UNIVERSIDADE FEDERAL DE OURO PRETO – UFOP

ESCOLA DE MINAS

DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO

JOÃO LUCAS MONTEIRO SAMPAIO SILVA

ANALISAR A FORMAÇÃO DE TIMES ÁGEIS PARA O DESENVOLVIMENTO DE SOFTWARE EM UMA EMPRESA

OURO PRETO - MG
2023

JOÃO LUCAS MONTEIRO SAMPAIO SILVA
joaolucasmon@gmail.com

ANALISAR A FORMAÇÃO DE TIMES ÁGEIS PARA O DESENVOLVIMENTO DE SOFTWARE EM UMA EMPRESA

Monografia apresentada ao Curso de Graduação em Engenharia de Produção da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do título de Engenheiro de Produção.

Orientador: Prof. Dr. Cristiano Luís Turbino de França e Silva

OURO PRETO – MG
2023

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA DE PRODUCAO, ADMINISTRACAO E ECON

FOLHA DE APROVAÇÃO

Monografia João Lucas Monteiro Sampaio Silva

Analisar a formação de times ágeis para o desenvolvimento de software em uma empresa

Monografia apresentada ao Curso de Engenharia de Produção da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Engenheiro de Produção

Aprovada em 27 de julho de 2023

Membros da banca

Mestre - Cristiano Luís Turbino de França e Silva - Orientador Universidade Federal de Ouro Preto
Doutora - Irce Fernandes Gomes Guimarães - Universidade Federal de Ouro Preto
Mestre - Pedro Saint Clair Garcia - Universidade Federal de Ouro Preto

Cristiano Luís Turbino de França e Silva, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 27/07/2023

Documento assinado eletronicamente por **Cristiano Luis Turbino de Franca e Silva**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 27/07/2023, às 20:45, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0563862** e o código CRC **CBED51C2**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591540 - www.ufop.br

Dedico este trabalho a toda minha família em especial meu pai Lair, minha mãe Luzia e meu irmão José

Dedico a todos meus amigos e pessoas que morei e estudei em Ouro Preto.

Por fim dedico a meu orientador Cristiano pela dedicação e paciência para me ajudar a concluir este projeto.

AGRADECIMENTO

Gostaria de agradecer primeiramente minha família por todo apoio e dedicação. A Tigrada por todo acolhimento. A todos os tigres que se tornaram minha família para toda a vida. Todos os amigos que fiz nessa caminhada, sem eles também não conseguiria chegar aqui. Ao CAEPRO que foi uma experiência de grande crescimento. A meu orientador por fazer com que essa pesquisa fosse possível se realizada com sucesso. E a UFOP pelo ensino de qualidade e gratuito.

A única segurança verdadeira na vida provém de saber que a cada dia você melhora de alguma maneira.

Anthony Robbins

RESUMO

A pesquisa abordou a formação de times ágeis para o desenvolvimento de software em uma empresa de médio porte, envolvendo uma revisão bibliográfica sobre formação de times ágeis, metodologias como Scrum e Kanban, e a entrega de valor no desenvolvimento de software. Um estudo de caso acompanhou um time em formação nos três primeiros ciclos de desenvolvimento, totalizando 45 dias. A metodologia aplicada foi quantitativa e explicativa, buscando identificar fatores para melhorar os resultados de times ágeis em formação. Os resultados incluíram análise de produtividade, coleta de dados de desempenho e tempo de entrega, recomendações para melhorias e planos de ação implementados com treinamentos e ajustes nas práticas de desenvolvimento. O estudo destacou a importância da formação de times ágeis e metodologias ágeis para o sucesso no desenvolvimento de software em empresas de médio porte.

Palavras-chave: Ágil, *Scrum, Squad, Scrum Master (SM), Product manager (Pm), User Story (Us)*

ABSTRACT

The research addressed the formation of agile teams for software development in a medium-sized company, involving a literature review on agile team formation, methodologies such as Scrum and Kanban, and value delivery in software development. A case study followed a team in formation through the first three development cycles, totaling 45 days. The applied methodology was quantitative and explanatory, aiming to identify factors to improve the results of agile teams in formation. The results included productivity analysis, collection of performance and delivery time data, recommendations for improvements, and action plans implemented through training and adjustments in development practices. The study highlighted the importance of agile team formation and agile methodologies for success in software development in medium-sized companies.

Keywords: *Agile, Scrum, Squad, Scrum Master (SM), Product Manager (PM), User Story (US)*

LISTA DE FIGURAS

LISTA DE SIGLAS

Agile	-	Metodologia que enfatiza a flexibilidade e a colaboração.
DoD	-	Definition of Done (Definição de Pronto).
Engineering Manager	-	Gerente de Engenharia.
Head of Product	-	Líder de Produto.
Jira	-	Software de gerenciamento de projetos.
Kanban	-	Método ágil para gerenciamento de fluxo de trabalho.
PM	-	Product Manager (Gerente de Produto).
Scrum	-	Framework ágil utilizado para gerenciamento de projetos.
Sprint Planning	-	Planejamento da Sprint.
Sprint Review	-	Revisão da Sprint.
Sprint Retrospective	-	Retrospectiva da Sprint.
Tech Leaders	-	Líderes Técnicos.
Team Canvas	-	Ferramenta para alinhar a equipe em relação a objetivos e valores.
User Stories (Us)	-	Histórias de Usuário.

SUMÁRIO

1 INTRODUÇÃO

Atualmente, há um interesse por parte de startups a utilização de metodologias ágeis, como os *frameworks*, que servem como estruturas ou guias para auxiliar equipes e organizações a implementarem metodologias de trabalho de forma mais eficiente e eficaz. Promovendo assim, ganho de visibilidade e escaladas para a empresa. Este ambiente desenvolveu a curiosidade e mostrou um vasto campo de estudo. Pretende-se desenvolver um estudo de caso mostrando uma experiência vivida sobre a formação de times ágeis de desenvolvimento.

Pontuando o papel de um agilista nos times de desenvolvimento de *software*, segundo experiência na área e em discussões com outros profissionais, o agilista é uma pessoa que tem o papel de ajudar organizações a implementar práticas ágeis para melhorar seus processos de trabalho. Ele tem conhecimento e experiência em métodos ágeis, como *Scrum* e *Kanban*, e pode ajudar as equipes a implementarem de maneira eficaz. Além disso, o agilista também atua como mentor e facilitador, ajudando a equipe a trabalhar de forma mais colaborativa e a alcançar seus objetivos de maneira mais eficiente.

O agilista também tem a responsabilidade de garantir que as práticas ágeis sejam aplicadas de maneira consistente e que sejam ajustadas às necessidades específicas de cada organização. Ele trabalha em estreita colaboração com as equipes e garante que todos estejam alinhados aos princípios ágeis. Ademais, o agilista também promove a cultura ágil na organização, ajudando a torná-la mais inovadora e adaptável à mudança.

Segundo (VARELA ARRUDA, 2012), “a maioria dos sistemas desenvolvidos hoje, no final do processo, não tem todas suas funcionalidades em uso ou sequer entraram em produção”. Por tanto, tais características demonstram um profissional que pode facilitar todo o processo de desenvolvimento de *software*. De forma que, através da automatização das atividades, esse processo seja aperfeiçoado.

Times de desenvolvimento de *software* ágeis são equipes que seguem uma abordagem ágil para o desenvolvimento de *software*. A formação desses times é feita pela avaliação de características que trabalhem de maneira colaborativa e iterativa, entregando pequenas partes do *software* regularmente. Os times ágeis seguem princípios como entrega contínua, comunicação aberta e foco no valor para o cliente, o

que permite que eles sejam mais flexíveis e adaptáveis às mudanças no mercado e nas necessidades do cliente.

Os times ágeis também valorizam a auto-organização e a autossuficiência, permitindo que as equipes tomem decisões de maneira independente e trabalhem de maneira mais eficiente. Eles também são mais transparentes, com reuniões regulares e processos claros para garantir que todos estejam alinhados aos objetivos da equipe. Isso leva a um ambiente de trabalho mais colaborativo e engajado, o que pode resultar em *software* de alta qualidade e entregue de maneira mais rápida.

