UNIVERSIDADE DE CAXIAS DO SUL

ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

GÉSSICA GROLLI

IMPLEMENTAÇÃO DE METODOLOGIAS ÁGEIS PARA O GERENCIAMENTO DE PROJETOS

BENTO GONÇALVES
2020

GÉSSICA GROLLI

IMPLEMENTAÇÃO DE METODOLOGIAS ÁGEIS PARA O GERENCIAMENTO DE PROJETOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador Prof.: Dr. Daniel Luis Notari

BENTO GONÇALVES
2020

GÉSSICA GROLLI

IMPLEMENTAÇÃO DE METODOLOGIAS ÁGEIS PARA O GERENCIAMENTO DE PROJETOS

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Prof. Dr. Daniel Luis Notari

Aprovado(a) em 04/12/2020

Banca examinadora

Prof. Dr. Daniel Luis Notari
Universidade de Caxias do Sul - UCS

Prof. Esp. Daniel Antônio Faccin
Universidade de Caxias do Sul - UCS

Profa. Ma. Gabriele Dani Bonetti
Universidade de Caxias do Sul - UCS

Aos meus pais que sempre me apoiaram e incentivaram nesta jornada. Ao meu namorado por sua paciência, dedicação, apoio e por sempre acreditar nas minhas competências

RESUMO

A velocidade das mudanças no mundo dos negócios exige que a Tecnologia da Informação (TI) acompanhe esse ritmo, sendo sempre uma facilitadora e não deve ser um obstáculo para a empresa. Desta forma, é preciso evoluir as tecnologias e os processos da TI para que possam entregar melhores resultados à toda organização. Este trabalho tem o objetivo de realizar um estudo de caso através da análise do Gerenciamento de Projetos em uma empresa de grande porte do ramo metalúrgico. No mercado há vários modelos de Métodos Ágeis sendo implantados, porém, como é conhecido, nenhum processo é totalmente perfeito para uma empresa ou organização e todos processos necessitam de tratamento especial. A partir da análise da cultura organizacional da empresa utilizada neste estudo, foi sugerida a criação e utilização de um *framework* sob medida, baseado na abordagem ágil do *Design Thinking* e associado ao método ágil Scrum. O estudo buscou aprimorar o desempenho, organização e gerenciamento de projetos, e através desta visão, auxiliou no planejamento do aumento de demandas na empresa, apoiando o desenvolvimento e acompanhamento do projeto. A metodologia garantiu um bom nível de assertividade na entrega do produto, de motivação aos envolvidos e de satisfação do cliente.

Palavras-chave: Gerenciamento de Projeto. Métodos Ágeis. Scrum. *Design Thinking*.

ABSTRACT

The speed of changes in the business world requires that Information Technology (IT) keeps pace with this, always being a facilitator and should not be an obstacle for the company. Thus, it is necessary to evolve IT technologies and processes so that they can deliver better results to the entire organization. This work aims realize a case study through the analysis of Project Management in a large company in the metallurgical industry. In the market there are several models of Agile Methods being implemented, however, as it is known, no process is totally perfect for a company or organization and all processes need special treatment. From the analysis of the organizational culture of the company used in this study, it was suggested the creation and use of a tailored framework, based on the agile approach of Design Thinking and associated with the agile Scrum method. The study seek to improve the performance, organization and management of projects, and through this vision, helped in planning the increase of demands in the company, supporting the development and monitoring of the project. The methodology ensured a good level of assertiveness in the delivery of the product, motivation to the involved people and customer satisfaction.

Keywords: Project Management. Agile Methods. Scrum. Design Thinking

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AR	Análise de Requisitos	-
EDI	*Electronic Data Interchange*	Troca Eletrônica de Dados
ERP	Sistema de Gestão Empresarial	*Enterprise Resource Planning*
GT	Grupo de Trabalho	-
PMBOK	Guia para o conjunto de conhecimentos de gerenciamento de projetos	*Project Management Body of Knowledge*
PMI	Instituto de Gerenciamento de Projetos	*Project Management Institute*
PO	Dono do Produto	*Product Owner*
TI	Tecnologia da Informação	-

SUMÁRIO

1 INTRODUÇÃO

De acordo com Xavier (2016), quando estamos à frente de um mercado que nos oferece dezenas de ofertas para cada produto, o cliente é quem dita o sucesso das empresas. Isso tem levado as organizações a se reinventarem e a viverem em permanente mudança, seja lançando novos produtos, modificando linhas de produção ou realizando mudanças administrativas. Cada uma dessas mudanças é um projeto, que tem como objetivo gerar um produto ou serviço.

De acordo com Gallotti (2016), é raro encontrar um sistema em que os requisitos não mudam. A indústria de *software* tem adequado os seus processos para realizar entregas de produtos, com o menor tempo possível aos seus clientes, visando não perder a qualidade, economia e satisfação.

Segundo Pressman (2016), um processo ágil pode reduzir o custo das alterações, porque o *software* é entregue de forma incremental e as alterações podem ser mais bem controladas, além disso, incentiva uma boa estruturação e atitudes da equipe, tornando-a de fácil comunicação. Nesse contexto, destaca-se o Scrum, um método de desenvolvimento ágil de *software* concebido por Jeff Sutherland no início dos anos 1990, que possui princípios coerentes com o Manifesto Ágil[1]. Os princípios do Scrum são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as atividades de: requisitos, análise, projeto, evolução e entrega.

O Manifesto Ágil[1] estudou melhores formas de como desenvolver software, e considerando este estudo, devem-se adotar os seguintes valores:

a. Indivíduos e interações mais que processos e ferramentas
b. Software em funcionamento mais que documentação abrangente
c. Colaboração com o cliente mais que negociação de contratos
d. Responder a mudanças mais que seguir um plano

Pressman (2016) ainda expõe que, além do Scrum ser uma metodologia, ele também possui maior ênfase nas pessoas e na comunicação, e estes são dois aspectos muito importantes dentro de uma organização ou departamento.

1 O Manifesto Ágil encontra-se no site https://agilemanifesto.org/iso/ptbr/manifesto.html

Schwaber e Sutherland (2017) citam que o Scrum é um *framework* estrutural, que está sendo usado para gerenciar o trabalho em produtos complexos, dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos.

Além do Scrum, outra abordagem que visa a comunicação e promove maior ênfase nas pessoas de uma organização é o *Design Thinking*. Para Camargo e Ribas (2019), o termo *Design Thinking* trabalha muito com a empatia, que é a capacidade de se colocar no lugar da outra pessoa, com o objetivo de enxergar as dificuldades e necessidades pelos olhos do outro.

De acordo com Makioszek (2019), sempre que precisamos saber o quanto um projeto ou ideia é eficiente e importante para o cliente, ou organização, aplicam-se os conceitos do *Design Thinking*.

Segundo Camargo e Ribas (2019), o *Design Thinking* não é uma solução padrão. No entanto, pode ser aplicada a qualquer projeto, necessidade ou oportunidade com o objetivo de buscar a melhor solução. Os autores ainda defendem que o *Design Thinking* é ainda mais recomendado para projetos de definição e circunstâncias incertas.

1.1 PROBLEMA DA PESQUISA

O desenvolvimento desse projeto será realizado em uma empresa metalúrgica que possui um departamento de desenvolvimento de *software*. Este departamento dispõe de uma equipe de desenvolvimento formada por vinte e sete participantes que desempenham papéis variados para desenvolver o *software Enterprise Resource Planning (*ERP), ou seja, o Sistema Integrado de Gestão Empresarial, que é utilizado em praticamente toda a estrutura organizacional. Para preservar o nome da empresa, iremos identificá-la ao longo deste trabalho como a empresa ABC.

O departamento de Tecnologia da Informação (TI) da empresa ABC trabalha junto dos Grupos de Trabalhos (GT`s), que são grupos de pessoas com conhecimento do negócio, divididos por módulos do sistema. Estes GT`s são os responsáveis pelo direcionamento e especificação das funcionalidades do sistema, e

pela priorização de ajustes ou desenvolvimento das demandas. Atualmente, o setor de TI, recebe projetos durante os primeiros seis meses do ano, caso houver algum projeto que demanda certa urgência e que não foi planejado no início do ano, pode-se realizar a inclusão deste projeto como Extra Plano, sempre sob aprovação do gerente de TI.

Os projetos são recebidos com um documento padrão que possui uma breve Análise de Requisitos (AR), e após aprovados são incluídos no portfólio de projetos. O fato de deixar o levantamento de requisitos apenas na visão do usuário, muitas vezes geram explicações superficiais, até porque, esses profissionais possuem conhecimento de negócio e não de sistema.

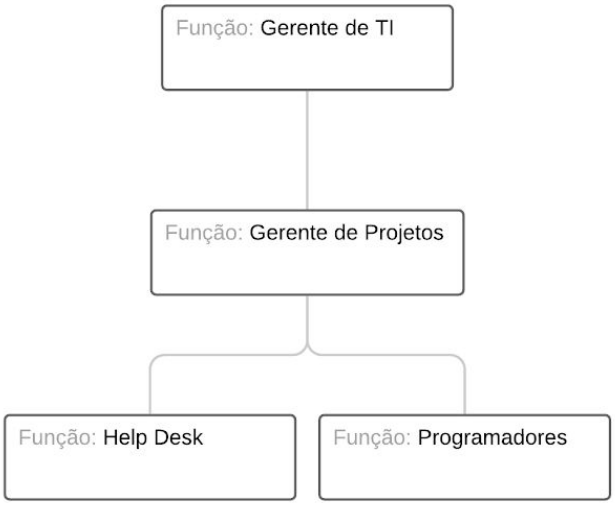
Neste contexto, alguns dos projetos entregues nem sempre atendem as dificuldades previstas pelos solicitantes. Quando esta situação ocorre, o projeto é abandonado logo após a sua entrega.

A programação e o andamento dos projetos do ano podem ser consultados em um sistema desenvolvido internamente, onde o analista do sistema sugere uma estimativa de planejamento através da informação de ano/mês. Contudo, dificilmente esta estimativa é condizente com o ano/mês de entrega.

Atualmente, o programador analisa o documento de análise de requisitos (AR) anexado ao projeto, realiza um breve levantamento de requisitos de sistema, e parte imediatamente para o desenvolvimento do código. Enquanto desempenha esse papel, o programador também realiza o atendimento diário de solicitações e/ou de dúvidas sobre regras e procedimentos.

A Figura 1 demonstra que o departamento de desenvolvimento de sistemas da empresa ABC possui uma estrutura enxuta, justificando a ligação de mais de uma responsabilidade em uma mesma função.

Figura 1 - Organograma atual do departamento de desenvolvimento



Fonte: Desenvolvido pelo autor (2020).

Atualmente o departamento não realiza o uso específico de uma metodologia de gerenciamento de projetos. Sendo assim, as atividades são realizadas conforme é conveniente para a equipe de desenvolvimento naquele momento. Neste contexto, as atividades e fases dos projetos não são realizadas de forma padronizada, e as fases existentes como definição e distribuição de tarefas são tratadas com base na experiência e disponibilidade dos programadores.

1.2 OBJETIVO GERAL

Este trabalho tem como objetivo realizar um estudo de caso no departamento de desenvolvimento de *software* da empresa ABC, utilizando as práticas *Design Thinking* e Scrum para sugerir uma metodologia ágil de gerenciamento de projetos.

1.3 ESTRUTURA DO TEXTO

Esta monografia está estruturada da seguinte forma: no Capítulo 2 apresenta-se uma conceituação referente ao gerenciamento de projetos e metodologias ágeis, abordando aspectos relevantes à avaliação das metodologias

propostas para este trabalho como o *Project Management Body of Knowledge (*PMBOK), Scrum e *Design Thinking, e* finalizará apresentando uma análise dos trabalhos relacionados acerca do tema. No Capítulo 3 será apresentado a metodologia científica para este estudo de caso, as ferramentas que auxiliam no controle e métrica do gerenciamento de projetos e a proposta de solução e de validação a ser aplicada. No Capítulo 4 serão apresentados os experimentos realizados neste estudo, bem como os detalhamentos a respeito da condução do experimento em cada uma das etapas do projeto, os resultados e as validações efetivadas. Por fim, no Capítulo 5, descreve-se a conclusão mostrando os resultados obtidos.

2 GESTÃO DE PROJETOS

Neste capítulo serão abordados aspectos relacionados ao gerenciamento de projetos, apresentando um referencial teórico dos conceitos das metodologias ágeis Scrum e Design Thinking. Na sequência, o capítulo também apresentará a análise dos trabalhos relacionados acerca do tema.

2.1 GERENCIAMENTO ÁGIL DE PROJETOS

O gerenciamento de projeto permite que as organizações executem projetos de forma eficaz e eficiente. De acordo com o PMBOK (2017), projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado único.

Xavier (2016), cita o relatório de estudo do *Project Management Institute* (PMI) de 2015, indicando que as empresas que possuem alto desempenho no gerenciamento de seus projetos alcançam os seus objetivos 2,5 vezes mais rápido do que seus pares com menor desempenho.

Kerzner (2015) destaca que um projeto pode ser considerado como sendo quaisquer séries de atividades e tarefas que:

a) possuem um objetivo específico a ser atingido dentro de determinadas especificações;
b) possuem datas de início e término definidas;
c) possuem limites de financiamento (se aplicável);
d) consomem recursos humanos e não humanos (ou seja, dinheiro, pessoas, equipamentos);
e) são multifuncionais (isto é, cruzam diversas linhas funcionais).

De acordo com o PMBOK (2017), os processos de gerenciamento de projetos são divididos em cinco grupos:

a) iniciação: definir um novo projeto ou fase;
b) planejamento: obtenção de autorização para iniciar o projeto;
c) execução: concluir o trabalho definido no plano de gerenciamento do projeto e satisfazer os requisitos;

d) monitoramento e controle: processos para acompanhar, analisar e controlar o progresso e desempenho do projeto;
e) encerramento: conclui formalmente um projeto.

Várias técnicas e ferramentas baseadas em métodos ágeis têm sido aplicadas ao gerenciamento de projetos, visando aproximar estes processos e atividades para a realidade das empresas. Segundo Foggetti (2014), às metodologias ágeis, na verdade, não trazem nada novo; elas têm a mesma estrutura básica das metodologias tradicionais, mas com outro enfoque e outros valores, conforme apresentado no Quadro 1.

Quadro 1 - Comparação entre metodologia ágil e tradicional

Metodologia ágil	Metodologia tradicional
Foco nas pessoas	Foco nos processos
Usa mais tempo mais implementação	Gasta mais tempo com documentação
Vai sendo adaptada no decorrer do projeto	Tenta prever tudo o que acontecerá no projeto
Aceita a mudança	Prevê o futuro
É usada quando os requisitos são mais dinâmicos	É usada para requisitos estáveis e previsíveis
É usada quando o cliente não sabe bem o que quer	É usada quando o cliente tem certeza do que quer
Entrega em partes	Entrega de uma só vez

Fonte: Foggetti (2014).

No Quadro 1 é possível verificar que a maior diferença entre as metodologias em questão está na possibilidade de planejamento antecipado que a metodologia ágil possibilita. A metodologia tradicional planeja com bastante antecedência, já na metodologia ágil o planejamento é feito de forma iterativa e incremental, descobrindo o percurso no caminho.

2.2 METODOLOGIAS ÁGEIS

Segundo Pressman (2016), a Engenharia de *Software* Ágil combina filosofia com princípios de desenvolvimento. A filosofia defende a satisfação do cliente e a entrega incremental antecipada; equipes de projeto pequenas e altamente motivadas; métodos informais; artefatos de engenharia de *software*; e principalmente, simplicidade no desenvolvimento geral. Já os princípios de

desenvolvimento priorizam a comunicação ativa e contínua entre desenvolvedores e clientes.

De acordo com Gallotti (2016), é muito comum encontrar sistemas em que os requisitos mudam constantemente. Essa mudança é praticamente uma característica geral dos sistemas, pois os envolvidos amadurecem e esclarecem a ideia do que desejam, as complicações são solucionadas, e assim as necessidades mudam. Por esse motivo, entende-se que as entregas fracionadas vem ao encontro desta dificuldade de manter-se os requisitos estáveis, para assim amenizar possíveis insatisfações do usuário final e também evitando retrabalho.

Segundo Pressman (2016), a agilidade pode ser aplicada a qualquer processo de *software*. Contudo, para isso ocorrer, é necessário projetar o processo de tal forma que a equipe possa adaptar e alinhar as tarefas, além de conduzir o seu planejamento.

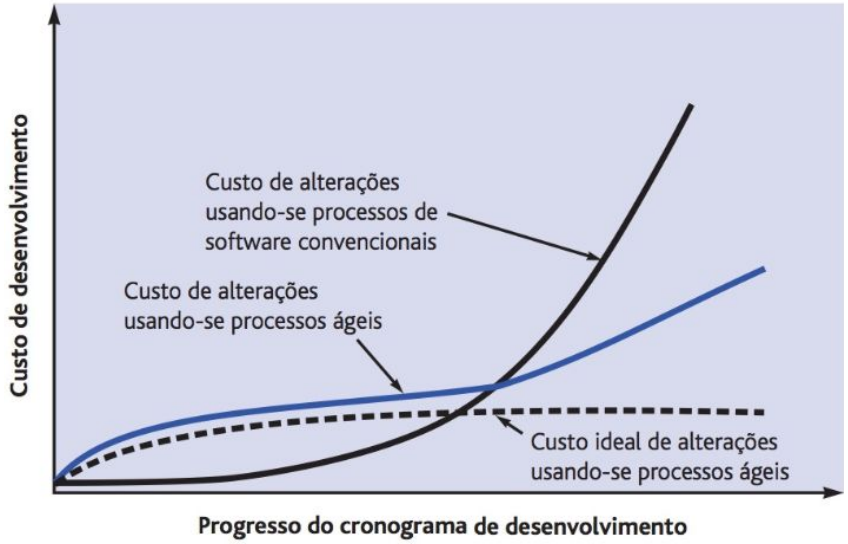
Condizente com este contexto, Vianna et al. (2018), justifica que ao utilizar-se da possibilidade de entregas fracionadas, tem-se um ganho de usabilidade por parte do usuário. O autor ainda destaca que somente após a entrega, quando o usuário está manuseando de fato o sistema, é que ele consegue identificar com maior clareza as reais necessidades e aprimoramentos. Por exemplo, quando o desenvolvedor entrega um produto pronto, depois de 6 meses ou 1 ano, possivelmente o nível de assertividade diminui, e também corre-se o risco de as necessidades e requisitos terem mudado. Quando isso ocorre, o programa já nasce com diversos ajustes.

