Pró-Reitoria Acadêmica
Escola de Educação, Tecnologia e Comunicação
Lato Sensu em Engenharia de Software

O IMPACTO DOS MÉTODOS ÁGEIS NA GESTÃO DA DISPONIBILIDADE DE SERVIÇOS HOSPEDADOS EM UM DATACENTER

Autor: Lincoln S. Carvalho
Orientador: Beatriz Araujo

Brasília – DF
2019

LINCOLN SILVA CARVALHO

O IMPACTO DOS MÉTODOS ÁGEIS NA GESTÃO DA DISPONIBILIDADE DE SERVIÇOS HOSPEDADOS EM UM DATACENTER

Trabalho de Conclusão de Curso apresentado na Universidade Católica de Brasília como requisito básico para a conclusão do Programa de Pós-Graduação Lato Sensu em Engenharia de Software

Orientador: Profª. Beatriz Araujo

Brasília

2019

Campus I - QS 07 – Lote 01 – EPCT – Águas Claras – Brasília – DF CEP: 71966-700 - (61) 3356-900
Campus Avançado Asa Norte - SGAN 916 Módulo B Avenida W5 - CEP: 70790-160 - Brasília/DF - Telefone: (6
Campus Avançado Asa Sul - SHIGS 702 Conjunto 2 Bloco A - CEP: 70330-710 - Brasília/DF - Telefone: (61)

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O cenário empresarial está cada vez mais dinâmico. A velocidade que as inovações tecnológicas são introduzidas são cada vez maiores, bem como o nível de competitividade e as mudanças nas necessidades dos clientes.

O mercado atual vive em constante mudança e elas ocorrem de forma rápida a ponto de não ser possível determinar antecipadamente. O mesmo acontece com os softwares, visto que estão diretamente integrados com os processos diários de um negócio ou empresa.

E em um mundo de rápidas mudanças e grande competitividade, a inovação se tornou essencial para a sobrevivência das empresas. É necessário estar sempre um passo à frente, prevendo as novas tendências do mercado e as próximas expectativas dos clientes.

Nesse contexto, tem aumentado a busca por adaptações das abordagens, técnicas e ferramentas até então utilizadas e propagadas ao redor do mundo em virtude do crescimento das teorias e métodos ágeis.

A adoção das metodologias ágeis cresceu exponencialmente em anos recentes. Diversas empresas e companhias acostumadas com os modelos tradicionais estão se rendendo e gradativamente tem adotado os métodos ágeis em seus ambientes de desenvolvimento, gestão, produção, pois um ambiente de crescentes mudanças exige inovações constantes e criativas. Além disso, há também as startups que já nascem utilizando o ágil desde sua criação. Isso tem gerado muitos resultados positivos: eficiência operacional, economia de recursos, melhoria na imagem das empresas, agilidade na entrega de um produto, entre outras.

Muitos tem apontado os métodos ágeis como uma alternativa eficaz às metodologias tradicionais. Isso acontece porque ela promete a entrega de software de qualidade com rapidez e satisfação aos clientes. Na verdade, como lembra Caroli (2015), ninguém quer desperdiçar tempo, dinheiro e esforço construindo um produto que não vai atender as expectativas.

Segundo Schwaber e Beedle (2002), as metodologias ágeis se adaptam a novos fatores decorrentes do desenvolvimento de software, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento. Ou seja, os métodos ágeis possuem como característica principal serem adaptativos, não preditivos.

Mas o que dizer de ambientes que precisam ser estáveis, seguros, com alta disponibilidade e que tem a responsabilidade de executar transações valiosas? Esse cenário pode ser encontrado em diversos datacenters que hospedam serviços e aplicações de empresas de diversos setores da economia. Será que o uso da metodologia ágil tem se mostrado eficiente também nesses ambientes? Tem gerado resultados positivos? Como empresas que hospedam seus serviços e aplicações em um datacenter podem se beneficiar dos métodos ágeis para tornar a gestão da disponibilidade mais eficiente?

A finalidade desse trabalho é apresentar o estudo de caso realizado em um datacenter que vem utilizando os métodos ágeis para automatizar processos e

apresentar os resultados dessa adoção. Além disso, será apresentado uma visão geral da metodologia ágil, tendo como foco o SCRUM.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é abordar a definição e as características dos métodos ágeis e analisar qual o impacto das metodologias ágeis e os resultados de sua utilização no processo de gestão da disponibilidade dos serviços hospedados em um datacenter.

1.2 OBJETIVOS ESPECÍFICOS

Como objetivos específicos, este trabalho pretende:

- Apresentar os conceitos da metodologia ágil;
- Apresentar o método ágil Scrum;
- Analisar o impacto dos métodos ágeis na automação da gestão de serviços hospedados em um datacenter;
- Avaliar os resultados da utilização dos métodos ágeis no desenvolvimento de ferramentas para gestão da disponibilidade de aplicações hospedadas em um datacenter.

1.3 ORGANIZAÇÃO DO TRABALHO

Este artigo está organizado em 5 capítulos.

O primeiro capítulo traz a introdução ao conteúdo a ser apresentado, com a contextualização, o objetivo geral e os objetivos específicos.

O segundo capítulo apresenta o referencial teórico.

O terceiro e quarto capítulo compreende a metodologia de pesquisa utilizada e o estudo de caso realizado, respectivamente.

O quinto capítulo compreende as interpretações do autor e considerações finais (conclusão) e o sexto capítulo as referencias bibliográficas.

2 REFERENCIAL TEÓRICO

2.1 METODOLOGIAS ÁGEIS

2.1.1 Definições e conceitos

As metodologias ágeis são um conjunto de processos projetados para o desenvolvimento rápido de um software. Estes processos costumam ser iterativos, intercalando as atividades de especificação, projeto, desenvolvimento e teste (Sommerville, 2011).

De acordo com Pressman (2007), a Engenharia de Software abrange três componentes básicos: métodos, que proporcionam os detalhes de como construir um software (planejamento, estimativas, análise de requisito, entre outros); ferramentas, que sustentam cada um dos métodos; e procedimentos, que definem a sequencia em que os métodos são aplicados e conectam os métodos as ferramentas.

Antes da popularização das metodologias ágeis havia a ideia de que era necessário planejamento cuidadoso, métodos de análise e processo de desenvolvimento de software rigoroso e controlado para chegar a um produto de qualidade. Essa concepção veio da comunidade de engenharia de software, responsável pelo desenvolvimento de sistemas de software grandes e duradouros.

Mas esse modelo não conversou muito bem com o desenvolvimento de sistemas corporativos de pequeno e médio porte. Muito tempo era perdido na análise de como o sistema deveria ser desenvolvido, em contraste com o pouco tempo empregado no desenvolvimento em si e nos testes.

Foi então que surgiram os métodos ágeis, no ano de 2001, quando um grupo de especialistas na área de computação se reuniram para discutir sobre métodos de desenvolvimento de software. Suas reuniões originaram o Manifesto Ágil, que foi definido como melhores maneiras de desenvolver softwares.

