Modelos de atuação de Suporte com *DevOps* em sistemas críticos

Daniel Ventura de Oliveira
TV Globo
Especialista de Governança Estratégica – daniel.veol@gmail.com

Resumo – A vasta adoção da cultura de desenvolvimento ágil para desenvolvimento de software (neste caso focando em *DevOps*) é uma realidade, e ganhou muito espaço no mercado para suprir a burocracia da gestão de mudança e dificuldade do planejamento assertivo de tarefas a longo prazo. O que não está sendo tratado de forma relevante seria como a equipe de desenvolvimento atua em conjunto com a equipe de suporte. A baixa priorização de documentação do projeto prejudica o time de suporte, pois o mesmo acaba não tendo as ferramentas necessárias e conhecimento para suportar tal aplicação. Tal problema se evidencia quando um software é posto em produção no modelo *MVP* cujo seu desenvolvimento continua após o *Go Live*. Através de entrevistas com outros profissionais, é percebido que o tema não é tratado de forma uniforme em algumas empresas e carece de modelos para ser tratado de forma eficiente.

Palavras-chave: *DevOps*. Metodologias ágeis. Suporte. Documentação.

Introdução

A motivação da escrita deste artigo surgiu a partir de dificuldades enfrentadas por mim, em experiências profissionais passadas, quando fiz parte de um time de desenvolvimento de software em uma cultura *DevOps*.

DevOps não é ferramenta, processo, ou se quer uma tecnologia. É uma cultura, oriunda de discussões de fóruns sobre desenvolvimento ágil. Damon Edwards e John Willis apresentaram em 2010 no congresso "*DevOpsDays Mountainview 2010*" os eixos do *DevOps*: *CAMS* (*Culture, Automation, Measurement and Sharing*) [1].

Há muita literatura sobre como implementar a cultura, como desenhar os processos a serem automatizados e como integrar as equipes de Desenvolvimento e Operações/Infraestrutura. Porém, não encontrei muitas referências mencionando a integração com o Suporte, o que, para certos projetos/empresa, é de extrema relevância.

Um dos cenários onde esta relevância se faz necessária é em produtos "*Beta* eternos", que constantemente recebem melhorias. Não necessariamente projetos que foram liberados com bugs conhecidos, mas sim soluções que nunca estão finalizadas, estão sempre evoluindo. Aplicações geralmente focadas na relação *B2C* (*Business to Consumer*), como Facebook, Instagram, Whatsapp, GMail e diversos outros, que necessitam de uma operação de suporte/monitoração 24/7. Como fazer com que a equipe de suporte tenha papel relevante em um cenário com diversas modificações, promovidas em uma cultura onde a documentação tem um papel inferior a projetos tradicionais[2]?

Contexto

DevOps está intimamente ligado com automação, com o objetivo de automatizar processos rotineiros, inerentes ao ciclo de vida de desenvolvimento de software. Com isso, conseguimos agilidade e facilidade na publicação de novos *releases*.

Esta facilidade dá mais autonomia e responsabilidade ao o desenvolvedor, que tem como função levar o seu código até a produção. Para conquistar isto, diversas técnicas devem ser implementadas para garantir o máximo de segurança neste processo, tais como: *TDD* (*Test Driven Development*), automação na replicação do ambiente de produção, monitoração constante em tudo que reflita à saúde do sistema, *logs* detalhados, definição e acompanhamento de *KPIs*, *feedback* constante, controle de versão, *blue green deployment* [3], *staged rollouts*[3], acompanhamento dos *deploys*, integração contínua, entrega contínua, implantação contínua [4], *rollback* de versão automatizado, dentre outras. Com tais ferramentas/técnicas implantados, o número de releases tende a aumentar consideravelmente.

Fazendo um paralelo a métodos tradicionais, para cada *release* deve-se ter uma documentação apropriada para orientar às equipes de suporte. Porém, em métodos ágeis como *DevOps*, a documentação não é algo priorizado no seu ciclo de vida. Isto pode não ser um problema para produtos que terão o seu *Go Live* ao fim do seu desenvolvimento, mas pode ser para aqueles que utilizam o conceito de *MVP* (*Minimum Viable Product*), onde os mesmos são publicados com o mínimo de funcionalidades necessárias para o lançamento e continuam evoluindo. Talvez algumas soluções não necessitam de um *SLA* (*Service Level Agreement*) agressivo, mas e para aplicações que são acessadas 24 horas por dia, ou mundialmente em diferentes fusos horários?

As equipes de suporte seriam responsáveis para tratar a maior parte dos problemas e incidentes do produto, reduzindo ao máximo o acionamento ao time de *DevOps*, para que ele esteja focado no desenvolvimento do projeto. Para tal, o suporte necessita de um mínimo de documentações e ferramentas para atuar na solução dos problemas. O que normalmente vemos em projetos ágeis, é um time de suporte desguarnecido de ferramentas devido ao foco do time de desenvolvimento em gerar valor para o produto. Isto pode ser extremamente benéfico para a sua evolução, mas seu efeito colateral é a dependência total do time de desenvolvimento para solucionar todo e qualquer tipo de problema.