> Uma das características mais convincentes da metodologia ágil é sua habilidade de reduzir os custos da mudança no processo de *software*, isso porque um processo ágil bem elaborado “achata” o custo da curva de mudança (Figura 2, curva em linha azul), permitindo que uma equipe de *software* assimile as alterações, realizadas posteriormente em um projeto de *software*, sem um impacto significativo nos custos ou no tempo (PRESSMAN, 2016, p. 68).

Através da Figura 2 percebe-se que como o processo ágil envolve entregas incrementais, mantendo o foco no ‘problema’ que o projeto visa solucionar, os

requisitos acabam ficando mais alinhados ao que o cliente deseja, fazendo com que o custo das mudanças diminua.

Figura 2 - Custo de alterações como uma função do tempo em desenvolvimento



Fonte: Pressman (2016).

De acordo com Pressman (2016), se a equipe ágil concorda que o processo funciona e essa equipe produz incrementos de *software* passíveis de entrega e que satisfaçam o cliente, então, o trabalho está correto.

2.3 Scrum

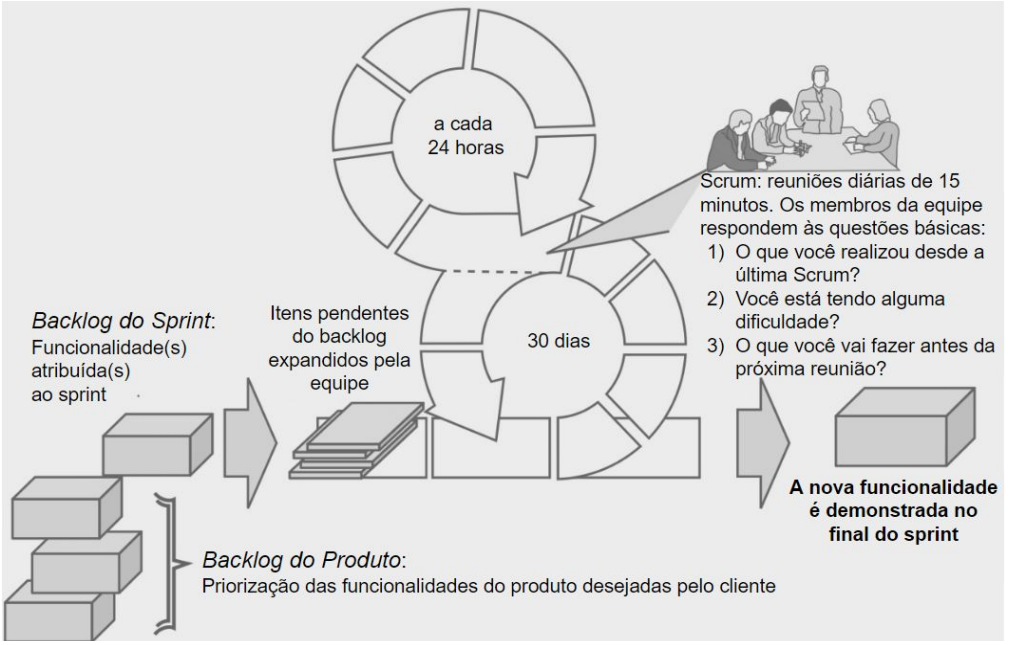
Segundo Schwaber e Sutherland (2017), o Scrum foi feito para ser uma maneira rápida, eficaz e confiável de criar *softwares* para o setor de tecnologia. Depois de ter ganho praticamente 100% do mercado de gerenciamento de projetos de *software*, o Scrum começa a ganhar a gerenciamento dos projetos tradicionais.

O Scrum é um método de gerenciamento de projetos ágil que ocorre de um jeito prático e favorável. Por este motivo, acaba sendo facilmente incorporado em qualquer projeto, principalmente projetos que tendem a ter mudanças repentinas de escopo. Pressman (2016), suporta este fundamento afirmando que o Scrum utiliza um conjunto de padrões de processos de *software* que provaram ser eficazes para

projetos com prazos de entrega apertados, com requisitos mutáveis e urgência do negócio.

Cohn (2011), aponta que parte do sucesso do Scrum justifica-se por produzir trabalho em ritmo constante, evitando o desleixo e aumentando a qualidade do *software*. O fato de não deixar erros e falhas para trás acaba elevando as hipóteses da equipe fazendo-a avançar com entusiasmo e de forma rápida e consistente. A Figura 3 demonstra como ocorre este ritmo de execução do trabalho. Estas etapas serão abordadas nas próximas seções.

Figura 3 - Fluxo do processo Scrum



Fonte: Pressman (2016).

2.3.1 Papéis do Scrum

Segundo Schwaber e Sutherland (2017), o Scrum é apresentado através de três papéis: O *Product Owner* (PO), o Time de Desenvolvimento e o Scrum Master.

O PO ou dono do produto, segundo Schwaber e Sutherland (2017), é responsável por gerenciar o *backlog* do produto, ele poderá até delegar algum trabalho para outra equipe do desenvolvimento, mas o PO continua sendo o responsável. O gerenciamento do *backlog* do produto inclui:

a) expressar com clareza os itens do *backlog* do produto;
b) ordenar os itens para alcançar melhor as metas e missões;
c) otimizar o valor do trabalho que o Time realiza;
d) garantir que o *backlog* do produto seja visível e claro a todos;
e) garantir que o Time entenda os itens do *backlog*.

O Time de Desenvolvimento, segundo Schwaber e Sutherland (2017) é composto pelos profissionais que realizam o trabalho de entregar o incremento. Esta equipe possui as seguintes características:

a) são auto-organizados, ninguém diz ao Time como transformar o *backlog* do Produto em incrementos;
b) são multifuncionais, possuem todas as habilidades necessárias para criar o incremento;
c) independentemente do trabalho realizado, não são reconhecidos títulos ou divisões de equipes no Time de Desenvolvimento;

O Scrum Master, de acordo com Schwaber e Sutherland (2017), é o responsável por ajudar a todos (PO, Time de Desenvolvimento e a Organização) a entenderem a teoria, as práticas, as regras e os valores do Scrum.

2.3.2 Eventos do Scrum

De acordo com Schwaber e Sutherland (2017), cada evento no Scrum é uma oportunidade de inspecionar e adaptar alguma circunstância. Abaixo relacionam-se os seguintes eventos: *Product backlog*, Sprints, Reuniões Scrum e Revisão da Sprint.

O *Product backlog*, segundo Foggetti (2014), é um documento definido pelo cliente no início do projeto, o qual pode ser modificado. Neste documento estão as funcionalidades esperadas e a ordem de prioridade.

Os Sprints, de acordo com Pressman (2016), são os períodos em que a equipe possui para a execução de um requisito estabelecido no registro de trabalho e que precisa ser ajustado dentro de um prazo já fechado. Alterações não são

introduzidas durante a execução da Sprint, possibilitando que a equipe trabalhe a curto prazo, porém com estabilidade.

Segundo Foggetti (2014), os Sprints servem para reduzir os riscos e revelar ao cliente como ocorre o passo a passo do desenvolvimento. No primeiro dia de cada Sprint é feita a reunião de planejamento do período, onde é desenvolvido o *Product backlog*.

> O Time de Desenvolvimento trabalha para prever as funcionalidades que serão desenvolvidas durante a Sprint. O *Product Owner* debate o objetivo que a Sprint deve realizar e os itens de *backlog* do produto que, se completados na Sprint, atingirão o objetivo da Sprint. Todo o Time Scrum colabora com o entendimento do trabalho da Sprint. (Schwaber e Sutherland, 2017).

De acordo com Pressman (2016), os itens de *backlog* do produto selecionados para a Sprint, com o plano de entrega destes itens, é chamado de *backlog* da Sprint. Segundo Schwaber e Sutherland (2017), quando os itens do *backlog* do produto já foram selecionados para a definição do objetivo da Sprint, então o Time de Desenvolvimento decide como produzir as funcionalidades da Sprint e transformá-las em incremento.

O Sprint *backlog* é redigido em forma de documento na reunião de planejamento, dividindo o *Product backlog* em pedaços para implementação. Os itens podem ser adicionados a esse registro a qualquer momento (é assim que as alterações são introduzidas). O gerente de produto avalia os registros e realiza ou atualiza a lista de prioridades.

De acordo com Schwaber e Sutherland (2017), a Sprint pode ser cancelada antes dela terminar, porém, apenas o PO possui essa autoridade. Caso uma Sprint for cancelada, qualquer item de *backlog* do produto completado e “Pronto” precisa ser revisado. Schwaber e Sutherland (2017), apontam ainda que o cancelamento de Sprints são muito incomuns, visto que consomem recursos e são frequentemente traumáticos para o Time Scrum.

As reuniões do Scrum são reuniões diárias e curtas, de acordo com Schwaber e Sutherland (2017), normalmente possuem duração de 15 minutos e são

realizadas pela equipe Scrum, onde são respondidas três perguntas-chave por todos os membros da equipe:

a) o que você realizou desde a última reunião de equipe;
b) quais obstáculos está encontrando;
c) o que planeja realizar até a próxima reunião da equipe.

Schwaber e Sutherland (2017) salientam que o Scrum Master assegura que o Time de Desenvolvimento realize a reunião diária, mas o responsável por conduzir esta reunião é o Time. Schwaber e Sutherland (2017), afirmam que as reuniões diárias aprimoram as comunicações, eliminam reuniões desnecessárias, apontam e removem impedimentos que haveriam no desenvolvimento, proporciona uma rápida resposta às decisões, aperfeiçoando o amadurecimento do Time de Desenvolvimento.

A revisão da Sprint, segundo Schwaber e Sutherland (2017), é realizada no final da Sprint para inspecionar o incremento e, se necessário, adaptar o *backlog* do produto. Durante a revisão da Sprint o time e as partes interessadas discutem e colaboram sobre o que foi feito e o que não foi desenvolvido na Sprint, discutem o que foi bem e os problemas solucionados, além de revisar a linha de tempo, orçamento e prazos. Nesta revisão o grupo todo colabora sobre o que fazer a seguir e assim, planejam a Sprint subsequente.

2.4 *DESIGN THINKING*

De acordo com a Fundação Instituto de Administração[2], a palavra da língua inglesa significa projetar, planejar, desenhar ou esboçar um projeto.

> A missão do *Design Thinking* é traduzir observações em *insights*, e estes em produtos e serviços para melhorar a vida das pessoas. (Brown 2010. p. 46).

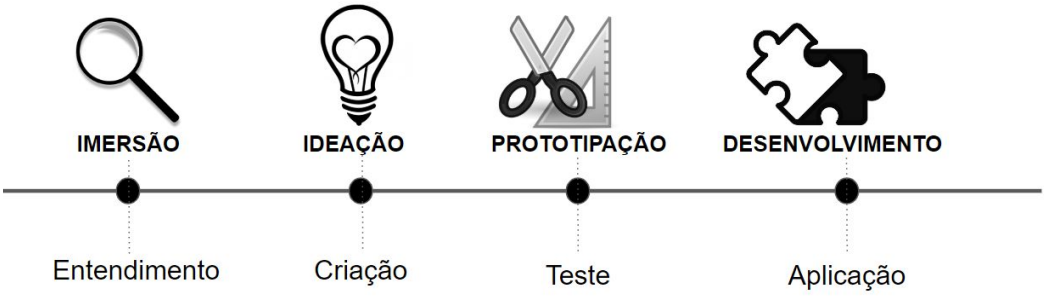
[2] Disponível em: <https://fia.com.br/blog/design-thinking/> Acesso em 18 de abril de 2020.

Ademais, Makioszek (2019) cita que o *Design Thinking* trata-se de uma perspectiva para encontrar os caminhos corretos na trajetória de resultados organizacionais.

Segundo Brown (2010), a metodologia *Design Thinking* atenta para criar soluções visando suprir às necessidades dos usuários, e isso, de certa forma, assemelha-se à algumas etapas do ciclo de vida de um *software*.

Vianna et al. (2018), divide o passo a passo do *Design Thinking* da seguinte forma: imersão, ideação e prototipação. A Figura 4 demonstra como este passo a passo pode ser aplicado a um processo.

Figura 4 - *Design Thinking* como processo



Fonte: Adaptado de Makioszek (2019).

De acordo com Vianna et al. (2018) e Makioszek (2019), a etapa de Imersão pode ter duas perspectivas: preliminar e profundidade.

Para Vianna et al. (2018), na fase preliminar geralmente a equipe não conhece o tema. Portanto, realiza-se uma Imersão Preliminar para aproximação do problema. Makioszek (2019), sugere realizar entrevistas com foco no cliente, de modo a conhecer as partes interessadas.

Em relação à perspectiva de profundidade, Vianna et al. (2018) descreve que se trata de um mergulho a fundo no contexto, com foco no ser humano. Neste momento, levantam-se informações de quatro questões:

a) o que as pessoas falam;
b) como agem;
c) o que pensam;

d) como se sentem;

Segundo Vianna et al. (2018), a etapa de ideação geralmente inicia com a equipe de projeto realizando reuniões do tipo *brainstorming* ao redor do tema a ser explorado.

De acordo com Makioszek (2019), a etapa de Prototipação é onde uma versão simples do produto pode ser lançada para testes e para verificar se a proposta atinge as necessidades. Para isso, utiliza-se protótipo em papel, modelo de volume, encenação, *storyboard* ou protótipo de serviços.

Makioszek (2019), utiliza-se do passo a passo do *Design Thinking*, incluindo ao final a etapa de desenvolvimento. Nessa etapa, Makioszek (2019), realiza a conclusão do processo *Design Thinking*, entrando no curso de desenvolvimento do produto, quando implementam-se as sugestões ou soluções verificadas no processo de *Design Thinking* de maneira incremental e contínua, com coparticipação permanente de todas as partes interessadas (*stakeholders*) do negócio.

2.5 MONITORAMENTO E DESENVOLVIMENTO DE PROJETOS ÁGEIS

Segundo Massari (2018), através de técnicas de monitoramento e controle é possível identificar desvios e tomar as ações necessárias para o sucesso do projeto. O autor indica também que uma das formas de controlar estes desvios é através dos gráficos *burndown* e *burnup*.

Do mesmo modo, Foggetti (2014) salienta a importância do monitoramento e gestão visual do projeto, quando afirma que o Kanban tem sido uma ferramenta adotada junto das metodologias ágeis.

2.5.1 *Planning Poker*

De acordo com Foggetti (2014), o *Planning Poker* é uma das técnicas mais comuns entre os desenvolvedores de projetos ágeis. A técnica é iniciada a partir de uma reunião com todos os membros da equipe do projeto, onde o responsável apresenta a funcionalidade, e cada um estima o esforço para desenvolvê-la. Todos

revelam suas estimativas ao mesmo tempo, o grupo discute os resultados apresentados e são feitas outras rodadas até que se chegue a um consenso do tempo.

Segundo Massari (2018), o *Planning Poker* evita coerção, pois todos exibem suas estimativas ao mesmo tempo, e evita estimativas superestimadas, pois o resultado é consenso do grupo, apresentando assim importantes vantagens desta técnica.

De acordo com Foggetti (2014), para ter maior assertividade pode-se utilizar a estratégia de valorizar a dificuldade ou esforço de uma funcionalidade. Para isso, pode-se iniciar as estimativas de tempo pela tarefa mais simples e dar-lhe o valor de "1 ponto". A partir daí, analisam-se as demais funcionalidades e estima-se o esforço em comparação a esta.

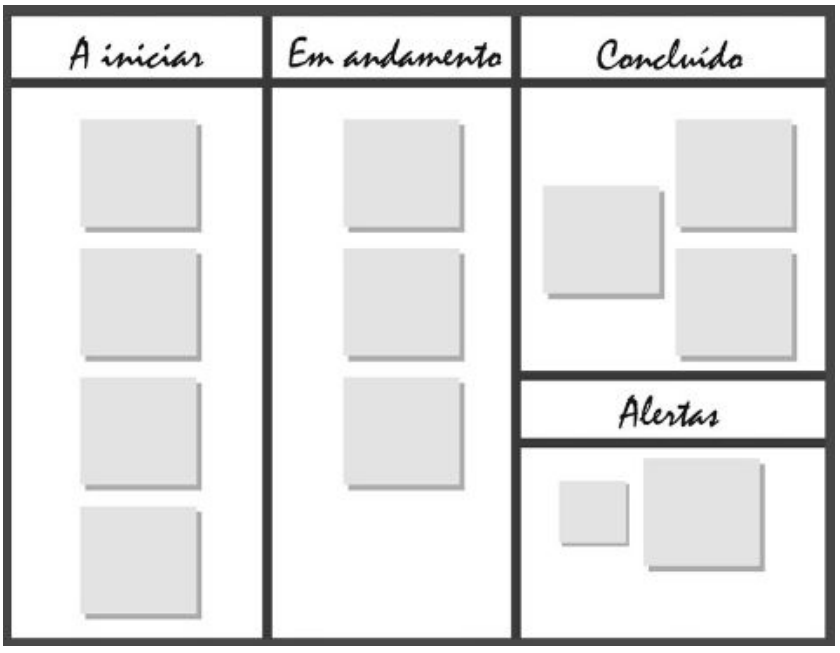
2.5.2 Kanban

Segundo Foggetti (2014), a palavra Kanban significa cartão ou "registro visível" em japonês. O autor apresenta o Kanban como uma técnica visual que mostra o fluxo a ser executado, com o objetivo de tornar visível quais são os problemas e os gargalos em suas atividades.