Os quatro valores defendidos pelo Manifesto Ágil são (SABBAGH, 2013):

1. **Os indivíduos e suas interações** acima de procedimentos e ferramentas;
2. **O funcionamento do software** acima de documentação abrangente;
3. **A colaboração dos clientes** acima da negociação de contratos;
4. **A capacidade de resposta às mudanças** acima de um plano preestabelecido.

Mesmo havendo valor nos itens à direita, os itens à esquerda são mais valorizados. Além disso, foram definidos doze princípios com o intuito de auxiliar as pessoas a compreenderem melhor o desenvolvimento de software ágil. Esses

princípios foram criados a partir do Manifesto Ágil por parte de seus autores (Beck et al, 2001). Sao eles:

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado. Foco no desenvolvimento do produto está na satisfação dos clientes.

2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente. Aceitar as mudanças como algo natural no processo de desenvolvimento.

3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo. Isso permite *feedback* sobre o que foi produzido.

4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho. Quem constrói o produto são pessoas. Então elas precisam estar em uma condição favorável para desempenharem bem suas funções.

6. O método mais eficiente e eficaz de transmitir informações para e entre um time de desenvolvimento é através de conversa face a face. Essa comunicação é a mais indicada por ser direta, apresentar a linguagem corporal e outros elementos importantes.

7. Software funcionando é a medida primária de progresso.

8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.

10. Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial. Isso evita o desperdício no desenvolvimento ao não realizar trabalho que não é necessário.

11. As melhores arquiteturas, requisitos e designs emergem de times auto organizados.

12. Em intervalos regulares, o time reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

O foco central dos métodos ágeis é o desenvolvimento de software, ao invés de sua concepção e documentação. Ele é baseado na abordagem incremental de especificação, desenvolvimento e entrega de software e é indicado para aplicativos que mudam rapidamente os requisitos de sistema durante o processo de construção.

Esse método de desenvolvimento busca “entregar o software rapidamente aos clientes, em funcionamento, e estes podem, em seguida, propor alterações e novos requisitos a serem incluídos nas iterações posteriores” (SOMMERVILLE, 2011).

Com isso é possível reduzir a burocracia do processo e evitar documentação que possivelmente não será usada. Segundo Sommerville (2011) os “entusiastas de métodos ágeis argumentam que é um desperdício de tempo criar documentação formal”. O segredo é a produção de códigos de alta qualidade, códigos bem estruturados. Ou seja, é necessário investir na melhoria do código.

O fato é que o Manifesto Ágil não abandona os processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, mas simplesmente mostra que eles têm importância secundária quando comparado com os indivíduos e interações, com o software funcionando, com a colaboração com o cliente e as respostas rápidas a mudanças e alterações (LIBORDI; BARBOSA, 2010).

Segundo Sommerville (2007), os métodos ágeis foram propostos pelos próprios desenvolvedores, devido à insatisfação que possuíam em utilizar as metodologias inadequadas no desenvolvimento de projetos pequenos onde o tempo gasto com o planejamento e documentação era superior ao tempo de desenvolvimento.

2.1.2 Agilidade

Segundo Pressman (2016), a agilidade: incentiva a melhora da comunicação, tornando-a mais fácil e abrangente entre os membros da equipe; enfatiza a entrega rápida do incremento útil do sistema; diminui a importância de artefatos intermediários; acolhe o cliente como um membro da equipe, eliminando as possíveis barreiras existentes entre ele e os outros membros da equipe; reconhece os limites existentes que um planejamento pode ter e que o plano de projeto deve ser flexível.

Agilidade ou ser ágil, não é simplesmente um adjetivo ou um método pronto, é uma competência de equipes em ambientes de gerenciamento de projetos. Logo, para criar esta competência nas organizações, alguns aspectos e elementos devem ser considerados como, a cultura e estrutura organizacional, as práticas, ferramentas e técnicas de gerenciamento, o ambiente de negócios, as experiências, habilidades e competências dos integrantes, ideologias e motivações (Conforto et al, 2014).

Para Highsmith (2002), agilidade é a habilidade de criar e responder às mudanças, de maneira lucrativa, em um ambiente turbulento de negócios. Já Abrahamsson et al. (2002) definem ágil como a qualidade de ser ágil, estar de prontidão para movimentar-se, vivacidade, atividade e destreza para movimentar-se. O aspecto principal das metodologias ágeis está em sua simplicidade e velocidade, se concentrando em funções necessárias, entregas rápidas e coletando feedback e reações das informações recebidas. As características que fazem de um método ágil são: Desenvolvimento incremental; cooperação entre clientes e desenvolvedores; direto e adaptativo.

“O termo ágil se refere a capacidade de se mover ou de responder de forma ágil e fácil. Como uma qualidade, ágil deve ser um objetivo a ser alcançado” (SATPATHY, 2017). A definição apresentada por Satpathy (2017) casa com a afirmação de Caroli (2015), quando ele diz que projetos ágeis enfatizam

entregas antecipadas e contínuas de acordo com os objetivos do negócio e as necessidades dos usuários.

Além disso, de acordo com Sabbagh (2013) o nome Ágil foi escolhido para representar um movimento que surgiu como resposta aos pesados métodos de gerenciamento de desenvolvimento de software, também conhecidos como métodos tradicionais. Segundo Sommerville (2011), os métodos ágeis são métodos de desenvolvimento em que os incrementos são pequenos e são criadas e disponibilizadas versões do produto rapidamente aos clientes para que os mesmos efetuem *feedbacks.* Esse método minimiza a documentação. Sendo assim, o objetivo é construir de forma rápida um produto útil.

2.2. *SCRUM*

Scrum não é um processo ou uma técnica para o desenvolvimento de produtos. Ele é um *framework* que permite utilizar diversos processos e técnicas. Tem sido utilizado para gerenciar o desenvolvimento de produtos complexos desde o início de 1990 (Schwaber e Sutherland, 2016).

O *Scrum* é uma metodologia de adaptação, iteratividade, rapidez, flexibilidade e eficiência, projetada para fornecer um valor significativo de forma rápida durante todo o projeto, garantindo transparência na comunicação e criando um ambiente de responsabilidade coletiva e progresso contínuo (Satpathy, 2017).

De acordo com Sabbagh (2013), "*Scrum* é um *framework* ágil, simples e leve, utilizado para a gestão do desenvolvimento de produtos complexos imersos em ambientes complexos". Ele pode ser classificado como ágil porque sua utilização deve seguir os princípios e valores do Manifesto para o Desenvolvimento Ágil de *Software.*

Como mencionado o *Scrum* é um *framework*, ou seja, uma estrutura básica que serve de suporte e guia para a construção. Visto que é um *framework*, o *Scrum* não define práticas específicas e detalhadas a serem seguidas, como devem ser levantadas as necessidades do negócios, como deve lidar com os clientes, nem como o time deve desenvolver o produto (Sabbagh, 2013).

Segundo Ribeiro e Ribeiro (2015), o "*Scrum"* é baseado em três pilares:

- Transparência: O processo deve estar visível para todos. A transparência se estende não somente aos processos como ao ambiente de trabalho e às pessoas;
- Inspeção: Anomalias e oportunidades de melhora devem ser rastreadas em inspeções frequentes;
- Adaptação: Adaptações devem ser realizadas o mais rápido possível caso sua necessidade seja identificada nas inspeções.