Criar times de desenvolvimento de *software* ágeis pode ser um desafio, pois requer mudanças significativas na cultura e nas práticas de trabalho. Muitas vezes, as equipes enfrentam resistência à mudança, especialmente de membros da equipe que são acostumados a trabalhar de maneira tradicional. Além disso, é preciso garantir que todos os membros da equipe estejam familiarizados com os princípios ágeis e saibam como aplicá-los em seu trabalho diário.

Outra dificuldade é garantir que as equipes tenham realmente auto-organização e autossuficiência. Isso requer que as equipes tenham claras as suas responsabilidades e objetivos, e que trabalhem de maneira colaborativa para alcançá-los. Além disso, é importante que as equipes tenham as ferramentas e recursos necessários para realizar seu trabalho de maneira eficiente.

Finalmente, o sucesso dos times de desenvolvimento de *software* ágeis depende de uma cultura de apoio e de uma liderança comprometida. A liderança precisa acreditar nas práticas ágeis e apoiá-las de maneira consistente, além de ajudar a equipe a superar desafios e fornecer recursos e suporte quando necessário. Sem esse apoio, pode ser difícil para as equipes adotarem práticas ágeis de maneira eficaz e sustentável.

Como o estudo e desenvolvimento da análise de formação de times ágeis de desenvolvimento de *software* pode melhorar e contribuir para a otimização de criação de times com a cultura ágil?

O estudo e desenvolvimento da análise de formação de times ágeis de desenvolvimento de *software,* pode ajudar a entender os fatores críticos para o sucesso desses times. Ao identificar as práticas e características que contribuem para a formação de times ágeis eficazes, é possível melhorar a criação de novos times e aumentar a probabilidade de sucesso. Além disso, essa análise pode fornecer *insights*

valiosos sobre como as equipes podem evoluir e melhorar continuamente ao longo do tempo.

Outra forma pela qual o estudo da formação de times ágeis de desenvolvimento de *software* pode melhorar a otimização da criação de times é ajudando a definir as competências e habilidades necessárias para os membros da equipe. Isso permite que a empresa selecione as pessoas certas para o time e garanta que as equipes tenham os recursos e habilidades necessárias para alcançar seus objetivos. Além disso, esse estudo pode ajudar a identificar as melhores práticas para treinar e desenvolver os membros da equipe, garantindo que eles estejam sempre atualizados e preparados para trabalhar de maneira eficaz.

Por fim, o estudo da formação de times ágeis e utilização de metodologias ágeis de desenvolvimento de *software*, como os *frameworks Scrum* e *Kanban*, podem contribuir para a otimização da criação de equipes, ao ajudar a promover uma cultura ágil. Ao compreender a importância da cultura ágil e como ela influencia o sucesso dos times, é possível criar uma cultura organizacional que apoie e valorize as práticas ágeis. Isso pode ajudar a garantir que as equipes trabalhem de maneira eficaz e que a organização esteja preparada para lidar com mudanças e desafios futuros. Em resumo, o estudo da formação de times ágeis de desenvolvimento de *software* pode ser uma peça-chave para o sucesso da criação de times e da cultura ágil na organização.

O objetivo geral é avaliar e aprimorar a formação de times ágeis em uma empresa de porte médio, para garantir eficiência e qualidade no desenvolvimento de *software*.

Os objetivos específicos foram:

- Realizar a revisão de conceitos de metodologias ágeis para a criação de times dentro dos *frameworks* utilizados pela empresa estudada.
- Avaliar a aplicação do planejamento feito para a formação dos times que se adaptem aos *frameworks* que melhor encaixam a cultura da organização empresarial.
- Realizar um estudo de caso acerca da formação de times ágeis de desenvolvimento de *softwares* demostrando a eficácia da metodologia aplicada do time com os primeiros resultados dos seus ciclos avaliativos de desenvolvimento.

No primeiro capítulo faz-se uma breve introdução acerca da formação de times ágeis, contendo a formulação do problema, a justificativa, bem como a estrutura do trabalho. O segundo capítulo refere-se a uma revisão bibliográfica sobreo desenvolvimento de *softwares*, entrega de valor e *frameworks* ágeis. O terceiro capitulo refere-se a metodologia. Já no quarto capítulo realizou-se um estudo de caso referente ao acompanhamento do time desde a decisão da sua formação, até o final da terceira *sprint*. Por fim, no quinto capítulo é conferida a conclusão deste trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 Desenvolvimento de Software

Segundo Sommerville (2011), um processo de desenvolvimento de software é uma série de atividades estruturadas e relacionadas entre si, com o objetivo de construir um artefato de software de alta qualidade. Valente (2020) acrescenta que no mundo moderno, tudo é software e explica que quando aplicado a engenharia, utiliza uma abordagem sistêmica, disciplinada e quantificáveis. Esses processos são considerados essenciais para cumprir com os contratos do desenvolvimento e garantir a qualidade do software produzido. Eles são estudados para aprimorar a maneira como as atividades são conduzidas e, assim, melhorar os resultados alcançados.

Há diversos processos de desenvolvimento, contudo todos devem possuir quatro atividades fundamentais para a Engenharia de *Software* (SOMMERVILLE, 2011):

- Especificação: Definição das funcionalidades e restrições do *software*.
- Projeto e implementação de software: Produção do *software* para atender a especificação.
- Validação de software: Garantia do atendimento às necessidades para as quais foi construído.
- Evolução de software: Capacidade de evoluir para atender novas demandas.

Um pilar adicional importante nas atividades fundamentais para a engenharia de software são os testes. Segundo Valente (2020), é fundamental introduzir atividades de teste em projetos de desenvolvimento de software, para evitar que tais erros cheguem aos usuários finais e causem prejuízos de valor incalculável. Essa etapa se torna essencial em metodologias ágeis, anterior ao processo de validação.

Ainda como diz Sommerville (2011), todas as atividades relacionadas ao desenvolvimento de software estão presentes de alguma forma em todos os processos. No entanto, esses processos são complexos e, como todos os processos intelectuais e criativos, são dependentes do fator humano e, portanto, suscetíveis a julgamentos subjetivos.

Diante disso, os processos, além de buscarem aderência aos requisitos da Engenharia de Software, também evoluíram no sentido de se adaptarem às condições do ambiente ao qual estão expostos e tirarem proveito dos recursos disponíveis. Logo, é possível compreender que as mudanças nos processos de *software* estão intimamente ligadas às mudanças da Tecnologia da Informação, da mentalidade das pessoas que trabalham com ela e do mundo (SOMMERVILLE, 2011).

2.1.1 Entregando valor

Nesta nova perspectiva, conceitos relacionados ao valor na visão do cliente emergem de maneira a representar uma orientação que parece estar cada vez mais presente nas organizações. A “jornada do cliente”, a “experiência do cliente” de projetos, os aspectos “comportamentais” na gestão de projetos, a “cocriarão de valor” (BIZARRIAS; PENHA; SILVA, 2021).

Como analisam Bizarrias, Penha, Silva (2021), quando falamos em entregas de valor, estamos entregando valor ao cliente, ele direciona o *roadmap* e estratégias das empresas de desenvolvimento de software consideradas ágeis.

O valor da entrega feita em cada etapa deve ser mensurado na perspectiva do cliente do projeto, no seu papel de usuário final do produto desenvolvido. (BIZARRIAS; PENHA; SILVA, 2021)

Ainda segundo a análise de Bizarrias, Penha, Silva (BIZARRIAS; PENHA; SILVA, 2021), o cliente final acompanha todo o processo como um consultor externo, ajuda o time a direcionar esforços para entregar corretamente o valor esperado e continuar a manter a empresa competitiva.

