Um Processo de Introdução de DevOps em Sistemas Legados - A DevOps Introduction Process for Legacy Systems

Vinícius Lima Cruz
Universidade de Fortaleza
Av. Washinton Soares, 1321, BL J, SL 30, CEP 60833-155, Fortaleza, Ceara, Brazil
+5585988907959
vinnycruz@gmail.com

Adriano Bessa Albuquerque
Universidade de Fortaleza
Av. Washinton Soares, 1321, BL J, SL 30, CEP 60833-155, Fortaleza, Ceara, Brazil
+5585988983676
adrianoba@unifor.br

***Abstract* — Legacy systems are a challenge for the operations of modern organizations as they limit the growth of business and their changing ability. However, they are often systems suited to their purpose, they deliver the expected value and investment in them is justifiable. The concept of DevOps has come to reduce the separation between development and operations teams in software development companies, decreasing the product lifecycle. With this concept comes the practice of continuous delivery, allowing teams to deliver and deploy any version of their software in any computing environment at any time through a fully automated process. This practice improves the feedback of the development process, so problems could be identified and resolved as early as possible. In this paper, we present an experience of deploying DevOps, and propose a structured process for deploying DevOps and its modifications necessary to adapt to legacy systems in order to the delivery process of the legacy systems has a short and high quality lifecycle, delivering frequent versions in an automated way.**

Keywords— Legacy systems; DevOps; Continuous Integration; Continuous Deployment; Deployment Pipeline.

I. INTRODUÇÃO

Com a crescente concorrência no mercado de software, as organizações estão dando atenção significativa e alocando recursos para desenvolver e entregar software de alta qualidade a um ritmo acelerado [13]. Mas mesmo com todo o avanço da tecnologia nos últimos anos, a maioria das organizações ainda tem que conviver com sistemas legados, que muitas vezes são sistemas antigos, mas muito importantes para o negócio, difíceis de serem evoluídos e com uma série de riscos associados a qualquer mudança nesses sistemas [9][10][15].

Existem diversas iniciativas ágeis que estão sendo descritas na literatura e amplamente adotadas na indústria de software, para entregar produtos de alto valor ao mercado, de maneira rápida e alinhados a novas tendências [7][11]. O conceito de Integração Contínua (IC) foi um primeiro passo, inspirado em técnicas ágeis, que acelerou significativamente o ciclo de vida de um produto, incentivando desenvolvedores a integrar mais frequentemente as alterações em um armazenamento compartilhado [8]. Isso ajudou a detectar problemas logo após qualquer alteração, reduzido o tempo de integração, e desenvolvendo software mais rápido e com mais produtividade.

A maioria dos projetos falha devido a problemas de pessoal, e não a problemas técnicos. Em nenhum outro ponto do processo de desenvolvimento isso é tão válido quanto na parte que envolve a implantação em ambientes de teste e produção [11]. Quase todas as empresas de médio e grande porte separam as atividades de desenvolvimento e de infraestrutura (ou operações) em dois grupos diferentes. Cada time tem que ser especializado em suas próprias ferramentas e gerenciar uma fase diferente e isolada do ciclo de vida do produto. Passar de uma fase para a próxima, muitas vezes exige processos complexos, passos manuais e aprovações, fazendo com que as entregas sejam muito lentas para as expectativas do negócio.

A necessidade de melhorar o ciclo de vida em todas as diferentes fases e, especialmente, a grande competitividade do mercado, tendo que lançar produtos rapidamente se alinhando às tendências, foi o que originou o conceito de "Entrega Contínua" (CD) que, como extensão da Integração Contínua, defende a automação dos testes, dos processos de lançamento e das etapas de implantação [11]. Para isso acontecer, teve que ser reduzida a separação dos papéis entre as diferentes equipes e levou ao surgimento do movimento DevOps. O movimento DevOps está centrado em um objetivo: encorajar uma colaboração maior entre todos os envolvidos no processo de entrega de software de maior valor com mais confiança e rapidez.

Neste trabalho, apresentamos um estudo de caso de adoção de DevOps, propomos um processo estruturado de introdução de DevOps e suas modificações necessárias para se adaptar a sistemas legados, visando a melhoria do processo de desenvolvimento de sistemas através da implantação da prática de entrega contínua. Assim, pretendemos facilitar a manutenção e evolução desses sistemas, através da automação dos testes, dos processos de lançamento e das etapas de implantação, detectando problemas logo após qualquer alteração, diminuindo os erros causados por passos manuais, diminuindo as aprovações necessárias para implantação, reduzindo o tempo de integração e entrega, fazendo as manutenções serem mais rápidas e aumentando a produtividade da equipe.

Este artigo consiste em seis seções: a seção II apresenta a revisão bibliográfica com os principais conceitos de sistemas legados e DevOps. A seção III apresenta o estudo de caso e lições aprendidas. A seção IV descreve o processo proposto. A seção V apresenta a avaliação do processo e, finalmente, a seção VI apresenta as considerações finais.

II. REVISÃO BIBLIOGRÁFICA

A. *Sistemas Legados*

Os autores dão diversas definições a sistema legados.Um sistema legado é um sistema sociotécnico que é útil ou até essencial para uma organização, mas que foi desenvolvido utilizando tecnologias ou métodos obsoletos. Pelo fato de os sistemas legados frequentemente executarem funções críticas para o negócio, eles precisam ser mantidos [22]. Sistemas se tornam legados quando começam a resistir a modificações e evolução [20].

Os sistemas legados não são uma questão transitória, mas contínua: "O novo sistema de hoje se tornará o sistema legado na próxima, e inevitável, rodada de mudanças" [4]. Sistemas legados são aqueles que não possuem testes automatizados (é uma definição útil e simples, no entanto, controversa) [6].

Os sistemas legados são um desafio conhecido para as operações de organizações modernas, pois limitam sua capacidade de mudança e crescimento de negócios [9][10]. As agências governamentais continuam a investir substancialmente em sistemas legados, com uma estimativa de US $ 35,7 bilhões gastos pelo governo federal dos Estados Unidos em 2010 [23].

Apesar das desvantagens bem conhecidas, como ser inflexível e difícil de manter, os sistemas legados ainda são de vital importância para as empresas, pois suportam complexos processos de negócios e armazenam lógica de negócios crítica e de grande valor para a empresa, por isso não podem simplesmente ser removidos [12].

O conhecimento incorporado ao sistema legado constitui um ativo organizacional significante, e, se o sistema provê valor para o negócio, então deve ser modernizado ou substituído por outro sistema mais moderno [20]. Muitos sistemas legados dão suporte a funções vitais de negócio e são indispensáveis. Se o software legado atende às necessidades de seus usuários e funciona de forma confiável, ele não está quebrado e não precisa ser consertado [15].

A maioria dos esforços de modernização de sistemas legados falham por razões de complexidade da modernização do sistema, falta de entendimento da tecnologia em que foi construído o software, falta de gestão de riscos, dificuldades surgidas da adoção de componentes comerciais e falta de alinhamento aos objetivos de negócio [20].