2.2.1 Valores do *Scrum*

Além dos já mencionados valores e princípios ágeis, o *Scrum* possui o seu próprio conjunto de valores a serem seguidos que complementam as regras do *Scrum* descritas em seus papéis, artefatos e eventos. Assim, os valores do *Scrum*

representam os pilares para todo o trabalho realizado pelas pessoas que trabalham no projeto. Os cinco valores são: foco, coragem, franqueza, compromisso e respeito (Sabbagh, 2013).

De acordo com Schwaber e Sutherland (2016), quando os valores do *Scrum* são assumidos e vividos pelo *Scrum Team*, os pilares do *Scrum* de transparência, inspeção, e adaptação tornam-se vivos e constroem a confiança para todos. À medida que o Time trabalha com os eventos, papéis e artefatos do *Scrum,* eles aprendem e exploram esses valores. O sucesso no uso do *Scrum* depende das pessoas se tornarem mais competentes no uso desses valores.

2.2.2 Papéis *Scrum*

O *Scrum* implementa sua estrutura iterativa e incremental através de três papéis: o *Scrum Master,* o *Product Owner (PO), e* o *Scrum Team,* como pode ser visualizado na Figura 1. Toda responsabilidade no projeto é dividida entre esses três papéis (SCHWABER, 2002).

Visto que são auto organizadas, escolhem a melhor forma de realizar seu trabalho. O modelo de equipe no *Scrum* é projetado para otimizar a flexibilidade, criatividade e produtividade. Suas entregas de produtos acontecem de forma iterativa e incremental (Sabbagh, 2013; Schwaber e Sutherland, 2016).

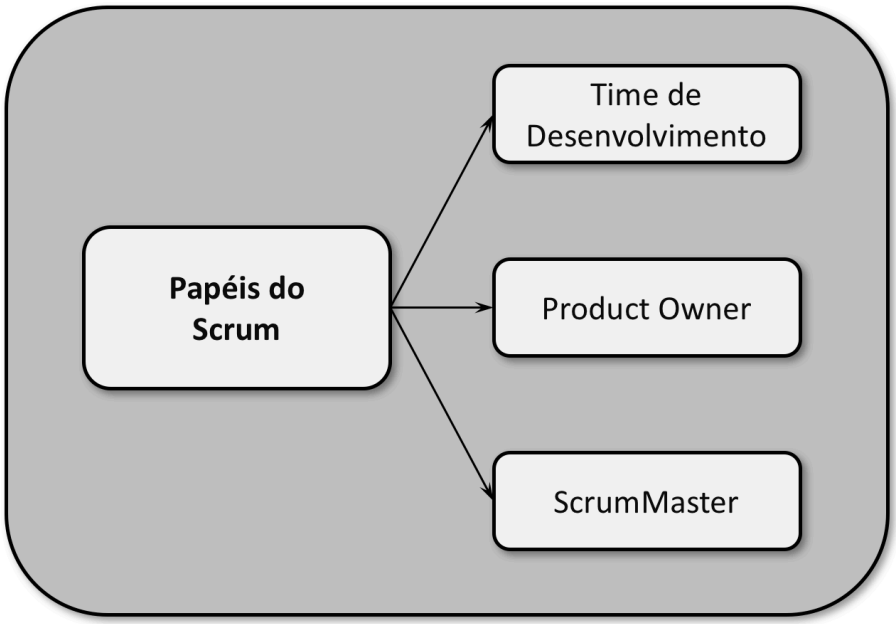

Figura 1 - Papéis *Scrum* (Sabbagh, 2013)

2.2.2.1 *Scrum Master*

O *Scrum Master* facilita e potencializa o trabalho do Time de *Scrum*. Utilizando seu conhecimento de *Scrum*, habilidade de lidar com pessoas, técnicas de facilitação e outras técnicas, ele ajuda o *Product Owner* e o Time de Desenvolvimento a serem mais eficientes na realização do seu trabalho. Além disso, é responsável por garantir que o *Scrum* seja compreendido e colocado em prática

por toda a equipe, fazendo dele uma cultura dentro da organização. Para alcançar esse objetivo o *Scrum Master* tem a responsabilidade de ensinar as pessoas envolvidas e implementá-lo (Sabbagh, 2013; Schwaber e Sutherland, 2016).

Abrahamsson et al. (2002) menciona que o *Scrum Master*, certifica que o projeto está seguindo as práticas, valores e regras do *Scrum* e dessa forma está progredindo como planejado.

Por ter a responsabilidade de moderar e facilitar a interação do time, o *Scrum Master* age como motivador e mentor do time. Além disso, garante que o time tenha um ambiente de trabalho produtivo, protege o time de influências externas, remove qualquer impedimento e aplica os princípios e aspectos do *Scrum* (Satpahty, 2017).

2.2.2.2 *Product Owner*

Segundo Sabbagh (2013) “o *Product Owner* é o responsável pela definição do produto” que é realizado de forma incremental ao longo de todo o projeto. Schwaber e Sutherland (2016) afirma que ele é “responsável por maximizar o valor do produto e do trabalho do Time de Desenvolvimento”.

O *Product Owner* é responsável por representar os interesses dos *stakeholders* no projeto e definir todos os itens de requisito do projeto numa lista chamada *Product Backlog*. Utilizando essa lista ele é responsável por garantir que as funcionalidades que agreguem maior valor ao produto sejam desenvolvidas primeiro. Isto é feito através da freqüente priorização do *Product Backlog* para que os itens de maior valor agregado sejam entregues a cada iteração (Satpahty, 2017).

Para Ribeiro e Ribeiro (2015), o *Product Owner* é o representante do cliente, aquele que deve compreender as necessidades do cliente e repassar seu entendimento para a equipe de desenvolvimento. Além disso, ele tem a responsabilidade de sanar dúvidas da equipe de desenvolvimento e receber feedbacks dos clientes. Já Schwaber e Sutherland (2016) afirma que o *Product Owner* é uma pessoa e não um comitê, mesmo que represente os desejos de um comitê. E para que tenha sucesso, toda a organização deve respeitar as suas decisões.

2.2.2.3 *Scrum Team*

O *Scrum Team* é formada por profissionais que realizam o trabalho de desenvolvimento do produto de ponta a ponta. É essa equipe que gera um incremento do produto que agrega valor para os clientes do projeto (Sabbagh, 2013).

Recomenda-se que o *Scrum Team* ideal tenha de 6 a 10 pessoas auto-organizáveis, autogerenciáveis e multifuncionais. Nela, não existe necessariamente uma divisão funcional através de papéis tradicionais, tais como programador, designer, analista de testes ou arquiteto. Todos no projeto trabalham juntos e são responsáveis por completar o conjunto de trabalho com o qual se comprometem a cada iteração. A equipe não pode ser alterada durante o *Sprint* (Satpahty, 2017).