2.2 Metodologias Ágeis

Provocadas pelo descontentamento com o uso destas maneiras antiquadas de criar software, tornaram-se mais frequentes as discussões sobre novos ‘métodos ágeis’, que possibilitam às equipes de desenvolvimento concentração no software e não em sua concepção ou registros em documentos (SOMMERVILLE, 2011).

A essência da agilidade é lançar sobre o desenvolvimento de software um novo olhar, orientado a adaptação, habilidades comunicativas e capacidade de ofertar novos produtos e serviços de valor ao mercado em breves períodos (KOSCHEVIC, 2001).

As metodologias ágeis são abordagens para o desenvolvimento de produtos que estão alinhadas com os valores e princípios descritos no Manifesto Ágil para Desenvolvimento de Software, assinado em 2001 em Utah. O Manifesto ágil trás valores e princípios que as metodologias devem se basear para nos afastarmos dos antigos modelos.

Os valores são:

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

Os doze princípios são:

- Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado;
- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente;
- Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo;
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto;
- Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho;
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face;
- Software funcionando é a medida primária de progresso;
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente;
- Contínua atenção à excelência técnica e bom design aumenta a agilidade;
- Simplicidade a arte de maximizar a quantidade de trabalho não realizado é essencial;
- As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis;
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

De acordo com APPELO, 2011, o Manifesto Ágil não foi apenas uma reação à burocracia das abordagens de desenvolvimento consideradas mais formais, foi também um posicionamento contra os programadores indisciplinados, processos caóticos e produtos de baixa qualidade, que dominavam o mundo dos softwares que eram desenvolvidos para demandas pontuais e específicas. Os autores buscavam difundir a ideia de que existia um meio-termo entre a estrutura e a falta dela, um ponto mediano entre a ordem e o caos.

Como a mentalidade ágil está sempre pronta para o inesperado, sendo continuamente movida por mudanças, é interessante adotar ferramentas e sistemas, como o *Scrum* e *Kanban*, que auxiliem a administração dos projetos. Assim, é possível promover a integração das informações relevantes e a organização de atividades, custos, riscos, entre outras questões.

2.2.1 Scrum

O Scrum é um framework leve que ajuda pessoas, equipes e organizações a gerar valor por meio de soluções adaptativas para problemas complexos.(SCHWABER; SUTHERLAND, 2020)

Segundo SCHWABER; SUTHERLAND (2020) ,o Scrum é uma metodologia simples que pode ser experimentada para ver se sua filosofia, teoria e estrutura ajudam a alcançar objetivos e criar valor. A estrutura do *Scrum* é intencionalmente incompleta e define apenas as partes essenciais para implementar a teoria do *Scrum*. O processo do Scrum é criado pela inteligência coletiva das pessoas que o utilizam. Ao invés de fornecer instruções detalhadas, as regras do *Scrum* orientam as relações e interações entre as pessoas.

O *Scrum* é baseado no empirismo e no pensamento enxuto. O empirismo afirma que o conhecimento vem da experiência e da tomada de decisões com base no que é observado. O pensamento enxuto reduz o desperdício e se concentra no essencial. (SCHWABER; SUTHERLAND, 2020)

Como orientado no Guia *Scrum* de SCHWABER; SUTHERLAND (2020) a equipe é composta por três papeis:

- O *Scrum Master*, que é responsável por estabelecer o *Scrum* conforme definido no Guia do Scrum. Eles fazem isso ajudando todos a entender

a teoria e a prática do Scrum, tanto dentro do Time *Scrum* quanto na organização. Além disso ele também é responsável pela eficácia do Time Scrum. Eles fazem isso permitindo que o Time *Scrum* melhore suas práticas, dentro do framework Scrum.

- O *Product Owner,* que é responsável por maximizar o valor do produto resultante do trabalho do Time Scrum. A forma como isso é feito pode variar amplamente entre organizações, equipes Scrum e indivíduos. O *Product Owner* também é responsável pelo gerenciamento eficaz do *Product Backlog*
- *Developers* são as pessoas do Time *Scrum* que estão comprometidas em criar qualquer aspecto de um Incremento utilizável a cada Sprint. As habilidades específicas exigidas pelos desenvolvedores geralmente são amplas e variam de acordo com o domínio do trabalho

O Scrum, por ser uma metodologia prescritiva, tem papeis e rituais bem definidos. Ela também possui os seus artefatos que segundo SCHWABER; SUTHERLAND (2020), representam o trabalho ou valor entregue pelo Time *Scrum*, e eles são projetados para maximizar a transparência das principais informações. Os artefatos são:

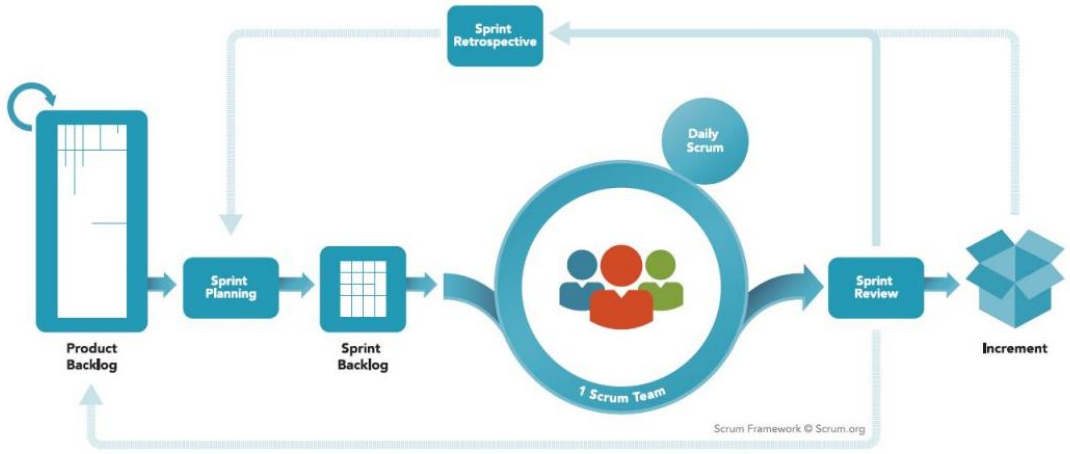
- O *Product Backlog,* que é uma lista emergente e ordenada do que é necessário para melhorar o produto. É a única fonte de trabalho realizada pelo Time *Scrum*.
- O *Sprint Backlog,* que é um plano feito por e para os desenvolvedores. É uma imagem altamente visível e em tempo real do trabalho que os desenvolvedores planejam realizar durante o Sprint para atingir o objetivo do Sprint
- O Incremento, que é um trampolim concreto em direção à meta do produto. Cada incremento é aditivo a todos os incrementos anteriores e completamente verificado, garantindo que todos os incrementos funcionem juntos. Para fornecer valor, o incremento deve ser utilizável.

O outro ponto importante a ressaltar sobre o *Scrum*, são seus ritos em todo o momento durante o processo. SCHWABER; SUTHERLAND (2020), falam sobre encontros que o time deve realizar e seus objetivos, para que o time mantenha o foco e continue em um processo ágil.

- O Daily Scrum é um evento de 15 minutos para os Desenvolvedores do Time Scrum. Para reduzir a complexidade, ela é realizada no mesmo horário e local todos os dias úteis da Sprint.
- O objetivo da Sprint Review é inspecionar o resultado do Sprint e determinar futuras adaptações. A Equipe Scrum apresenta os resultados de seu trabalho aos principais interessados e o progresso em direção à Meta do Produto é discutido.
- O objetivo da Retrospectiva da Sprint é planejar maneiras de aumentar a qualidade e a eficácia.
- O Planejamento do *Sprint* inicia o *Sprint* estabelecendo o trabalho a ser executado para o *Sprint*. Este plano resultante é criado pelo trabalho colaborativo de todo o Time *Scrum*.

Observa-se na Figura 1 o funcionamento geral da Scrum

Figura 1: Framework Scrum.



Fonte:(SCRUM, 2023).