Os especialistas argumentam que, muitas vezes, a migração ou transformação desses sistemas legados não é possível, mas melhorando o processo de desenvolvimento, testes e entrega, facilitaria a manutenção e evolução desses sistemas, diminuindo muitos dos problemas existentes.

B. *DevOps*

As empresas normalmente separam as atividades de desenvolvimento e operações em duas equipes diferentes que possuem objetivos conflitantes. Enquanto o time de desenvolvimento é incentivado a entregar software mais rápido, o time de operações busca estabilidade do ambiente de produção. Mas todos os envolvidos têm um objetivo comum: tornar a entrega de software de valor aos usuários uma atividade de baixo risco. Com essas necessidades surgiu o movimento DevOps. O movimento DevOps está centrado em um objetivo: encorajar uma colaboração maior entre todos os envolvidos no processo de entrega de software de maior valor com mais confiança e rapidez. Não há uma definição precisa do termo DevOps na literatura, nem um processo específico a ser seguido, ou seja, não há uma solução única que sirva para todas as empresas [5].

O movimento DevOps pode ser considerado uma extensão do movimento ágil, um movimento que surgiu em 2001 devido a novos valores exigidos pelo desenvolvimento de software e coletados no "Manifesto Ágil" [24]. Esses valores incentivam uma resposta rápida às mudanças e a colaboração entre equipes. Entre as metodologias ágeis, podemos destacar Scrum e Extreme Programming (XP), que incentivam a necessidade de pequenos ciclos de desenvolvimento e equipes multifuncionais. O XP foi a primeira metodologia que abraçou plenamente o conceito de Integração Contínua [1], defendendo a necessidade de entregar e testar pequenas mudanças, reduzindo o risco de erros e facilitando a depuração[8].

Com a necessidade de melhoria contínua do ciclo de vida nas diferentes fases de desenvolvimento e de reduzir o tempo de entrega dos produtos ao mercado, o conceito de Integração Contínua foi estendido e foi criado o conceito de Entrega Contínua. A Entrega Contínua é o ponto central do movimento DevOps, e defende a automação dos testes e das etapas de implantação. Espera-se que a Entrega Contínua ofereça vários benefícios, tais como: (1) obter feedback maior e mais rápido dos clientes e do processo de desenvolvimento de software; (2) com lançamentos freqüentes e confiáveis, levar à melhoria da satisfação do cliente e da qualidade do produto; (3) a ligação entre equipes de desenvolvimento e operações ser reforçada e as tarefas manuais poderem ser eliminadas [3][12][18].

Muitos casos industriais relatados indicam que a Entrega Contínua está sendo cada vez mais utilizada nas práticas de engenharia de software em empresas de vários tamanhos e domínios [13][14][16]. Existe um estudo em uma empresa que adotou Entrega Contínua e teve o ciclo de vida de liberação de uma evolução no software reduzido de vários meses para 2 a 5 dias, desde a criação das histórias de usuários até a liberação da versão em produção [3]. Mas poucos trabalhos abordam os benefícios trazidos na aplicação de Entrega Contínua aos processos de manutenção, evolução e implantação de sistemas legados. Mesmo que os sistemas legados não necessitem ter versões liberadas frequentemente, também poderão ser beneficiados com a diminuição do ciclo de vida de liberação das evoluções.

C. *Trabalhos relacionados*

Um dos trabalhos que aborda os desafios de se implantar de DevOps em empresas que trabalham com sistemas legados foi desenvolvido por Schaller [19], onde são abordados os desafios nas áreas de segurança, cultura organizacional, arquitetura de sistemas legados em larga escala, automação em larga escala. Schaller fez uma vasta pesquisa na literatura sobre as experiências de empresas que adotaram DevOps, apontando os desafios específicos encontrados na literatura ao lidar com sistemas legados. Outro trabalho foi desenvolvido por Rao [17], onde são abordadas modificações em um roteiro típico de

DevOps para se adaptar a sistemas legados. Segundo o autor, um típico roteiro de DevOps envolve construir um pipeline de implantação, que é uma implementação automatizada do processo de compilar todas as partes de uma aplicação, implantá-la em um ambiente qualquer, testá-la e efetuar sua entrega final.

O pipeline de implantação tem três objetivos. Em primeiro lugar, ele torna cada parte do processo de compilação, implantação, teste e entrega de versão visível a todos os envolvidos, promovendo a colaboração. Em segundo, melhora o feedback do processo, de modo que os problemas são identificados e resolvidos o mais cedo possível. Finalmente, permite que equipes entreguem e implantem qualquer versão de seu software a qualquer momento e em qualquer ambiente necessário (teste, desenvolvimento ou produção) por meio de um processo completamente automatizado [11].

Ao adotar DevOps em um projeto novo, os times irão achar mais fácil que aplicar a sistemas legados [17], pois em um projeto novo, a Entrega Contínua seria considerada na etapa de análise, projeto e definição da arquitetura. Em sistemas legados, que evoluíram com o tempo sem considerar a automação, a adoção de DevOps pode levar à necessidade de reprojetar o sistema e uma refatoração de código em larga escala. Com isso, introduzir Entrega Contínua e um pipeline de implantação em sistema legado pode ser um grande desafio.

Nosso trabalho se diferencia dos trabalhos desenvolvidos por Schaller [19] e Rao [17], pois nós apresentamos um estudo de caso de adoção de DevOps e propomos um processo estruturado de introdução de DevOps com modificações necessárias para se adaptar a sistemas legados, detalhando atividades e fluxos necessários.

III. Estudo de Caso: Introdução De Devops Em Sistemas Legados

Um estudo de caso é uma investigação empírica que investiga um fenômeno contemporâneo dentro de seu contexto da vida real, especialmente quando os limites entre o fenômeno e o contexto não estão claramente definidos. [25]. O estudo de caso foi realizado em uma instituição financeira não identificada neste trabalho durante 2 anos, começando em março de 2016 e sendo implantada e refinada até o momento da produção desse trabalho. Primeiramente, foi identificado que o processo de implantação de sistemas demandava muito esforço e havia uma grande burocracia, então foi formada uma equipe na área de arquitetura de software da instituição para melhorar e automatizar o processo. Foram estudados os benefícios da aplicação de práticas ágeis e Entrega Contínua.

A. Integração contínua e Automação de Implantação

Em um primeiro momento, foram pesquisadas ferramentas que automatizassem o processo de Integração Contínua, tanto na compilação e construção do sistema, quanto na implantação automática do sistema. A partir disso, foi identificada uma ferramenta, chamada Jenkins, que serviria como um servidor de integração contínua. Essa ferramenta então foi instalada pela equipe. Jenkins é uma plataforma de integração contínua de código aberto, altamente personalizável e flexível graças à sua arquitetura de plug-ins e sua enorme comunidade Open Source, cujo objetivo inicial foi a automação dos processos de compilação e teste [21]. A tarefa básica do Jenkins é executar algumas etapas predefinidas conhecidas como tarefas com base em um acionador.