O *Scrum Team* é responsável por realizar as atividades necessárias para as entregas de cada *sprint*, tendo o respaldo da organização para organizar e gerenciar seu próprio trabalho. A equipe que forma o *Scrum Team* deve ser pequena o bastante para ser ágil, porém deve ser grande o suficiente para completar o trabalho necessário dentro do tempo definido para o *Sprint* (Schwaber e Sutherland, 2016).

A equipe de desenvolvimento tem autoridade para agir e se auto organizar com a finalidade de cumprir as metas de cada *"Sprint".* ela é envolvida nas estimativas de esforço das atividades, na criação do *"Sprint backlog"*, revisão do "*backlog* do produto" e indicando impedimentos do projeto (ABRAHAMSSON et al., 2002).

2.2.3 Eventos

Os eventos *Scrum* são usados para minimizar a necessidade de reuniões não definidas no *Scrum*. Todos os eventos têm uma duração máxima estipulada. Além do próprio *Sprint*, que é um recipiente para todos os outros eventos, cada evento em *Scrum* é uma oportunidade formal para inspecionar e adaptar (Schwaber e Sutherland, 2016). A Figura 2 apresenta os eventos *Scrum* - alguns deles serão detalhados neste trabalho.

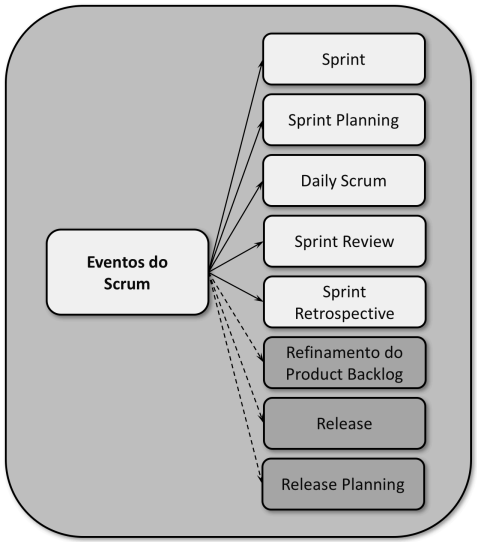

Figura 2 - Eventos do *Scrum* (Sabbagh, 2013)

2.2.3.1 *Sprint*

O *Sprint* é o ciclo de desenvolvimento, a iteração, que se repete ao longo de todo o projeto, um atrás do outro. Em cada *Sprint*, o *Scrum Team* planeja, desenvolve um incremento do produto, coleta feedback e busca melhorar sua forma de trabalhar (Sabbagh, 2013).

A *Sprint* é o coração do *Scrum.* Cada *Sprint* tem um intervalo fixo de tempo, com duração de até quatro semanas, no qual uma versão incremental utilizável do

produto é desenvolvida. As *Sprints* são cíclicas, sempre começando ao término da anterior. Além disso, cada *Sprint* pode ser considerada um pequeno projeto, com seu planejamento e suas metas do que deve ser construído. (Schwaber e Sutherland, 2016).

De acordo com Ribeiro e Ribeiro (2015), fazem parte da *Sprint* uma reunião de planejamento no seu início, reuniões diárias, as atividades de desenvolvimento, uma reunião de revisão da *Sprint* e uma reunião de retrospectiva. Já Schwaber e Sutherland (2016) apontam como principal benefício do uso da *Sprint* a previsibilidade, pois garante ao menos uma vez por mês uma inspeção e a adaptação do progresso em direção aos objetivos, além de limitar os riscos de retrabalho ao trabalho de um mês.

O cancelamento da *Sprint* antes de seu final pode ser realizado apenas pelo *Product Owner.* Embora raro, pode ocorrer quando os objetivos do *Sprint* não agregarem mais valor ao produto, em casos de mudança das estratégias empresariais ou alterações do mercado ou tecnológicas (Schwaber e Sutherland, 2016).

2.2.3.2 *Sprint Planning Meeting*

No primeiro dia de cada *Sprint* é realizada a *Sprint Planning Meeting*, que deve ter uma duração máxima de 5% do tempo definido para o *Sprint* (Reunião de 8 horas para *Sprint* de quatro semanas), sendo esta, dividida em duas partes: a primeira, quando é debatido o que será entregue ao final da *Sprint* e a segunda, como será realizado o trabalho para esta entrega (Ribeiro e Ribeiro, 2015).

- Primeira parte: nas primeiras quatro horas o PO apresenta os itens de maior prioridade do *Product Backlog* para a equipe. O *Scrum Team* faz perguntas para entender melhor as funcionalidades e de forma colaborativa define o que poderá entrar no desenvolvimento do próximo *Sprint*, considerando o tamanho da equipe, a quantidade de horas disponíveis e a produtividade do *Scrum Team.*
- Segunda parte: durante as quatro horas finais o *Scrum Team* planeja seu trabalho, definindo o *Sprint Backlog*, que são as tarefas necessárias para implementar as funcionalidades selecionadas no *Product Backlog* e revisam as estimativas de esforço para conferir se a *Sprint* está adequada.

A responsabilidade de realizar a reunião é do *Scrum Master*, que também é responsável pela duração e pelo entendimento de seu propósito por todos os participantes. Faz-se necessário a presença do PO, bem como do *Scrum Master* e do *Scrum Team* que recebem do PO uma descrição das funcionalidades de maior prioridade para a equipe (Schwaber e Sutherland, 2016).

2.2.3.3 *Daily Scrum Meeting*

A cada dia do *Sprint* o *Scrum Master* realiza uma reunião de 15 minutos com o *Scrum Team*. Essa reunião, chamada *Daily Scrum Meeting*, tem como objetivo disseminar informações sobre o estado do projeto. De acordo com Schwaber e

Sutherland (2016), a *Daily Scrum Meeting* visa sincronizar as atividades e criar um plano para as 24 horas seguintes.

As *Daily Scrum Meetings* devem ser realizadas no mesmo lugar, na mesma hora do dia. Idealmente são realizados na parte da manhã, para ajudar a estabelecer as prioridades do novo dia de trabalho. Embora qualquer pessoa possa participar da reunião, somente os membros do *Scrum Team* estão autorizados a falar (Schwaber e Sutherland, 2016).

Ribeiro e Ribeiro (2015) indicam três questões comuns que todos os integrantes da equipe devem responder durante a *Daily Scrum Meeting:*

- O que realizei desde ontem?
- O que irei realizar até a próxima reunião?
- Há algum impedimento que esteja atrapalhando meu trabalho?

Schwaber e Sutherland (2016) afirma que “reuniões diárias melhoram a comunicação, eliminam outras reuniões, identificam e removem impedimentos para o desenvolvimento, destacam e promovem rápidas tomadas de decisão, e melhoram o nível de conhecimento do Time de Desenvolvimento”.

A Daily Scrum Meeting não é uma reunião de status na qual um chefe fica coletando informações sobre quem está atrasado. Ao invés disso, é uma reunião na qual os membros da equipe assumem compromissos perante os demais. Também não deve ser usada como uma reunião para resolução de problemas. No entanto, os impedimentos identificados na *Daily Scrum Meeting* devem ser tratados pelo *Scrum Master* o mais rapidamente possível.