> As ferramentas Kanban começaram a ser adotadas com o surgimento dos métodos ágeis. São utilizados quadros visuais, nos quais os desenvolvedores marcam com cartões físicos as funcionalidades ainda não iniciadas, as que estão em desenvolvimento e as prontas, por exemplo. (Foggetti, 2014. p. 80)

Conforme Massari (2018), o Kanban é considerado como um sistema *pull* (puxar), pois conforme mostra a Figura 5, as tarefas são puxadas do primeiro quadro à esquerda até o estágio final mais à direita.

Figura 5 - Kanban



Fonte: Massari (2018).

2.5.3 Gráficos *burnup* e *burndown*

Para Massari (2018), os gráficos *burndown* e *burnup* demonstram o progresso, auxiliando na identificação do andamento e tendências do projeto.

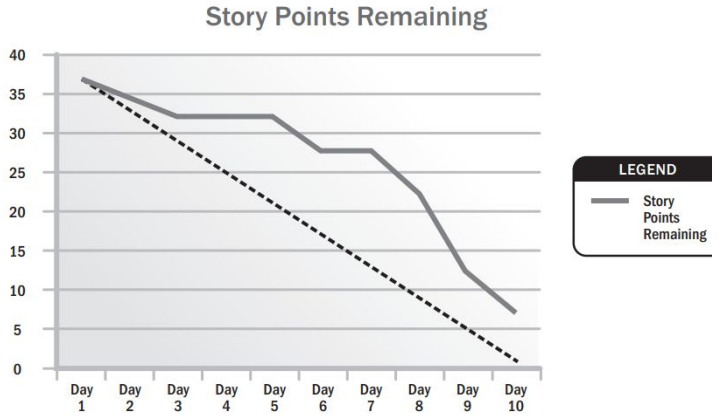
Conforme PMI (2017), este gráfico é uma ferramenta utilizada nos projetos ágeis, onde as equipes conseguem, ao final da iteração, analisar o desempenho de entregas concretas e indicar ações de ajuste no que julgarem necessário.

Os gráficos *burndown* exibem o esforço remanescente na linha do tempo para finalizar o projeto, a *release* ou a iteração (MASSARI, 2018, p.114). O autor ainda descreve que este gráfico possibilita acompanhar se o projeto está dentro ou fora da linha de base prevista, representando o progresso do trabalho em desenvolvimento em comparação com o projeto total.

Na Figura 6, verifica-se um exemplo que há 37 pontos de história no dia 1, já no dia 3 percebe-se que a linha de trabalho restante distanciou-se da linha de

trabalho programado, apresentando o risco de entrega atrasada, podendo não ser finalizado até o dia 10.

Figura 6 - Representação do gráfico *burndown*

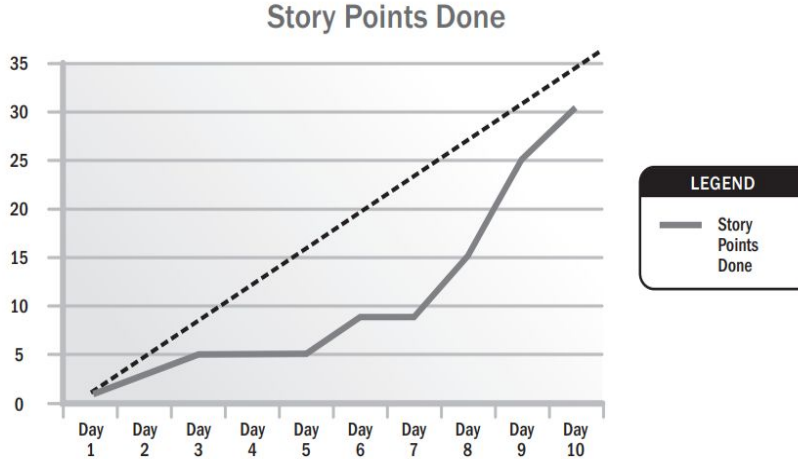


Fonte: PMI (2017).

Para PMI (2017), algumas equipes preferem utilizar o gráfico *burnup* para realizar este acompanhamento. Segundo Massari (2018) e PMI (2017), o gráfico *burnup* exibe o esforço do projeto, *release* ou iteração do que já foi concluído.

Na Figura 7, tem-se os mesmos dados do gráfico *burndown* anterior, porém percebe-se que a análise das linhas é realizada de forma contrária, indicando o trabalho concluído ao invés do faltante. Verifica-se que no dia 1 o trabalho é iniciado e que já no dia 2 percebe-se que a equipe não pode concluir o trabalho que havia sido projetado para ser entregue, ficando abaixo do nível desejável de conclusão.

Figura 7 - Representação do gráfico *burnup*



Fonte: PMI (2017).

2.6 TRABALHOS RELACIONADOS

Diversos trabalhos já foram publicados os quais possuem relação com esta pesquisa. O cuidado das empresas em como controlar os seus projetos acabam sendo uma maneira de garantir prazos, retorno do investimento e gerenciamento de custos.

Segundo Cohn (2011), a empresa Salesforce.com percebeu as vantagens ao adotar o Scrum. A Saleforce.com foi fundada em 1999 e é um caso de sucesso duradouro da era das ponto.com. Em 2006 a empresa com 2 mil funcionários apresentava uma receita de mais de 450 milhões de dólares, neste momento, a empresa percebeu que a frequência do lançamento de suas versões tinha diminuído de quatro por ano para uma por ano. Os clientes estavam recebendo menos, mas esperando mais. Para resolver esta situação a empresa precisava fazer alguma coisa, e então decidiu realizar a transição para o Scrum. No primeiro ano com Scrum, a Salesforce.com teve um aumento de 94% nas versões de seu sistema, distribuiu 38% mais recursos por desenvolvedor e 500% a mais de valor a seus clientes. Dois anos depois, a receita mais do que duplicou, ultrapassando 1 bilhão de dólares.

Andrade et al. (2011), abordaram um estudo de caso em uma empresa do ramo de desenvolvimento de *software* utilizando Scrum, e observaram que o projeto foi entregue dentro do prazo e do orçamento melhor do que os previstos. Os autores constataram uma maior participação e satisfação do cliente, principalmente pelo tempo fixo estimado para as Sprints. Andrade et al. (2011), também destacaram que a equipe pode evoluir profissionalmente, tornando-se mais auto-confiante e com autogerenciamento, e que este crescimento foi gradativo, no decorrer das Sprints.

De acordo com Andrade et al. (2011), a utilização do Scrum proporcionou menor retrabalho e aumento no comprometimento e foco com o projeto. Também verificou-se uma facilidade em extrair informações gerenciais do projeto através dos quadros do Scrum, e com isso foi possível identificar que a equipe teve uma boa produtividade durante a execução do projeto. Por fim, os autores concluíram que o

uso do Scrum levou a maior satisfação de todos os integrantes, desde equipe até o cliente.

Moraes et al. (2019), no que lhe concernem, realizaram um estudo de caso em uma filial de uma grande empresa brasileira de *software*, a qual, em 2017, passou por uma mudança de processos com a implantação do *framework* Scrum. O objetivo dos autores foi implementar as metodologias ágeis presentes na empresa com a adoção do *Design Thinking*. Para este objetivo, notou-se a necessidade de considerar a opinião dos funcionários visando a aproximação com a realidade no dia a dia da empresa, auxiliando assim na compreensão das inquietações das pessoas.

Moraes et al. (2019), observaram a necessidade de uma combinação de planos, pessoas e processos para que pudesse ter uma evolução. Os benefícios percebidos com a implantação do *Design Thinking* foram: conhecimento compartilhado, tempo na resolução de problemas, poder fazer mais com menos, manutenção junto de inovação e a forma de trabalho (sem retrabalho). Como desafios, Moraes et al. (2019), citaram a cultura, papéis, dimensionamento, maturidade, planejamento, estimativa, comunicação e gerenciamento. De acordo com os autores, a utilização do *Design Thinking* dividiu o trabalho em diferentes etapas. Foi realizado com o intuito de ouvir a voz dos colaboradores e entendê-los diante da cultura organizacional presente na empresa, e este foi um dos fatores determinantes para o sucesso da pesquisa.

3 PROPOSTA DE SOLUÇÃO

Neste capítulo serão abordados os procedimentos metodológicos utilizados neste estudo de caso, bem como a explanação do estudo de ferramentas com a recomendação de ferramenta a ser empregada. Posteriormente, será especificada a proposta de solução e validação.

3.1 MÉTODO CIENTÍFICO

De acordo com Ferrari (1974), o método científico é um traço característico da ciência, um mecanismo básico que organiza o pensamento em sistemas e traça os procedimentos para atingir o objetivo. Já Lakatos e Marconi (2017) defendem que é possível utilizá-la também para a resolução de problemas do cotidiano.

Para a elaboração deste trabalho será utilizado o método científico dialético, que de acordo com Gil (2008), busca interpretar a realidade partindo do pressuposto de que todos os fenômenos apresentam características contraditórias organicamente unidas e indissolúveis. Lakatos e Marconi (2017), concordam com este pensamento ao afirmar que sempre existe uma forma de se transformar, tornando o fim de um processo o início de outro.

Por se tratar de um estudo de caso com experimento, o meio de investigação adotado será através da abordagem experimental, que segundo Gil (2008), consiste em definir e observar variáveis para então determinar formas de controle.

O método específico utilizado será o método comparativo, que se dará entre dois projetos da empresa ABC, e que conforme Gil (2008) procede pela investigação de indivíduos, classes, fenômenos ou fatos, com vistas a ressaltar as diferenças e as similaridades entre eles.

Para a elaboração da pesquisa será utilizado a pesquisa de campo, que conforme Lakatos e Marconi (2017), é utilizada para obter informações sobre um problema ou hipótese, que se queira solucionar ou comprovar. O autor ainda comenta que esta pesquisa consiste na observação de fatos e fenômenos

espontâneos, na coleta de dados e no registro de variáveis relevantes para a análise que se procura realizar.

Para validar a qualidade do produto criado, o trabalho da equipe e a satisfação do usuário, a pesquisa de campo utilizada será do tipo quantitativo-descritiva. Para Lakatos e Marconi (2017), esta pesquisa empregam artifícios quantitativos e tem a finalidade de delinear as características de fatos e a avaliação de programas, com a finalidade de fornecer dados para a verificação de hipóteses.

3.2 FERRAMENTAS

Existem ferramentas que possuem funcionalidades que auxiliam o gerenciamento de projetos de forma prática e ágil. No presente trabalho serão analisadas as seguintes ferramentas: Trello, *ActiveCollab,* Jira Software e Service Desk.

A ferramenta Trello é um aplicativo de gerenciamento de projetos que opera um modelo gratuito com opção de assinatura para acesso a recursos avançados. O Trello utiliza o Kanban, representando os projetos por quadros (*boards*) que contém conjunto de listas de tarefas (cartões) que podem ser designados a usuários de um time e movidos de um *board* para outro[3].

A ferramenta ActiveCollab é uma ferramenta que permite organizar os projetos em tarefas e subtarefas, e com hierarquia entre elas. Esta ferramenta oferece a possibilidade de elencar prioridades em seus cartões e, além disso, fornece alguns controles como cronômetro, quadros e relatórios de horário, para visualizar o tempo que a equipe está gastando nas tarefas. O *ActiveCollab* também possibilita analisar a rentabilidade do projeto, atribuindo taxas por hora a todos os membros da equipe[4].

A ferramenta Jira Software permite organizar os projetos em tarefas e subtarefas, e com dependência entre elas. Esta ferramenta oferece a possibilidade

[3] Disponível em: <https://blog.trello.com/br/metodo-Kanban> Acesso em: 03 de maio 2020.
[4] Disponível em: <https://activecollab.com/> Acesso em: 03 de maio 2020.

de criar versões diferentes por projetos, elencar prioridades e vários controles como tempo estimado, realizado, categorias e *branches*. Possui quadro Scrum já desenvolvido pensando-se na divisão em Sprints, além de variados relatórios e gráficos de tempos do projeto, carga de trabalho e tarefas, *burndown* e *burnup*.

O Jira Software também possibilita a criação de um *backlog* separado da Sprint, possibilitando a organização dos próximos projetos ou tarefas na mesma ferramenta e também detém a possibilidade de integrar o gerenciamento de projeto de *software* junto aos repositórios com os *commits* dos códigos. Outro diferencial desta ferramenta é a possibilidade de configurar um *chatbot* e algumas regras de funcionamento pelo próprio usuário

A ferramenta Service Desk é uma ferramenta desenvolvida internamente pelo departamento da empresa ABC, que visa gerenciar as tarefas (solicitações) e a lista de projetos solicitados ao setor. Com esta ferramenta, é possível que os usuários nos encaminham solicitações de incidentes e dúvidas e também cadastrar tarefas de melhorias ou tarefas vinculadas a um projeto previamente cadastrado e atribuí-las aos analistas ou desenvolvedores. No Service Desk é possível priorizar as tarefas através de três prioridades: normal, alta e crítica, também é possível identificar o *status* do projeto, a quantidade de horas investidas, e a sua porcentagem de execução.

Quadro 2 - Comparação entre ferramentas de acompanhamento de projetos

	Trello	ActiveCollab	Jira Software	Ferramenta própria
Controle de Status	✓	✓	✓	✓
Dashboard	✗	✗	✓	✓
Alertas por e-mail	✓	✓	✓	✗
Armazenamento de arquivos	✓	✓	✓	✓
Kanban	✓	✓	✓	✗
Facilidade	✓	✗	✗	✓
Software Gratuito	✗	✗	✗	✓
Criação de lista de subtarefas	✓	✓	✓	✗
Identificação de prioridade	✓	✓	✓	✓
Lançamento de horas trabalhadas	✗	✓	✓	✓
Chat online para o grupo do projeto	✗	✓	✓	✗
Velocidade rápida	✓	✓	✓	✗
Possibilidade de criar etiquetas ou categorias	✓	✓	✓	✗
Gráfico de Gantt	✗	✓	✗	✗
Gráfico de Burndown	✗	✗	✓	✗

Fonte: Desenvolvido pelo autor (2020).

O Quadro 2 apresenta uma análise das características das ferramentas apresentadas. As características analisadas foram:

a) controle de *status*, está presente em todas as ferramentas e é uma característica indispensável na indicação das diferentes fases do projeto;
b) *dashboard*, não está presente nas ferramentas Trello e ActiveCollab, mas é um diferencial importante para demonstrar em uma tela, visualizações rápidas dos principais indicadores de desempenho relevante aos projetos;
c) alerta por correio eletrônico, esta característica não existe na ferramenta própria, mas pode vir a ser algo vantajoso para notificar os interessados sobre o andamento e vencimentos dos prazos das tarefas;
d) armazenamento de arquivos, todas as ferramentas possuem, é uma característica bem importante, pois um projeto poderá ter diferentes documentos ou imagens;
e) Kanban, não está presente na ferramenta própria, porém é um aspecto bem importante para o controle do fluxo de execução das tarefas;
f) facilidade, é um importante diferencial para a fácil operação na manutenção das informações levando a uma interface intuitiva, em comparação às demais ferramentas percebeu-se maior necessidade de adaptação na ferramenta ActiveCollab *e* Jira Software;
g) *software* gratuito, é um atributo pertencente apenas na ferramenta própria, mas de certa forma é relevante para indicar os custos fixos que a empresa poderá vir a ter com novas ferramentas;
h) criação de lista de subtarefas, atributo não identificado na ferramenta própria, mas que poderia auxiliar na identificação da hierarquia de tarefas e dependências;
i) identificação de prioridade, é um atributo presente em todas as ferramentas analisadas e é necessário para identificar as tarefas ou projetos mais importantes para a organização, ficando claro o que deverá ser executado por primeiro;

j) lançamento de horas trabalhadas, é uma característica não identificada na ferramenta Trello, porém é indispensável para dimensionar o custo do projeto;
k) *chat* online para o grupo do projeto, presente nas ferramentas *ActiveCollab e* Jira Software, porém não se demonstrou ser um ponto tão importante visto a empresa já viabiliza uma fácil comunicação entre as equipes;
l) velocidade rápida, houve menor eficiência desta característica na ferramenta própria, poder ser julgado como um ponto importante, pois garante que os processos sejam realizados com maior agilidade e eficiência;
m) possibilidade de criar etiquetas ou categorias, embora a ferramenta própria não disponibiliza desta possibilidade, entende-se como um ponto importante na subdivisão do planejamento do projeto em diferentes formatos, podendo ser readequado de acordo com a demanda;
n) gráfico de Gantt, está presente apenas na ferramenta *ActiveCollab*, não é uma necessidade essencial, mas tende a ser um diferencial ao auxiliar na estimativa de prazo das demandas de uma forma visual e de fácil interpretação;
o) gráfico de *burndown*, apenas a ferramenta Jira Software possui. Entende-se ser importante para demonstrar a situação entre a estimativa proposta e o trabalho realizado. Através deste gráfico, é possível identificar o nível de esforço necessário para garantir o prazo estipulado e identificar os pontos em que houve falha de estimativa para posterior análise e melhoria.

