1 Introdução

Um dos desafios em projetos de desenvolvimento de *software* utilizando o modelo em cascata é o de gerenciar as mudanças de escopo durante o ciclo de vida do projeto (Sutherland, 2019). A justificativa é que clientes e *stakeholders* não conseguiam predizer todo o escopo nas etapas de levantamento de requisitos de *software* do projeto (Sommerville, 2007). Tal cenário pode estar relacionado as incertezas originadas pelas constantes mudanças nas organizações desse segmento. Em relação às incertezas, Sutherland (2019) destacou que o cone da incerteza é uma clara demonstração de que o escopo do projeto de desenvolvimento de *software* se conhece ao longo do projeto e não somente no início. Assim, as mudanças de estratégias de produto, mudanças legais e tributárias passaram a ocorrer com mais velocidade nas organizações (Van Casteren, 2017).

Em um ambiente dos negócios ambíguo e complexo, como o de Tecnologia da Informação (TI), o modelo em cascata deixou de fazer sentido em projetos de desenvolvimento de *software* devido às incertezas e constantes mudanças de requisitos que ocorrem nesses ambientes (Lichtenthaler, 2020; Ries, 2011). Nesse sentido que engenheiros de *software* se reuniram na década de 90 para discutir sobre a maneira como os projetos de *software* eram conduzidos, tornando-se um evento conhecido mundialmente por Manifesto Ágil para o Desenvolvimento de *Software* (*Agile Manifesto*, 2001). A partir desse encontro, muitos modelos de gestão de projetos ágeis foram criados, tais como o *Scrum, Extreme Programming* (XP), *Feature-Driven Development* (FDD) e o *Dynamic System Development Method* (DSDM) (Zafar, Nazir & Abbas, 2017; Nawaz, Aftab & Anwer, 2017).

Os novos modelos de desenvolvimento de ágil de *software* são baseados na entrega do melhor valor ao cliente sustentado no desenvolvimento de produtos de *software* certos e na hora certa ou como *just in time*. Para isso, faz-se necessário determinar hipóteses que devem ser validadas o mais rapidamente possível e com o menor esforço de desenvolvimento (Lichtenthaler, 2020; Ries, 2011). Além disso, obter o *feedback* de clientes e *stakeholders*, aprender com esse *feedback* e adaptar o escopo do projeto com o menor tempo possível, assim, esses desafios apoiam à definição de produto de *software* certo estando alinhados com o produto que os clientes querem e que realmente geram valor para eles (Lichtenthaler, 2020; Ries, 2011; Sutherland, 2019).

Para adotar os princípios de entrega de valor a partir do projeto de desenvolvimento ágil de *software*, torna-se necessário mudar a cultura da organização, rever processos que

onerem a produtividade e a adoção de métodos, técnicas e *frameworks* como o *Scrum* e o *Kanban* (Sutherland, 2019). Desse modo, o objetivo do uso desses métodos é em apoiar na transformação ágil em equipes e organizações que sejam ou não orientadas a projetos de desenvolvimento de *software* (Jyothi & Rao, 2017).

Nesse contexto em que as organizações orientadas a projetos de desenvolvimento de *software* iniciam seu processo de adoção inicial de modelos ágeis para apoio à gestão de projetos, o objetivo desse relato técnico foi de apresentar a adoção de *framework*, métodos e técnicas ágeis e de gerenciamento de projetos para uma equipe de desenvolvimento de *software* em uma organização multinacional.

2 Referencial Teórico

Esta seção apresenta brevemente o referencial teórico que balizou o desenvolvimento deste relato técnico.

2.1 *Framework*, Método e Elicitação de Requisitos

A seguir, serão apresentados os principais conceitos que serviram de base para a elaboração deste relato técnico em relação aos *frameworks* e ferramentas ágeis.

2.1.1 *Scrum*

O *Scrum* foi concebido a partir do Controle de Processos Empíricos que possui três pilares estabelecidos na transparência, inspeção e na adaptação. Assim, o empirismo e o desenvolvimento ágil de *software* são as bases de definem os propósitos do *Scrum framework* (Schwabe & Sutherland, 2020; Sutherland, 2019; Lichtenthaler, 2020; Penha et al., 2020).