2.2.2 Kanban

O *Kanban*, ou mais precisamente o “Sistema *Kanban* para desenvolvimento de *software*”, representa uma implementação mais direta dos princípios de desenvolvimento Lean de produtos para o desenvolvimento de software que os métodos tradicionais. Com foco consistente no fluxo e contexto, o Kanban oferece uma abordagem menos prescritiva comparada ao *Agile* do *Scrum* e *XP* (BOEG et al., 2010).

Existem dois conceitos predominantes quando se trata de ritmo de produção: Produção Puxada e Produção Empurrada. A Produção Empurrada é o método mais antigo, onde o produtor tenta antecipar as demandas do mercado. Esse método é aplicado pelo fordismo. Um exemplo de Produção Empurrada é a indústria da moda, que produz várias coleções ao longo do ano, esperando que o mercado as absorva. O objetivo é "empurrar" o produto até as prateleiras, na esperança de que o cliente o compre por impulso. No entanto, esse modelo pode resultar em superprodução, já que a demanda antecipada pode não se concretizar, gerando custos com estoque e desperdício de mercadorias (SOUZA, 2021).

As vantagens encontradas para adotar o *Kanban* são grandes. A equipe pode fazer entregas a qualquer instante para o cliente e ele pode modificar a importância das atividades quando desejar. O desenvolvimento fica mais transparente, já que é possível visualizar o fluxo de trabalho sem preocupações com as iterações e estimativas, como em outros métodos. Caso isso seja relevante para um projeto, no qual o foco vai estar na implementação do produto ou que tenha a necessidade de ter papéis bem definidos, não é indicada a aplicação do *Kanban*.(VARELA ARRUDA, 2012).

Ainda segundo Valela Arruda (2012), outro ganho é a gestão visual do quadro *kanban*, para acompanharmos o andamento de todas as atividades da equipe, possíveis impedimentos.

Como citado por BOEG et al., ([s.d.]), a metodologia Kanban trabalha com o princípio de fluxo contínuo e limitação de *WIP (Work in Progress)*. O time de desenvolvimento tem de realizar as entregas enquanto time antes de iniciar uma nova demanda.

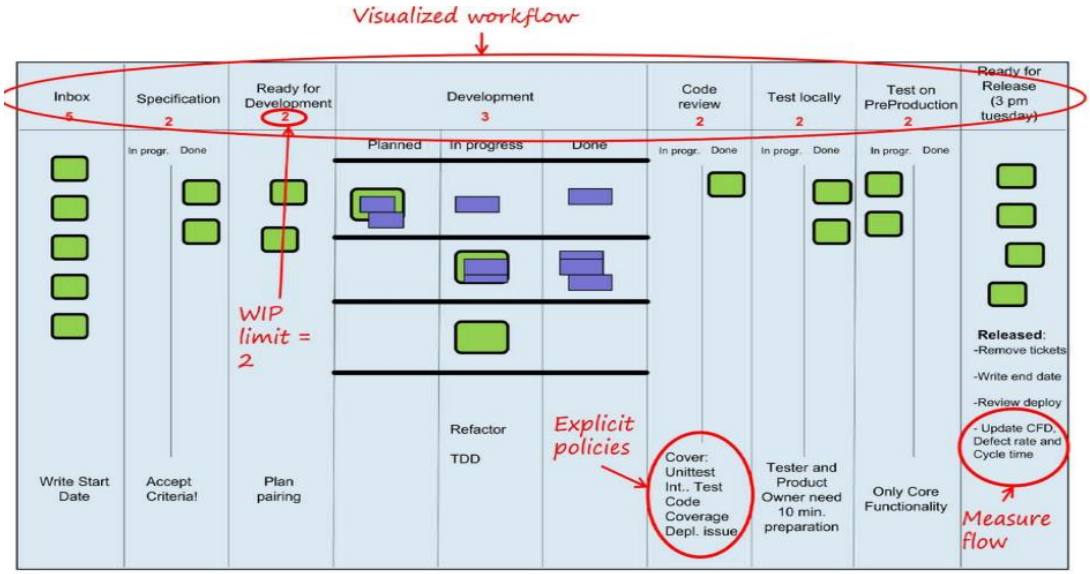
Outro conceito do *Kanban* são as classes de serviço. As demandas a serem feitas primeiro no *Kanban*, são as que estiverem na parte superior do quadro. Mas como elas foram priorizadas? Existe alguma tão urgente que me permite ultrapassar meu *Wip*? A essa classificação de urgência e o modo como tratar cada tipo de demanda, chamamos de classe de serviço. Ela deve estar alinhada com as políticas da empresa e com os demandantes do time de desenvolvimento.

Segundo BOEG et al., ([2010]), o *Kanban*, nos oferece uma visão simplificada do fluxo e da forma de se lidar com os gargalos, ajuda a entender a importância de ver o sistema como um todo, e de aplicar esforços onde geram mais valor.

Como os rituais do Scrum, o *Kanban* possui as suas cadências, uma serie de reuniões para acompanhamento e monitoramento. Mas diferente do Scrum elas não são obrigatórias, caso o time sinta a necessidade deve realiza-las.

Observa-se na Figura 2 o *board Kanban:*

Figura 2: Funcionamento do Kanban.



Fonte (BOEG et al., 2010, p. 6)

3 METODOLOGIA

Devido ao objetivo ser a aplicação prática visando solucionar o problema da empresa que é objeto de estudo, a natureza é de metodologia aplicada. Será adotada uma abordagem quantitativa, na qual serão observados dados de produtividade do time de desenvolvimento no início de sua formação e serão exploradas possibilidades para melhorar os resultados. O tipo de pesquisa abordada neste trabalho é explicativa, buscando identificar os fatores que determinam ou contribuem para a melhoria dos resultados de times ágeis em formação.

Durante o período de setembro à novembro de 2022 foi realizado o aprimorando de uma *feature* para desenvolver um *software* já existente na empresa. Todas as sugestões foram levantadas e avaliadas pelo time, buscando o ideal de melhoria contínua, evolução e agilidade.

Primeiramente, realizou-se uma revisão da literatura, buscando fontes confiáveis e atualizadas sobre formação de times ágeis para o desenvolvimento de software, incluindo princípios e boas práticas, utilizando-se de artigos científicos e base de dados disponíveis na literatura atual.

Posteriormente fez-se um estudo de caso em uma empresa de médio porte do setor de tecnologia, acerca da formação de times ágeis para o desenvolvimento de software, onde foram coletados dados em forma de reunião com o líder da equipe, gerente de projeto e outros envolvidos no processo de desenvolvimento de *software* na empresa. Além disso, coletou-se dados quantitativos sobre o desempenho dos projetos e o tempo de entrega dos produtos durante o período de setembro a novembro de 2022. Por fim, analisou-se os resultados obtidos da análise, incluindo recomendações para melhorias na formação de times ágeis.

Foi realizado um comparativo dos dados obtidos com a empresa e os padrões levantados na revisão da literatura, onde foram criados planos de ação para a melhoria dos resultados da equipe de desenvolvimento de *software*.

Em conjunto com a empresa estudada, foi realizada a implementação de melhorias sugeridas identificadas na análise, incluindo treinamentos e ajustes nas práticas de desenvolvimento de *software*.

4 ESTUDO DE CASO

Foi realizado um estudo de caso com um time de desenvolvimento que está se formando para trabalhar em um projeto de software para incremento de *features* em um dos produtos da empresa, em uma empresa de médio porte que desenvolve software e que já desenvolve outros produtos.

O acompanhamento foi realizado em um único projeto, determinado e planejado pelo time estratégico da empresa que visava atender clientes entregando uma melhoria que seria de maior valor para o cliente. A equipe apresentada trabalhou junto durante todo desenvolvimento do projeto e foi acompanhada pelos 45 dias. Após o estudo a equipe continuou o projeto, por tanto foi acompanhado somente o início do desenvolvimento do projeto. O período de 45 dias foi escolhido pois representa as 3 sprint iniciais, onde cada sprint teve duração de 2 semanas. Na literatura é falado que é indicado ter a duração de 2 a 4 semanas para cada sprint, o time optou por 2 semanas.