3.3 ESTUDO DE CASO

Com base no estudo da literatura, anteriormente descrito, foi elaborada uma proposta de metodologia de gerenciamento de projetos de TI, que visa auxiliar

gestores e/ou programadores, no desenvolvimento de projetos rentáveis e de sucesso.

Entende-se que, organizando a maneira como os projetos e os trabalhos são gerenciados e, principalmente realizando a divisão das etapas, poderemos entender melhor a complexidade dos projetos solicitados e estimar custos e prazos. Também entende-se que irá aumentar a objetividade, motivação e concentração dos envolvidos no projeto, diminuindo assim possibilidade de erros e aumentando a produtividade, qualidade e assertividade.

Visando a forte cultura enraizada na empresa ABC, resolveu-se criar um *framework* sob medida baseado no *Design Thinking*, incorporando algumas técnicas do Scrum que possam auxiliar na gerenciamento do portfólio de projetos.

O Service Desk, *software* utilizado atualmente pela empresa, possui uma estruturação simples para acompanhar as fases de um projeto, que são: não iniciado, em andamento e concluído. Na Figura 8, pode-se verificar como é o funcionamento do processo destas três fases.

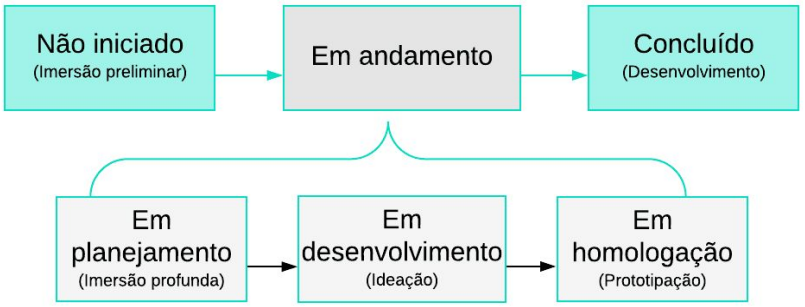
Figura 8 - Processo atual das fases de um projeto

Fonte: Desenvolvido pelo autor (2020).

Baseado no método *Design Thinking*, a proposta deste trabalho é incorporar duas novas divisões nas fases do projeto, que são: imersão de profundidade e prototipação. Para o caso em questão, por motivo de padronização, serão utilizadas as nomenclaturas “Em planejamento” e “Em homologação” respectivamente.

A proposta implica em subdividir a fase “Em andamento” em três etapas, sendo a primeira “Em planejamento”, a segunda “Em desenvolvimento” e por fim “Em homologação”. A Figura 9 representa de forma ilustrativa esta proposta de adequação.

Figura 9 - Ilustração da proposta de subdivisão do processo atual



Fonte: Desenvolvido pelo autor (2020).

A fase de imersão preliminar será chamada de “Não iniciado”, esta fase é realizada através das reuniões de GT’s que ocorrem no primeiro semestre do ano e definem os projetos que farão parte do portfólio. Para que isso ocorra, o projeto precisa ser aprovado pelo GT e pelo setor de TI, além de possuir o documento AR preenchido, onde consta o real problema a ser resolvido, as partes interessadas, os objetivos do projeto, escopo e prioridades.

A fase de imersão profunda será chamada de “Em planejamento” e irá compreender a atividade de montar a equipe que irá trabalhar neste projeto e nivelar as informações entre eles. Para isso será realizada uma reunião para validar os objetivos, restrições, especificações, prazos, cronograma, recursos, divisão de tarefas, documentos do projeto e outras questões afins. Esse encontro será denominado reunião de iniciação.

A fase de ideação será chamada de ‘Em desenvolvimento”, esta fase irá realizar as etapas de monitoramento e controle do que está sendo desenvolvido, e verificará se a evolução está de acordo com o planejado. Em caso de problemas, é nesta fase que serão gerenciadas as possíveis soluções, e quando necessário serão realizadas reuniões com a equipe para manter o projeto alinhado.

A fase de prototipação será chamada de “Em homologação” e será responsável pela análise e *feedback* da entrega ao cliente. Como se trata da primeira entrega, poderá haver adequações de requisitos que não ficaram claros ou ainda ajustes de falhas do *software*. Quando isso ocorrer, será realizado um novo planejamento para desenvolvimento das premissas e entrega do projeto.

A fase de desenvolvimento será chamada de “Concluído”, nesta fase o projeto é encerrado e o produto é entregue para utilização do cliente. Nesta etapa também será analisado o desempenho dos envolvidos, onde o gerente poderá apresentar os indicadores de desempenho de produtividade do projeto, o *feedback* e a documentação.

Cada fase do projeto terá um conjunto de tarefas a ser seguido, estas tarefas serão monitoradas através do Scrum *Board* ou Kanban. Através do Quadro 3 identifica-se o resumo destas etapas separadas por fases, podendo ser observada através da mudança das cores, iniciando na cor mais clara até a mais escura.

Quadro 3 - Resumo das etapas de cada fase do projeto

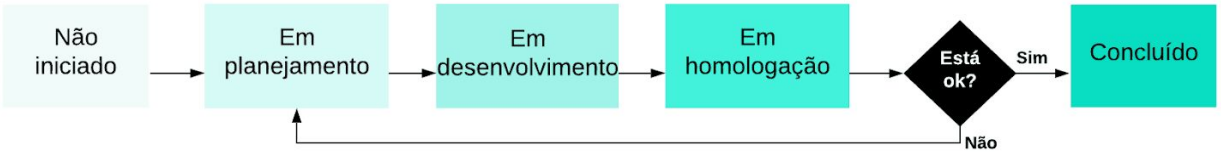
PROPOSTA DE FASES E ETAPAS DO PROJETO		
1ª FASE	**NÃO INICIADO**	
		Etapa A - Determinar o real problema a ser resolvido
		Etapa B - Identificar Stakeholders
		Etapa C - Definir objetivos
		Etapa D - Definir escopo, recursos e tarefas prioritárias
2ª FASE	**EM PLANEJAMENTO**	
		Etapa A - Montar a equipe do projeto
		Etapa B - Nivelar informações
		Etapa C - Reunião de iniciação
		Etapa D - Discriminar tempos
		Etapa E - Definir agenda do trabalho
3ª FASE	**EM ANDAMENTO**	
		Etapa A - Reunião frequentes com a equipe
		Etapa B - Monitorar e controlar o planejamento do desenvolvimento
		Etapa C - Reportar a evolução do projeto
		Etapa D - Gerenciar problemas
4ª FASE	**EM HOMOLOGAÇÃO**	
		Etapa A - Análise do feedback para possível replanejamento
5ª FASE	**CONCLUÍDO**	
		Etapa A - Concluir documentação
		Etapa B - Análise da performance
		Etapa C - Feedback
		Etapa D - Encerrar

Fonte: Desenvolvido pelo autor (2020).

Já a Figura 10 está ilustrado o processo final, ao nível macro, do funcionamento da proposta de gerenciamento de projeto, nesta figura também é

possível observar a evolução do processo através da mudança das cores, iniciando na cor mais clara até a mais escura.

Figura 10 - Proposta final das fases de um projeto



Fonte: Desenvolvido pelo autor (2020).

4 PROTOTIPAÇÃO

Neste capítulo serão apresentados os experimentos realizados neste estudo de caso, em cada uma das etapas do projeto, bem como os detalhamentos a respeito da condução do experimento. Também serão apontados os resultados obtidos e as validações realizadas.

4.1 NÃO INICIADO

Nesta fase preenche-se o documento com a análise de requisitos, que tem como objetivo atender as etapas iniciais do projeto, garantindo assim uma imersão preliminar ao assunto. A Figura 11 apresenta o modelo de documento utilizado.

Figura 11 - Documento criado para a análise de requisitos (AR)

ID Projeto	Nome do projeto		
GT ou Empresa		Líder/Solicitante	□ Sistemas □ Infraestrutura

Histórico de Revisão

Data	Versão	Descrição	Autor

1. Qual o problema a ser resolvido

2. Como é feito hoje

3. Qual a sugestão do grupo para resolver isso

4. Qual o ganho desta mudança (custo/benefício)

5. Especificar requisitos do que está sendo solicitado

Passo a passo/ Regras e restrições/ O que deve mostrar ou aparecer/ opções/ filtros

6. Quais as pessoas/departamentos afetados

7. Atende quais empresas

□ Todas fábricas
□ Fábrica 1 □ Fábrica 2 □ Fábrica 3 □ Fábrica 4 □ Fábrica 5

□ Todos CDs (Centros de Distribuição)
□ CD 1 □ CD 2 □ CD 3 □ CD 4 □ CD 5

□ Outras, Qual? _____

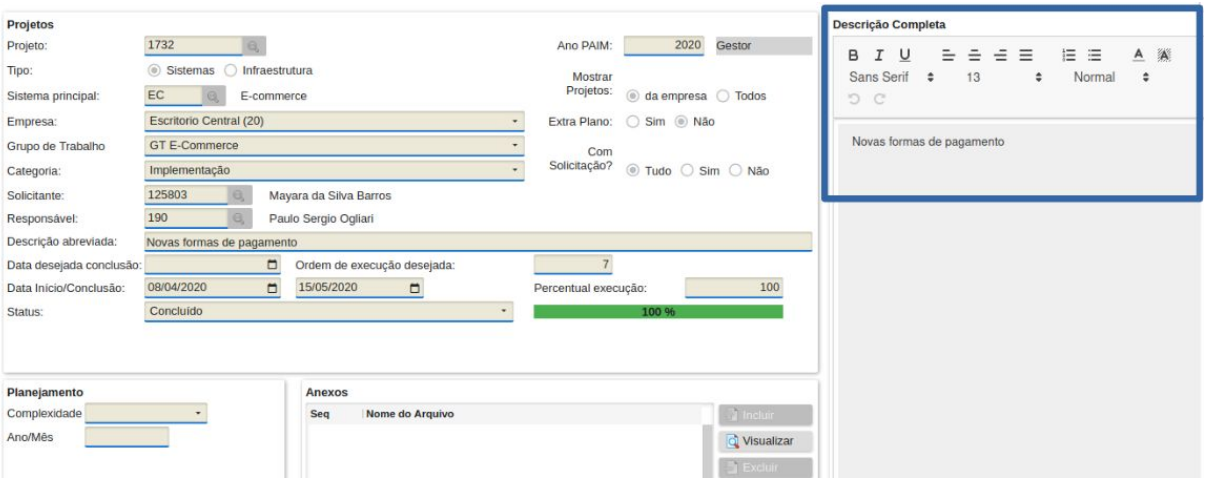
8. Anexos e/ou telas (de exemplo)

Fonte: Desenvolvido pelo autor (2019).

Este levantamento prévio de requisitos auxiliou para que o departamento de TI identificasse qual a direção e caminho a percorrer para chegar ao que o cliente necessita. Com base neste documento, é possível identificar requisitos funcionais e não funcionais, e prever a dimensão e complexidade do projeto antes mesmo de sua análise.

Anteriormente, utilizava-se um campo de texto livre definido como ‘descrição completa’, onde o usuário relatava informações do projeto, e se houvesse interesse, também poderia anexar arquivos ou telas de exemplo. Porém, muitas vezes o cliente não sabia o que relatar e a descrição recebida não detinha informação suficiente para o entendimento e dimensão do assunto. A Figura 12 ilustra o processo atual de um destes projetos, onde não estão sendo apontados maiores detalhamentos.

Figura 12 - Tela de detalhamento do projeto atual



Fonte: Autor (2020).

4.2 EM PLANEJAMENTO

Nesta fase definiram-se os projetos a serem trabalhados, comprovou-se o recebimento do AR que detém a imersão preliminar do assunto e também definiram-se os integrantes que farão parte do time para o desenvolvimento do projeto.

Os projetos desenvolvidos neste experimento foram:

a) Projeto 1 - Melhorias no processo de controle de documentos de prestadores de serviço, o qual terá integração com o controle de acesso às fábricas e demais unidades da empresa ABC.
b) Projeto 2 - Atualização da versão do EDI (*Electronic Data Interchange*), que em português significa Troca Eletrônica de Dados

O segundo passo foi o treinamento da equipe de envolvidos no experimento nivelando-se o conhecimento sobre métodos ágeis. No treinamento, que teve duração de 30 minutos, foram repassados os ciclos do *framework* e apresentados em detalhes cada evento a ser executado, assim como as vantagens de sua utilização.

No terceiro passo realizou-se a separação do projeto em pequenas tarefas e uma reunião de iniciação. Esta reunião teve o objetivo de aprofundar o levantamento de requisitos e o entendimento da necessidade do usuário, além de realizar a estimativa de tempo para cada uma das tarefas. Também foram ordenadas as prioridades elegendo primeiramente as tarefas que agregam maior valor ao usuário, e após isso, priorizou-se a tarefa mais fácil, usando-a assim como referência na estimativa das próximas.

Com o *backlog* ordenado, estimou-se a duração das tarefas através do método *Planning Poker* convertido em horas, seguindo as necessidades e a realidade do setor. O funcionamento ocorreu da seguinte forma:

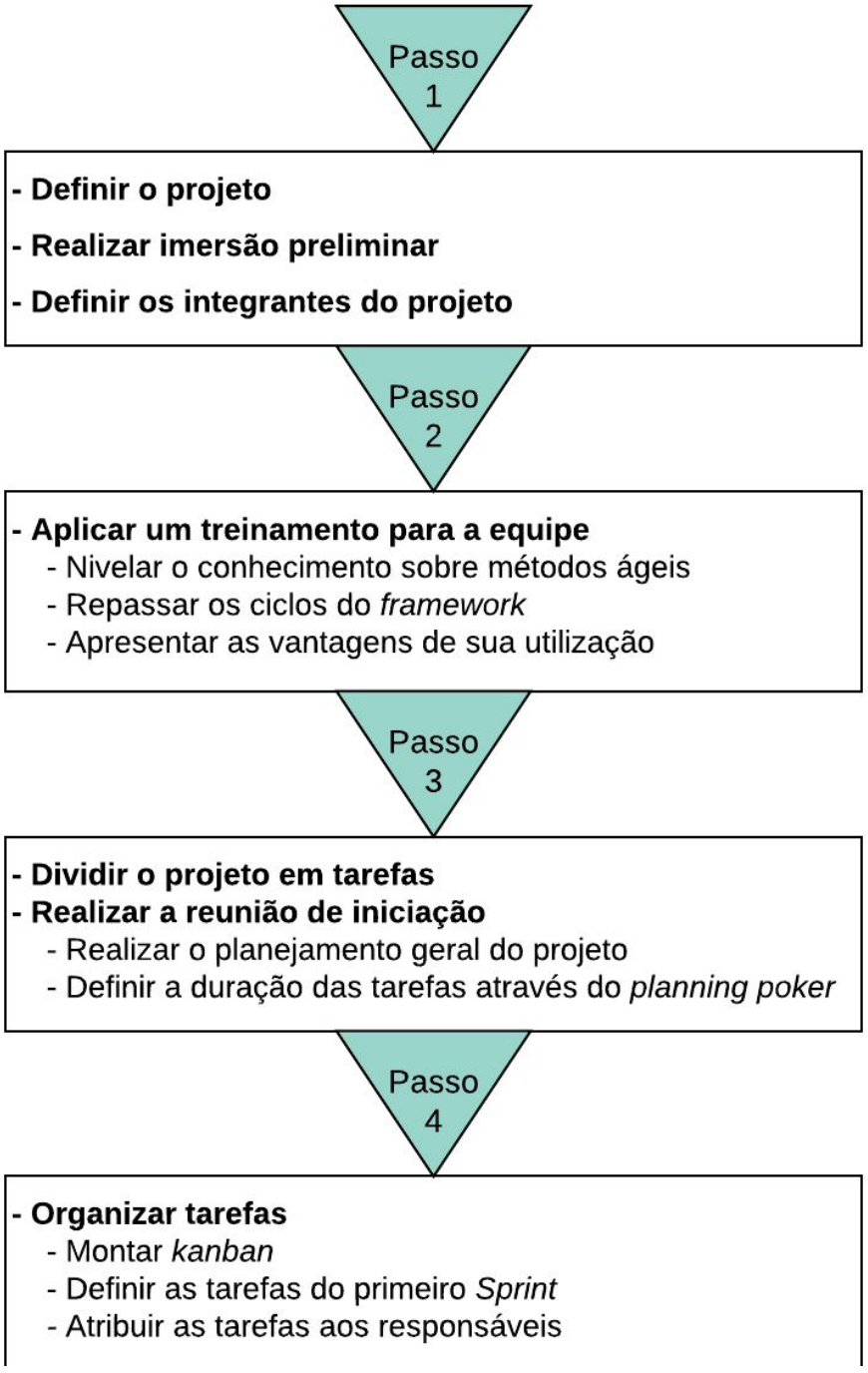
a) O Scrum Master leu a tarefa em voz alta
b) Os integrantes do grupo aproveitaram para esclarecer algumas dúvidas a respeito da tarefa, realizando uma análise e entendimento mais profundo na necessidade do usuário.
c) Em segredo, cada desenvolvedor jogou a carta com a quantidade de horas que acreditava ser a mais assertiva, baseado no quanto complexo eles entenderam ser a tarefa. Quando os desenvolvedores indicarem diferentes cartas, é feito uma discussão para entender o motivo e é realizada uma segunda rodada para esta tarefa. O baralho de *Planning Poker* contou com as seguintes cartas e definições:

a) 0 horas: a tarefa não precisa ser feita por algum motivo (talvez já esteja pronta)
b) Até 1 hora: a tarefa é muito simples, provavelmente menos de 1 hora de desenvolvimento.
c) Até 3 horas: a tarefa é simples, provavelmente leve menos de um turno de trabalho, como por exemplo, uma manhã.
d) Até 5 horas: a tarefa é um pouco mais trabalhosa e deve ocupar no mínimo um turno de trabalho.
e) Até 8 horas: tarefa de complexidade mediana, provavelmente demande um dia de trabalho.
f) Até 24 horas: a tarefa é complexa, demanda de algum estudo ou muito desenvolvimento, provavelmente tomará alguns dias da semana, sendo no máximo 3 dias.
g) Até 40 horas: a tarefa é muito complexa, demandará estudo e bastante desenvolvimento, levará em média uma semana (5 dias úteis).
h) Mais de 44 horas: a tarefa é complexa demais e não vale a pena ser estimada. Sugere-se quebrar a necessidade em tarefas menores, para que a estimativa seja realizada com maior exatidão.