Uma aplicação simples em Java foi escolhida como projeto piloto, e foi pesquisado o que seria necessário para automatizar o processo de compilação e geração de build (que iremos chamar de processo de construção) dessa aplicação. Para a construção automática de projetos Java, foi visto que seria necessário utilizar o Maven. Maven é uma ferramenta de automação de compilação utilizada em projetos Java que utiliza um arquivo XML (POM) para descrever o projeto de software sendo construído, suas dependências sobre módulos e componentes externos, a ordem de compilação, diretórios e plug-ins necessários. A aplicação Java escolhida foi então modificada para utilizar Maven, com ajuda da documentação e da equipe de desenvolvimento responsável. Então foram reunidas todas as dependências necessárias e cadastradas nos arquivos XML do Maven e criado um script para realizar a construção automática. Esse script foi testado e aprovado. Após o sucesso dessa execução, esse script de construção automática já estava passível de ser acionado e rodado automaticamente pelo Jenkins.

O próximo passo foi verificar como poderia ser realizada a implantação automática dessa aplicação Java no ambiente de desenvolvimento. As aplicações Java eram implantadas em um servidor IBM Websphere Application Server (WAS) seguindo uma série de passos manuais. Então foi descoberto na internet um script que poderia utilizar Maven para realizar a implantação automática de aplicações Java em servidores WAS. Esse script foi então estendido e criado um plugin para a plataforma Jenkins que, depois de testado e aprovado, se tornou o script padrão para implantações automáticas de aplicações Java em servidores WAS.

A partir do sucesso da construção e implantação automática para um projeto Java, foram surgindo iniciativas para se utilizar a plataforma Jenkins para automatizar a construção e implantação automática de sistemas de outras linguagens. A plataforma Jenkins foi apoiada pela gerência como uma solução corporativa que seria utilizada por toda a instituição. Então, a equipe de desenvolvimento começou uma análise dos sistemas que possuíam mais demandas de evolução e foi criada uma fila de sistemas, por prioridade, e os plugins para outras tecnologias foram sendo criados.

Paralelamente, a ferramenta de controle de versão também sofreu mudanças. A ferramenta de gerência de configuração precisou ser atualizada. Decidiu-se então migrar o controle de versão para a ferramenta IBM Rational Team Concert (RTC). Primeiramente, a nova ferramenta de controle de versão foi instalada e, em um segundo passo, o código fonte e os artefatos de todos os sistemas e projetos foram migrados para a nova ferramenta. O terceiro passo foi o treinamento de toda a equipe de desenvolvimento na nova ferramenta. Eentão, todos os sistemas foram migrados para nova plataforma, que começou a ser amplamente utilizada.

Com a adoção da ferramenta RTC, teve que ser desenvolvido um plugin do Jenkins para acessar o RTC, então os scripts de construção dos sistemas foram modificações para apontar para a nova ferramenta de controle de versão RTC.

Após a instalação e configuração da ferramenta de controle de versão, foi iniciado um trabalho de regularização nas ramificações dos sistemas. Foi identificado que alguns sistemas possuíam muitas ramificações, o que dificultava a junção do código (merge) dos diversos ramos de desenvolvimento do sistema, quando era necessário gerar uma versão para ser testada. Então, foi definida uma padronização das ramificações de todos os sistemas em 3 tipos de ramificações: integração, que continha o código atual do sistema utilizado em produção; projeto, que continha o código candidato a nova versão; desenvolvimento, que continha o código de cada evolução. Em seguida, foi definida uma baseline para todos os sistemas, que seria o código de cada sistema utilizado em produção e ficaria na ramificação de integração, sedo utilizada como marco para a criação de qualquer projeto a partir de então.

Finalmente, a gerência da área de tecnologia da instituição decidiu impor regras para a implantação de aplicações nos ambientes. As implantaçõesdeveriam ser todas feitas automaticamente através do Jenkins, e não mais serem feitas solicitações pelos analistas, anexando a build do sistema e documentos necessários para implantação. O prazo de 1 mês foi imposto para que todas as aplicações se adequassem à nova regra de implantações.

Essa decisão foi importante para garantir a integridade dos sistemas e que todas as aplicações, que evoluíssem a partir de então, passariam por todo o processo automatizado. Isso diminuiu os erros causados por compilações, construções com dependências erradas, disponibilizações de artefatos incorretos e erros humanos em geral, pois apenas processos automatizados fariam as implantações nos ambientes. Com isso, todas as alterações feitas nos sistemas e pessoas que atuaram em cada etapa do pipeline de implantação poderiam ser rastreadas, verificadas, otimizadas e auditadas.

B. Automação de Testes

As validações eram realizadas manualmente, através de requisições da equipe de desenvolvimento para a equipe de qualidade, que aplicava alguns scripts de validação de código fonte. Para automação dos testes, primeiramente foi implantada a ferramenta SonarQube. SonarQube é uma plataforma aberta para controle de qualidade de código, que é basicamente uma ferramenta web baseada em plugins, que manipula os resultados de várias ferramentas de análise de código, gerando resultados em forma de relatório. Os especialistas da equipe de qualidade responsáveis pelas validações definiram as métricas de qualidade (Duplicação de código, Padrões de codificação, cobertura de código e Bugs em potencial) seriam utilizadas na ferramenta SonarQube. Foi criado um plugin para a plataforma Jenkins se integrar com o SonarQube. Então se tornou possível ser realizada a análise da qualidade do código automaticamente inclusive para sistemas cobol, powerbuilder e outras linguagens comuns em sistemas legados.

Com os defeitos apontados na validação automática feita pelo SonarQube, as equipes de desenvolvimento podiam verificar o código fonte de suas aplicações e ajustá-los aos padrões de código fonte definidos. Cada gerente de projeto definiu um prazo para adequação dos defeitos apontados pela ferramenta SonarQube em seus sistemas.

Em seguida, foi implantada a ferramenta IBM Rational Quality Manager (RQM), que é uma ferramenta para planejamento de teste e gerenciamento de ativos de teste. Foi baixado um plugin de integração entre o jenkins e o RQM, e então foi possível serem criados testes funcionais para serem executados automaticamente.

C. Automação da Monitoração

A automação da monitoração foi um processo iniciado independente das iniciativas de automação feitas pelas equipes de desenvolvimento de sistemas. Primeiramente, foram identificados 24 sistemas críticos para o negócio, então foi formada uma equipe no ambiente de infraestrutura, para implantar uma monitoração mais efetivas para esses sistemas.

