Universidade Federal do Paraná
Programa de Pós-Graduação Lato Sensu
Engenharia Ágil de Projetos

LIVIA FABRIN SOMERA
NATHALIA HELOISA DULLIUS
RITA DE CÁSSIA MIRANDA DA COSTA
RUI ROSSI DOS SANTOS

SCRUM NO DESENVOLVIMENTO DE SOFTWARE: ESTUDO DE CASO.

TOLEDO
2023

LIVIA FABRIN SOMERA
NATHALIA HELOISA DULLIUS
RITA DE CÁSSIA MIRANDA DA COSTA
RUI ROSSI DOS SANTOS

SCRUM NO DESENVOLVIMENTO DE SOFTWARE: ESTUDO DE CASO.

Monografia apresentada como requisito parcial à obtenção do grau de Especialista em Engenharia Ágil de Projetos, Curso de Pós-graduação Lato Sensu, Setor de Tecnologia, Departamento de Engenharia Mecânica, Universidade Federal do Paraná.

Orientador: Prof. Dr. Fernando Deschamps e Prof. Dr. Alessandro Marques

TOLEDO
2023

RESUMO

O Scrum é o framework ágil mais utilizado no mundo para o desenvolvimento de softwares e a sua utilização tem ajudado a melhorar os resultados dos times e das organizações. Um benefício central que é produzido com a adoção do Scrum é a antecipação da entrega de valor, mas isso só ocorre quando ele é aplicado adequadamente. Há muitos estudos de casos acerca do Scrum, mas a maioria se limita a descrever os resultados positivos e não explora os desafios enfrentados e tampouco os fracassos e as suas causas. Este artigo apresenta um estudo de caso de aplicação do Scrum no desenvolvimento de software com foco na avaliação dos resultados obtidos e na análise de fatores que contribuíram positiva e negativamente. Esse estudo de caso refere-se ao desenvolvimento de um software para um escritório de projetos e que se estendeu por seis meses. Um dos resultados negativos mais significativos foi exatamente a demora na entrega de valor e avaliou-se a relação disso com a dinâmica de trabalho estabelecida comparativamente ao que é preconizado pelo Scrum nos seus pilares, valores, papéis, eventos e artefatos.

Palavras-chave: Scrum. Software. Ágil.

CONTEÚDO

1. INTRODUÇÃO E SITUAÇÃO-PROBLEMA

O Scrum é o framework ágil mais utilizado no mundo e sua aplicação no processo de desenvolvimento de softwares é especialmente difundido. A sua adoção tem ajudado a melhorar os resultados dos times e das organizações. Um benefício central que ele ajuda a produzir é a antecipação da entrega de valor. Mas isso só ocorre quando ele é aplicado adequadamente.

Há muitos estudos de casos acerca do Scrum, mas a maioria se limita a descrever os resultados positivos e não explora os desafios enfrentados ou os fracassos e as suas causas. Este artigo centra-se exatamente na aplicação do Scrum no desenvolvimento de software com foco na avaliação dos resultados obtidos em um caso específico e na análise de fatores que contribuíram positiva e negativamente.

O segundo tópico deste artigo apresenta a fundamentação teórica do estudo realizado. Ele inclui uma visão panorâmica acerca do processo de desenvolvimento de software, conceitos e exemplos de framework e de ferramentas ágeis e uma apresentação analítica dos fundamentos e da estrutura do framework Scrum.

O terceiro tópico indica os materiais e métodos empregados para conduzir o presente estudo. Optou-se por um estudo de caso utilizando abordagem qualitativa e descritiva e fez-se o levantamento e registro de dados via observação participante, análise de documentos e entrevistas.

O quarto tópico apresenta o estudo de caso selecionado e faz a análise dos resultados. Ele descreve o projeto de desenvolvimento de software escolhido, como foi composto o Time Scrum, como se deu o desenvolvimento da solução, o relato das observações e a análise dos dados coletados através de entrevistas.