Líderes técnicos e líderes de produtos tomaram a decisão de criar esse time em acordo com a alta gerência. Foi uma decisão estratégica visando a evolução dos produtos em que o time focará e uma oportunidade dos desenvolvedores que têm uma baixa senioridade mostrarem seu potencial.

Com essas decisões em mãos, foi realizada uma reunião estratégica com participação das pessoas chaves para o funcionamento do time. Onde foram alinhados as expectativas e os primeiros passos.

Após a reunião citada acima, foi iniciado o trabalho do time e foi realizado o acompanhamento do mesmo durante o tempo de três *Sprints*, 45 dias, onde será possível acompanhar a aplicação dos métodos ágeis e ver a evolução do time.

Um ganho do time pela utilização de metodologias ágeis é buscar cada vez mais tração de entrega e entender a sua realidade para quando assumir que irá entregar a demanda em determinado de tempo, ser cada vez mais assertivo em sua mensuração. O que trará confiabilidade do time estratégico da empresa e dos clientes, uma vez que a expectativa está sempre alinhada.

Foi realizada inicialmente uma reunião de alinhamento. Nessa reunião estratégica estavam presentes o *Engineering Manager* e o *Tech Lead* como representantes da parte técnica do *software*, o *Head* de produto e o *Product Manage*r

representantes da por parte estratégica do produto, a líder da área de agilidade (Gestora dos *Agile Master* e *Scrum Mast*er da empresa), o *Scrum Master* e o desenvolvedor de maior senioridade que fará parte do *squad*.

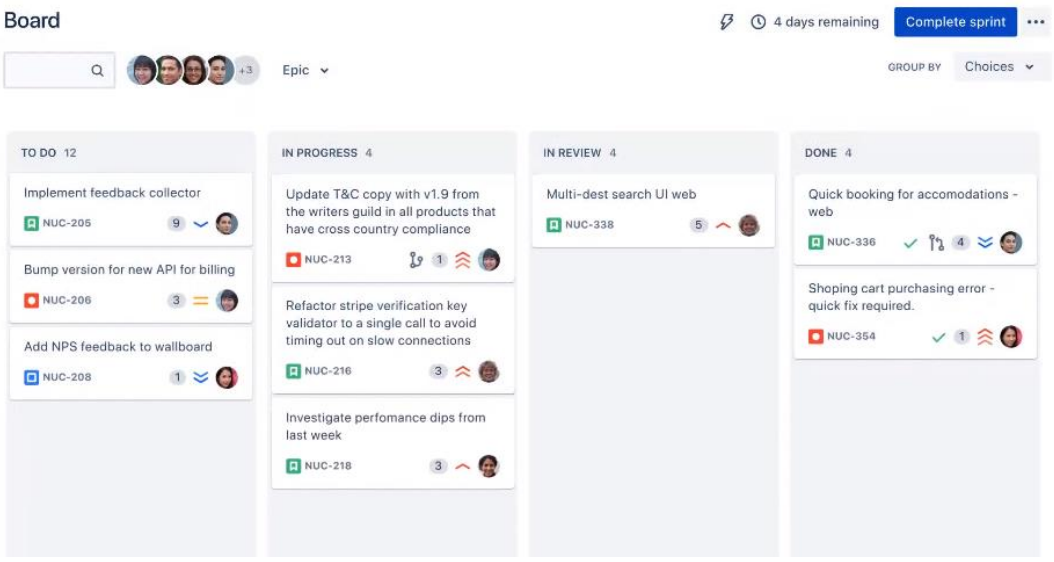
A reunião foi para esclarecer a todos os porquês da criação do time e quais são as expectativas da gerência com essa mudança. Durante a reunião foi confirmada a composição do *squad*, como será a dinâmica inicial de funcionamento do time, e os papéis esperados que cada um exerça e como será o início dos trabalhos.

O *squad*, recentemente criado, terá como composição técnica dois desenvolvedores *front end*, dois desenvolvedores *back end*, além do *Product Manage*r, *Tech Lead* e *Scrum Master*.

Por ser um squad que está iniciando agora, seus integrantes ainda não trabalharam juntos e não têm um alto grau de conhecimento do sistema que irão trabalhar, foi decidido utilizar o framework *Scrum*. Como dito anteriormente, essa metodologia de trabalho é um método prescritivo e com um sistema de produção empurrada, com o ciclo de desenvolvimento e avaliação em duas semanas.

A empresa utiliza o software de gestão de trabalho chamado *Jira*, desenvolvido pela empresa *Atlassian*, com ele que será realizado o acompanhamento do *squad* em estudo. Abaixo temos um exemplo do quadro de acompanhamento de algum time, fornecido pela empresa desenvolvedora do software, Figura 3.

Figura 3: Exemplo de quadro de acompanhamento do software Jira.



Fonte:(“Jira Software”, 2023)

Na imagem acima são observáveis as colunas que dizem o *status*, a evolução da tarefa. Os cartões representam o que deve ser feito. No canto superior direito, quantos dias ainda tem de *sprint*.

Com as definições acertadas e todas as preparações concluídas para o que o time comece a trabalhar, será dado início a primeira *Sprint.*

4.1 Sprint 1

O primeiro acordo feito pelo time é que a sprint será iniciada sempre na quarta à tarde e na parte da manhã acontecerá o primeiro ritual previsto pelo *Scrum*, a *planning.* Outro acordo inicial foi o horário do ritual *daily*, que ocorrerá sempre no mesmo horário como prevê o *framework*, o time votou e decidiu sempre a realizar as nove horas da manhã. Todos os integrantes devem participar e caso não consigam, devem justificar e procurar saber o que foi discutido com o *tech leader* ou com o *scrum master.* Ela não será realizada em dias que ocorram a *planning* ou *review* e ou a retrospectiva.

Como citado nas definições sobre o *scrum*, será iniciada nossa primeira *planning.* O *product managmament*(Pm) apresenta ao time as *User Stories (Us).*

As *User Stories* são obtidas por meio de narrações dos usuários (histórias) e compiladas como descrições breves a partir da perspectiva do usuário. Tais descrições demonstram uma funcionalidade que o sistema/produto deve atender, estas funcionalidades envolvem aspectos que vão além de características técnicas, abrangendo fatores que retornam valor ao usuário (ZACHARIAS; CUNHA; COSTA, 2017)

Nesse primeiro ciclo as *Us* são mais voltadas as adaptações iniciais a serem feitas no sistema, para que possam ser realizadas as incrementações no produto. Alterações em bancos, pequenas correções de bug no código e no produto, e os desenvolvedores conhecerem melhor esse produto com o qual vão trabalhar, parear com quem trabalhava antes com ele.

As *Us* foram apresentadas ao time de desenvolvimento e ele irá decidir quais devem entrar para desenvolvimento na *sprint*. Primeiro elas são lidas e ganham um ponto que representa complexidade e esforço de resolução dela. Essa pontuação segue o modelo de Fibonacci, onde a nota um representa uma tarefa de muito baixa

complexidade e esforço para resolução, e demandas que tenha uma pontuação acima de oito devem ser restruturadas pois seu esforço provavelmente excederá o tempo de duas semanas, sua complexidade real beira ao desconhecido, podendo haver contratempos que não podem ser resolvidos e atrasando o tempo previsto de entrega. Essa pontuação é votada pelos desenvolvedores presentes, cada um expõe seu ponto por dar aquela nota e chegam a um senso de nota.