Um grande equívoco que muitas pessoas cometem está em classificar o *Scrum* como uma metodologia de desenvolvimento. A justificativa desse equívoco é que o *Scrum* não prescreve em seu guia as fases de como construir um *software*, como realizar testes unitários, testes integrados e de aceitação de usuário. Desse modo, o *Scrum* não pode ser classificado como método, mas sim como um *framework* leve e simples que prescreve apenas os Eventos ou Cerimônias, Artefatos e Papéis para orientar o projeto no que precisa para dar foco na construção de um produto (Schwaber & Sutherland, 2020; Sutherland, 2019).

Os artefatos e eventos são definidos por Bosch e Bosch-Sijtsema (2011). Para os autores, os artefatos do *Scrum* são o *Backlog* do Produto (BP), *Backlog* da *Sprint* (BS) e o Incremento de *Software*. O BP é um artefato vivo contendo requisitos priorizados em constante refinamento para responder as mudanças legais, de tecnologia, de negócio e organizacionais. O BS contém as atividades de desenvolvimento de *software* como, por exemplo, análise, desenvolvimento e testes. O incremento de *software* é parte do *software* construído e liberado ao final da *Sprint* com qualidade funcional para ser utilizado em produção (Schwaber & Sutherland, 2020; Sutherland, 2019).

Já os Eventos são a *Sprint Planning, Daily Meeting, Sprint Review* e *Sprint Retrospective* onde cada evento tem a sua função importante dentro do *framework* que devem se repetir em cada ciclo de desenvolvimento ou iteração. Essa iteração é denominada por *Sprint* que não pode ser maior que um mês, assim como os eventos citados não podem exceder aos tempos pré-definidos conforme o tempo da *Sprint*. Esse tempo pré-definido é denominado por *time box* (Schwaber & Sutherland, 2020; Sutherland, 2019).

Os papéis são importantes para a *Sprint* seja efetiva, ou seja, cumpra a sua meta preestabelecida acordada na *Sprint Planning*. Os papéis são o *Scrum Master* (SM), *Product Owner* (PO) e *Developers* (Schwaber, Sutherland, 2020). O SM é o líder servo responsável por orientar os demais na execução correta do *Scrum*, na facilitação de decisões e eventos que os demais papéis tenham dificuldade. Por fim, o SM também deve ajudar na remoção dos impedimentos que atrapalham o time e na disseminação do *Scrum* em toda a organização (Beck et al., 2001).

O PO é o responsável pela gestão do escopo do produto determinando o que será desenvolvido primeiro através de Itens do BP (*Product Backlog Items* – PBI). Essa priorização dos PBIs pode ser definida pelo valor que eles podem trazer ao negócio e clientes, ponto fundamental para agregar mais valor ao trabalho dos desenvolvedores. O PO também é responsável pela Gestão do Produto, pela Gestão de Lançamentos ao Mercado e pelo *Minimum Viable Product* (MVP) (Beck *et al.*, 2001).

Os *Developers* são responsáveis pelo BS e por cumprir a Meta da *Sprint* determinada na *Sprint Planning*. Além disso, o cumprimento da Meta da *Sprint* requer que os *Developers* entreguem um incremento de *software*. O *Scrum Team* deve ser composto de dez ou menos membros para reduzir a complexidade do trabalho e para facilitar a comunicação. Os valores do *Scrum* são o Compromisso, Foco, Abertura, Respeito e Coragem (Beck *et al*., 2001).

O *Scrum* é considerado um *framework* leve e simples de compreender. A dificuldade está em adotar o *Scrum* nas organizações sem modificá-lo e sem que haja a mudança estrutural de processos, nos times departamentais e mudanças culturais. (Hajjdiab, Taleb & Ali, 2012).

2.1.2 *Kanban*

O termo *Kanban* é um termo japonês que significa "quadro visual" (*Kan* – quadro, *Ban* – visual). O quadro *Kanban* foi criado pela Toyota através de Taiichi Ohno em 1953. O quadro *Kanban* é uma ferramenta visual para acompanhamento do fluxo de produção da linha de montagem do chão de fábrica da Toyota (Gaete *et al*., 2021). É de fundamental importância diferenciar o quadro *Kanban* como uma ferramenta visual do método *Kanban* (Jyothi, Rao, 2017; Dos Santos et al., 2018).