Um dos resultados negativos mais significativos foi a demora na entrega de valor para a organização. Avaliou-se que isso esteve diretamente relacionado à baixa aderência da dinâmica de trabalho que foi estabelecida em relação ao que é prescrito pelo Scrum nos seus pilares, papéis, eventos e artefatos.

2. FUNDAMENTAÇÃO PARA A SOLUÇÃO

Este tópico explora referências teóricas levantadas durante o esforço de revisão da literatura relacionada ao objeto deste estudo: o uso do Scrum no desenvolvimento de softwares. As próximas seções apresentam a compilação do conteúdo que foi produzido para sustentar a análise do estudo de caso analisado.

A primeira seção apresenta uma visão panorâmica acerca do processo de desenvolvimento de software. Ela inicia explorando as características do processo de desenvolvimento de software, indica as fases típicas nas quais ele se divide, os modelos mais utilizados para estruturar esse trabalho e como se faz a gestão desse tipo de projetos.

A segunda seção contempla os conceitos de framework e de ferramentas ágeis. Além disso, ela explora alguns dos frameworks e ferramentas mais utilizados no mercado para conduzir esforços empreendidos no entorno de projetos de desenvolvimento de softwares. Dentre esses recursos estão o Extreme Programming, o Lean Software Development, o Jira, o Trello e o Asana.

A terceira seção explora o tema central do presente estudo: o framework Scrum. Aqui encontra-se uma visão geral dos fundamentos do Scrum, incluindo os seus pilares e valores. Ela também apresenta como o Time Scrum é composto e como cada um dos seus eventos e artefatos se encaixam para organizar o trabalho do time e potencializar a entrega de valor.

2.1. PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O processo de desenvolvimento de software é um campo vasto e complexo dentro da Engenharia de Software. Ele envolve uma série de fases interconectadas, modelos e práticas que visam criar software de alta qualidade que atenda às necessidades dos usuários. A escolha do modelo de processo adequado e uma gestão de projetos eficaz são fundamentais para o sucesso no desenvolvimento de software.

O processo de desenvolvimento de software é uma parte fundamental da Engenharia de Software, uma disciplina que lida com os princípios, métodos e técnicas utilizados para projetar, construir, testar e manter sistemas de software de alta qualidade. Neste contexto, este tópico irá explorar as principais características, fases e modelos de processo de desenvolvimento de software, a importância da gestão de

projetos e da qualidade do software, bem como apresentar as principais ferramentas e frameworks ágeis.

O desenvolvimento de software é uma atividade complexa que exige um planejamento cuidadoso, uma abordagem sistemática e uma compreensão profunda das necessidades do cliente. Neste contexto, este texto irá explorar as principais características, fases e modelos de processo de desenvolvimento de software, bem como a importância da gestão de projetos e da qualidade do software.

2.1.1. Características do processo de desenvolvimento de software

O processo de desenvolvimento de software é uma série de atividades e etapas que são seguidas para criar um software de alta qualidade. Existem várias metodologias e abordagens para o desenvolvimento de software, mas há características fundamentais que são comuns à maioria dos processos. As principais características do desenvolvimento de software são: Iteração e Incrementalismo, Gerenciamento de Requisitos, Controle de Qualidade, Flexibilidade e Adaptabilidade, Comunicação e Colaboração, Documentação Adequada e Testes Abrangentes.

De acordo com PRESSMAN & MAXIM (2016), o desenvolvimento de software adota uma abordagem Iterativa e Incremental, em que o progresso não ocorre de forma linear, mas sim por meio de ciclos repetidos e progressivos. Cada iteração adiciona funcionalidades ao sistema, permitindo um feedback contínuo e ajustes à medida que o projeto avança. Isso é vital para lidar com requisitos mutáveis e garantir a adaptabilidade às necessidades do cliente.