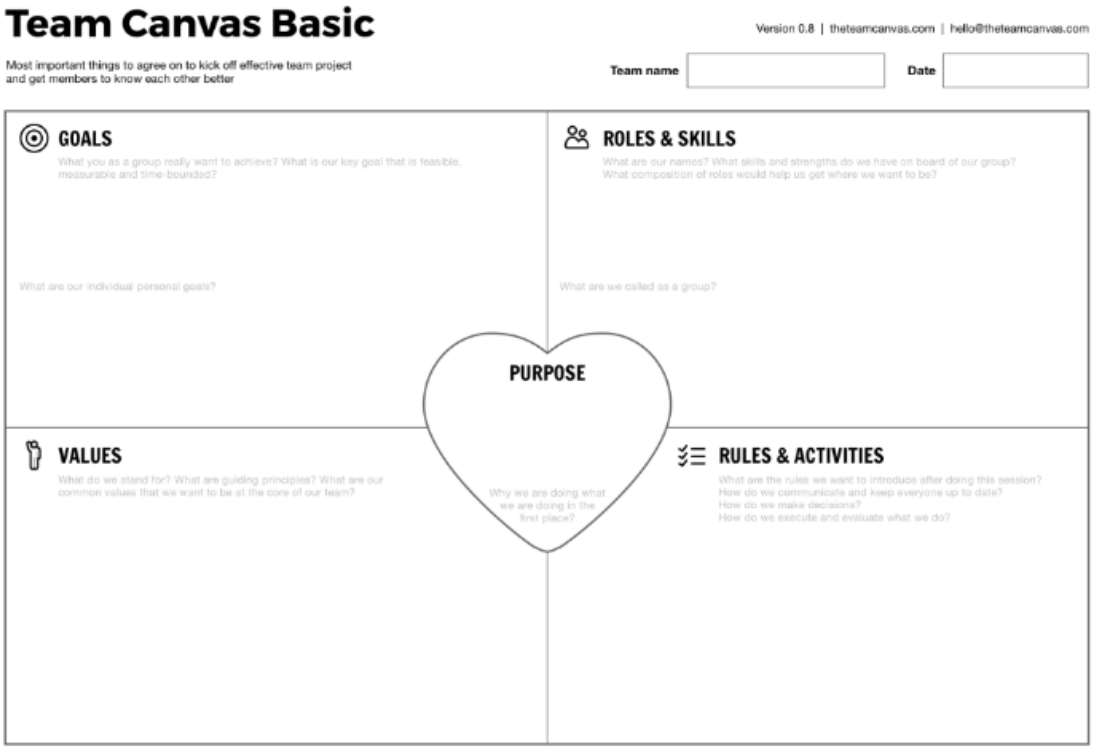
Após todas as *Us* estarem pontuadas, o time de desenvolvimento decide com quais eles iram se comprometer para entregarem no prazo de ciclo, duas semanas. Eles se comprometeram com a sprint com set Us, que somadas tem um total de vinte e nove pontos.

Agora finalizando a *planning*, o *Pm* ressalta novamente o objetivo principal dessa primeira sprint. Repassamos nossos primeiros acordos de time e é iniciada a primeira sprint do time.

Sobre as *dailys*, elas têm acontecido todos os dias como acordados. O foco é manter a todos atualizados sobre a evolução do trabalho e caso exista algum impedimento para prosseguir com e demanda ele seja identificado o mais rápido e tratado para que o time de desenvolvimento prossiga com as demandas.

Para continuar com a evolução ágil do time na empresa, squads recém-formados realizam alguns *Team Building*, para se identificarem enquanto time, alinharem alguns propósitos, dentre outros. Nessa primeira sprint, o *Scrum Master (SM)* utilizará o *Team Canvas* para que o time entenda melhor o projeto e crie seu próprio caminho para resolução de problemas e crescimento dos membros, Figura 4.

Figura 4: Exemplo de Team Canva aplicado ao time.



Fonte:(“The team Canvas”, 2023).

Toda a dinâmica consiste em uma discussão do time, cada integrante coloca o seu ponto de vista e depois chegamos a um acordo em comum, onde todos devem concordar com o que foi falado. Toda a dinâmica acontece em um tempo pré-estabelecido, essa teve o tempo de duas horas para conclusão.

No quadrante *goal*, o time define os objetivos em comum e compartilham seus objetivos pessoais. Os objetivos sempre relacionados ao projeto que estão envolvidos, no caso a evolução do produto para o qual o *squad* foi criado.

No quadrante *Roles and Skills*, uma oportunidade do time se conhecer melhor. É adicionado o nome e especialidades de cada um. Eles falam discutem e entendem como se completam para a evolução do produto.

No quadrante *Values*, a equipe compartilha quais os valores importantes para que o projeto siga e para que o time tenha um ambiente acolhedor e desenvolva seus membros. Principais valores para que todos os objetivos sejam alcançados.

No quadrante *Rules and Activities*, a equipe deve realizar acordos para o seu dia a dia. Deve descrever o que acham importante para evolução do projeto e dos membros e todos devem estar de acordo. Nesse momento foram levantados alguns pontos, mas a

discussão se estendeu e o time pediu para ter um momento próprio para discutir somente esse ponto.

No ponto central, o *Porpose,* o time define após todas as discussões qual o propósito do time e como eles vão seguir para que o projeto obtenha sucesso e todos os membros do time tenham sucesso em seus objetivos.

Lembrando que tudo que foi decidido hoje não é algo definitivo, caso o time mude de projeto ou a grande maioria dos membros sejam trocados, é importante revisitar essa dinâmica.

Após duas semanas, em uma terça-feira à tarde, será realizado os rituais de fechamento da sprint: A review e a retrospectiva.

A review é um ritual para que os desenvolvedores apresentem os resultados da sprint a pessoas interessadas nos resultados, os chamados *stakeholders*. Eles podem ser *tech leader* de outros times, gerentes de projetos, até o *CEO* ou cliente interessado naquela evolução do produto. Nessa primeira review estava presente o time, o *Engineering Manager* e o *Head* de produto.

Durante a *review* os desenvolvedores passaram por todas as Us, mostrando modificações feitas no código e documentações geradas. No caso não houve evolução no produto, mas se houvesse seria necessário mostrar a nova funcionalidade em funcionamento. Relataram experiencias e algumas dificuldades por estarem iniciando em um produto desconhecido por todos. Relatam oportunidades observadas nessa primeira sprint.

Durante a review foi constatado que o time entregou todas as Us com as quais se comprometeram. Cada vitória é comemorável, mas foi uma sprint exploratória e de preparação para iniciar os incrementos no produto.

Para finalizar o primeiro ciclo, falta somente o ritual da retrospectiva. Este será conduzido pelo *Sm* ainda na terça-feira, para que o próximo dia se inicie com a *planning* e logo após inicie uma nova *sprint*.

A retrospectiva é um momento para o time rever pontos positivos e pontos que não foram tão bem durante a sprint e se comprometer em como melhorar a partir do próximo ciclo.

Nessa primeira retrospectiva, o time ressaltou comprometimento e foco de todos para começarem a trabalhar com essa parte desconhecida do produto. Sobre pontos que não correram tão bem, o time trouxe que a evolução será mais rápida se todos pareassem mais, que deveriam tirar as dúvidas mais rápido e avisar ao demais do time mais rápido.

Sobre como o time vai se comprometer a melhorar, todos entraram em acordo em ficar mais presentes em uma sala online para parearem mais e as dúvidas serem esclarecidas mais rápidas. Sobre facilitar o *onboarding*, para que a adaptação ao time seja mais rápida, a equipe pediu para o *Sm* realizar um dinâmica para realização de acordos do time, sobre boas práticas de programação e documentação que todos devem seguir, e até atitudes durante o dia a dia.

Após a retrospectiva, foi encerrada a primeira sprint. Tivemos 100% das entregas com as quais nos comprometemos entregues. Foi avaliada como boa pelos líderes, mas lembrando que foi uma *sprint* exploratória.

4.2 Sprint 2

Para começar a nova sprint, serão seguidos os mesmos passos da anterior. Tem início com a realização da *planning.* Seguindo a mesma ordem, em que as *Us* são apresentadas ao time e logo após elas são pontuadas de acordo com o mesmo critério utilizado na *planning* anterior. Como foram feitos todos os pontos da sprint anterior, o time resolve assumir nessa sprint oito Us, somando um total de trinta e sete pontos.

