliam a satisfação do cliente.
- Qualidade de Código: estudos que avaliam a qualidade do código que está sendo desenvolvido.

Quadro 1 - Comparativo entre trabalhos

Trabalhos	**Critérios**					
	Estudo Experimental	**Indústria**	**Defeitos em produção**	**Quantidade entregue**	**S*atisfação* dos Usuários**	**Qualidade de Código**
Este estudo experimental	√	√	√	√	√	√
Chen (2015)	√	√	√	-	√	-
Neely e Stolt (2013)	√	√	√	~	-	-
Krusche, Alperowitz, Bruegge e Wagner (2014)	√	-	∅	-	√	-
Akerele, Ramachandran e Dixon (2014)	∅	-	~	-	~	∅

Notas: (-) Não atende (~) Atende parcialmente (√) Atende Plenamente (∅) Não Aplicável

Fonte: elaborado pela autora.

6 CONSIDERAÇÕES FINAIS

Este estudo foi realizado na organização Alpha com uma amostra de 10 meses de dados para métricas de qualidade de código, sendo 5 meses antes da utilização do processo de entrega contínua e 5 após. E uma amostra de 12 meses de dados para métricas de qualidade dos produtos entregues, sendo 6 meses antes da utilização e 6 após. Foram analisados os impactos da utilização da entrega contínua em uma organização da indústria, onde entende-se como impacto a qualidade de código e de produtos entregues. Analisando os resultados, pode-se verificar em quais aspectos pode-se observar impacto, respondendo assim às duas questões de pesquisa que nortearam este estudo.

Como contribuição, este estudo apresenta os conhecimentos empíricos adquiridos na utilização do processo de entrega contínua para outras empresas que se interessem também pela adoção. Além disso contribui com o aumento de publicações a respeito deste assunto.

Conforme os resultados obtidos a respeito de qualidade dos produtos entregues, conclui-se que o processo de entrega contínua, que inclui a automatização de testes, teve influência na diminuição de defeitos disponibilizados no software, uma vez que tem possibilitado a correção de defeitos antes do usuário ter contato com o software. O processo inclui também o planejamento de uma demanda por vez, que resultou influência na quantidade de demandas entregues aos usuários. E que por entregar aos usuários pequenos lotes influenciou também na resistência dos usuários na utilização do sistema, que se refletiu numa diminuição de número de feedbacks dados. Quanto a qualidade de código observou-se, em sua maioria, uma estabilidade na incidência de ocorrência de problemas, porém não houve uma diminuição.

Uma oportunidade para trabalhos futuros, onde pode-se realizar esta pesquisa em uma amostragem maior, seria avaliar outras métricas que podem estar relacionadas a qualidade de código e qualidade de produtos entregues. Além disso,

são encontrados poucos estudos que tratam dos impactos da entrega contínua nas organizações e isto reforça a necessidade de mais abordagens sobre o assunto.

IMPACT ANALYSIS OF THE USE OF CONTINUOUS DELIVERY OF SOFTWARE IN AN ORGANIZATION:

An Experimental Study

Abstract: The process of continuous delivery was adopted by organizations to provide the software for their users at any time. The transition between traditional methodology of software delivery to continuous delivery can cause impacts in the results generated by organizations, like the quality of the products developed. But little is known about the impacts of continuous delivery in the organizations that adopt. For this reason, this experimental study, analyzes the quality of products developed in an organization before and after the adoption of the process, through of the following perspectives: quality of software and quality of products delivered. The results presented demonstrate which aspects of quality is impacted by the transition to continuous delivery. This job contributes both for the academic environment and practice, through of the empirical knowledge acquired during the experimental study.

Keywords: Software Development. Software Engineering. Continuous Delivery.

REFERÊNCIAS

SABBAGH, R. **Scrum: Gestão ágil para projetos de sucesso**. 1. ed. São Paulo: Casa do Código, 2013.

DAIRTON, B. Programação Extrema (XP). In: PRIKLADNICKI, R.; WILLI, R.; MILANI, F. (Org.). **Métodos ágeis para desenvolvimento de software**. 1. ed. Porto Alegre: Bookman, 2014. p. 37-57.

ANDERSON, D. **Kanban**: successful evolutionary change for your technology business. Sequim, Washington: Blue Hole Press, 2011.

BASILI, V., CALDIERA, G., ROMBACH, H.: **The goal question metric paradigm**. In: Encyclopedia of Software Engineering, vol. 2, pp. 528–532. Wiley, Hoboken, 1994.

PRESSMAN, R. **Engenharia de Software**. McGraw-Hill, 2001.

MANIFESTO for agile software development, [S.l.], 2001. Disponível em <http://agilemanifesto.org>. Acesso em: 27 setembro 2017.

HUMBLE, J., FARLEY, D. **Entrega Contínua: Como Entregar Software de Forma Rapida e Confiavel.** bookman, 2014.

RIES, E. **The lean startup.** New York: Crown Business, 2011.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson, 2011. 9. ed. 532 p.

MAXIM, B.; PRESSMAN, R. **Engenharia de Software.** bookman, 2016.

PRIKLADNICKI, R.; AUDY, J. L. N.; EVARISTO, R. **Global Software Development in Practice Lessons.** Software Process Improvement and Practice Learned, 2003. pp. 267-281 . Disponível em: <http://www.inf.pucrs.br/munddos/docs/JournalGSD2003_camera_ready_1column.pdf> Acesso em: 23 maio. 2017.

FUGGETTA, A., 2000, **Software Process: A Roadmap**. In: FINKELSTEIN, A. (ed.), The Future of Software Engineering.

FOWLER, M. **ContinuousDelivery**. Disponível em: <https://martinfowler.com/bliki/ContinuousDelivery.html> Acesso em 21 jun. 2017.

CHEN, L. **Continuous Delivery: Huge Benefits, But Challenges Too**, 2015. Disponível em <https://www.researchgate.net/publication/271635510_Continuous_Delivery_Huge_Benefits_but_Challenges_Too>. Acesso em: 21 jun. 2017

NEELY, S.; STOLT, . **Continuous Delivery? Easy! Just Change Everything (well, maybe it is not that easy)**, 2013. Disponível em

<http://ai2-s2-pdfs.s3.amazonaws.com/e488/7a8d007ff8360b3e9d882cf97818fe60f4e2.pdf> Acesso em: 21 jun. 2017

KRUSCHE, S.; ALPEROWITZ, L.; BRUEGGE, B.; WAGNER, M. **Rugby: An Agile Process Model Based on Continuous Delivery**, 2014. Disponível em <https://www.researchgate.net/publication/262450962_Rugby_An_Agile_Process_Model_Based_on_Continuous_Delivery> Acesso em: 21 jun. 2017

AKERELE, O.; RAMACHANDRAN, M; DIXON, M; **Evaluating the Impact of Critical Factors in Agile Continuous Delivery Process: A System Dynamics Approach**, 2014. Disponível em <https://thesai.org/Downloads/Volume5No3/Paper_19-Evaluating_the_Impact_of_Critical_Factors_in_Agile_Continuous_Delivery_Process_A_System_Dynamics_Approach.pdf> Acesso em: 24 nov. 2017

VASCONCELOS, A. M., ROUILLER, A. C., MACHADO, C. A., MEDEIROS, T. M. **Introdução à Engenharia de Software e à Qualidade de Software.** Universidade Federal de Lavras - UFLA, 2006. Disponível em: <http://www.cin.ufpe.br/~if720/downloads/Mod.01.MPS_Engenharia&QualidadeSoftware_V.28.09.06.pdf> Acesso em: 16 abr. 2017.