Após identificados os sistemas críticos, foram adquiridas e instaladas um conjunto de ferramentas da CA conhecida como CA Service Operations Insight (SOI). Para iniciar o processo, foram solicitadas as documentações dos sistemas, verificado com os analistas de cada sistema o que devia ser monitorado e verificado o funcionamento de cada sistema (linguagem, integrações, banco de dados, etc). Inicialmente, cada sistema foi monitorado durante 15 dias, para verificar o desempenho médio, então, baseado no desempenho medido e no desempenho esperado pelos analistas e usuários, foram criados na ferramenta os limites, alertas e configuradas as regras de tratamento de ocorrências de alertas.

A maioria dos sistemas críticos eram legados, então a adequação ficou bem mais difícil, pois não existiam scripts de monitoração construídos para monitorar sistemas legados. Foram então tomadas decisões de se monitorar a infraestrutura utilizada pelas aplicações, e serem criados scripts específicos que simulavam emonitoravam a experiência de usuário em sistemas legados, como tempo para abrir sistema, tempo de resposta, tempo para realizar consultas, etc.Os primeiros scripts criadospuderam servir como padrão para aquele tipo de sistema, e foram reutilizados em outros sistemas de mesmo tipo ou mesma linguagem.

Finalizada a implantação da monitoração para todos os 24 sistemas críticos, então foi criada uma fila para serem incluídos na monitoração todos os sistemas utilizados na instituição, de acordo com a quantidade de demanda e criticidade dos sistemas.

D. Automação da Infraestrutura

Para automação da infraestrutura, foi utilizada a plataforma System Center Orchestrator, que fornece ferramentas para desenvolver, testar, depurar, implantar e gerenciar automação de tarefas utilizando uma interface gráfica. Primeiramente foram identificados quais os tipos de servidores existentes na infraestrutura de produção e quais desses servidores seguiam o mesmo padrão de criação e provisionamento. Então foram definidos padrões de servidores (servidor de banco de dados, servidor Web, servidor de componentes) e foram criados templates de criação desses servidores na ferramenta VMware. Por fim, foram criadas as tarefas completas no System Center Orchestrator necessárias para criação de máquinas virtuais utilizando esse templates de criação máquinas virtuais criados no VMware e então o processo de criação de servidores foi automatizado.

E. Resultados obtidos pela revisão teórica e estudo de caso

Com a execução do estudo de caso e leitura dos artigos referenciados na seção B da parte II, nós pudemos entender alguns pontos relevantes, alguns semelhantes aos relatados em [14], no contexto de uma instituição que possua sistemas legados e deseje implantar um processo de DevOps, entre eles: a) DevOps é uma cultura e não se refere apenas à utilização de ferramentas. Todos devem se sentir responsáveis por mudar a mentalidade, contribuir para a melhoria do processo como um todo e tornar a entrega de software mais rápida e com qualidade; b) A implantação gradativa da cultura e ferramentas DevOps é interessante para todos irem aos poucos se adequando e o impacto da mudança não gerar resistência por parte de alguma equipe; c) O treinamento da equipe no uso das ferramentas deve ser feito pouco antes de começarem a utilizá-las. Se for muito antes, quando forem utilizar no dia-a-dia talvez tenham esquecido o que aprenderam no treinamento. Se for depois, as equipes vão ter as ferramentas disponíveis, mas não saberão utilizá-las; d) A gerência de TI tem que apoiar e estar comprometida com a implantação do processo, estabelecendo controles, quando o processo estiver implantado e maduro, para que aplicações possam ser implantadas apenasse passarem por todo o processo; e) Muitos plugins, scripts padrão de compilação, construção e integração entre as ferramentas estão disponíveis na internet, facilitando a implantação do processo; f) A introdução das ferramentas é repetível, portanto é interessante pessoas que já tenham participado da implantação do processo de integração ou Entrega Contínua em outras oportunidades participem da equipe de implantação do processo na empresa que irá aderir a DevOps; g) Provavelmente existirão pessoas resistentes à mudança pra nova mentalidade de DevOps, que não conseguirão enxergar os benefícios da implantação do processo. Porém, após a implantação do processo e utilização diária, provavelmente irão notar a agilidade adquirida nos processos; h) Inicialmente existe a impressão que alguns sistemas legados e tecnologias não conseguirão ser automatizados ou beneficiados pelo processo de Entrega Contínua, mas, no caso da instituição do estudo de caso, até sistemas COBOL e Powerbuilder foram beneficiados com automatização de processos.

IV. Processo De Introdução De Sistemas Legados Em DevOps

Como resultado da revisão teórica e estudo de caso, foi modelado um processo de introdução de sistemas legados em DevOps visando auxiliar outras instituições que trabalhem com desenvolvimento de software e sistemas legados.

Os sistemas legados não possuem um padrão, cada empresa tem sistemas legados que, muitas vezes, não se assemelham em nada com o de outras empresas da mesma área de atuação. Para adotar uma abordagem que beneficie empresas que tenham sistemas tão distintos, temos que nos esforçar para padronizar ao máximo os processos e fases pelas quais passam esses sistemas durante o desenvolvimento. Portanto, o processo se baseia na ideia de padronização de como os sistemas são desenvolvidos, testados, monitorados e implantados. Primeiramente, devem ser priorizados sistemas de um tipo de linguagem, desenvolvidos os padrões (de compilação, construção, integração, etc) e depois os outros sistemas de mesmo tipo irem se adequando a essa padronização e os processos e fases sendo automatizados.

No início da adoção do processo, é necessário um esforço inicial para configurar as ferramentas, treinar a equipe e implantar os subprocessos, o que irá gerar um grande impacto na velocidade do desenvolvimento e manutenções durante o início da adoção processo. Portanto, é importante o apoio da gerência e dos clientes, pois, embora a configuração inicial seja demorada, uma vez feita, evoluir os sistemas será um processo muito mais simples e natural.

O fluxo proposto para o processo modelado, representado graficamente na Figura 1, pode ser dividido em cinco subprocessos:

- *Integração Contínua* – subprocesso em que, ao ser entregue o código pelos desenvolvedores, serão automatizados os processos de integração do código gerado, gerencia de configuração e geração da build.
- *Automação da Implantação* – subprocesso em que será automatizada a implantação de qualquer build gerada em qualquer um dos ambientes (desevolvimento, homologação e produção).
- *Automação dos Testes* - subprocesso em que serão automatizadas as aplicações dos testes unitários, funcionais e qualquer outro tipo de teste de verificação e validação da build gerada.
- *Automação da Monitoração* - subprocesso em que será automatizada a monitoração, alertas e regras de tratamento dos sistemas presentes no ambiente de produção.
- *Automação da Infraestrutura* - subprocesso em que será automatizada a criação e disponibilização de servidores de todos os ambientes.

Esses subprocessos podem ser feitos em paralelo, quando feitos por equipes diferentes.