Após identificar os tempos de cada tarefa, estimou-se os itens que fariam parte do Sprint e definiu-se a duração do Sprint.

O primeiro *Sprint* teve a duração de uma semana, que é o período padrão de atualização de versão no departamento em questão. Já no segundo experimento, optou-se por realizar um experimento com um Sprint quinzenal. De forma objetiva, a Figura 13 ilustra o passo a passo realizado.

Figura 13 - Passo a passo para implementação da metodologia



Fonte: Desenvolvido pelo autor (2020).

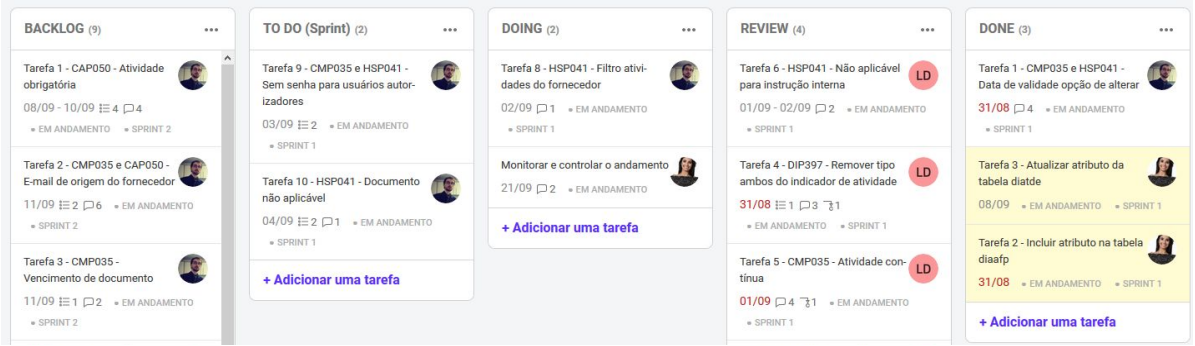
Cada tarefa dos projetos experimentados teve a determinação de métricas e controles como a data de entrega, etiquetas que indicavam a Sprint *e* a etapa do projeto, a estimativa de horas e o total de horas trabalhadas.

Para o primeiro experimento, montamos uma equipe pequena, de apenas 3 integrantes, visando realizar o amadurecimento e o aprendizado da proposta. Esta equipe contou com:

a) Um Scrum Master, que garantiu que a metodologia fosse aplicada e adaptada a realidade do setor;
b) Um analista de negócio, que não fazia parte do setor de desenvolvimento, porém compreendia o negócio do cliente. Esta figura mapeou as necessidades e em alguns momentos até a especificação da solução, além de desempenhar o papel de testar a funcionalidade logo após o desenvolvimento;
c) Um desenvolvedor/analista, que realizou o levantamento dos requisitos (imersão profunda), analisou os impactos das necessidades propostas, indicou as melhores oportunidades para resolução da proposta e realizou o desenvolvimento do *software*.

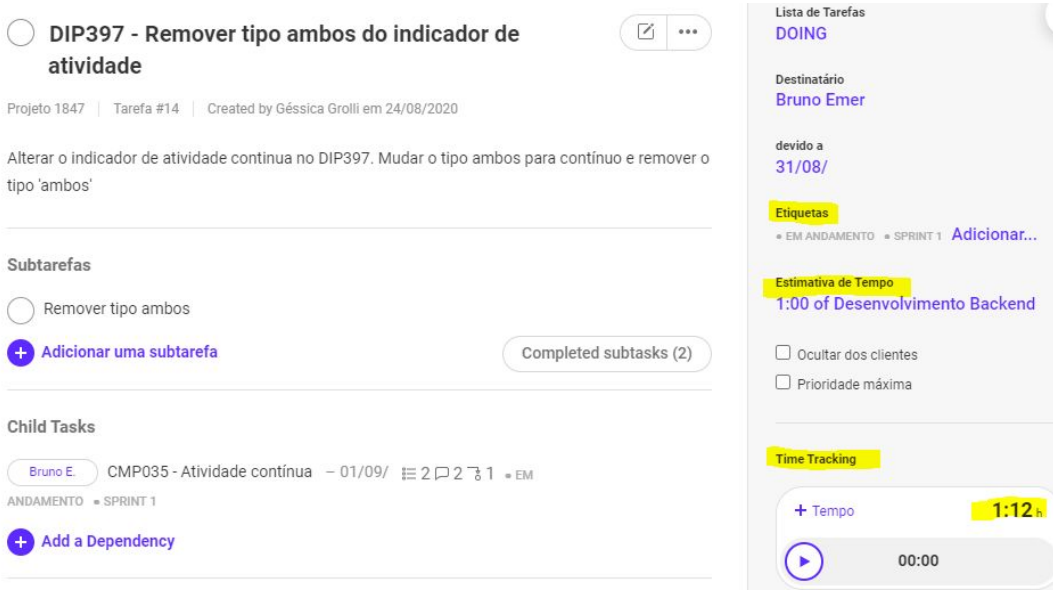
Para os controles, métricas e montagem do Kanban, neste primeiro experimento utilizou-se a ferramenta ActiveCollab. As imagens 14 e 15 demonstram a utilização deste *software* e as tarefas com as métricas propostas.

Figura 14 - Kanban do projeto 1 com as tarefas desmembradas



Fonte: Autor (2020).

Figura 15 - Exemplo de tarefa demonstrando as métricas



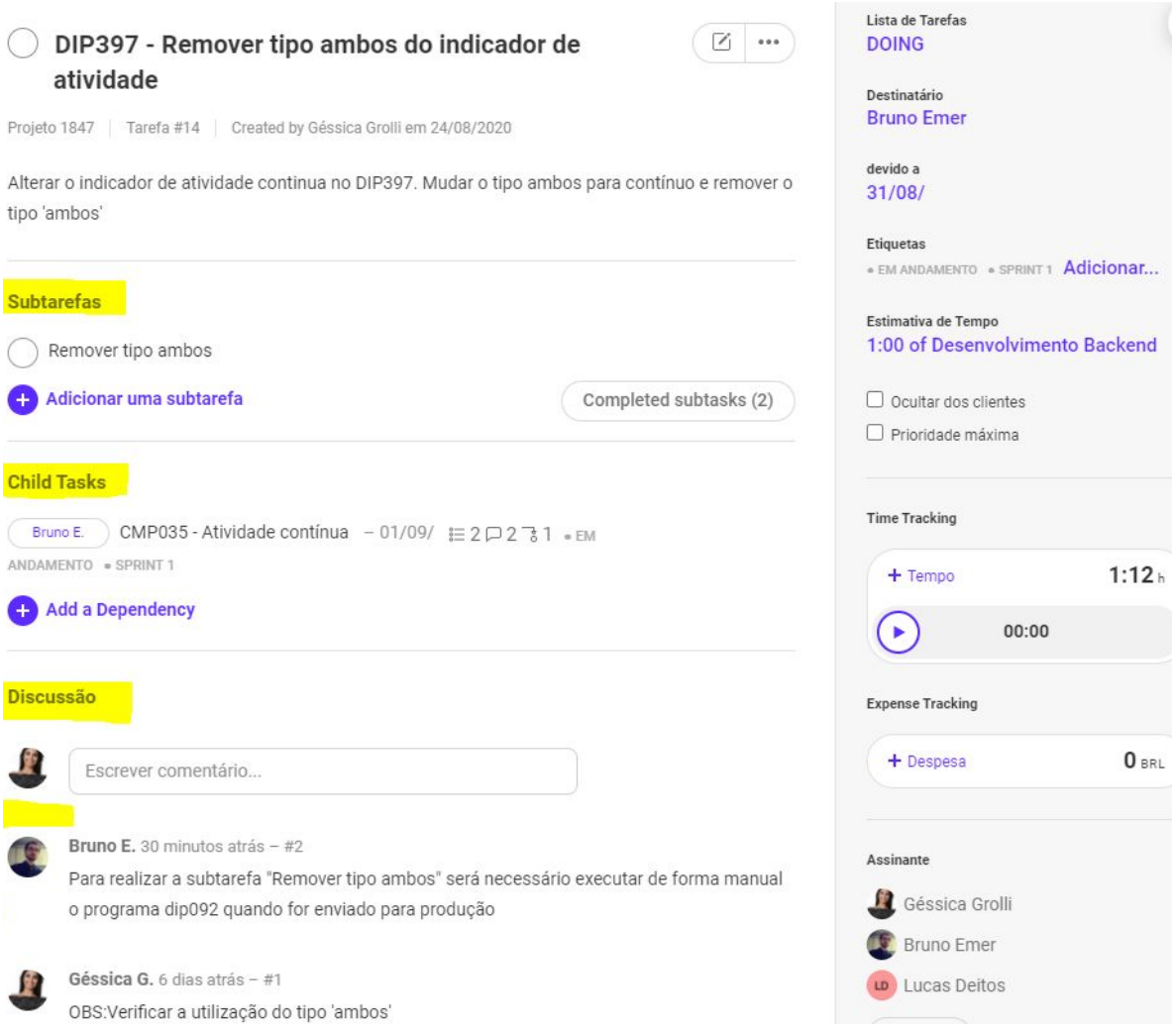
Fonte: Autor (2020).

Através do ActiveCollab também foi possível trocar mensagens através das tarefas, mantendo o histórico de todas conversas do grupo em um único local. Também foi possível configurar lembretes por e-mail para tarefas que necessitam de algum tratamento especial. As Figuras 16 e 17 ilustram os lembretes e a troca de mensagens, respectivamente.

Figura 16 - Exemplo de lembrete

Fonte: Autor (2020).

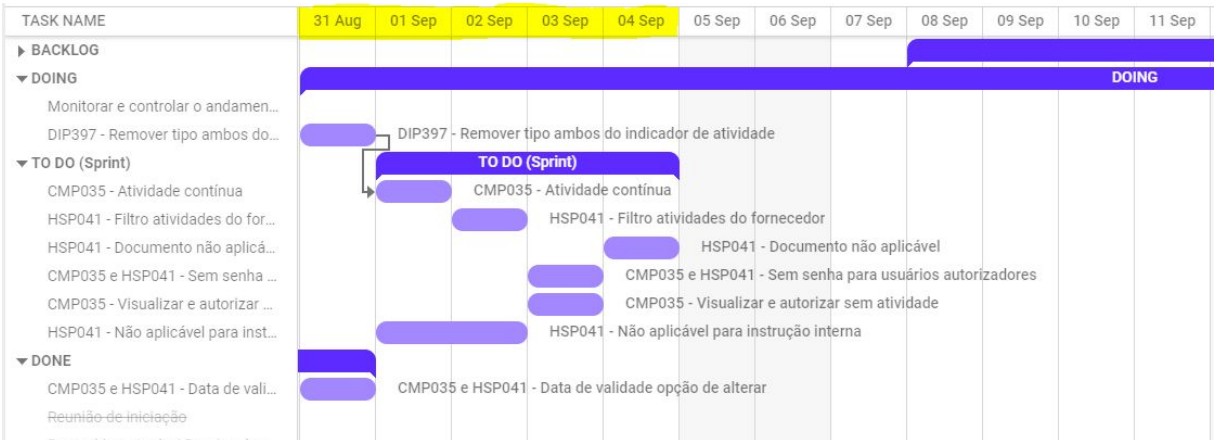
Figura 17 - Exemplo de troca de mensagens



Fonte: Autor (2020).

Através do *software* Activecollab foi possível visualizar de forma prática a divisão das tarefas durante a Sprint e a carga de trabalho dos envolvidos. Isso possibilitou uma melhor identificação dos trabalhos dependentes e dos tempos, garantindo assim, o bom planejamento das Sprints. A seguir, a Figura 18 demonstra o gráfico de Gantt do primeiro Sprint com as tarefas distribuídas entre as datas propostas e a dependência existente entre cada uma delas.

Figura 18 - Gráfico de Gantt do 1º Sprint



Fonte: Autor (2020).

Já a Figura 19 demonstra a carga de trabalho do programador envolvido no projeto, possibilitando identificar os tempos livres ou superlotados. Percebe-se que não foi utilizado todo o tempo de trabalho com tarefas da Sprint, pois como é um experimento isolado em um ambiente que não trabalha com metodologias ágeis, previu-se a possibilidade de interrupções urgentes.

Figura 19 - Carga de trabalho do 1º e 2° Sprint

Fonte: Autor (2020).

Para o segundo experimento, montou-se uma equipe de 4 integrantes, visando criar o papel do PO e, principalmente, retirar a responsabilidade da análise do desenvolvedor. Para este experimento também foram utilizadas as aprendizagens identificadas no primeiro projeto. Esta equipe contou com:

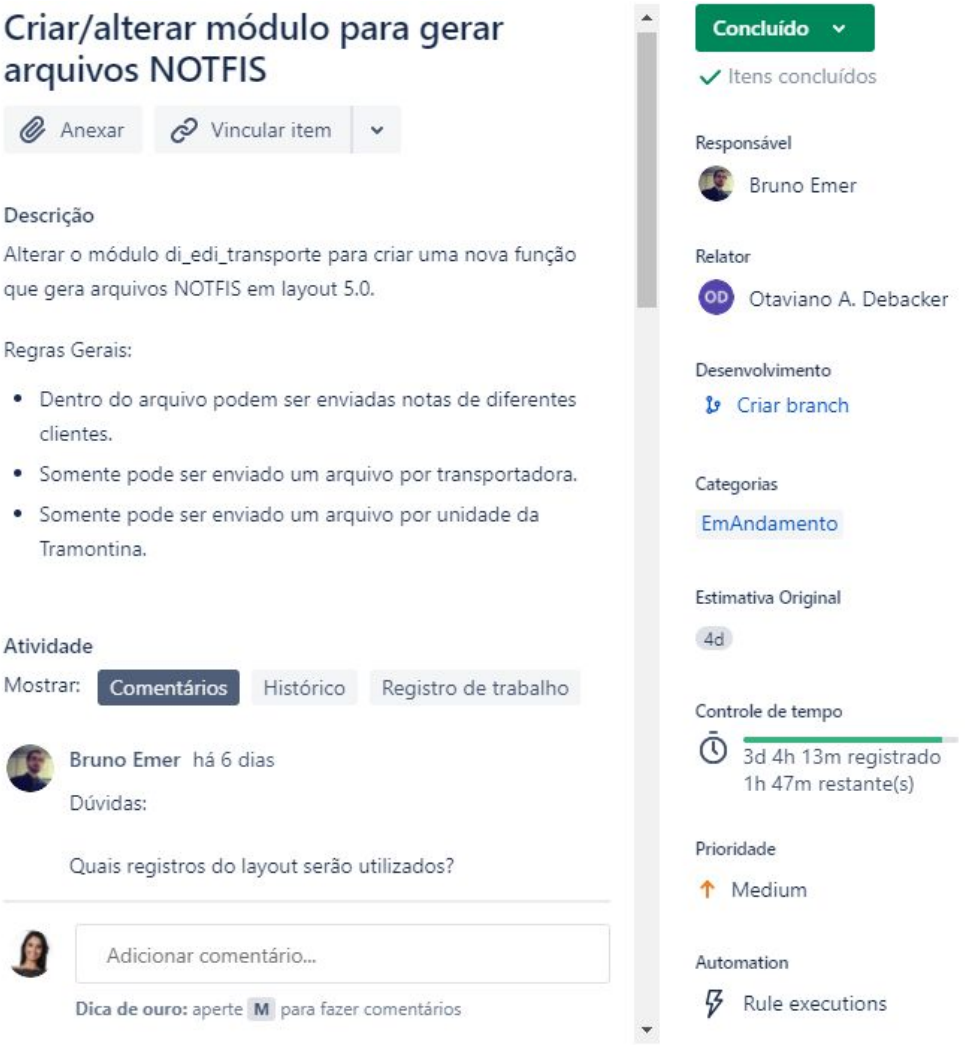
a) Um Scrum Master, que garantiu que a metodologia fosse aplicada e adaptada a realidade do setor;
b) Um PO - que neste primeiro momento, por falta de recursos humanos no setor, desenvolveu mais de um papel. Além de PO, esta figura realizou a análise junto ao usuário e a análise detalhada de todo o projeto, dividindo-o em tarefas. Durante a execução da Sprint, esta figura irá, inicialmente, resolver dúvidas e bugs do sistema legado em que lhe compete, visando blindar a equipe de desenvolvimento da Sprint que estarão desenvolvendo.
c) Time de dois desenvolvedores, para o desenvolvimento da aplicação.

As Figuras 20 e 21 demonstram, respectivamente, a montagem do Kanban, os controles e métricas do projeto através da ferramenta Jira Software.