O método *Kanban* foi criado por David J. Anderson vai além de um quadro visual, pois possui princípios e valores claros para ser adotado em qualquer linha de produção, apesar que atualmente, ser muito utilizado em desenvolvimento de *software* (Majchrzak & Stilger, 2017). O objetivo, tanto o quadro *Kanban* e o método *Kanban* é o de limitar o trabalho em progresso (*Work In Progress* – WIP) para evitar a multitarefa que afeta diretamente no desempenho das pessoas. Além disso, o WIP ajuda a promover o foco, a localizar problemas no processo, reduzir os desperdícios e os custos relacionados com mudanças durante o desenvolvimento de *software* (Jyothi, Rao, 2017; Dos Santos et al., 2018).

O método *Kanban* possui cinco práticas como: Visualizar o Fluxo de Trabalho, limitar o Trabalho em Progresso, gerenciar o Fluxo, Tornar a Política dos Processos Explícita e Melhorar Colaborativamente. Também possui quatro princípios como o "Comece com o que você faz agora", "Busque mudanças evolucionárias e incrementais", "Respeite a estrutura atual, seus papéis, responsabilidades e cargo" e "Incentive atos de liderança em todos os níveis" (Jyothi, Rao, 2017; Dos Santos *et al*., 2018).

O método *Kanban* utiliza as métricas tais como a do *Lead time, Cycle time, Throughput* e o *Flow Efficiency* (Jyothi, Rao, 2017; Dos Santos *et al*., 2018). O *Work Item Age* tem como objetivo de medir o tempo em dias que um PBI está em uma determinada fase do desenvolvimento. Por exemplo, se um PBI está há muitos dias na fase de Construção, fazemos a leitura que o PBI está envelhecendo nessa fase, o que sinaliza um alerta para a equipe no sentido de discutirem o motivo do PBI estar envelhecendo e a possível solução ou

colaboração para resolver o problema e destravar o PBI no fluxo de valor (Budacu, Pocatilu, 2018; Jyothi, Rao, 2017).

2.1.3 História de Usuário

A história do usuário (HU) é uma técnica muito utilizada que busca a simplicidade e eficiência na escrita de requisitos funcionais baseados na narrativa de um usuário e sobre os critérios de aceite utilizados como argumento para o aceite da HU pelo PO, *stakeholders* e clientes (Budacu, Pocatilu, 2018; Jyothi, Rao, 2017; Sutherland, 2019).

As HUs possuem uma estrutura semântica para identificar o usuário beneficiado pela funcionalidade de *software*, necessidades a serem atendidas e resultados esperados. A estrutura semântica é composta pela identificação do usuário (ator), necessidade do usuário e o resultado esperado (Budacu, Pocatilu, 2018; Jyothi, Rao, 2017).

Os Pontos de História (*story points*) definidos durantes a primeira parte da *Sprint Planning* para mensurar o tamanho ou o volume da complexidade da HU. Assim é possível medir a quantidade de pontos de histórias selecionadas na *Sprint Planning* e entregues no Incremento de *Software* (Budacu, Pocatilu, 2018; Jyothi, Rao, 2017; Sutherland, 2019).

Normalmente os pontos de HU são definidos pelos números da sequência de *Fibonacci*, que são eles 1, 2, 3, 5, 8, 13, 21 ou mensurados como complexidade utilizando o *T-Shirt Sizing* como o P (pequeno), M (médio) ou G (grande). Os números menores definem que o escopo é de baixa complexidade, os intermediários de média complexidade e os maiores de alta complexidade. Também são utilizados os números 40 e 100 que não fazem parte da sequência de *Fibonacci*, mas para sinalizar a dificuldade de um ou mais membros do time em determinar a complexidade da HU. Nesse caso, o SM pode facilitar uma nova discussão sobre o requisito entre os *Developers* e o PO ou o time poderá decidir pelo consenso ou até fazer uma média os números para chegar à um valor (Budacu, Pocatilu, 2018; Jyothi, Rao, 2017; Sutherland, 2019).

Apesar de ser uma boa proposta, essa métrica tem perdido o sentido, pois a quantidade de pontos de história pode variar de um time para outro. Além disso, o que mais importa na agilidade não é a quantidade de pontos de história ou histórias de usuário entregues ao final da *Sprint* e sim se estamos entregando o produto certo e na hora certa do ponto de vista do cliente (Budacu, Pocatilu, 2018; Jyothi, Rao, 2017; Sutherland, 2019).