2.2.3.4 *Sprint Review Meeting*

A Sprint Review Meeting acontece no último dia do *Sprint.* Nessa reunião, planejada para ser realizada em no máximo 4 horas, o *Scrum Team* mostra o que foi desenvolvido durante o *Sprint*. Tipicamente, isso tem o formato de uma demonstração das novas funcionalidades. Durante a *Sprint Review Meeting*, o projeto é avaliado em relação aos objetivos do *Sprint*, determinados durante a *Sprint Planning Meeting*. O ideal é que a equipe tenha finalizada cada um dos itens do *Sprint Backlog* (Schwaber e Sutherland, 2016).

Sprint Review é uma reunião onde o Time de Desenvolvimento e o *Product Owner* trabalham em conjunto, com a facilitação do *Scrum Master*, para demonstrar o trabalho pronto produzido durante o *Sprint*, para clientes do projeto e demais partes interessadas. Seu principal objetivo é a obtenção de *feedback* antecipado dessas pessoas sobre o Incremento do Produto produzido, já que um *feedback* mais concreto e profundo só poderá ser obtido após a entrega para seus usuários. O *feedback* obtido será usado pelo PO como matéria-prima para modificar o *Product Backlog* em *Sprints* futuras (Sabbagh, 2013).

Sabbagh (2013) acrescenta que “o propósito da reunião de *Sprint Review* não é o de se obter a aprovação formal dos clientes sobre o que foi feito no *Sprint*… não é uma sessão de testes de aceitação”. Essa aprovação que concretiza uma entrega deve ser realizada em outra ocasião, fora do contexto do *Scrum*.

2.2.3.5 *Sprint Retrospective Meeting*

A *Sprint Retrospective Meeting* é uma reunião de três horas que ocorre ao final de um *Sprint,* após a *Sprint Review Meeting,* e serve para identificar o que funcionou bem, o que pode ser melhorado e que ações serão tomadas para melhorar.

Sabbagh (2013) declara que durante a *Sprint Retrospective Meeting* o *Scrum Team* "inspeciona o *Sprint* que está se encerrando quanto a seus processos de trabalho, dinâmicas, pessoas, relacionamentos, comportamentos, práticas, ferramentas utilizadas e ambiente, e planeja as melhorias necessárias".

Para Schwaber e Sutherland (2016), a Sprint *Retrospective Meeting* é uma reunião para o *Scrum Team* inspecionar a si mesmo e criar um plano de melhorias a serem executados no próximo *Sprint.*

Segundo Sabbagh (2013), a *Sprint Retrospective Meeting* não deve ser utilizada para se identificarem ações de melhoria no produto, mas sim identificar o que precisa ser melhorado na forma de trabalho do *Scrum Team.*

2.2.4 Artefatos

Os artefatos do *Scrum* representam trabalho ou valor que fornecem oportunidades para inspecionar e adaptar. Artefatos definidos pelo Scrum são especificamente desenhados para maximizar a transparência de informação chave para que todas as pessoas tenham o mesmo entendimento sobre o artefato (Schwaber e Sutherland, 2016). A Figura 3 apresenta os artefatos do *Scrum.*

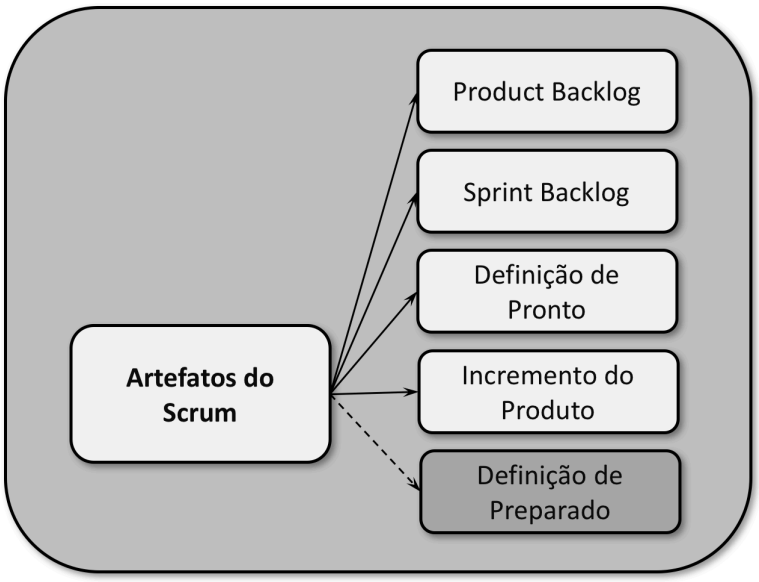

Figura 3 - Artefatos do *Scrum* (Sabbagh, 2013)

2.2.4.1 *Product Backlog*

O *Product Backlog* é uma lista ordenada de itens necessários no produto sobre os quais o Time de Desenvolvimento trabalhará no decorrer do projeto, e é a única fonte de requisitos para as alterações efetuadas a esse produto. O responsável pelo *Product Backlog* é o *Product Owner* (Sabbagh, 2013; Schwaber e Sutherland, 2016).

De acordo com Abrahamsson et al. (2002), o *Product Backlog* contém o trabalho a ser realizado durante o projeto, incluindo uma lista priorizada e constantemente atualizada dos requisitos do sistema, que pode ser elaborada por diversos interessados, tais como clientes, equipe de projeto, equipe de marketing, gerentes ou equipe de apoio ao cliente.

Ribeiro e Ribeiro (2015) adiciona que ao longo do projeto ocorrem modificações no *Product Backlog*. Por exemplo, pode ocorrer mudança nas prioridades ou inclusão/exclusão/alteração de itens. No entanto, o único que tem autoridade para realizar estas modificações é o *Product Owner*.

Segundo Schwaber e Sutherland (2016), o *Product Backlog* deve conter todas as características, funções, requisitos, melhorias e correções necessárias às versões futuras do produto além de estimativa de tempo, custo, descrições, prioridades e valores.

O *Product Owner* utiliza as estimativas de tempo do *Product Backlog* para acompanhar o andamento do projeto, avaliando e comparando com informações anteriores quanto de trabalho falta para se chegar no produto final. Este acompanhamento, normalmente é realizado nas reuniões de revisão do *Sprint*. *(Schwaber e Sutherland, 2016)*

2.2.4.2 *Sprint Backlog*

O *Sprint Backlog* é uma lista de itens do *Product Backlog* selecionados para o desenvolvimento do incremento do produto no Sprint, adicionada de um plano de como esse trabalho será realizado. O *Sprint Backlog* torna visível tudo que o Time de Desenvolvimento julga necessário para atingir o objetivo do *Sprint*. O *Sprint Backlog* é uma importante ferramenta utilizada para organização do trabalho durante o *Sprint* (Sabbagh, 2013; Schwaber e Sutherland, 2016).

De acordo com Sabbagh (2013), o *Sprint Backlog* “existe apenas no contexto de seu Sprint correspondente”, ou seja, nasce na reunião de *Sprint Planning* e deixa de existir após a realização da *Sprint Review* Meeting e *Sprint Retrospective Meeting*. Schwaber e Sutherland (2016) acrescenta que somente o “Time de Desenvolvimento pode alterar o *Sprint Backlog* durante a *Sprint*”.