Figura 20 - Kanban do projeto 2 com as tarefas e subtarefas

Fonte: Autor (2020).

Figura 21 - Exemplo de subtarefa com detalhamentos e métricas



Fonte: Autor (2020).

4.3 EM ANDAMENTO

Nesta fase iniciou-se o desenvolvimento da Sprint e realizou-se o monitoramento do trabalho realizado pelo programador. Para isso, utilizou-se a prática das reuniões diárias para *feedback* e entendimento do andamento do projeto, bem como a apresentação do gráfico de *burndown* para demonstrar a relação entre o que foi estimado e o que foi realizado.

No primeiro experimento as reuniões diárias ocorreram no início do dia, com horário e local predefinidos. Esta interação com o desenvolvedor teve uma duração

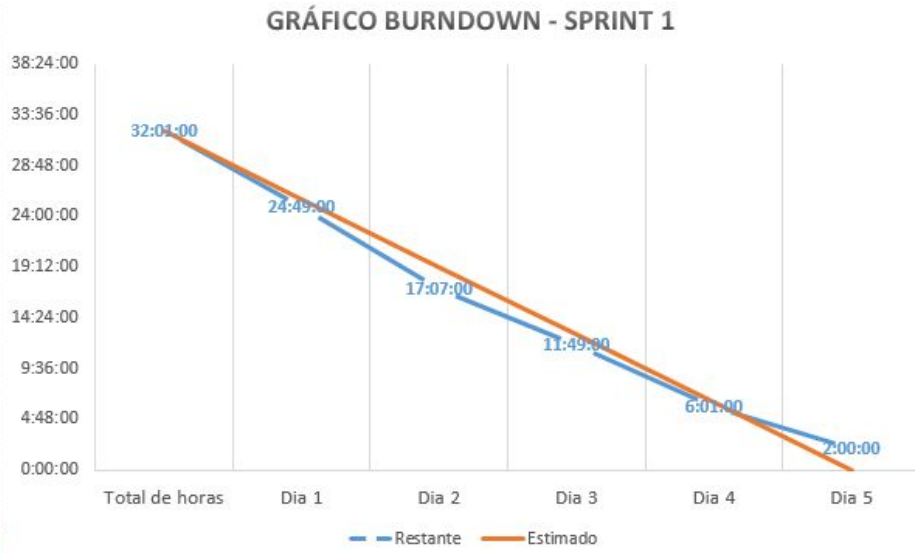
média maior que o esperado nos primeiros 2 Sprints, porém a partir do terceiro Sprint, os envolvidos se adaptaram ao método e pode-se realizar reuniões diárias de em média 15 minutos, que foram determinante para o bom andamento do projeto.

Foi realizado o acompanhamento das tarefas diariamente através do gráfico de *burndown*, os quais foram utilizados para medir a velocidade da execução das tarefas em comparação ao estimado, e que puderam ser analisados durante o andamento da Sprint.

Os primeiros dois Sprints tiveram uma carga média de 31 horas semanais. Já o terceiro e último Sprint teve uma carga de trabalho estimada em 24 horas, mas que demandou apenas 13 horas de desenvolvimento, ficando assim mais compacta se comparada às anteriores.

A primeira Sprint teve a divisão do trabalho em 10 tarefas. Avaliando o gráfico do demonstrado através da Figura 22, percebe-se que houve um pequeno atraso no desenvolvimento das tarefas dos primeiros 3 dias. Contudo, este tempo pode ser recuperado a partir do quarto dia. Embora tenha ocorrido esse pequeno atraso no início, ainda obteve-se um tempo remanescente de 2 horas ao final da Sprint. De uma forma geral o tempo estimado ficou muito próximo ao realizado.

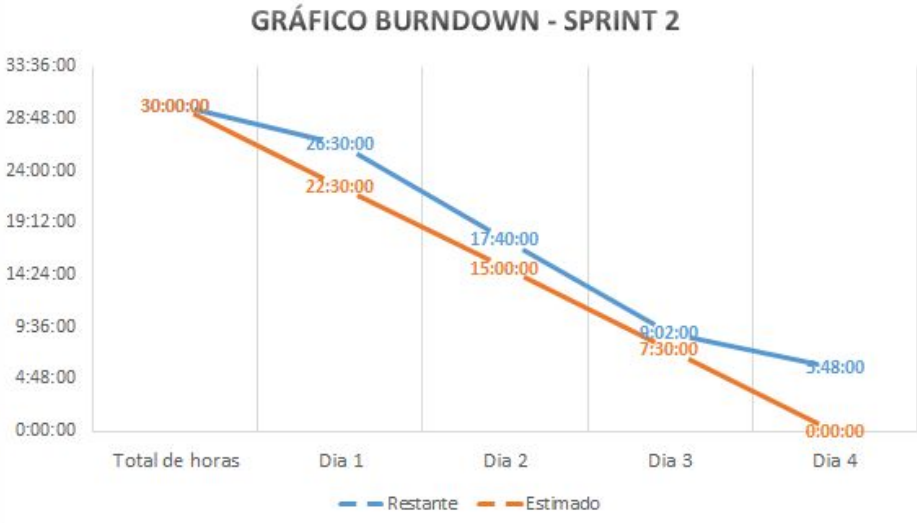
Figura 22 - Projeto 1: tempo estimado X realizado do 1º Sprint



Fonte: Do autor (2020).

Na Figura 23 verifica-se o andamento do segundo Sprint o qual foi repartido em 4 tarefas. Neste Sprint também incluiu-se o tempo das reuniões diárias que foi um aprendizado da primeira Sprint, e incluiu-se um *bug* que ocorreu em consequência da primeira Sprint. O gráfico da Figura 23 demonstra que o desenvolvimento manteve-se adiantado em relação ao estimado, e finalizou-se com um tempo remanescente de 5 horas e 48 minutos.

Figura 23 - Projeto 1: tempo estimado X realizado do 2º Sprint

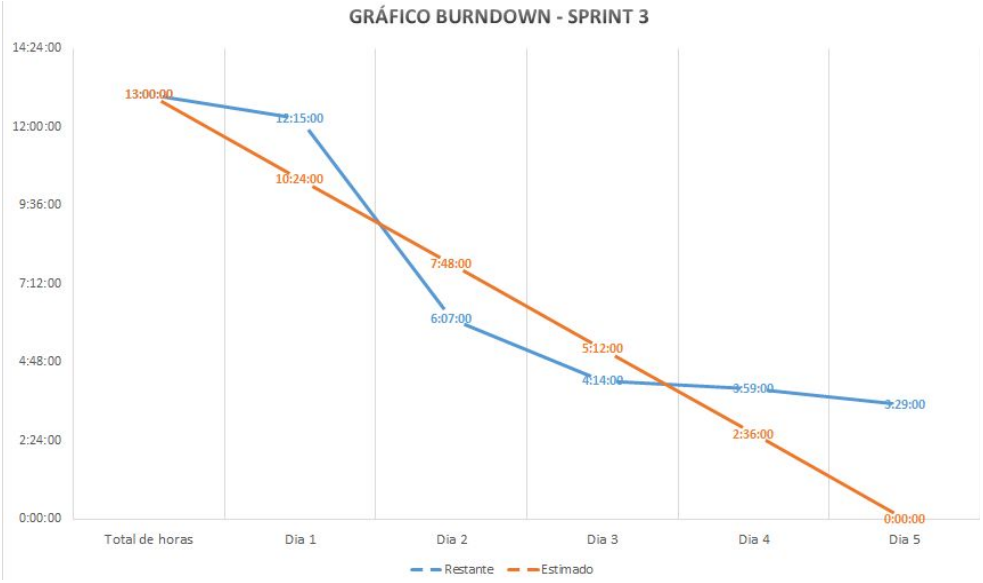


Fonte: Do autor (2020).

Na Figura 24 verifica-se o andamento do terceiro Sprint o qual possui uma tarefa de desenvolvimento e mais 3 tarefas de monitoramento e/ou reuniões. Através dos Sprints anteriores, houve o aprendizado da necessidade de incluir o tempo das reuniões de revisão e planejamento da *Sprint* na estimativa e tempos.

No gráfico da Figura 24 percebe-se um atraso considerável no primeiro dia da Sprint, que ocorreu em detrimento de *bugs* do sistema legado que ocorreram e precisaram ser resolvidos urgentemente. Independente disso, no segundo dia o objetivo foi retomado e como esta Sprint possuía uma carga menor de trabalho o projeto não sofreu nenhuma consequência e ainda finalizamos com um tempo remanescente de 3 horas e 29 minutos.

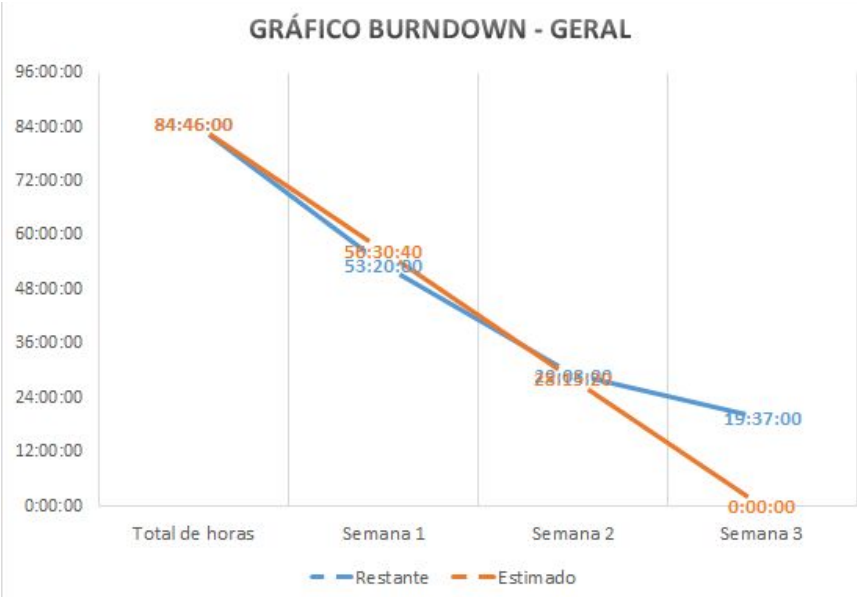
Figura 24 - Projeto 1: tempo estimado X realizado do 3º Sprint



Fonte: Do autor (2020).

Na Figura 25 verifica-se o andamento do projeto na totalidade, onde há a divisão levando-se em consideração os três Sprints semanais realizados. Verifica-se que os primeiros Sprints ficaram poucas horas atrasados em relação à velocidade total do projeto, e como o terceiro Sprint era menor, pode-se recuperar e ainda adiantar-se em torno de 19 horas do total estimado.

Figura 25 - Relação do tempo estimado X realizado do projeto 1



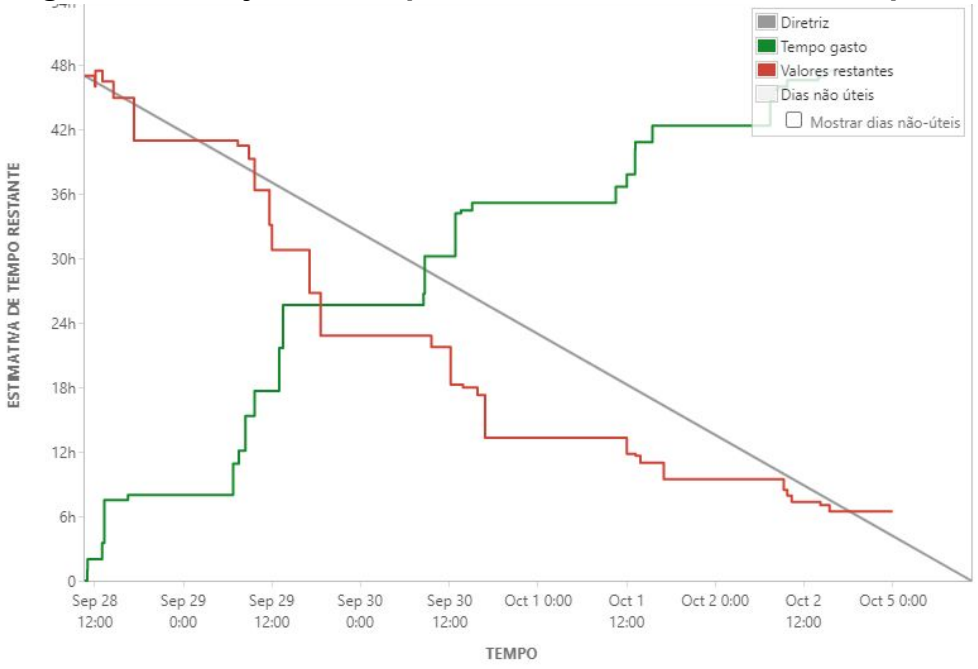
Fonte: Do autor (2020).

No segundo experimento, as reuniões diárias também ocorreram no início do dia, sempre no início do expediente e em local predefinido. Esta interação com o time teve uma duração média de 15 minutos, que foi determinante para o bom andamento do projeto e principalmente para a identificação dos problemas e aprendizagens contínuas.

Neste segundo momento, também foi realizado o acompanhamento das tarefas através do gráfico de *burndown*. A primeira Sprint teve a divisão do trabalho em 3 histórias de usuários, que foram subdivididas em 9 tarefas, totalizando pouco mais de 47 horas de estimativa de trabalho.

O importante a ser verificado no gráfico do primeiro Sprint demonstrado através da Figura 26, é a linha vermelha em relação à linha cinza (diretriz). Quanto mais próximas estas duas linhas estiverem, mais assertiva foi a estimativa das tarefas pelo time. Se a linha vermelha ficar abaixo da cinza significa que estamos adiantados, se a linha ficar acima, significa atraso. Percebe-se assim que na maior parte do tempo a Sprint esteve adiantada em relação ao tempo estimado, finalizando com um tempo remanescente de mais de 6 horas, para uma sprint semanal.

Figura 26 - Projeto 2: tempo estimado X realizado do 1º Sprint

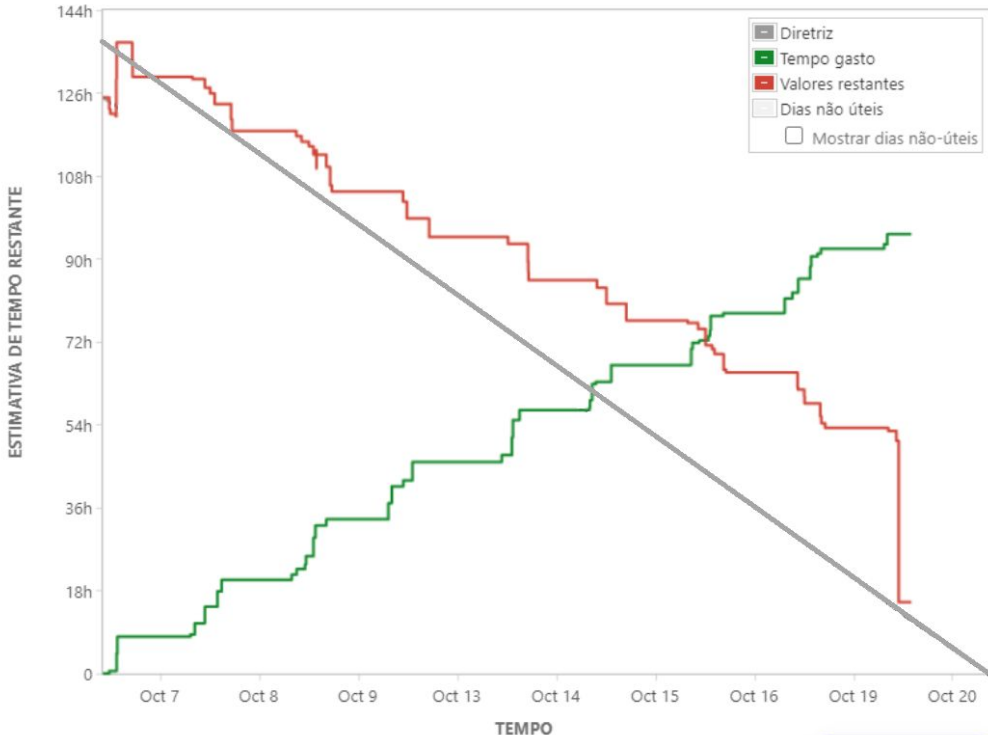


Fonte: Do autor (2020).

A segunda Sprint teve a divisão do trabalho em 4 histórias de usuários, que foram subdivididas em 15 tarefas, totalizando 141 horas de estimativa de trabalho.

A Figura 27 demonstra o gráfico de *burndown* do 2º Sprint. Através da linha vermelha em relação à linha cinza, verifica-se que no início da Sprint o time esteve bem alinhado ao que foi proposto, ao passar dos dias houve um pouco de atraso em relação ao estimado, contudo, o time esforçou-se para realizar a entrega no prazo e ainda foi possível finalizar o projeto com um tempo remanescente de mais de 15 horas e 30 minutos, para uma Sprint de 15 dias.