2.1.4 Behavior Driven Development

O *Behavior Driven Development* (BDD) foi criado por Dan North em 2003 como prática de desenvolvimento de *software* orientado por comportamento (Alferez *et al*., 2019). A ideia é que o comportamento possa induzir os testes necessários para validar o comportamento do ponto de vista do negócio. O modelo auxilia na clareza das regras de negócio, pois incentiva à construção de cenários por meio da colaboração de múltiplos membros do time (De Souza *et al*., 2017). O BDD utiliza-se da sintaxe *Gherkin* que é baseada em linguagem natural com o formato estruturado como Dado Que – Quando – Então: Dado Que (Contexto inicial), Quando (Evento ou Ação) e Então (Resultados esperados) (Alferez *et al*., 2019)**.**

A HU deve ter uma especificação de critério de aceitação descreve um cenário de teste de aceitação que podem ser capturados através da estrutura do BDD. A sintaxe do Gherkin permite a geração automatizada de casos de teste executáveis com base na correspondência de texto de critérios de aceitação com *Application Program Interfaces* (APIs), levando assim à rastreabilidade do caso de teste e aumentando a cobertura do código-fonte testado (Alferez *et al.*, 2019). Além disso, o BDD pode ser combinado com o *Scrum* para auxiliar o PO na escrita de regras de negócio juntamente com a HU. Os processos do BDD envolvem a colaboração entre o PO e os *Developers* na criação e no refinamento dos PBIs, além de contribuir para a gestão mais eficiente do escopo do projeto (De Souza *et al.*, 2017).

3 Metodologia

O estudo foi realizado em uma empresa multinacional líder no setor de serviços no Brasil e América Latina. Apesar da organização possuir diversas empresas em seu conglomerado, a escolha do departamento de TI – *BackOffice* da sede corporativa foi intencional, visto que a organização está aberta para a adoção de boas práticas de gerenciamento de projetos e obteve-se também o apoio da gestão de TI para a aplicação da proposta do uso de fluxo de valor ágil. Os dados foram obtidos a partir de profissionais membros da equipe de projeto e de ferramentas de apoio ao gerenciamento de projetos da organização.

Este relato técnico foi criado conforme o protocolo de Biancolino *et al,* (2012) para elaboração de relatos de produção técnica. O objetivo desse relato técnico é apresentar a adoção de *framework*, métodos e técnicas ágeis e de gerenciamento de projetos para uma equipe de desenvolvimento de *software* em organização multinacional. Assim, o autor esteve em todas as reuniões que envolveram desde a aprovação do modelo de fluxo de valor ágil para a gerência executiva de TI quando nas reuniões com a equipe de projeto. Além disso, o autor atuou na configuração da ferramenta de gerenciamento de projetos, nos processos de *coaching* e acompanhamento do projeto e da equipe.

No que tange aos documentos produzidos como artefatos e para a gestão do projeto, eles foram mantidos nas ferramentas *on-line* de gerenciamento de projetos e de repositório de documentos eletrônicos da organização.

4 Resultados Obtidos e Análise

Esta seção apresenta a empresa, o projeto analisado e os resultados obtidos.

4.1 Caracterização da organização

Esse relato técnico foi realizado em uma empresa multinacional líder no setor de serviços no Brasil e América Latina. A empresa conta com outras 11 (onze) empresas que fazem parte do seu conglomerado e possui quase 50 anos de existência com 3,5 mil colaboradores no Brasil e exterior. Possui mais de 13 milhões de clientes atendidos por ano com mais de 1,2 mil lojas franqueadas e com um faturamento em torno de R$ 17,1 bilhões em 2019.

Com relação ao departamento de TI – *BackOffice*, a partir da estrutura do diretor de TI, o departamento conta com 30 funcionários que são responsáveis por sistemas nas áreas do contábil e financeiro. A equipe de projeto é responsável pela gestão de pontos e fidelização de um produto de cartão de crédito e composta por quatro profissionais envolvendo um PO, dois desenvolvedores e um coordenador de desenvolvimento de *software*. Essa equipe atende duas analistas de negócios especializadas em programa de pontos e que pertencem a área de Novos Negócios da organização.

4.2 Caracterização do problema analisado

O projeto teve início em novembro de 2020 e está em andamento até o presente momento. O projeto envolve o relacionamento entre o departamento de TI – *BackOffice*, uma instituição bancária responsável pela gestão do cartão de crédito, uma empresa como adquirente e outra empresa responsável pela solução de *software* para comunicação com o equipamento *Pinpad* utilizado nas lojas próprias e franqueadas. O *Pinpad* ou leitor de cartões, é um aparelho para empresas receberem seus pagamentos através cartão de crédito ou débito, ou até mesmo vale refeição ou alimentação.