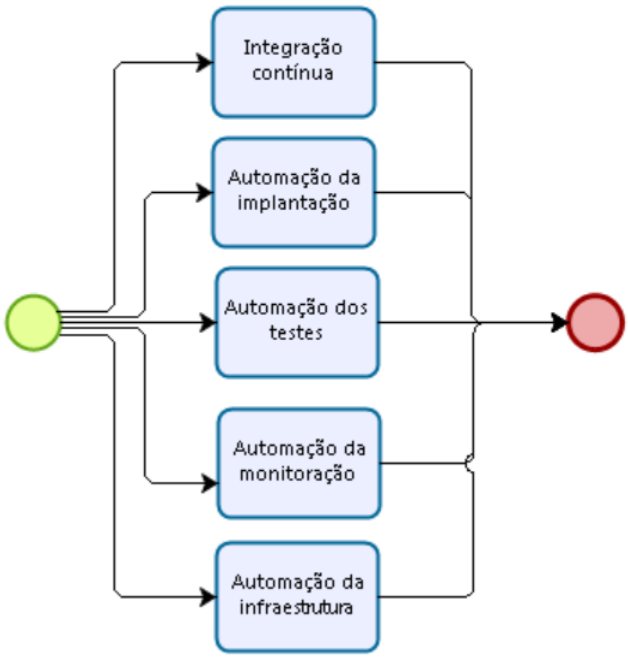

Figura 1 – Representação gráfica do Processo de Introdução de Sistemas Legados em DevOps

A. Integração contínua

O subprocesso de integração contínua, apresentado na Figura 2, começa com a configuração da ferramenta de automação da aplicação de scripts. Essa é a ferramenta principal do processo, pois é com ela que todas as outras ferramentas irão se integrar para serem automatizadas todas as etapas da Entrega Contínua, e na qual as outras atividades de criação de scripts irão se basear. Após a configuração da ferramenta de automação da aplicação de scripts, teremos o micro processo de gerência de configuração, onde será configurada a ferramenta de controle de versão compatível com integração contínua e serão trabalhadas as ramificações existentes nos sistemas, além de definida uma baseline de integração que será ponto de partida para a evolução dos sistemas.

Em seguida, teremos o micro processo de automação de build, onde serão configuradas as ferramentas de compilação e construção, criados os padrões e scripts de construção do sistema, além de configuradas outras ferramentas que otimizem o processo de construção da build do sistema, e, por fim, todos os scripts criados serão automatizados na ferramenta de automação da aplicação de scripts.

Por fim, teremos a atividade de fechamento do processo de integração contínua, em que são aplicadas as restrições de integridade no processo como um todo. Nessa atividade é onde há a determinação, pela gerência de TI, que todos os sistemas que forem implantados em produção, precisarão seguir o processo automatizado que foi definido. Assim, poderão ser rastreadas quais alterações foram feitas em uma determinada versão do sistema, o solicitante, o responsável, artefatos modificados, ativos gerados, ativos implantados e os responsáveis pela implantação. Com isso, garantimos a integridade e rastreamento de todo o processo pelos quais os sistemas implantados passaram.

A seguir, descreveremos as atividades do subprocesso de integração contínua e seus principais objetivos:

- *Configurar Ferramenta de Automação de Aplicação de Scripts* - Introduzir e configurar uma ferramenta de automação de aplicação de scripts a ser utilizada por todos os sistemas.
- *Introduzir Controle de Versão Compatível ao Sistema* - Atualizar a ferramenta de controle de versão para uma versão compatível com o processo de IC.
- *Diminuir Ramificações* - Verificar quais ramos de desenvolvimento estão ativos e diminuir a quantidade de ramos.