Antes de iniciar a sprint, o Pm ressalta o objetivo principal desse ciclo. O *tech leader* ressalta sobre a sala de reunião para tirar dúvidas e caso tenha algum impedimento no desenvolvimento, entrar em contato o mais rápido para que o trabalho não seja comprometido. O Sm ressalta que nessa sprint haverá um feriado e que tem uma reunião para definições de acordo marcada, como foi solicitado na retrospectiva.

Em busca de continuar a evolução ágil do time, foi realizado uma reunião para que o time faça seus acordos e documente eles na central de conhecimento da empresa na página do time. Isso irá auxiliar quando novos integrantes forem adicionados ao time, pois eles entenderão com maior facilidade o método de trabalho da equipe.

Nessa dinâmica, todos os membros escrevem combinados que acham importantes e logo após todos os cards serão lidos. Todos os cards que forem aceitos

por todos os membros do time se tornam um combinado. Essa prática chamamos de Regras de Ouro do *squad*.

O *squad* entrou em acordo sobre compromissos com reuniões, como deixar o código de mais fácil entendimento para todos, regras de teste e sobre maneiras corretas de documentar e onde salvá-las para fácil acesso.

Um dos desenvolvedores ficou responsável para salvar esses acordos na página do time e repassar esses acordos em algumas *dailys* dessa sprint. Esse acordo também deve ser revisitado e atualizado quando necessário.

Seguindo, chegou o dia de encerramento da sprint, primeiro será realizado a review e logo após a retrospectiva.

Na *review* estão participando o time e novamente o *Engineering Manager* e o *Head* de produto. O time repassa as Us entregues, que foram quatro, com um total de dezesseis pontos.

O time mostrou as evoluções entregues e documentadas. Trouxe pontos de dificuldades durante essa sprint. Houve alguns questionamentos técnicos e foi encerrada a review.

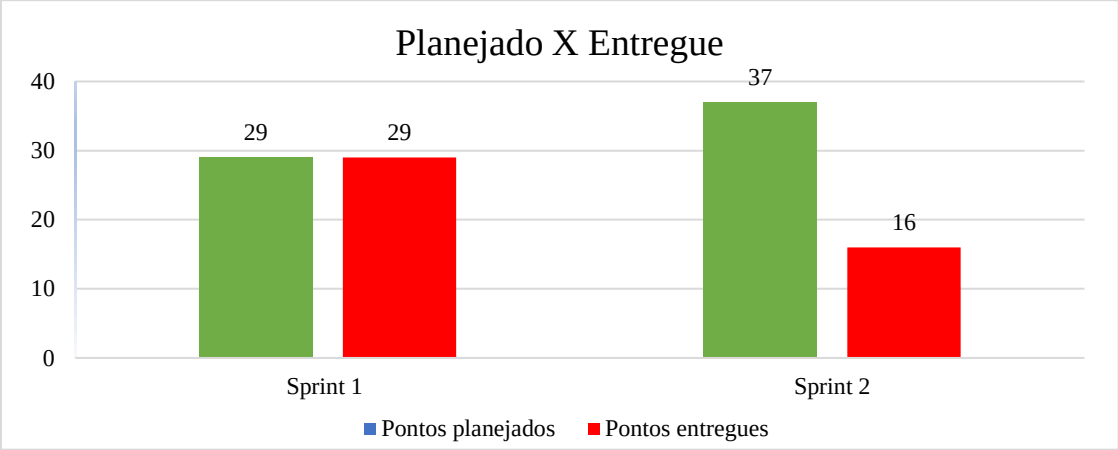
Na retrospectiva o time trouxe como ponto positivo a evolução em trabalhar junto, como a sala de reunião funcionou bem, onde sempre tinha alguém para tirar as dúvidas.

A respeito de pontos que não ocorreram tão bem, foi falado sobre algumas Us não estarem bem claras, o seu critério de aceite e o que era para ser desenvolvido nelas. Foi falado sobre a inexperiência do time em *planning*. A necessidade de entender melhor o que a Us está pedindo, dedicar um tempo a mais ao planejamento. Outro ponto que o time trouxe é sobre procurar mais rápido os *tech leaders* e o *pm* quando encontrar problemas mais sérios durante o desenvolvimento.

Sobre como o time pode melhorar, o mesmo pediu para realizar uma dinâmica para definir melhor como as Us devem estar escritas. Definir o mínimo que Us deve ter para entrar no *sprint* e como ela é considerada finalizada.

Antes de finalizar, foi mostrado ao time algumas métricas do software *Jira* para acompanhamento. E nesse trabalho iremos acompanhar a de velocidade Figura 5.

Figura 5: Gráfico de barra com o histórico do sprint 1 e 2 do time.



Fonte: (AUTOR, 2023).

4.3 Sprint 3

Para começar a nova sprint, foi seguido os mesmos passos das anteriores. Tem início com a realização da *planning*. Seguindo a mesma ordem, em que as Us são apresentadas ao time e logo após elas são pontuadas de acordo com o mesmo critério utilizado nas *plannings* anteriores. Foi ressaltado que na última sprint poucos pontos foram entregues. O time argumenta que muitas das Us estão com a metade do desenvolvimento finalizado e por isso vão adicionar mais algumas ainda sem iniciar. A sprint contará com nove Us e um total de trinta e quatro pontos.

Antes de iniciar a sprint, o *Pm* ressalta o objetivo principal desse ciclo. O *tech leader* pede para sempre procurarem ele ou o *Pm* quando houver dúvidas e não esperar as reuniões. O *Sm* avisa que nessa sprint tem uma reunião para definições de DOR e DOD.

O *DoR (Definition of Ready)* é um conjunto de critérios definidos pelo time, que detalha as condições que uma *story* precisa atingir/conter para que possa ser trabalhada pelo time. Basicamente, é também um critério de entrada da sprint (MACHER TECNOLOGIA, 2023).

O *DOD (Definition of Done)* ajudará a equipe a identificar os critérios que uma *task/story* precisa atender para que seja classificada como concluída *(“Done”).* Como cada equipe tem produto, maturidade e cultura diferenciadas, não há uma definição padronizada. (MACHER TECNOLOGIA, 2023)

Para o DoR, o time citou ter todos os critérios de aceite bem definidos, caso tenha as telas, já tenha aprovação do *Ux* antes de seguir o modelo chamado *Invest* para escrita de *Us.*

O modelo chamado *Invest* consiste em: Dar valor (atributo mais importante), expressa como determinada funcionalidade traz valor ao usuário; Independente – desenvolvida e testada por si só; Negociável, frente as rodadas de desenvolvimento de acordo com os feedbacks dos usuários, há colaboração e mudanças nos requisitos; Estimável, percepção da dificuldade de implementação; Testável, consegue ser validada por meio de testes para garantir que um determinado requisito tenha sido devidamente trabalhado (ZACHARIAS; CUNHA; COSTA, 2017)

Para o DoD, o time entende que *feature* está entregue, quando já está testada e aprovada, todos os requisitos foram cumpridos e o código já *deployado* em ambiente de produção

Finalizamos com os acordos acima aprovados todos os membros do time e documentados em nossa central do conhecimento.

Prosseguindo com a sprint, como prevê o framework que está sendo seguido, o próximo ritual é a review. Novamente participam da review o time e o *Engineering Manager* e o *Head* de produto. O time repassa as Us entregues. Do planejamento inicial somente uma não foi entregue, ou seja, vinte e nove pontos entregues. Foram apresentadas as alterações, onde toda a documentação está e dúvidas foram respondidas.