O projeto envolve o desenvolvimento de *software* para contabilizar as vendas dos serviços oferecidos aos clientes através do uso do cartão de crédito e a gestão de pontos recebidos com o uso do cartão. Essa pontuação ainda permite que o cliente faça compras de outros serviços. Além disso, o projeto realiza o desenvolvimento de relatórios para a área de negócios e micro serviços para aplicações *web* para que clientes possam acompanhar os benefícios do cartão de crédito e extratos de acúmulo de pontos e de utilização deles.

O projeto não utilizava do modelo preditivo ou ágil de gerenciamento de projetos para condução do projeto. Assim, o atendimento ao escopo do projeto era feito através da execução de tarefas definidas pela área de negócios e organizadas em uma lista de tarefas com prazos pré-definidos.

5. Tipo de Intervenção e Mecanismos adotados

Em março de 2021, o autor deste relato técnico assumiu a gestão da equipe de projeto. Assim, observou-se que a equipe de projeto de desenvolvimento estava atendendo às demandas conforme eram solicitadas e priorizadas pela área de novos negócios, mas sem planejamentos mínimos, sem uso de algum de um fluxo de valor, metodologia ou *framework* e ferramentas de gerenciamento de projetos e engenharia de *software*. Além disso, observou-se que a equipe de projeto não utilizava de artefatos importantes e padronizados para o desenvolvimento de *software*. Para citar alguns, a equipe de projeto não utilizava de artefatos para documentar, de maneira padronizada, o escopo do projeto, os riscos e pendências do projeto, além do esforço e prazos estimados dos entregáveis, planejamento de testes de *software* e documentação de lições aprendidas durante o andamento do projeto.

5.1 Planejamento de Mudanças Estruturais do Projeto e Escolha de *Framework*

Com o projeto estava em andamento, o primeiro passo do planejamento foi de desenhar o fluxo de valor para que a equipe de projeto pudesse seguir como o processo de desenvolvimento de *software* para as entregas previstas no BP. O fluxo de valor que atendesse as necessidades do projeto e da organização deveria cobrir desde a concepção das ideias sobre o escopo até a entrega dos PBIs em forma de incremento de *software* funcional no ambiente produtivo da TI.

O mapeamento de fluxo de valor realizado para o projeto buscou identificar as etapas importantes para a ideação e elicitação de requisitos de *software* do projeto. Essa primeira parte do fluxo de valor trouxeram as preocupações com as etapas de análise de sistemas e de refinamento de requisitos funcionais viabilizando a participação da área de negócios, da gestão do projeto, do PO e dos *Developers*. Na segunda parte do fluxo de valor, buscou-se criar as etapas que envolvessem o desenvolvimento de *software* como o BS, desenvolvimento, testes integrados, homologação e gestão de mudanças para o ambiente produtivo.

Dadas as tendências do mercado e das pesquisas científicas publicadas em relação aos projetos de desenvolvimento ágil de *software*, o segundo passo do planejamento foi para a escolha do *framework* ágil e do método de fluxo para suportar as etapas do fluxo de valor e que viabilizasse os processos iterativos relacionados ao desenvolvimento do *software* durante o projeto.

5.2 Planejamento de Artefatos do Projeto e Ferramentas de Gestão

Os artefatos do projeto são documentos importantes para especificação de requisitos previstos no escopo do projeto, bem como para formalizar o entendimento do escopo e para comunicar para as partes interessadas sobre o escopo projetado para a atual iteração do projeto e visando a entrega de um incremento de *software* com qualidade funcional. Além disso, os artefatos apoiam e suportam a equipe de projeto para que façam a construção das funcionalidades que atenda o valor esperado pelos clientes e *stakeholders*. O planejamento e escolha dos artefatos do projeto foram feitos para atender a elicitação de requisitos, critérios de aceite funcionais e não funcionais, cenários de testes funcionais e não funcionais, definição de preparado para os requisitos e a definição de pronto (*Definition of Done* – DoD) para

alinhamento e transparência de quando é considerando o *software* entregue e com a qualidade funcional para o ambiente de produção da organização.