2.2.4.3 Definição de Pronto

Definição de Pronto é um acordo formal entre *Product Owner* e *Scrum Team* sobre o que é necessário para se considerar que um trabalho realizado no *Sprint* está “pronto”. Todos precisam entender o que significa ter o trabalho concluído (Sabbagh, 2013; Schwaber e Sutherland, 2016).

De acordo com Sabbagh (2013), a “mesma Definição de Pronto se aplica a cada item do *Sprint Backlog* e ao Incremento de do Produto como um todo, que é o conjunto de dos itens desenvolvidos no *Sprint*”.

2.2.4.4 Incremento de Produto

O Incremento do Produto é a soma de todos os itens completos em um *Sprint*. É composto por novas funcionalidades e por melhorias no que foi produzido em *Sprints* anteriores (Sabbagh, 2013; Schwaber e Sutherland, 2016).

Sabbagh (2013) afirma que os itens não prontos ao final do *Sprint* nao fazem parte do Incremento do Produto. Na realidade, de acordo com a avaliação do *Product Owner* esses itens devem voltar ao *Product Backlog*.

O ideal é que ao final do *Sprint* o *Product Owner* possa decidir se o Incremento do Produto pode ser entregue aos clientes do projeto. Embora ele pode decidir acumular Incrementos do Produto para fazer uma entrega de uma só vez.

A apresentação do Incremento do Produto para os clientes é realizada pelo *Scrum Team* e o *Product Owner* no final de cada *Sprint*, na *Sprint Review Meeting*. O objetivo é receber feedback sobre o trabalho realizado e medir o progresso do projeto. No entanto, o Incremento do Produto deve conter funcionalidades que possam ser experimentadas e usadas pelos presentes na *Sprint Review Meeting* para que seja viável o feedback (Sabbagh, 2016). A Figura 4 traz uma ideia do ciclo do *Scrum.*

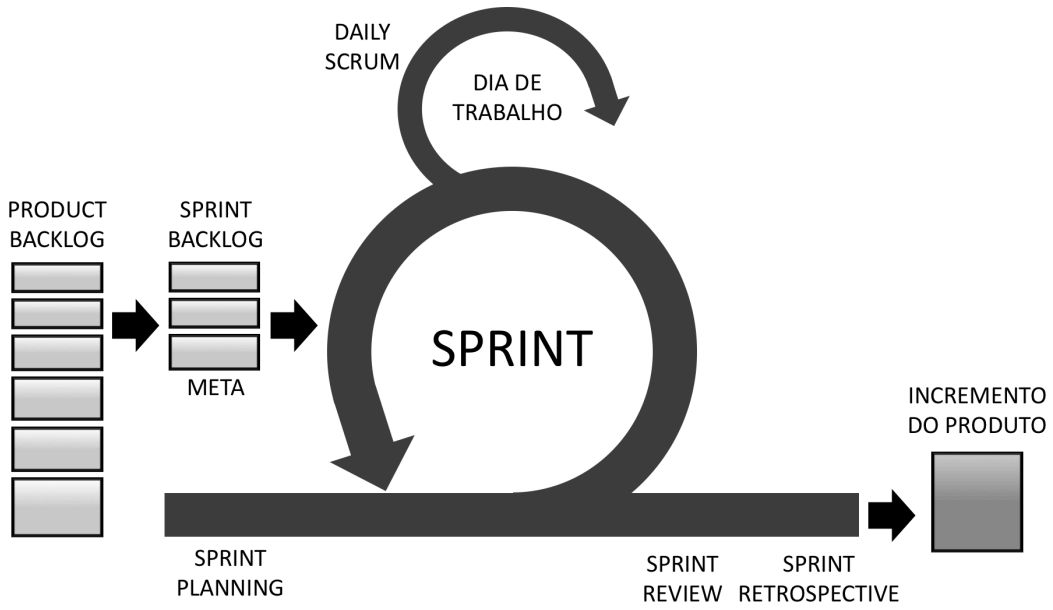

Figura 4 - O ciclo do *Scrum* (Sabbagh, 2013)

3 METODOLOGIA

Este trabalho buscou analisar o impacto da utilização do método ágil Scrum em um datacenter. Dessa forma, é caracterizado como um estudo de caso, com fins exploratórios, utilizando-se da pesquisa qualitativa como abordagem e tendo a observação como método de coleta de dados.

Conforme Yin (2010), o estudo de caso permite que sejam investigadas situações da vida real e que, delas, sejam absorvidas todas as características significativas dos eventos cotidianos. De acordo com Hartley (1994), o estudo de caso é uma investigação detalhada de uma ou mais organizações, ou grupos dentro de uma organização, com o objetivo de prover uma analise do contexto e dos processos envolvidos no fenômeno em estudo.

O estudo de caso, neste trabalho, foi usado como forma de exploração sobre o tema definido, não gerando dados para amostragem, e sim como uma abordagem qualitativa quanto à própria utilização do Scrum em sua capacidade de possibilitar mudanças positivas a uma empresa.

Por fim, foi realizada uma avaliação do uso do Scrum visando identificar e avaliar os impactos causados. A avaliação foi desenvolvida, utilizando a comparação entre o processo antes da implantação e o processo após a implantação, considerando-se os aspectos de gestão de escopo, tempo e qualidade.

4 O ESTUDO

Este capítulo apresenta uma caracterização do ambiente de estudo no qual a pesquisa foi realizada. Com o objetivo de analisar o impacto da metodologia Scrum nos processos que afetam a disponibilidade de um datacenter optou-se por desenvolver tal experimento aplicado a uma instituição financeira. Nesse sentido, procura-se apresentar uma introdução geral da empresa, bem como a identificação e comparação do processo anterior a implementação da metodologia e do processo posterior.

É importante destacar que, embora tenha sido permitido o pesquisador observar e obter informações dos processos e tirar conclusões do que foi observado, não foi autorizado utilizar ou apresentar nenhuma imagem, dado ou informação da instituição observada devido a natureza da atividade que exerce e da estratégia corporativa que a mesma possui frente aos concorrentes. Portanto, em respeito ao responsáveis que permitiram esse estudo de caso, serão utilizados termos e nomes genéricos na caracterização e apresentação das informações.

4.1 *A EMPRESA*

A Instituição Financeira (IF) pesquisada possui diversos produtos e serviços para sua base de clientes e utilizando a tecnologia é possível atendê-los de maneira mais eficiente e tempestiva. Portanto, pode-se afirmar que a tecnologia é uma grande aliada tanto em satisfazer os atuais clientes como prospectar novos.

Os clientes e potenciais clientes podem tanto se autoatender como serem atendidos por funcionários da Instituição Financeira. Devido a isso, a área de tecnologia precisa fornecer tecnologia tanto para os clientes externos (Clientes da IF) como os clientes internos (funcionários da IF).

4.2 *PRODUTO OBSERVADO*

Um dos recursos que a IF oferece aos seus clientes é a possibilidade de realizar transações pela internet utilizando tanto o navegador web pelo computador como o aplicativo do smartphone desenvolvido pela própria IF. Independente da forma utilizada pelo cliente, o datacenter onde a IF hospeda seus recursos tecnológicos é acessado.