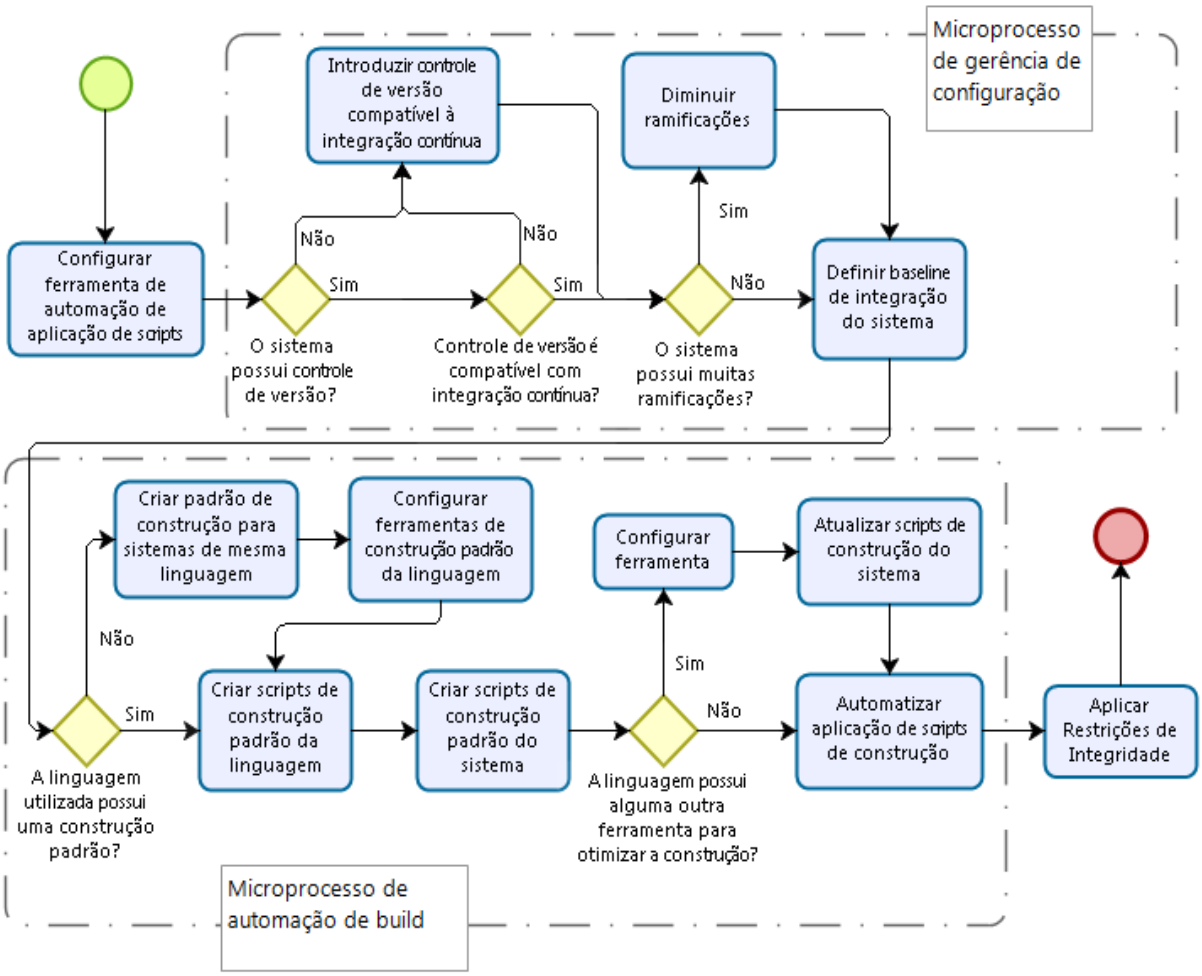

Figura 2– Representação gráfica do subprocesso de Integração Contínua

- *Criar Padrão de Construção para Sistemas de mesma Linguagem* - Verificar como os sistemas da mesma linguagem fazem a construção e criar uma maneira padrão de construção, para ser utilizada por todos.
- *Configurar Ferramentas de Construção Padrão da Linguagem* - Introduzir e configurar uma ferramenta de construção padrão a ser utilizada por todos os sistemas de uma mesma linguagem.
- *Criar Scripts de Construção Padrão da Linguagem* - Codificar o padrão de construção criado anteriormente e criar scripts de construção padrão da linguagem.
- *Criar Scripts de Construção do Sistema* - Modificar os scripts de construção padrão da linguagem e produzir os scripts de construção do sistema.
- *Configurar Ferramenta* - Introduzir e configurar uma qualquer ferramenta adicional para otimizar a construção do sistema.
- *Atualizar Scripts de Construção do Sistema* - Modificar os scripts de construção do sistema e adicionar as dependências e artefatos na geração da build
- *Automatizar Aplicação de Scripts de Construção e Construção* - Automatizar a aplicação dos scripts de construção e construção do sistema.
- *Aplicar Restrições de Integridade* - Aplicar restrições para a implantação do sistema no ambiente de produção para que apenas sistemas gerados automaticamente sejam implantados.

B. Automação da Implantação

O subprocesso de automação da implantação, apresentado na Figura 3, começa com a criação dos padrões de implantações para os sistemas de mesma linguagem. Em seguida, serão criados os padrões de implantações para os sistemas de mesmo tipo (batch, web, cliente-servidor, etc), depois serão criados os padrões de implantações para os tipos de ambiente (desenvolvimento, testes, homologação, produção). Então, a ferramenta de automação da implantação, que pode ser a mesma ferramenta de automação da aplicação de scripts, ou uma ferramenta especializada em automação da implantação, será configurada. A seguir, os scripts de implantação padrão para cada tipo de linguagem, sistema e ambiente serão criados. Por fim, serão criados os scripts de implantação específicos de cada sistema e esses scripts serão automatizados pela ferramenta de automação da aplicação de scripts.

A seguir, descreveremos as atividades do subprocesso de automação da implantação e seus principais objetivos:

- *Criar Padrão de Implantação para Linguagens* - Verificar como é feita a implantação das aplicações de uma dada linguagem e padronizar a implantação para ser feita da mesma maneira.
- *Criar padrão de implantação para tipos de sistema* - Padronizar como é feita a implantação de um tipo de sistema (aplicação web, batch, cliente servidor).

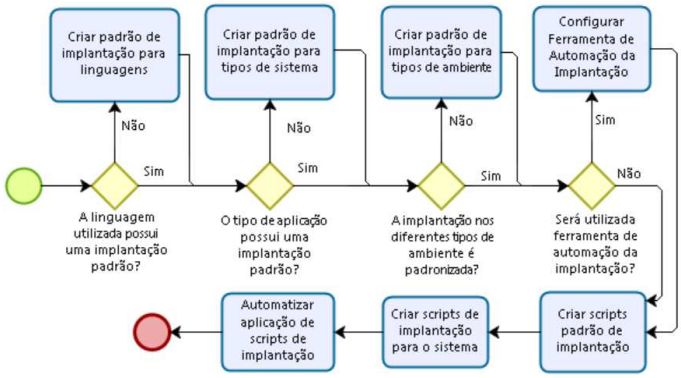

Figura 3 – Representação gráfica do subprocesso de Automação da Implantação

- *Criar padrão de implantação para tipos de ambiente* - Padronizar como são feitas as implantações de cada tecnologia nos ambientes de desenvolvimento, testes, homologação e produção.
- *Configurar Ferramenta de Automação da Implantação* - Introduzir e configurar uma ferramenta de automação de implantação a ser utilizada por todas as implantações feitas.
- *Criar Scripts Padrão de Implantação* - Baseado nos padrões implantação criados, criar os scripts de implantação padrão da linguagem, por tipos de sistema e por tipo de ambiente.
- *Criar Scripts de Implantação para o Sistema* - Modificar os scripts de implantação padrão e produzir os scripts de implantação do sistema para cada ambiente.
- *Automatizar Aplicação de Scripts de Implantação* - Automatizar, com a ajuda da ferramenta de automação de scripts, a aplicação dos scripts de implantação do sistema para cada ambiente.

C. Automação de Testes

O subprocesso de automação dos testes, apresentado na Figura 4, começa com a definição da baseline de métricas a serem analisadas para cada linguagem. Em seguida, será configurada a ferramenta de análise estática do código automatizada. Então serão criados os scripts padrão de análise estática do código fonte por linguagem, depois os scripts de análise estática para cada sistema. Serão então configuradas as ferramentas de testes unitários e testes funcionais automatizados. Finalmente, os scripts de testes criados serão automatizados pela ferramenta de automação da aplicação de scripts.

A seguir, descreveremos as atividades do subprocesso de automação de testes e seus principais objetivos:

- *Definir Baseline de Métricas de Qualidade para a Linguagem* - Definir, de acordo com a linguagem do sistema, quais serão as métricas aplicadas para fazer a análise estática de qualidade do código fonte (Exemplos: Duplicação de código, Padrões de codificação, cobertura de código e Bugs em potencial).

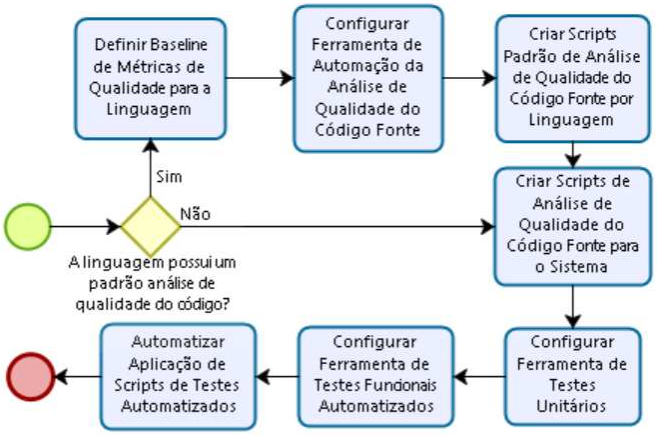

Figura 4 – Representação gráfica do subprocesso de Automação de Testes

- *Configurar Ferramenta de Automação da Análise de Qualidade do Código Fonte* - Introduzir e configurar uma ferramenta de análise estática do código automatizada.
- *Criar Scripts Padrão de Análise de Qualidade do Código Fonte por Linguagem* - Baseado na baseline de métricas de qualidade do projeto para a linguagem e na ferramenta de análise estática do código escolhida, criar os scripts de análise estática do código fonte da linguagem.
- *Criar Scripts de Análise de Qualidadedo Código Fonte para o Sistema* - Modificar os scripts de análise estática do código fonte da linguagem e produzir os scripts de análise estática do código fonte do sistema.
- *Configurar Ferramenta de Testes Unitários* - Introduzir e configurar a ferramenta de testes unitários que será utilizada por todas as linguagens compatíveis com essa ferramenta.
- *Configurar Ferramenta de Testes Funcionais Automatizados* - Introduzir e configurar a ferramenta de testes funcionais automatizados que será utilizada por todas as linguagens compatíveis com essa ferramenta.
- *Automatizar Aplicação de Scripts de Testes Automatizados* - Automatizar, com a ajuda da ferramenta de automação de scripts a aplicação dos scripts de testes automatizados do sistema.