6. Resultados Obtidos e Análise

Os resultados foram sendo confirmados iniciando com a substituição de artefato de Tarefas por HUs, e com isso a equipe de projeto passou a padronizar a escrita de requisitos funcionais do projeto e definir os critérios de aceite de negócio para que a funcionalidade seja aceita pela área de negócios. Os critérios de aceite formam estruturados a partir da sintaxe do BDD com objetivo de fornecer outro nível de padronização de regras de negócio associadas para atender a narrativa da HU. Além disso, com a adoção das HUs foi possível medir o esforço realizado por *Sprint* através dos pontos de história.

O artefato de definição de preparado apoiou a equipe de projeto em determinar o que eles entendem como uma HU pronta para iniciar o desenvolvimento dentro de uma *Sprint*. Isto é, a equipe do projeto precisa ter o entendimento se a HU possui a narrativa que determina o ator, sua funcionalidade e os resultados esperados pelo ator. E, se os critérios de aceite estão claros em termos de contexto, eventos e resultados esperados pela área de negócios.

A partir das HUs, o artefato de cenários de teste foi criado para seguir com a estrutura dos BDDs e detalhando os testes para atender requisitos de qualidade do *software* produzido pela equipe de projeto. O artefato de definição de pronto tornou transparente para a equipe de projeto e partes interessadas sobre os critérios que torna o software funcional para ser entregue em produção para os usuários ou clientes.

Foi feita a adoção do fluxo de valor *Upstream Kanban* (UK) e *Downstream* Kanban (DK) que está representado pela Figura 1 e do *Scrum framework*. Os princípios do controle de processos empíricos ou empirismo foram obtidos nos resultados, ou seja, os princípios de transparência, adaptação e inspeção foram alcançados a partir do UK, DK e do *Scrum framework*. Como discutido anteriormente, o *Scrum* prescreve que suas Sprints ocorram em ciclos de iteração de até 30 dias, o que ajuda o time na inspeção e adaptação constante e a cada iteração. A adoção do *Kanban* no UK e no DK forneceu maior transparência dos trabalhos realizados pela equipe de projeto, além de dar visibilidade do fluxo de trabalho desde a ideação, levantamento de requisitos, desenvolvimento até o ambiente de produção.

Upstream Kanban					*Downstream Kanban*							
Backlog (15-∞)	Em Análise (5-8)	Pronto para Refinar (4-6)	Em Refinamento (3-5)	Pronto para Desenvolvimento (3-5)	Selecionados (4)	Em Desenvolvimento (2)	Pronto para Teste (2)	Em Teste (1)	Pronto para Homologação (2)	Em Homologação (1)	Pronto para Produção	Em Produção

Figura 1 – Fluxo de Valor *Upstream* e *Downstream Kanban* adotado na equipe de projeto.
Fonte: O autor (2021).

O quadro *Kanban* (Figura 1) também apresentou, em cada coluna do fluxo de valor, o trabalho em progresso (*work in progress* – WIP) que sinaliza o número máximo de HUs que podem ser puxadas através das colunas do fluxo de valor. Isso ajuda a evitar que os membros da equipe de projeto executem HUs paralelas, contribuindo negativamente no desempenho da equipe. Normalmente, no UK utiliza-se o WIP em intervalos de valores mínimos e máximos, enquanto no DK utiliza-se somente o número máximo no WIP.

Na primeira coluna do UK (Figura 1), temos o *Backlog* que pode conter o número infinito de ideias, suposições e hipóteses a serem validadas podendo virar *software* funcional ou ser descartado. As colunas Análise, Pronto para Refinar e Em Refinamento, são responsáveis por acomodar HU em fase de elicitação e detalhamento de requisitos. Esse ponto do *Kanban* é importante para unir a visão de negócios com a visão da equipe técnica. Quando a equipe de projeto move as HUs para a coluna Pronto para Desenvolvimento, entende-se que as HUs movidas atendem aos critérios da definição de preparado. O DK sinaliza as etapas de desenvolvimento de *software* a partir da coluna *Backlog* da *Sprint* que contém as HUs que foram selecionados para a próxima *Sprint*. A partir desse ponto que os eventos do *Scrum framework* são executados. Assim, diariamente a equipe se reuni para acompanhar as HUs em desenvolvimento, testes, homologação e subindo para a produção. Ao final do ciclo da *Sprint*, a equipe de projeto passou a refletir sobre o processo do UK e DK e das entregas realizadas e *feedback* recebido da área de negócios.