Para atender esse produto específico há 10 instâncias de uma aplicação, rodando em 50 máquinas virtuais (VMs), hospedadas em um cluster com 8 servidores físicos. É fundamental que essa estrutura esteja disponível, funcional, acessível para os clientes. Do contrário a perda financeira para a IF pode ser gigantesca. Devido a isso, há diversos mecanismos, procedimentos, roteiros e funcionários para monitorar e manter esse ambiente funcionando dentro do datacenter.

Após uma avaliação por parte da IF foi identificado a necessidade de otimizar a forma como esse ambiente era monitorado e mantido disponível. Foi sugerido a automatização da monitoração e dos procedimentos executados para manter o ambiente em alta disponibilidade - alguns levavam horas para serem executados manualmente por uma pessoa. Para automatizar os processos seria necessário desenvolver um novo sistema.

4.3 PROCESSO DE DESENVOLVIMENTO COM SCRUM

O processo se inicia com a requisição da melhoria de uma funcionalidade existente ou a requisição de desenvolvimento de uma nova funcionalidade.

Todas as requisições, obrigatoriamente, devem passar pelo *Product Owner*, o qual é responsável pelo *Product Backlog*. O *Product Backlog* é refinado e priorizado diariamente com base no ROI de cada item. Os itens que possuem um ROI maior serão priorizados.

Com os itens do *Product Backlog* priorizados, é realizado o *Sprint Planning*. Esta reunião é dividida em duas partes. Na primeira parte, o *Product Owner*, *Scrum Master* e *Team* se reúnem com objetivo de identificar os itens de maior prioridade do *Backlog*, ou seja, o *Product Owner* passa para a equipe a visão do produto, informando maiores detalhes de cada demanda. Nessa reunião, é importante que o *Team* tire todas as dúvidas referentes às demandas com maior prioridade e o *Product Owner* detalha as demandas mais importantes, quebrando-as em partes menores.

Na segunda parte da reunião, o *Scrum Master* e a equipe se reúnem para detalhar as demandas priorizadas na primeira parte da reunião e o *Team* verifica o que poderá ser realizado durante o *Sprint* e se compromete a realizá-las.

Ao final do *Sprint Planning*, o *Sprint Backlog* está composto e a equipe poderá começar a realizar a implementação das demandas no *Sprint*. Durante o *Sprint*, diariamente, é realizado o *Daily Meeting*, quando o *Team* expõe as principais dificuldades encontradas no processo ao *Scrum Master*. Sendo o *Scrum Master* responsável por resolver os problemas descritos pelo *Team*.

Cada membro do *Team* verifica o *Sprint Backlog*, identifica uma demanda priorizada e, de comum acordo, às implementam. Após finalizar, o membro do *Team* responsável, atualiza o status do *Sprint Backlog*, informando que a demanda foi realizada. Havendo itens pendentes, o membro do *Team* verifica uma nova a ser realizada e reinicia o processo. A cada demanda realizada, o membro do *Team* atualiza o status no quadro para “Revisão” e a mesma entra na fila de testes.

Ao final do *Sprint*, com todas as demandas realizadas, o *Team* e o *Scrum Master* se reúnem e realizam o *Sprint Review* e *Sprint Retrospective,* cujo principal objetivo é analisar o que foi realizado no *Sprint* e indicar melhorias aos processos. Nesta última reunião, é verificado o que foi implantado e decidido se será ou não disponibilizado um release.

4.4 MELHORIAS EM RELAÇÃO AO MODELO ANTERIOR

A principal mudança que o uso do *Scrum* trouxe foram as entregas contínuas. A cada *Sprint* uma nova funcionalidade era entregue, um novo incremento que aprimorava a automação proposta. Ou seja, a cada 2 semanas uma nova funcionalidade estava pronta para entrar em produção. No modelo anterior, esperava-se meses para ver um conjunto de novas funcionalidades.

Com o novo modelo as equipes responsáveis pela saúde e disponibilidade do ambiente passaram a receber uma nova funcionalidade a cada *Sprint.* Foi possível reduzir esforço e automatizar tarefas já nas primeiras *Sprints*, o que possibilitou um ganho operacional para a IF. Isso significou utilizar melhor os recursos humanos que passavam horas executando comandos e procedimentos, pois com as primeiras automações foi possível se concentrar em outras atividades.

O modelo tradicional é eficaz e foi muito utilizado dentro da IF. Não há nada de errado com ele e ainda é utilizado em outros projetos dento da empresa objeto de estudo. No entanto, ficou evidente que a utilização da metodologia ágil trouxe melhorias no processo de desenvolvimento, tanto para os desenvolvedores que passaram trabalhar focados em pequenas tarefas; para os clientes que a cada 2 semanas recebiam uma nova funcionalidade; e para a IF que otimizou o processo e economizou recursos.

4.5 AVALIAÇÃO DO NOVO MODELO

Ao final da pesquisa foi possível perceber que o novo modelo de desenvolvimento utilizando o *Scrum* trouxe muitos benefícios para a IF. Entre os benefícios podem ser mencionados:

- Ganho operacional: como pode ser visto na Figura 5, os desenvolvedores passaram a focar em uma parte do projeto ao invés de olhar para o todo. O foco em módulos menores permitiu que o tempo entre desenvolvimento, teste e colocar em produção fosse reduzido significativamente. Dessa forma, funcionalidades que agregam valor são disponibilizadas a cada 2-4 semanas. Com esse modelo a IF conseguiu reduzir em 30% o custo com horas trabalhadas no período de 1 ano.

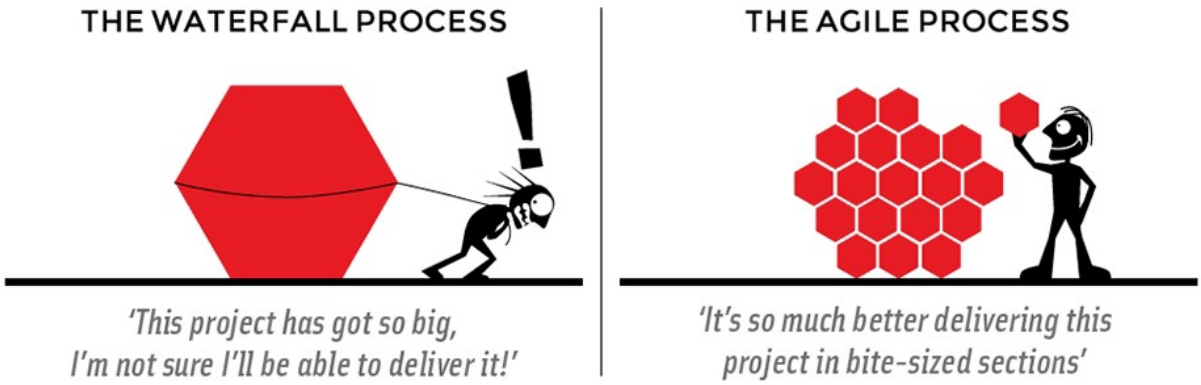

Figura 5 - Diferença entre Modelo Tradicional e Método Ágil

- Melhor gestão do tempo: no modelo tradicional perde-se muito tempo analisando, discutindo, desenvolvendo e testando funcionalidades que talvez nunca serão utilizadas devido alguma mudança na regra de negócio ou necessidade dos clientes. Visto que o método ágil de desenvolvimento é mais flexível no que diz respeito às mudanças, o tempo gasto com uma funcionalidade que pode vir a ser descartada é consideravelmente menor. A Figura 6 mostra como o uso do método ágil traz melhoria na gestão do tempo gasto na criação de conteúdo, permitindo a redução do desperdício de tempo.