D. Automação da Monitoração

O subprocesso de automação da monitoração, apresentado na Figura 5, começa com a configuração da ferramenta de monitoração. Em seguida, serãodefinidas as métricas de monitoração de desempenho de aplicativos (apm) por tipo de sistema, para definir o que deve ser monitorado e qual o desempenho esperado. Então serão criados os scripts padrão de monitoração por tipo de sistema, depois os scripts de monitoração para cada sistema. Serão então configuradas os limites, alertas e regras de tratamento para cada sistema. Finalmente, os scripts de monitoração criados serão automatizados pela ferramenta de monitoração.

A seguir, descreveremos as atividades do subprocesso de automação da monitoração e seus principais objetivos:

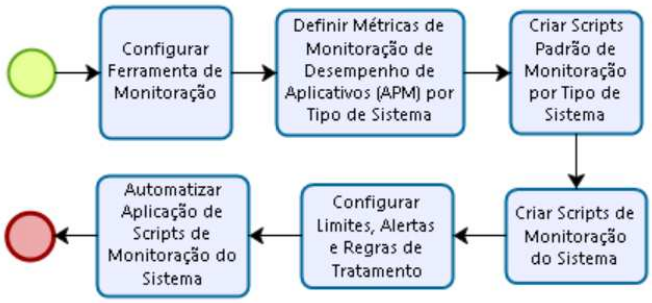

Figura 5 – Representação gráfica do subprocesso de Automação da Monitoração

- *Configurar Ferramenta de Monitoração* - Introduzir e configurar uma ferramenta de monitoração que será utilizada por todos os sistemas em produção.
- *Definir Métricas de Monitoração de Desempenho de Aplicativos (APM) por Tipo de Sistema* - Definir, de acordo com o tipo de sistema, quais métricas de desempenho de aplicativos serão monitoradas (Exemplos: tempos de carregamento de página, taxas de erro, transações lentas, etc.)
- *Criar Scripts Padrão de Monitoração por Tipo de Sistema* - Criar os scripts de monitoração padrão para aquele tipo de sistema, baseado na métricas de monitoração de desempenho de aplicativos do tipo de sistema,.
- *Criar Scripts de Monitoração do Sistema* - Modificar os scripts de monitoração padrão para o tipo de sistema e produzir os scripts de monitoração específicos do sistema.
- *Configuradas Limites, Alertas e Regras de Tratamento* - Configurar na ferramenta de monitoração, para cada sistema, os limites de cada métrica definida, os alertas a serem gerados e regras de tratamento para cada alerta.
- *Automatizar Aplicação de Scripts de Monitoração do Sistema* - Automatizar, com a ajuda da ferramenta de monitoração, a aplicação dos scripts de monitoração do sistema.

E. Automação da Infraestrutura

O subprocesso de automação da infraestrutura, apresentado na Figura 6, começa com a identificação da configuração atual da infraestrutura dos ambientes. Em seguida, serão padronizados os requisitos de infraestrutura em categorias, separando servidores e componentes do mesmo tipo em padrões. Em seguida, será configurada a ferramenta de automação da infraestrutura. Então serão criados os scripts padrão de criação de servidores, e, baseado nos scripts padrão, serão criados os scripts de criação de ambientes do sistema, para poder ser criado automaticamente cada ambiente em que o sistema irá rodar. Finalmente, os scripts de criação da infraestrutura serão automatizados pela ferramenta de automação.

A seguir, descreveremos as atividades do subprocesso de automação da infraestrutura e seus principais objetivos:

Figura 6 – Representação gráfica do subprocesso de Automação da Infraestrutura

- *Identificar Configuração Atual da Infraestrutura dos Ambientes* - Identificar, com ajuda da equipe de desenvolvimento, qual a configuração atual da infraestrutura de cada ambiente, se baseando em documentos de arquitetura, topologia de rede, ferramentas de gerenciamento de ambiente, monitoração e controle de versão.

- *Padronizar os Requisitos de Infraestrutura em Categorias* - Definir categorias de servidores e tamanhos das instâncias (por exemplo: servidor de aplicativos e servidor de banco de dados com instâncias pequenas, médias e grandes) para todas as aplicações poderem ter um padrão de servidores que utilizam.
- *Configurar Ferramenta de Automação da Infraestrutura* - Introduzir e configurar uma ferramenta de automação da infraestrutura que será utilizada para gerenciamento e criação dos ambientes
- *Criar Scripts Padrão de Criação de Servidores* - Criar os scripts padrão de criação de servidores que serão utilizados para todos os ambientes, baseado no padrão de servidores identificados.
- *Criar Scripts de Criação de Ambientes do Sistema* - Modificar os scripts padrão de criação de servidores e produzir os scripts de criação de ambientes para o sistema.
- *Automatizar a Aplicação dos Scripts de Criação de Infraestrutura* - Automatizar, com a ajuda da ferramenta de automação da infraestrutura, a aplicação dos scripts de criação de ambientes do sistema.

V. AVALIAÇÃO DO PROCESSO

A metodologia utilizada para validação do processo foi a revisão por pares, que consiste na inspeção dos produtos de trabalho por um par, ou seja, um colaborador com habilidades semelhantes ao autor dos artefatos [3]. Neste trabalho, a revisão por pares foi realizada por 10 (dez) participantes que, em sua maioria, são arquitetos e desenvolvedores que possuem nível superior completo, formação em tecnologia da informação, mais de 6 anos de experiência profissional, conhecem o conceito de DevOps e já trabalharam com sistemas legados.

A revisão por pares foi dividida em três estágios: 1 – Apresentação do processo aos envolvidos; 2 – Preenchimento do questionário do processo pelos entrevistados; e 3 – Análise dos resultados pelo pesquisador. Como resultado, obtivemos as respostas presentes na Tabela 1.

Algumas sugestões foram registradas pelos avaliadores. Dentre elas, o ponto forte mais destacado foi a facilidade de entendimento do processo organizado em forma de fluxo e separado em atividades. Outro ponto comentado pela maioria foi a possível necessidade de esforço para implantar o processo de DevOps, pois está diretamente atrelado à cultura da empresa, necessitando de apoio da gerência, habilidades dos executores, principalmente quando se tratam de aplicações legadas, além de treinamento da equipe, implantação de ferramentas de automação. Entre os pontos fracos foi destacado que as atividades do processo são factíveis de implantação nas empresas, mas que dependem do apoio da gerência.