Figura 27 - Projeto 2: tempo estimado X realizado do 2º Sprint



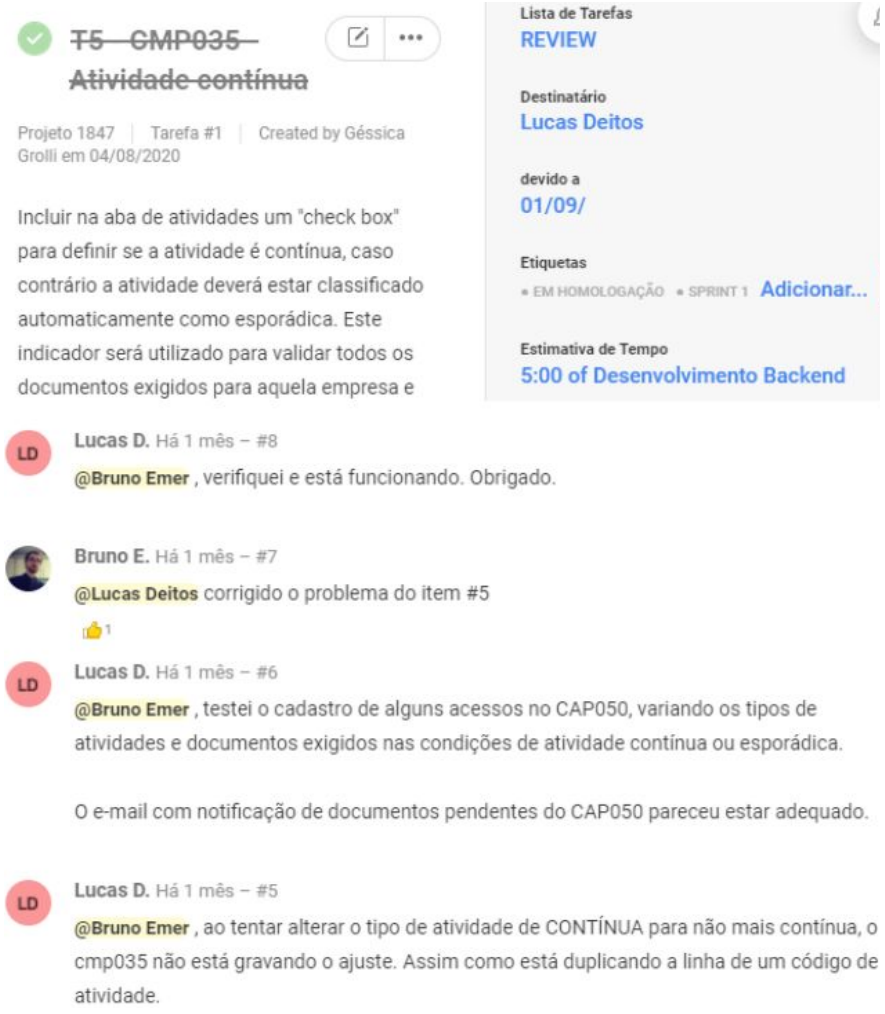
Fonte: Do autor (2020).

4.4 EM HOMOLOGAÇÃO

No primeiro experimento, os testes foram realizados pelo líder de GT, que realizou este trabalho em uma base específica para testes. O controle desta fase foi realizada no quadro Kanban através da coluna *review* e através da etiqueta

identificada como homologação. A Figura 28 demonstra um exemplo de tarefa em que houve ajustes após homologação.

Figura 28 - Projeto 1: exemplo de tarefa sendo homologada



Fonte: Do autor (2020).

No segundo experimento, na primeira Sprint os testes foram realizados pelo *PO,* que realizou este trabalho em uma base específica para testes, e pode observar se as demandas solicitadas concordavam com o tratado com o usuário.

No segundo Sprint, experimentou-se realizar os testes através da equipe de desenvolvimento, de tal forma que o programador que escreveu o código repassa a tarefa de teste para outro programador do mesmo time que não teve participação no desenvolvimento, evitando assim testes viciados de programação. Como resultado

desta alternância de tarefas de testes identificou-se situações de ajustes em programas e bugs que o programador não identificaria sozinho, por estar realizando os testes de sua maneira e ainda tendo a visão da programação previamente realizada.

O controle desta fase e destas tarefas foi realizado no quadro de *Kanban* através de tarefas de testes e através da etiqueta identificada como homologação. A Figura 29 demonstra um exemplo de tarefa de teste.

Figura 29 - Projeto 2: exemplo de tarefa sendo homologada

Fonte: Do autor (2020).

Em ambos experimentos, e ao final de cada Sprint, realizaram-se reuniões de revisão. Nesta reunião foi possível coletar o *feedback* dos *stakeholders*, garantindo assim a entrega do projeto alinhada e de acordo com o esperado pelo cliente. A

Sprint somente é enviada para produção após a apresentação das mudanças e a confirmação do cliente. Caso contrário, são revisados os ajustes necessários e replanejado um novo ciclo.

4.5 CONCLUÍDO

No primeiro experimento decidiu-se em não realizar a documentação, pois o projeto terá uma segunda etapa que será tratada posteriormente. Assim, como ainda haverão algumas mudanças de regras e de funcionamento do programa, para evitar retrabalhos, deixou-se a etapa de documentação para um segundo momento, quando o programa estiver estável. Esta decisão foi tomada em conjunto com o analista de negócio e cliente.

No segundo experimento, a documentação foi realizada pelo PO. O controle desta fase foi realizada no quadro Kanban através da coluna ‘Documentação’, que antecede a coluna concluído, garantindo assim que a tarefa somente seja disponibilizada ao usuário após a revisão da documentação.

Em ambos experimentos realizaram-se reuniões de retrospectiva a cada finalização de Sprint. Na reunião de retrospectiva, colheu-se o *feedback* do time de desenvolvimento e traçaram-se ações para os pontos negativos identificados, garantindo assim a evolução e o aprendizado contínuo em relação à metodologia de trabalho realizada.

Após a conclusão de todas as Sprints referenciadas ao projeto e a documentação de todos os elementos acima mencionados, o projeto finalmente é encerrado, contabilizando-se as horas de todo o processo.

4.6 RESULTADOS

Através das reuniões de retrospectiva da sprint foi possível identificar os resultados da metodologia, bem como os itens que precisavam de uma atenção para obterem melhor resultado. Estas reuniões são muito importantes, pois realizam um feedback do time e são o caminho para implementação da melhoria contínua.

Os Quadros 4, 5 e 6 demonstram, respectivamente, os aprendizados, problemas e pontos positivos identificados ao longo da execução dos três Sprints do projeto do primeiro experimento. Na seção problemas identificados, alguns itens possuem a palavra resolvido entre colchetes, e na sequência descreve-se o que foi feito para resolver o problema.

Quadro 4 - Aprendizados do primeiro experimento

Sprint 1	Sprint 2	Sprint 3
Deixar um tempo no início da Sprint para eventuais *bugs* da Sprint anterior, e no final para revisões; Lembrar de colocar as tarefas relacionadas a alteração de tabelas; Incluir um quadro 'bloqueio', para tarefas que possam ficar bloqueadas por conta de algum retorno do terceiro	Lembrar de colocar as tarefas que são de maior importância ao cliente como prioritárias, e na sequência, as mais fáceis; Verificar se haverá feriados, férias ou saídas programadas de qualquer um do time; Além da tarefa de análise, incluir as tarefas de reunião de planejamento, monitoramento e revisão/encerramento da Sprint	Sugestão de utilizar um método de priorização das histórias, por exemplo kano ou moscow.

Fonte: Do autor (2020).

Quadro 5 - Problemas identificados no primeiro experimento

(continua)

Sprint 1	Sprint 2	Sprint 3
Dificuldade do líder de GT (analista de negócio) saber o que está com ele. [Resolvido] Através da criação do quadro *'review'*, atribuindo as tarefas ao interessado; A Sprint de 1 semana é muito curta. [Resolvido] Realizar teste com um grupo maior, que enviará a atualização quinzenalmente;	É preciso padronizar a data e hora da atualização da base de testes; Aumento da carga de trabalho (sprits) inicialmente visualizada. [Resolvido] Criada uma terceira *Sprint*;	Não foi possível executar a tarefa de atualização da documentação, pois atualmente não há uma equipe para isso. Tornou-se um trabalho muito custoso para ser realizado pelo analista ou desenvolvedor, visto que não existe documentação atual, e o programa ainda está em fase de ajustes;

(conclusão)

Investir mais tempo de análise, pois o AR não é suficiente e gera muitas dúvidas quando a Sprint é iniciada.	Identificamos 3 tipos de *bugs* que precisam ter indicadores separados: - *Bug* da Sprint atual. [Resolvido] Conta na Sprint através do quadro *'review'*; - *Bug* da Sprint anterior. [Resolvido] Aberta uma tarefa não prevista; - *Bug* do sistema legado. [Resolvido] Controle pelo software Service Desk; Nem todas as tarefas designadas ao analista de negócio para testes teve o retorno dentro do prazo da Sprint. [Resolvido] Conclui-se a tarefa e caso tenham ajustes serão considerados na nova Sprint.	A Sprint teve atraso de 1 dia, iniciando-se no dia 2, devido a solicitações de incidentes e de código legado.

Fonte: Do autor (2020).

Quadro 6 - Pontos positivos do primeiro experimento

Sprint 1	Sprint 2	Sprint 3
Método do líder de GT estar em constante contato com o TI auxilia no entendimento dos requisitos e agiliza o processo; Cliente percebeu a importância de alinhar ainda mais as necessidades antes de iniciar, evitando retornos ao assunto durante a execução; A reunião diária foi indicada como um ponto positivo e essencial para o foco e a motivação de quem está desenvolvendo o projeto; As estimativas via *Planning Poker* deram certo, mantendo-os sempre dentro dos prazos.	Houve aumento da carga de trabalho (Sprints) se comparada ao inicialmente visualizado. [Resolvido] As necessidades foram replanejadas nas Sprints seguintes; Diminuição das dúvidas em relação à metodologia de trabalho; Novamente verificou-se que a forma como o *Planning Poker* foi utilizado auxiliou para que o Sprint ficasse dentro do prazo; Algumas tarefas puderam ser esclarecidas na reunião de *review*, evitando assim dúvidas futuras.	*Software* mais prático e ágil; Tivemos ganho nas entregas em relação à qualidade e prazos propostos; O usuário pode entender com maior facilidade o que estava sendo entregue e programar-se para repassar a informação aos departamentos envolvido; Em virtude da análise prévia realizada, o desenvolvedor percebeu melhora e facilidade no trabalho diário.

Fonte: Do autor (2020).

Os Quadros 7, 8 e 9, demonstram respectivamente, os aprendizados, problemas e pontos positivos identificados ao longo da execução dos dois Sprints do

projeto realizado no segundo experimento. Na seção problemas identificados, alguns itens possuem a palavra resolvido entre colchetes, e na sequência descreve-se o que foi feito para resolver o problema.

Quadro 7 - Aprendizados do segundo experimento

Sprint 1	Sprint 2
Verificar as dependências entre as tarefas; Verificar as necessidades ou preenchimentos obrigatórios e/ou condicionais no momento da análise.	O fato de fazer a Sprint de 15 dias foi algo positivo, pois possibilitou fazer ajustes necessários em meio a Sprint e ficou mais fácil de entregar no prazo.

Fonte: Do autor (2020).

Quadro 8 - Problemas identificados no segundo experimento

Sprint 1	Sprint 2
Desenvolvedores tiveram dificuldade no entendimento da regra de negócio e das tabelas; O PO precisou responsabilizar-se por muitos itens por fora, inclusive solicitações de outros GT's e direção, portanto ele não programa na Sprint, mas programa nas demais situações; Desenvolvedores com dificuldade de entender o legado;	Inúmeros conflitos na compilação de módulos, que demandam bastante tempo para resolução; A base de teste atual dificultou a rápida execução dos testes, pois as informações já existiam; O teste, por ficar a cargo dos programadores, acaba sendo mais uma revisão de código do que um teste de negócio. [Resolvido] Convida-se o líder do GT no início da Sprint para que ele faça os testes de negócio, caso ele recuse, será realizado o envio apenas com os testes unitários; Conflito dos códigos fontes, recebem-se tarefas de *bugs* críticos de um mesmo programa que possui tarefa da Sprint e já foi alterado, porém, não está pronto para ser enviado. [Resolvido] Criação de *branches*, garantindo que o código instável não seja mesclado nos arquivos do projeto principal; Dependência de funções e tarefas entre os desenvolvedores do time.

Fonte: Do autor (2020).

Quadro 9 - Pontos positivos do segundo experimento

Sprint 1	Sprint 2
A estimativa via *Planning Poker* com as cartas auxiliam os desenvolvedores a repensarem suas estimativas em alguns casos em que as cartas são apresentadas com valores diferentes, garantindo assim maior assertividade; Oportunidade de criar um *backlog* de refatoração e ajustes de código legado para os casos de ociosidade não estimados.	A alternância das tarefas de testes (um desenvolvedor programa e outro testa), possibilitou identificar situações de ajustes em programas e *bugs* que o programador não identificaria sozinho; A questão de estimativas e prazos da metodologia foi boa, e o cliente também elogiou. Pode-se entregar de acordo com o estimado no início da Sprint; As reuniões diárias aproximaram a equipe e facilitaram o controle do trabalho e dos prazos; As tarefas chegam mais “polidas” para o desenvolvimento, devido à organização do trabalho realizada antes de designar a tarefa.

Fonte: Do autor (2020).

4.7 VALIDAÇÃO

Para a validação deste *framework*, no que diz respeito ao atendimento de requisitos e ao funcionamento da metodologia, foi realizada uma pesquisa através da escala desenvolvida por Likert (1932), que consiste em desenvolver um conjunto de afirmações relacionadas à sua definição, para as quais os respondentes emitirão seu grau de concordância.

Na escala Likert, o nível de concordância com a afirmação realizada foi proposto através de cinco níveis de resposta: discordo totalmente, discordo parcialmente, indiferente, concordo parcialmente e concordo totalmente.

As afirmações que foram aplicadas aos participantes dos projetos contaram com as seguintes questões:

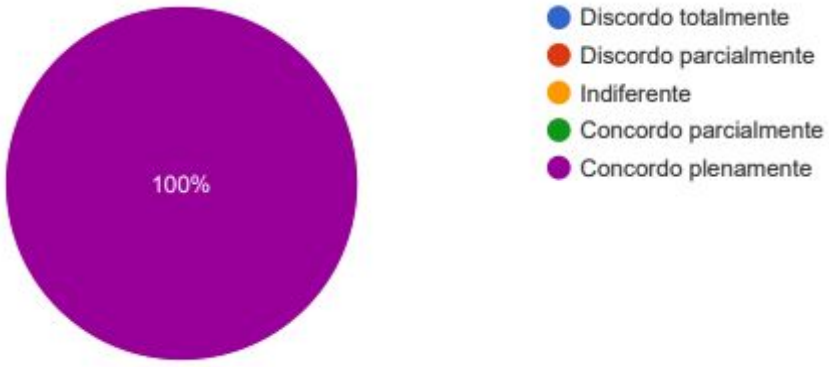
a) os requisitos do sistema foram entregues;
b) as necessidades do cliente foram atendidas;
c) estive motivado ao desenvolver este projeto através desta metodologia;

d) indicaria a utilização desta metodologia para outros projetos;
e) senti confiança ao utilizar esta metodologia;
f) esta técnica foi útil para o bom andamento do trabalho;
g) esta técnica é de fácil adaptabilidade;
h) o planejamento do trabalho demonstrou ser adequado;
i) fiquei satisfeito com o serviço que foi entregue;
j) fui ouvido ou atendido pelos demais integrantes do projeto;
k) a metodologia me auxiliou no desenvolvimento do projeto.

Seguem os resultados da pesquisa, que foi aplicada para o time e clientes que trabalharam nos dois projetos experimentados neste trabalho.

A Figura 30 representa o gráfico com o retorno da pesquisa quando o time e o cliente foram questionados a respeito da indicação e utilização da metodologia para outros projetos e se foram ouvidos pelos demais integrantes do projeto. Para ambas questões todos responderam que concordam plenamente.

Figura 30 - Afirmações que o time concordou



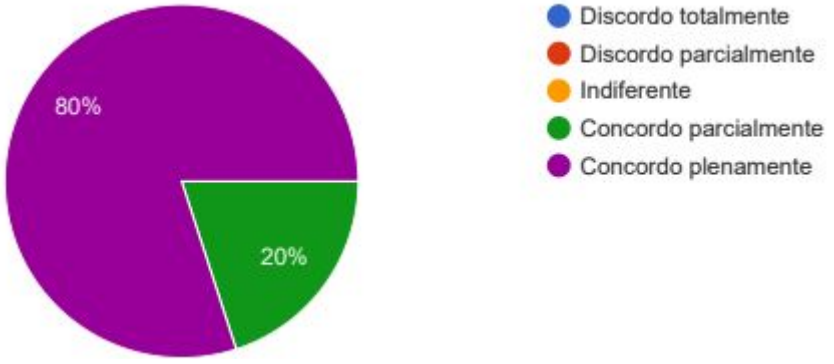
Fonte: Do autor (2020).

Na maior parte das questões aplicadas o resultado mostrou-se conforme a Figura 31, onde 80% dos respondentes concordam plenamente com a afirmação e 20% concordam parcialmente. O resultado deste gráfico foi composto através das respostas das afirmações abaixo:

a) os requisitos do sistema foram entregues;
b) as necessidades do cliente foram atendidas;

c) estive motivado ao desenvolver este projeto através desta metodologia;
d) senti confiança ao utilizar esta metodologia;
e) esta técnica foi útil para o bom andamento do trabalho;
f) o planejamento do trabalho demonstrou ser adequado;
g) fiquei satisfeito com o serviço que foi entregue;
h) a metodologia me auxiliou no desenvolvimento do projeto.