Para suportar toda essa reestruturação do *modus operandi* da equipe de projeto, foram utilizadas as ferramentas *Atlassian JIRA*, *Atlassian Confluence*, *Reetro.app* e *ScrumPocker*. O *Atlassian JIRA* foi utilizado para realizar a gestão do projeto com a implementação do fluxo de valor, do UK e DK, as HUs, pontos de história e critérios de aceite com BDD. No

Atlassian Confluence são mantidos os artefatos do *RoadMap*, Definição de Preparado e Pronto e os Cenários de Teste criados com planilha eletrônica do LibreOffice. O *Reetro.app* é a ferramenta que a equipe de projeto utiliza para realizar suas retrospectivas e para a elaboração do plano de ação para as próximas *Sprints*. Conforme a Figura 2, um recorte da ferramenta Reetro.app que utiliza o Kanban com as colunas Continuar, Começar a Fazer, Melhorar e Parar de Fazer. Em cada coluna, os membros da equipe do projeto incluem os assuntos referente a última *Sprint* para serem discutidos como lições aprendidas. Após as discussões, os itens que devem ser melhorados são movidos para o plano de ação na própria ferramenta.

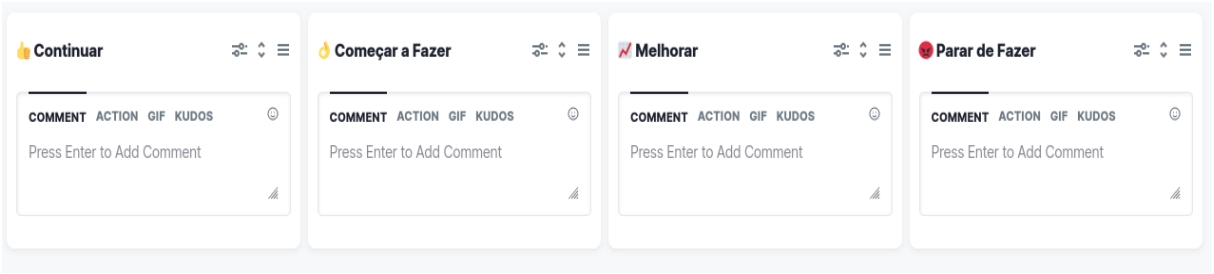

Figura 2 – Ferramenta *Reetro.app* e o *Kanban* para as reuniões de retrospectiva da equipe do projeto. Fonte: O autor (2021).

O *ScrumPocker* foi a ferramenta para as estimativas de esforço por meio dos pontos de história e possui uma interface simples utilizando a sequência de *Fibonacci*. A equipe se reúne para fazer a votação do esforço de cada HU escolhida pelo PO para a próxima *Sprint*, onde os *Developers* realizam as estimativas em pontos de história no formato das reuniões da técnica de *Planning Pocker*.

7. Conclusões/Considerações Finais

O objetivo desse relato técnico foi de apresentar a adoção de *framework*, métodos e técnicas ágeis e de gerenciamento de projetos para uma equipe de desenvolvimento de *software* em organização multinacional. Com base nos resultados apresentados, a reestruturação em apenas três meses do *modus operandi* da equipe de projeto trouxe o uso do fluxo de valor, do *Scrum framework*, do *Kanban*, de artefatos e ferramentas de apoio ao gerenciamento de projetos. Com essa reestruturação foi possível observar os ganhos na capacidade da equipe de projeto em relação à transparência, inspeção e adaptação, que também beneficiou a área de negócios com entregas mais alinhadas com as expectativas dos

clientes. Além disso, os benefícios da padronização ao se utilizar de técnicas, métodos, *frameworks* e ferramentas para apoio ao projeto de desenvolvimento ágil de *software* são fatores cruciais para a qualidade das entregas, para entrega do valor esperado por clientes e *stakeholders*, e para a motivação da equipe de projeto.

Visando a realização de estudos futuros, espera-se aplicar métricas de fluxo de valor para analisar os resultados da equipe de projeto em relação ao tempo de entrega em produção, tempo de ciclo de desenvolvimento, capacidade de entrega, esforço por HU, idade das HUs, entre outras métricas. Para métricas de qualidade, pode-se medir a quantidade de falhas de *software* identificadas em homologação, cobertura de código, entre outras. Espera-se aplicar também em trabalhos futuros a automação de testes funcionais com a ferramenta *Cucumber* em critérios de aceite com BDD.