Tabela 1 – Resultados do processo de avaliação

Questão	Discordo fortemente	Discordo	Nem concordo nem discordo	Concordo	Concordo fortemente
O processo proposto é de fácil entendimento para o leitor.	0%	10%	0%	80%	10%
As atividades citadas no processo estão adequadas para empresas passarem a utilizar práticas de DevOps	0%	0%	10%	60%	30%
A documentação gerada no processo é adequada, sem apresentar excessos	0%	0%	20%	80%	0%
O processo é adaptável para empresas de desenvolvimento de software	0%	0%	10%	70%	20%
O processo pode ser utilizado em diferentes cenários de empresas que possuem sistemas legados	0%	0%	0%	70%	30%
O processo é eficiente e pode ser executado com pouco esforço	0%	40%	30%	30%	0%

VI. Considerações Finais

Com a revisão teórica e o estudo de caso, nós aprendemos que é desafiador a melhoria do processo de desenvolvimento de software, principalmente se estamos lidando com sistemas legados, que possuem muitos riscos envolvidos e qualquer mudança que acarrete em erros podem custar muito às empresas. Porém, aplicando os conceitos de DevOps e práticas como a Entrega Contínua, significará que as evoluções serão entregues mais rápidamente, tendo um feedback antecipado por parte dos usuários, erros identificados em fases anteriores do ciclo de vida de desenvolvimento do sofwtare, e maior qualidade nas entregas, por causa dos processos de integração contínua e automação de implantação.

Conseguimos também criar um processo estruturado de introdução de DevOps para sistemas legados, que pode ser utilizado como um passo a passo em empresas que possuam sistemas legados e decidam aderir às práticas de DevOps, sendo beneficiadas pela padronização da maneira como os sistemas são evoluídos e mantidos, além da automação dos processos de desenvolvimentos, testes, entrega e monitoração, além da melhora do relacionamento entre as equipes de desenvolvimento e operações.

Como trabalhos futuros, identificamos: (i) Aplicar o processo de introdução de DevOps para sistemas legados em outras empresas de diversos contextos, para identificarmos quais atividades podem ser modificadas, adaptadas e estendidas; (ii) Expandir cada parte do processo focando nas ferramentas existentes que poderiam ser utilizadas para cada atividade, vantagens e desvantagens de cada ferramenta, tipos de sistemas legados suportados, segurança de cada ferramenta e adaptações no processo para utilizá-las. (iii) Fazer estudos do tempo do ciclo de vida do produto, e tempo desde o início da evolução até a entrega do produto, antes e depois da introdução do processo.

Referências

[1] BECK, Kent; FOWLER, Martin. Planning extreme programming. Addison-Wesley Professional, 2001.

[2] BEIZER, Boris. Software system testing and quality assurance. Van Nostrand Reinhold Co., 1984.

[3] CHEN, Lianping. Continuous delivery: Huge benefits, but challenges too. IEEE Software, v. 32, n. 2, p. 50-54, 2015.

[4] COAKES, Elayne; ELLIMAN, Tony. Focus issue on legacy information systems and business process engineering: the role of stakeholders in managing change. Communications of the AIS, v. 2, n. 1es, p. 4, 1999.

[5] ERICH, Floris; AMRIT, Chintan; DANEVA, Maya. Report: Devops literature review. University of Twente, Tech. Rep, 2014.

[6] FEATHERS, Michael. Working effectively with legacy code. Prentice Hall Professional, 2004.

[7] FITZGERALD, Brian; STOL, Klaas-Jan. Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, v. 123, p. 176-189, 2017.

[8] FOWLER, Martin; FOEMMEL, Matthew. Continuous integration. Thought-Works) http://www.thoughtworks.com/ContinuousIntegration.pdf, v. 122, p. 14, 2006.

[9] GONG, Yiwei; JANSSEN, Marijn. From policy implementation to business process management: Principles for creating flexibility and agility. Government Information Quarterly, v. 29, p. S61-S71, 2012.

[10] HALACHMI, Arie. Imagined promises versus real challenges to public performance management. International Journal of Productivity and Performance Management, v. 60, n. 1, p. 24-40, 2011.

[11] HUMBLE, Jez; FARLEY, David. Continuous delivery: reliable software releases through build, test, and deployment automation. Pearson Education, 2010.

[12] LAGUNA, Miguel A.; CRESPO, Yania. A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. Science of Computer Programming, v. 78, n. 8, p. 1010-1034, 2013.

[13] LEPPANEN, Marko et al. The highways and country roads to continuous deployment. Ieee software, n. 2, p. 64-72, 2015.

[14] OLSSON, Helena Holmström; ALAHYARI, Hiva; BOSCH, Jan. Climbing the" Stairway to Heaven"--A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software. In: Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on. IEEE, 2012. p. 392-399.

[15] PRESSMAN, S. Roger. Engenharia de Software: Uma Abordagem Profissional. 7ª. Edição. Rio de Janeiro–RJ. Editora McGraw-Hill, 2011.

[16] RAHMAN, Akond Ashfaque Ur et al. Synthesizing continuous deployment practices used in software development. In: Agile Conference (AGILE), 2015. IEEE, 2015. p. 1-10.

[17] RAO, G. DevOps for legacy systems – The demand of the changing applications landscape. Infosys limited. 2015. Disponível em: https://www.infosys.com/it-services/application-development-maintenance/white-papers/documents/DevOps-legacy-systems.pdf

[18] ROCHE, James. Adopting DevOps practices in quality assurance. Communications of the ACM, v. 56, n. 11, p. 38-43, 2013.

[19] SCHALLER, Amy E. DevOps transformation challenges facing large scale legacy systems. 2016. Tese de Doutorado. Utica College.

[20] SEACORD, Robert C.; PLAKOSH, Daniel; LEWIS, Grace A. Modernizing legacy systems: software technologies, engineering processes, and business practices. Addison-Wesley Professional, 2003.

[21] SMART, John Ferguson. Jenkins: The Definitive Guide: Continuous Integration for the Masses. " O'Reilly Media, Inc.", 2011.

[22] SOMMERVILLE, Ian; ARAKAKI, Reginaldo; MELNIKOFF, Selma Shin Shimizu. Engenharia de software. Pearson Prentice Hall, 2008.

[23] Tobin, M; Bass, B. Federal Application Modernization Road Trip: Express Lane or Detour Ahead, UNISYS and MeriTalk, January 2011.

[24] Waters,K. What is Agile? (10 Principles of Agile), All About Agile. 2007. Disponível em: http://www.allaboutagile.com/what-is-agile-10-key-principles

[25] Yin, R. K. (2015). Estudo de Caso-: Planejamento e Métodos. Bookman editora.