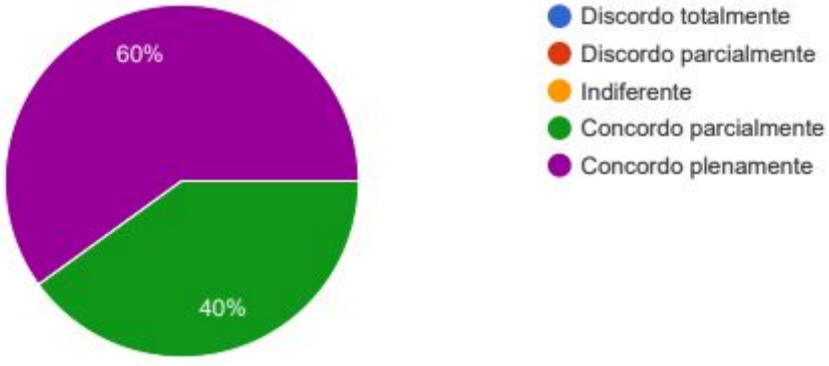
Figura 31 - Afirmações que a maior parte do time concordou



Fonte: Do autor (2020).

Quando os respondentes foram questionados a respeito da facilidade de adaptabilidade da metodologia, houve um retorno de 60% concordando plenamente com a facilidade de adaptação e 40% concordando parcialmente. A Figura 32 mostra, em gráfico de pizza, este percentual.

Figura 32 - Facilidade de adaptabilidade da metodologia



Fonte: Do autor (2020).

Para medir a assertividade das estimativas do time e analisar se o planejamento foi cumprido, utilizou-se o artefato gráfico de *burndown* com a técnica do *Planning Poker*. Este gráfico que é uma indicação da metodologia Scrum e possibilitou uma representação gráfica relacionando o que foi realizado em comparação ao que foi planejado.

Conforme apresentado nos gráficos dos dois experimentos realizados, houve uma sobra de tempo ao final das Sprints. Este resultado pode ocorrer por um dos seguintes motivos: a equipe se motivou com a metodologia e rendeu mais que o esperado, ou, o time realizou uma estimativa muito pessimista.

4.8 CONSIDERAÇÕES FINAIS

Após o desenvolvimento de algumas Sprints, sentiu-se a necessidade de montar um passo a passo, visando auxiliar, pelo menos durante os primeiros meses, a indicar as atividades mais importantes e necessárias para trabalhar com a metodologia.

O Quadro 6 lista este passo a passo, bem como a pessoa responsável, a fase do projeto em que se encontra e se este passo deve ou não ser contabilizado nas tarefas da Sprint.

Quadro 10 - Passo a passo para iniciar a implementação da metodologia

(continua)

Nº	O que	Responsável	Fase	Conta na Sprint?
1	**Descrever as histórias junto ao usuário**	PO	Planejamento	Não
2	**Quebrar as histórias em subtarefas menores**	PO/Analista	Planejamento	Não
2.1	Descrever o detalhamento em cada subtarefa, colocando anexos, *mockups,* exemplos e arquivos necessários.	PO/Analista	Planejamento	Não
2.2	Verificar dependência entre as tarefas	PO/Analista	Planejamento	Não
2.3	Verificar as necessidades ou preenchimentos obrigatórios e/ou condicionais.	PO/Analista	Planejamento	Não

(continuação)

2.4	Verificar necessidades de alterações de tabelas e criar tarefas a parte para isso	PO/Analista	Planejamento	Não
3	**Realizar reunião de iniciação e planejamento da Sprint**	Time	Planejamento	Sim
3.1	**Reunião de Iniciação**	Time	Planejamento	Sim
3.1.1	Passar tarefa por tarefa para discutir dúvidas e realizar estimativa do tempo (*Planning Poker*)	Scrum Master	Planejamento	Sim
3.1.2	Registrar a estimativa em cada tarefa	Time	Planejamento	Sim
3.1.3	Definir horário e local das reuniões diárias	Scrum Master	Planejamento	Sim
3.1.4	Verificar como serão realizados os testes: 1 - Testes de programação: através da criação de tarefas para este fim (que podem ser designadas ao desenvolver, PO, testador, ou outro, conforme a estrutura e decisão do time); 2 - Testes de negócio: criar uma coluna teste no Kanban e empilhar as tarefas para o líder do GT (especialista do negócio) testar em uma base específica.	Time	Planejamento	Sim
3.1.5	Montar o Kanban e as colunas necessárias (não iniciado/ em andamento/ documentação/ bloqueado/ testes/ concluído)	Time	Planejamento	Sim
3.1.6	Incluir tarefa de análise e as horas gastas do analista	PO/Analista	Planejamento	Não
3.2	**Reunião de planejamento da Sprint**	Time	Planejamento	Sim
3.2.1	Tarefas de monitoramento que devem ser abertas em cada Sprint 1 - Reunião de planejamento da Sprint (1 hora de reunião para cada semana de trabalho) 2 - *Daily* Scrum (máximo de 15 minutos por dia) 3 - Reunião de *review* 4 - Reunião de retrospectiva da Sprint Obs: Para as reuniões em que o time participa, multiplicar o tempo pela quantidade de desenvolvedores	Scrum Master	Planejamento	Sim
3.2.2	Realizar o planejamento da Sprint definindo primeiramente o objetivo da Sprint	Time	Planejamento	Sim
3.2.3	Dependendo de como os times estão estruturados no setor, definir um tempo diário, dentro da Sprint, para ajustes de *bugs* e de código legado.	Time	Planejamento	Não
3.2.4	Verificar possíveis folgas, feriados ou saídas que possam ocorrer com algum integrante e possa mudar a quantidade de horas da Sprint	Time	Planejamento	Sim

(continuação)

3.2.5	Determinar um tempo no início da Sprint para eventuais *bugs* da Sprint anterior (até 1 dia para Sprint quinzenal)	Time	Planejamento	Sim
3.2.6	Determinar um tempo no final da Sprint para revisões (até 1 dia para Sprint quinzenal)	Time	Planejamento	Sim
3.2.7	Calcular quantidade de horas disponíveis para desenvolvimento da Sprint multiplicando pelo número de desenvolvedores no time.	Time	Planejamento	Sim
3.2.8	Colocar as tarefas que são de maior importância ao cliente como prioritárias no *backlog*, e na sequência, as tarefas de menor complexidade (O nível de importância pode ser numerado, por exemplo, de 10 a 150, assim se alguma prioridade entrar em frente as demais, não teremos prioridades negativas, exemplo prioridade 0, -1 -2...)	PO	Planejamento	Sim
3.2.09	Definir o tempo (dias) da Sprint	Time	Planejamento	Sim
3.2.10	Atribuir as tarefas entre o time	Time	Planejamento	Sim
3.2.11	Registrar as horas da reunião	Scrum Master	Andamento	Sim
3.2.12	Iniciar a Sprint	Scrum Master	Andamento	Sim
4	**Desenvolvimento da Sprint**	Time	Andamento	Sim
4.1	Acompanhamento através das reuniões diárias e gráfico *burndown*	Scrum Master	Andamento	Não
4.2	Resolução de empecilhos e problemas	Scrum Master	Andamento	Não
4.3	Acompanhamento e solução de dúvidas	PO	Andamento	Não
5	**Revisão da Sprint**	Scrum Master	Andamento	Não
5.1	Confirmar com o time que todos os *commits* foram realizados	PO	Andamento	Não
5.2	Apresentar o resultado da Sprint para o usuário. Perguntas feitas pelo PO: O que foi feito (itens prontos)? O que não foi feito? Trabalho adicionado? Trabalho removido? Obs: Item pronto é um item desenvolvido, testado, homologado e documentado	PO	Homologação	Não
5.3	Deixar o usuário manusear as novas funcionalidades do sistema. Isso possibilita verificar se o usuário tem o comportamento esperado e/ou se há ajustes que possam auxiliar na experiência do mesmo.	PO	Homologação	Não
5.4	Receber o *feedback* do usuário	PO	Homologação	Não

(conclusão)

5.5	Decidir se serão realizados ajustes ou se o envio pode ser realizado	PO	Homologação	Não
5.6	Comentar de uma forma bem objetiva e o que será trabalhado nos próximos dias e quais serão as próximas liberações (próximo Sprint)	PO	Concluído	Não
6	**Retrospectiva da Sprint**	Scrum Master	Concluído	Não
6.1	Pontuar o que correu bem no Sprint	Time	Concluído	Não
6.2	Pontuar o que poderia ser melhorado	Time	Concluído	Não
6.3	Pontuar o que nos comprometemos a melhorar para o próximo Sprint (descrever ações)	Time	Concluído	Não
6.4	Revisar ações de melhorias das interações anteriores	Time	Concluído	Não

Fonte: Desenvolvido pelo autor (2020).

5 CONCLUSÃO

A aplicação de *Design Thinking* foi eficiente nas etapas iniciais de elicitação, prototipagem, e principalmente, de relacionamento com o cliente. No entanto, o método tem suas limitações no que diz respeito ao gerenciamento do tempo, desempenho e entregas do projeto. Para suprir estas limitações, utilizaram-se práticas e artefatos do Scrum que agregaram efeitos positivos e complementaram as fases de condução desta metodologia.

De forma geral, o *Design Thinking* foi utilizado como estratégia do projeto, contribuindo para a definição do escopo e compreensão das necessidades do usuário final, e o Scrum foi utilizado para a implementação do projeto, através da abordagem dos prazos, entregas em etapas e priorização dos trabalhos. A combinação destas abordagens ofereceu uma metodologia enxuta, focada em resultados e no entendimento das necessidades do cliente.

Para o correto funcionamento do *framework* é imprescindível que a empresa ou departamento siga e acredite nos valores do manifesto ágil, do contrário, é insuficiente implementar os ritos do Scrum, realizar a divisão de papéis, utilizar artefatos como o *Kanban* ou demais práticas indicadas pelo gerenciamento ágil.

Para este experimento realizou-se um treinamento de 30 minutos. Notou-se que este tempo foi suficiente quando ministrado a uma equipe pequena de até 4 pessoas, para treinamentos em equipes maiores, entende-se que haverá a necessidade de um tempo superior, visto as dúvidas que surgirão durante e após a explanação.

Através das ferramentas Jira Software e ActiveCollab foi possível ver, de forma clara e simplificada, o planejamento do aumento de demandas na empresa. Dentre as duas ferramentas experimentadas, o Jira Software mostrou-se mais completo e preparado para trabalhar com gerenciamento de projetos de *software*, possui a criação de versionamentos, quadro Kanban, quadro Scrum, e muitos relatórios e gráficos de acompanhamento, inclusive o gráfico de *burndown*.

Verificou-se que a estimativa do tempo via *Planning Poker* foi bem aceita pelos times e em algumas situações auxiliou a repensar no planejamento dos

tempos. Esta técnica, utilizada junto do gráfico de *burndown*, pôde medir o progresso em relação às tarefas e assim garantir maior assertividade no planejamento do projeto.

Outro item importante a ser ressaltado é que a Sprint quinzenal demonstrou maior facilidade de planejamento e menor pressão de trabalho ao time, concluindo-se assim que este tempo de 15 dias é mais adequado para o método.

Durante o desenvolvimento das tarefas, o fato de passar a responsabilidade de testes para um programador que não tenha desenvolvido a solução, ajudou na identificação precoce de erros que antes não seriam encontrados, resultando num ganho de qualidade no software entregue ao cliente.

O aumento na concentração e no foco foi identificado através das etapas de análise e nos relatos das reuniões diárias. Com as tarefas chegando bem analisadas aos programadores e com a aproximação da equipe, houve maior controle da execução do projeto, cumprimento dos prazos e aumento de produtividade.

Durante a execução do experimento identificaram-se dificuldades que deverão ser sanadas para uma expansão bem sucedida do método. As sugestões de resolução para cada tópico identificado são:

a) Utilização do versionamento de código com *branches* - O lançamento de uma versão do programa não pode bloquear a correção de erros. Com a utilização dos *branches*, cada equipe trabalharia numa ramificação independente, diminuindo a ocorrência de conflitos e a possibilidade de atrasos.
b) Base de dados de testes - Sugere-se automatizar o procedimento de atualização da base de dados de testes, de forma que a atualização ocorra sempre no mesmo dia do mês, para que a carga dos dados não atrapalhe os testes que estão em execução.
c) Usar a lista de projetos atuais como *backlog* - Todos as tarefas devem ser detalhadas na inclusão, os que não forem detalhados perdem prioridade no *backlog*.
d) Aumentar o número de responsáveis pela alteração de tabelas - Hoje apenas uma pessoa realiza esta tarefa, sugere-se indicar uma pessoa por time,

aumentando a autonomia da equipe e diminuindo os gargalos durante a execução.

e) Realizar uma reestruturação do setor - Será necessário montar os times, definir os produtos que cada um ficará responsável, definir os papéis de cada integrante e separar o suporte ao usuário do desenvolvimento.

Ao final do experimento, através da pesquisa aplicada, verificou-se que 80% dos respondentes sentiram-se motivados em utilizar a metodologia proposta. O aumento na motivação dos programadores foi um dos objetivos pelo qual se optou pelo uso de uma metodologia ágil.

Com a pesquisa foi possível identificar que houve um bom entendimento da complexidade dos projetos, isso auxiliou na assertividade da estimativa de prazos e automaticamente fez com que a satisfação dos clientes aumentasse. Estes aspectos também possibilitarão realizar a estimativa de custos do projeto, quando for necessário.

De forma geral, a contribuição deste trabalho foi de extrema importância para a empresa rever conceitos, métodos de trabalho, e até mesmo tópicos ligados a engenharia de software. A equipe inicial deste trabalho mostrou-se engajada e receptiva a melhorias e o resultado do primeiro experimento estimulou os gestores do setor a expandir e aplicar o método proposto.

Através da oportunidade de expansão, realizou-se o segundo experimento para uma equipe ainda maior. Esta nova equipe demonstrou uma boa maturidade em relação à busca de resultados, e o método foi de fato aplicado para o time, o qual vem trabalhando na busca de melhoria contínua.

Com as contribuições deste trabalho, a empresa está buscando adaptar-se às sugestões do método proposto, visando realizar a expansão da metodologia aos demais times, auxiliando não somente no gerenciamento de projetos como no controle e no gerenciamento do setor em sua totalidade.

REFERÊNCIAS

ANDRADE, Antônio José F. *et al*. **Gestão de Projeto com Scrum:** Um Estudo de Caso. Aracati, CE: Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), 2011. Disponível em: https://www.enucomp.com.br/2012/conteudos/artigos/Scrum.pdf. Acesso em: 10 abr. 2020.

BROWN, Tim. Design Thinking: uma metodologia poderosa para decretar o fim das velhas ideias. Rio de Janeiro. Elsevier, 2010.

CAMARGO, Robson; RIBAS Thomaz. **Gestão ágil de projetos**. 1. ed. São Paulo, SP: Saraiva Educação, 2019.

COHN, Mike. **Desenvolvimento de software com Scrum:** aplicando métodos ágeis com sucesso. 1. ed. Porto Alegre, RS: Bookman, 2011.

FERRARI, Alfonso Trujillo. **Metodologia da ciência**. 2. ed. Rio de Janeiro: Kennedy, 1974. Cap. 2.

FOGGETTI, Cristiano. **Gestão ágil de projetos**. São Paulo, SP: Pearson Education do Brasil, 2014.

FUNDAÇÃO INSTITUTO DE ADMINISTRAÇÃO**. Design Thinking:** O que é, como aplicar e passo a passo. São Paulo,12 jul. 2018. Disponível em: https://fia.com.br/blog/design-thinking/. Acesso em: 18 abr. 2020.

GALLOTTI, Giacondo Marino Antonio. **Arquitetura de software**. São Paulo, SP: Pearson Education do Brasil, 2016.

GIL, Antonio Carlos. **Métodos e técnicas de pesquisa social**. 6. ed. São Paulo: Atlas, 2008.

KERZNER, Harold R. **Gerenciamento de projetos:** uma abordagem sistêmica para planejamento, programação e controle. 2. ed. New York, NY: Edgard Blücher, 2015.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade**. Fundamentos de metodologia científica**. 8. ed. São Paulo, SP: Atlas, 2017.

LIKERT, Rensis**. A Technique for the Measurement of Attitudes**. 1. ed. New York, NY: 1932.

MAKIOSZEK, Anderson Andellon. **Organização, sistemas e métodos (OSM) e design organizacional:** novas práticas. Curitiba, PR: Intersaberes, 2019.

MASSARI, Vitor L. **Gerenciamento Ágil de Projetos.** 8. ed. São Paulo, SP: Brasport, 2018.

MORAES, Iur Cristine *et al*. Design Thinking Transformando a Cultura Organizacional. In: CONGRESSO INTERNACIONAL DE CONHECIMENTO E INOVAÇÃO, 2019, Porto Alegre. Anais [...]. Porto Alegre: Ciki, 2019. Disponível em: https://www.researchgate.net/publication/337090054_DESIGN_THINKING_TRANSFORMANDO_A_CULTURA_ORGANIZACIONAL. Acesso em: 12 jun. 2020.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software:** uma abordagem profissional. 8. ed. Porto Alegre, RS: AMGH, 2016.

PROJECT MANAGEMENT INSTITUTE. **Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK)**. 6 ed. Newtown Square, PA, 2017.

PROJECT MANAGEMENT INSTITUTE. **Agile Practice Guide**. 6 ed. Newtown Square, PA, 2017.

SCHWABER, Ken; e SUTHERLAND, Jeff. **Guia do Scrum**. Estados Unidos: Creative Commons, 2017. Disponível em: https://www.Scrumguides.org/docs/Scrumguide/v2017/2017-Scrum-Guide-Portuguese-European.pdf

VIANNA, Maurício. *et al.* **Design thinking:** inovação em negócios. 2. ed. Rio de Janeiro, RJ: MJV Press, 2018.

XAVIER, Carlos Magno da Silva, **Gerenciamento de projetos:** como definir e controlar o escopo do projeto. 3. ed. São Paulo, SP: Saraiva, 2016.