Referências

Agile Manifesto (2001). Manifesto for Agile Software Development. Disponível em <http://agilemanifesto.org/:> Acesso em: 07 jun. 2021.

Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2017). Agile software development methodologies: survey of surveys. Journal of Computer and Communications, 5(05),74.

Alferez, M., Pastore, F., Sabetzadeh, M., Briand, L., & Riccardi, J. R. (2019, September). Bridging the gap between requirements modeling and behavior-driven development. In 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS) (pp. 239-249). IEEE.

Anwer, F., Aftab, S., Shah, S. M., & Waheed, U. (2017). Comparative analysis of two popular agile process models: Extreme Programming and Scrum. International Journal of Computer Science and Telecommunications, 8(2), 1-7.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Thomas, D. (2001). The agile manifesto.

Biancolino, C. A., Kniess, C. T., Maccari, E. A., & Rabechini Jr, R. (2012). Protocolo para elaboração de relatos de produção técnica. Revista de Gestão e Projetos, 3(2), 294-307.

Bosch-Sijtsema, P. M., Fruchter, R., Vartiainen, M., & Ruohomäki, V. (2011). A framework to analyze knowledge work in distributed teams. *Group & Organization Management*, *36*(3), 275-307.

Budacu, E. N., & Pocatilu, P. (2018). Real Time Agile Metrics for Measuring Team Performance. Informatica Economica, 22(4), 70-79.

De Souza, P. L., do Prado, A. F., de Souza, W. L., Pereira, S. M. D. S. F., & Pires, L. F. (2017). Combining Behaviour-Driven Development with Scrum for Software Development in the Education Domain. In ICEIS (2) (pp. 449-458).

Dos Santos, P. S. M., Beltrão, A. C., de Souza, B. P., & Travassos, G. H. (2018). On the benefits and challenges of using kanban in software engineering: a structured synthesis study. Journal of Software Engineering Research and Development, 6(1), 1-29.

Gaete, J., Villarroe, R., Figueroa, I., Cornide-Reyes, H., & Muñoz, R. (2021). Enfoque de aplicación ágil con Serum, Lean y Kanban. Ingeniare. Revista chilena de ingeniería, 29(1), 141-157.

Genari, J. O. S., & Ferrari, F. C. (2016). Times de alto desempenho no contexto das metodologias Scrum e Kanban. Revista TIS, 4(3).

Hron, M., & Obwegeser, N. (2018, January). Scrum in practice: an overview of Scrum adaptations. In Proceedings of the 51st Hawaii International Conference on System Sciences.

Jyothi, V. E., & Rao, K. N. (2017). Effective Implementation of Agile Software Development with a Framework, Metric Tool, and in Association with Cloud and Lean Kanban. International Journal of Advanced Engineering Research and Science, 4(3), 237085.

Lichtenthaler, U. (2020). Agile innovation: the complementarity of design thinking and lean startup. International Journal of Service Science, Management, Engineering, and Technology (IJSSMET), 11(1), 157-167.

Nawaz, Z., Aftab, S., & Anwer, F. (2017). Simplified FDD Process Model. International Journal of Modern Education & Computer Science, 9(9).

Nugroho, F. A., & Sugiarto, T. (2017). Extreme Programming Methodology Approach to Pamulang University. In Prosiding Seminar Nasional Informatika Vol. 2549, p. 4805.

Penha, R., da Silva, L. F., & Russo, R. D. F. S. M. (2020). Escalando as práticas ágeis. Revista de Gestão e Projetos, 11(2), 1-11.

Ries, E. (2011). The Lean Startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. New York: Crown Business.

Schwaber, K., & Sutherland, J. (2020). Scrum Guide. Disponível em <https://www.scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf> Acesso em: 07 jun. 2021.

Sommerville, I. (2007). Engenharia de Software, 9. Edição. Pearson, Addison Wesley, 8(9), 10, p. 44-45.

Sutherland, J. (2019). Scrum: A arte de fazer o dobro do trabalho na metade do tempo. Rio de Janeiro: Sextante, 2019.

Van Casteren, W. (2017). The Waterfall Model and the Agile Methodologies: A comparison by project characteristics. Research Gate, 1-6.

Zafar, I., Nazir, A. K., & Abbas, M. (2017). The impact of agile methodology (DSDM) on software project management. In Circulation in Computer Science: International Conference on Engineering, Computing and Information Technology (ICECIT 2017) (pp. 1-6).