Com o intuito de entender como tal ponto é tratado, realizei entrevistas com alguns profissionais do mercado.

Fabio (líder de desenvolvimento de um time *DevOps*) menciona que o time de suporte deve ficar mais próximo ao time de *DevOps*, acompanhando o dia a dia da criação do produto. Já Diogo (líder de um time de suporte) menciona que estão criando uma posição similar a um *SRE* (*Site Reliability Engeneering*) [5]. O papel do cargo em questão seria acompanhar, de forma parcial, o time de *DevOps*, principalmente nas definições de *sprints* e reuniões de equipe. Ele será o responsável por levar a equipe de suporte informações relevantes (desde a arquitetura implementada, novas funcionalidades, *bugs* conhecidos, gerar as documentações necessárias, dentre outras). Diogo acredita que este modelo trará grandes benefícios para que a equipe de suporte gere mais valor ao produto.

Para Eduardo (Gerente de Tecnologia) há de se haver uma medição da efetividade dos diferentes níveis de suporte (1, 2 e 3), para identificarmos os problemas existentes e entendermos as necessidades de cada nível, e então fornecer ferramentas apropriadas. Em paralelo, ter uma monitoração constante de todo o ambiente utilizando: *Health Check*, *Dashboards*, *Logs*, *Watchdog*, *UX* (*User Experience*), dentre outros. Tuane (*Scrum Master*) define um mínimo de documentações a serem geradas a cada *release*, sempre focadas na arquitetura de novas funcionalidades. Flaviana (*Agile Coach*) faz a passagem para o time de sustentação no final de cada *sprint*, e está em busca de minimizar a documentação gerada para o time de sustentação, além de trazer um representante para as reuniões de *planning poker*.

Além dos possíveis modelos de atuação, observei também práticas que não deveriam ser adotadas. Absorver o time de suporte na equipe de *DevOps* não seria recomendado, pois o time acabaria sendo acionado frequentemente pelo suporte para temas corriqueiros do dia a dia. Outra prática que não foi recomendada é a de ter uma equipe *DevOps* operando 24x7, pois um dos problemas seria o esforço na comunicação de toda equipe (dividida em turnos), além de que o aumento do time vai contra os princípios da cultura, que preza por uma equipe enxuta. Uma saída para a necessidade de ter a sua equipe de prontidão em horários não comerciais, seguindo o modelo de sobreaviso.

Conclusões e Lições aprendidas

Apesar da cultura estar amplamente difundida no ambiente de desenvolvimento de *software*, vejo que não existe um modelo de atuação com o suporte que é padrão de mercado. As empresas ainda estão se adequando a novos processos e novas culturas, e diferentes propostas estão surgindo para relacionar ambos os times.

Dentre as soluções propostas para este problema, vejo com bons olhos a aproximação do time de *DevOps* com o suporte através do papel do *SRE*, mas sem que ele seja absorvido para atividades do time de desenvolvimento. Além disto, uma documentação mínima a ser gerada para a equipe se suporte é vital para que ela cumpra o seu papel, do contrário ela ficará “de mãos atadas”.

Fato é que aplicando *DevOps* com as técnicas pregadas pela sua cultura, o número de incidentes diminui substancialmente, mas mesmo assim continuaremos com problemas que só podem ser tratados com a própria equipe de *DevOps*. As urgências que necessitam de atuação imediata fora do horário de trabalho da equipe (que se tornam bem menos frequentes), devem ser tratadas como exceção, seja concedendo acesso remoto à equipe, ou em caráter de sobreaviso.

Referências

1. CAMS – DEVOPS DICTIONARY. Disponível em :<http://devopsdictionary.com/wiki/CAMS>. Acesso em 04 mar. 2018.
2. MANIFESTO ÁGIL PARA DESENVOLVIMENTO DE SOFTWARE. Disponível em: <http://www.manifestoagil.com.br/>. Acesso em 04 mar. 2018.
3. KIM G. *et al*,. **The DevOps Handbook: How to create world-class agility, reliability, & security in technology organizations.** 1 ed., Portland, USA: IT Revolution, 2015
4. DIFERENÇAS ENTRE INTEGRAÇÃO, DEPLOY E ENTREGA CONTÍNUA | 4Linux. Disponível em: <https://www.4linux.com.br/diferencas-entre-integracao-deploy-e-entrega-continua>. Acesso em 04 mar. 2018.
5. GOOGLE – SITE RELIABILITY ENGENEERING. Disponível em: <https://landing.google.com/sre/book/index.html>. Acesso em 04 mar. 2018.

Sobre os Autores:

Daniel Ventura de Oliveira
Especialista de Governança Estratégica – TV Globo
Daniel.veol@gmail.com

Mini currículo
Graduado em Ciência da Computação pela UCP, com MBA em Gestão de Projetos pela FGV, certificado COBIT 5 e PMP desde 2017, tendo feito 2 intercâmbios. 10 anos de experiência com desenvolvimento de software. Atualmente PMO de projetos de Mídias Digitais e Negócios na TV Globo e especialista de Governança Estratégica, focado em Inovação e Mídias Digitais.