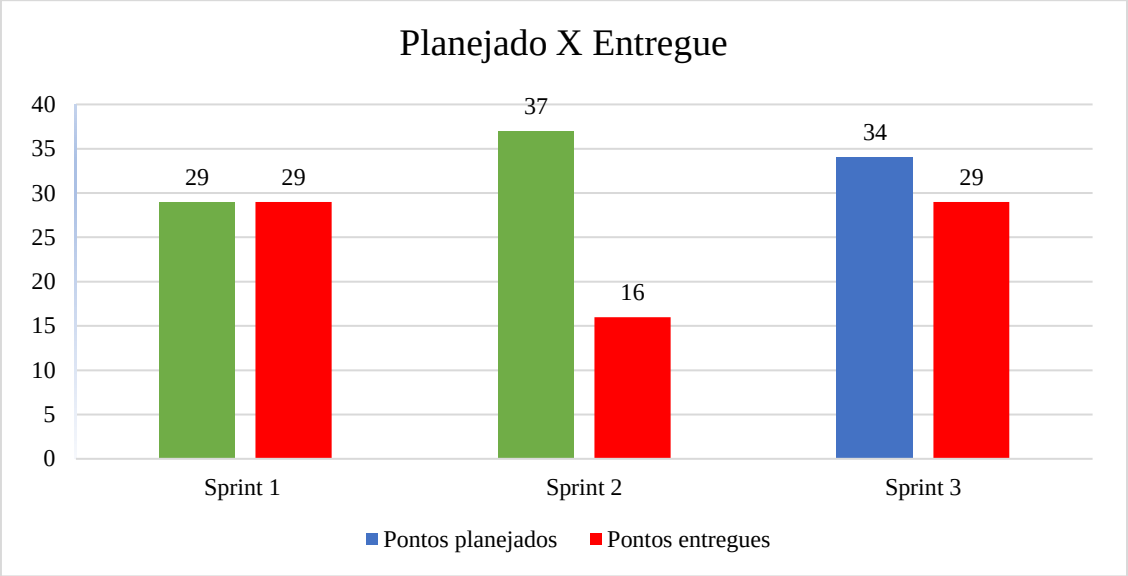
Na retrospectiva, o time ressaltou que o melhor refinamento das Us facilitou as entregas. A dedicação de alguns membros que se dedicaram a mais durante essa sprint.

Em pontos a melhorar, algumas Us ainda não estavam claras e dependiam de outras que ainda não foram desenvolvidas. Deveria mapear melhor as dependências.

Como o time se compromete a melhorar, antes da realização da *planning*, o time pede para que o Pm traga as Us que serão trabalhadas, e seja feito um “refinamento” (o passo a passo para a entrega da Us seja mapeado antes). Assim todos os contratempos e dependências serão mapeados.

Novamente passamos pelas métricas de acompanhamento da equipe e encerramos a sprint, Figura 6.

Figura 6: Gráfico de barra com o histórico do sprint 1, 2 e 3 do time.



Fonte: (AUTOR, 2023).

4.4 O que vem a seguir.

O time que foi acompanhado ainda tem um longo processo de evolução, assim como todos os times da empresa. A metodologia ágil não é somente aplicação de frameworks e canvas, mas sim a alteração da cultura da empresa e de todos os colaboradores.

O time iniciou sua jornada utilizando o Scrum, mas na agilidade é falado que existe o framework que é melhor que o outro, mas sim o que melhor se adapta ao momento. Esse time por ter integrantes de baixa senioridade e estar iniciando sua jornada com um produto desconhecido, necessitava de uma metodologia mais prescritiva, que diminui o espaço para dúvidas.

Outros frameworks como o *Kanban*, trazem excelentes resultados também. Porém o time já deve ter um histórico de produtividade e já saber como é o ciclo de atividades (bugs, incidentes). Ideal para times de mais alta senioridade e que já estão acostumadas a trabalhar com os códigos do produto.

Além do que já foi implantado com o time, foi aconselhado os próximos passos, como por exemplo, classes de serviços, definir outros tipos de demandas e trabalhar senso de urgência com o time.

Também com o passar das sprints, as métricas mostraram uma tendencia de estabilização, se não houver trocas de membros ou de propósito, o time conhecerá sua capacidade de entrega e o *Pm* poderá trabalhar com previsibilidade no seu *roadmap* e negociar os prazos para realizar entregas.

5 CONCLUSÃO

Este estudo teve como tema central a análise da formação de times ágeis em uma empresa de médio porte, com o objetivo de entender e aprimorar a eficácia na entrega de software. A pesquisa começou com uma revisão da literatura sobre o que caracteriza um time ágil, seguida pelo acompanhamento das sprints iniciais do time, onde se dedicou tempo para identificar e implementar melhorias contínuas.

Os resultados obtidos demonstraram que a aplicação de metodologias ágeis, como Scrum e Kanban, não apenas aumentou a velocidade de entrega, mas também melhorou a eficiência e a qualidade das entregas. A empresa em questão mostrou um comprometimento significativo com a agilidade, evidenciado pela presença de um scrum master e pela dedicação do time às práticas e frameworks ágeis. No entanto, a pesquisa também destacou que a agilidade vai além da mera aplicação de frameworks; é essencial que a organização como um todo adote uma cultura ágil, focada no cliente e em processos saudáveis.

Para dar continuidade a esta pesquisa, sugere-se a realização de estudos longitudinais que acompanhem a evolução dos times ágeis ao longo do tempo, permitindo uma análise mais profunda das práticas adotadas e dos resultados alcançados. Além disso, a implementação de treinamentos regulares e workshops sobre User Stories e Sprint Planning pode ser uma estratégia eficaz para aprimorar a compreensão e a aplicação das metodologias ágeis.

Em suma, a formação de times ágeis e a adoção de metodologias ágeis são fundamentais para o sucesso no desenvolvimento de software, especialmente em um ambiente dinâmico e em constante mudança. A busca por melhorias contínuas deve ser um esforço coletivo, envolvendo todos os colaboradores da organização, com o objetivo de gerar valor real para os clientes e garantir um processo saudável e sustentável para a empresa e seus colaboradores.

REFERÊNCIAS

APPELO, J. Management 3.0: leading Agile developers, developing Agile leaders. Upper Saddle River, Nj: Addison-Wesley, 2011

BIZARRIAS, F. S.; PENHA, R.; SILVA, L. F. VALOR E PROJETOS: A CONTRIBUIÇÃO DA PERSPECTIVA DE MARKETING. **GeP)**, v. 12, n. 2, p. 1, 2021.

BOEG, J. et al. **Kanban em 10 Passos Tradução para português e revisão**. [s.l: s.n.]. Disponível em: <http://infoq.com/br>.

Jira Software.

MACHER TECNOLOGIA. **DoD e DoR, você sabe a diferença?**

SCHWABER, K.; SUTHERLAND, J. **The Scrum Guide The Definitive Guide to Scrum: The Rules of the Game**. [s.l: s.n.]. Disponível em: <https://scrumguides.org/scrum-guide.html>. Acesso em: 12 mar. 2023.

SCRUM. **What is Scrum?**

SOMMERVILLE, IAN. **Engenharia de software**. [s.l.] Pearson Prentice Hall, 2011. v. 9º edição

SOUZA, B. L. **METODOLOGIAS ÁGEIS: ANÁLISE E COMPARAÇÃO DO SCRUM, KANBANELEANAPLICADOS AO DESENVOLVIMENTO DE SOFTWARE**. [s.l: s.n.]. Disponível em: <http://app.uff.br/riuff/handle/1/24054.>. Acesso em: 7 fev. 2023.

The team Canvas. Disponível em: <https://theteamcanvas.com/use/>. Acesso em: 29 jun. 2023.

VARELA ARRUDA, L. **DESENVOLVIMENTO ÁGIL DE SOFTWARE: UMA ANÁLISE SINTÉTICA A PARTIR DA METODOLOGIA KANBAN**. [s.l: s.n.].

ZACHARIAS, I. C. S.; CUNHA, L. P. DA; COSTA, J. M. H. DA. **USER STORIES: QUEM, QUANDO E COMO DEVE SER USADO?** Editora Edgard Blucher, Ltda., 22 nov. 2017.

VALENTE. M. T. **ENGENHARIA DE SOFTWARE MODERNA: PRINCÍPIOS E PRÁTICAS PARA DESENVOLVIMENTO DE SOFTWARE COM PRODUTIVIDADE**, Editora: Independente, 2020.