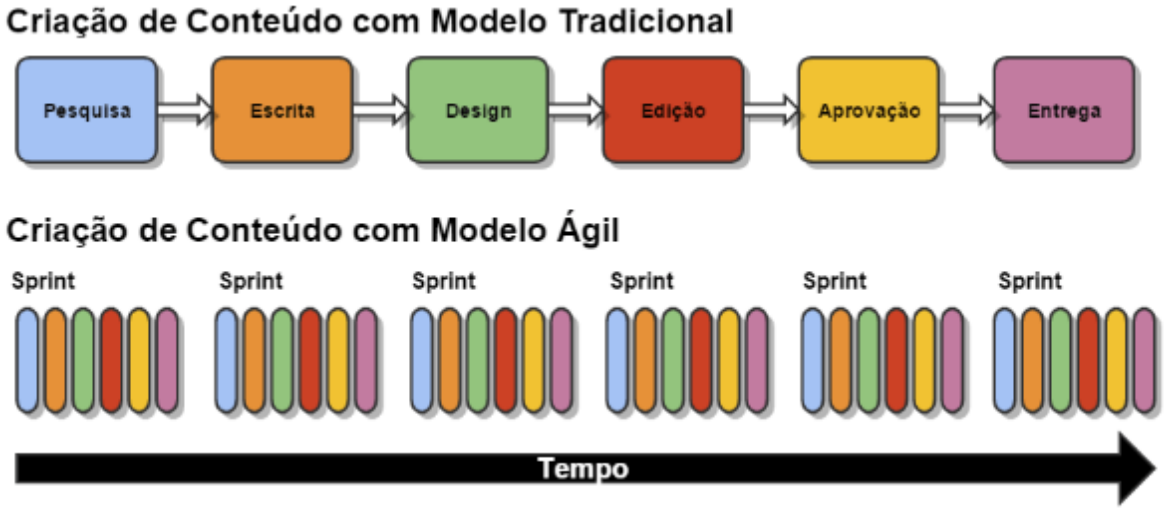

Figura 6 - Criação de conteúdo: diferença entre Modelo Tradicional e Método Ágil

- Melhoria na gestão de riscos: diferentemente do modelo tradicional que avalia os riscos ao final do projeto, no método ágil a avaliação dos riscos é realizada periodicamente, ao final de cada *Sprint.* Assim, é possível diminuir os riscos associados com as mudanças nas regras de negócio.

Como citado acima, o uso da metodologia ágil trouxe alguns benefícios ao processo de desenvolvimento. Visto que há diversas aplicações críticas em execução no Datacenter e que necessitam ser monitoradas e mantidas em alta disponibilidade, a IF decidiu extender o uso do Scrum no desenvolvimento de soluções que melhor atendam essa estrutura.

Sim, o modelo tradicional foi muito útil e ainda continua sendo utilizado. Mas não há como negar que o uso dos métodos ágeis trouxeram inúmeros ganhos para aqueles que o utilizam.

5 CONCLUSÃO

Todos os objetivos estabelecidos previamente foram cumpridos, realizando-se a revisão do referencial teórico, a teorização dos conceitos abordados, análise do modelo estudado e avaliação dos resultados.

Pode-se verificar que a implantação do *Scrum* apresentou resultados positivos para a área de desenvolvimento de software da Instituição Financeira observada.

Percebeu-se que a agilidade nos processos de desenvolvimento de software e a entrega constante de builds contribuem positivamente para a qualidade do software, tendo em vista que o *feedback* é recebido pela equipe horas após o seu lançamento, contribuindo para a melhoria contínua imposta pela forma iterativa de implementação.

O modelo de desenvolvimento utilizando o *Scrum* ainda é algo novo dentro da IF observada. Mas, devido aos ganhos obtidos ela tem avançado na implantação do uso do *Scrum* em outros projetos de desenvolvimento.

6 REFERÊNCIAS

ABRAHAMSSON,Pekka et al. **Agile Software Development Methods: Review and Analsis.** Espoo: Otamedia Oy, 2002.

BECK et al. **Manifesto for Agile Software Development.** 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 15 ago. 2019.

CAROLI, Paulo. **Direto ao ponto:** Criando produtos de forma enxuta. São Paulo: Casa do Código, 2015.

CONFORTO, Edivandro C.; REBENTISCH, Eric; AMARAL, Daniel. **Project Management Agility Global Survey**. Massachusetts Institute of Technology, Consortium for Engineering Program Excellence – CEPE, Cambridge, Massachusetts, 2014.

HIGHSMITH, Jim. **What is agile software development?** CrossTalk magazine, Hill AFB, UTAH, v.15 n.10, p. 4-9, Out. 2002.

LIBORDI, P. L. O.; BARBOSA, V. **Métodos Ágeis**. 2010. 35f. Artigo (Especialização em TI) - Faculdade de Tecnologia – FT, Universidades Estadual de Campinas, 2010.

PRESSMAN, Roger S. **Engenharia de Software:** Uma abordagem profissional. 7 ed. Porto Alegre: Mcgraw-Hill, 2011.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software:** Uma abordagem profissional. 8 ed. São Paulo: Bookman, 2016.

RIBEIRO, Rafael Dias; RIBEIRO, Horácio da Cunha e S. **Métodos ágeis em gerenciamento de projetos.** Rio de Janeiro: Horácio da Cunha e Sousa Ribeiro, 2015.

SABBAGH, Rafael. **Scrum:** Gestão Ágil para Projetos de Sucesso. São Paulo: Casa do Código, 2013.

SATPATHY, Tridibesh. **A guide to the Scrum Body of Knowledge**. 3 ed. Avondale, AR, USA: VMEdu, 2017.

SOMMERVILLE, Ian. **Engenharia de Software**. 8 ed. São Paulo: Pearson, 2007.

SOMMERVILLE, Ian. **Engenharia de Software**. 9 ed. São Paulo: Pearson Prentice Hall, 2011.

SCHWABER, Ken; BEEDLE, Mike. **Agile Software Development with Scrum.** Upper Saddle River: Prentice Hall, 2002.

SCHWABER, Ken; SUTHERLAND, Jeff. **O guia do Scrum:** O guia definitivo para o Scrum: As regras do jogo. 2016. Disponível em: <http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Portuguese-European.pdf>. Acesso em 15 ago. 2019.

YIN, Robert K. **Estudo de caso:** planejamento e métodos. 3a ed. Porto Alegre: Bookman, 2005.