Para SOMMERVILLE (2011), o gerenciamento de requisitos é uma pedra angular do processo de desenvolvimento de software, envolvendo a identificação, documentação e validação de requisitos. Uma compreensão sólida das necessidades dos stakeholders é crucial para o sucesso do projeto. Técnicas e ferramentas de gerenciamento de requisitos, como diagramas de caso de uso e entrevistas, são amplamente empregadas.

A qualidade é um princípio inegociável no desenvolvimento de software. O processo incorpora verificações e validações constantes para garantir que o software atenda aos padrões de qualidade estabelecidos. Isso se estende à qualidade do código, da documentação e da experiência do usuário (PRESSMAN & MAXIM, 2016).

Os processos de desenvolvimento de software devem ser flexíveis e adaptáveis. À medida que novos insights surgem e os requisitos evoluem, a equipe deve ser capaz de ajustar suas abordagens e metas. A capacidade de mudar de direção é crucial para evitar rigidez e ineficiência (SOMMERVILLE, 2011).

A comunicação eficaz e a colaboração são fundamentais em todas as etapas do processo. A equipe, os stakeholders e os usuários finais devem estar alinhados e manter uma comunicação transparente. Isso ajuda a evitar mal-entendidos, identificar problemas precocemente e manter a equipe engajada e motivada (PRESSMAN & MAXIM, 2016).

Segundo SOMMERVILLE (2011), a documentação desempenha um papel integral no processo de desenvolvimento de software. Ela não apenas registra requisitos, design e código, mas também serve como um recurso valioso para manutenção e compreensão contínua do software. Documentação clara e abrangente facilita a transição entre equipes e a resolução eficiente de problemas.

Para PRESSMAN & MAXIM (2016) testar é crucial para garantir que o software funcione corretamente e atenda aos requisitos e isso engloba desde testes de unidade até testes de aceitação do cliente.

2.1.2. Fases do processo de desenvolvimento de software

O processo de desenvolvimento de software é uma jornada complexa que envolve várias fases interconectadas, cada uma desempenhando um papel crucial na criação de um sistema de software robusto e de alta qualidade. As principais fases desse processo são: levantamento de requisitos, análise, projeto, implementação, testes e implantação.

A atividade de levantamento de requisitos consiste na busca pela compreensão do domínio de uma aplicação, através da interação com os stakeholders do projeto, com o intuito de identificar os requisitos desejados para o software. Não obstante sua essencialidade no contexto do desenvolvimento, o processo de levantamento de requisitos frequentemente encontra desafios decorrentes da falta de adoção de técnicas adequadas para a extração das informações necessárias, bem como da ausência de uma clara descrição dos requisitos por parte das partes interessadas (GALDINO, 2021).

O processo de levantamento de requisitos, por sua vez, é composto por quatro etapas: obtenção de requisitos, classificação e organização dos requisitos, priorização/negociação dos requisitos e documentação dos requisitos (SOMMERVILLE, 2011).

Com base nos requisitos coletados, as fases de análise e projeto se concentram em criar um plano para a implementação do software. Os analistas de sistemas projetam a arquitetura do sistema, definem estruturas de dados, algoritmos e interfaces de usuário. Essa fase também envolve a criação de modelos de design que servirão como guias para a implementação.

A fase de implementação é um estágio crítico no processo de desenvolvimento de software. Nesta etapa, a equipe de desenvolvimento traduz os projetos e documentos de design em código-fonte executável. Isso envolve a escrita, testes e integração de todas as partes do software para criar um sistema funcional. Após a implementação, o software está pronto para passar para a fase de testes, onde sua funcionalidade e qualidade serão avaliadas minuciosamente (SOMMERVILLE, 2011; PRESSMAN & MAXIM, 2016).

De acordo com Pressman apud SOUZA (2016), realizar os testes é de grande importância. É nessa etapa que se aplicam os meios de detecção e correção de defeitos e seu principal objetivo é detectar erros ainda não descobertos.

A fase de implantação é o estágio final e crítico no processo de desenvolvimento de software e é nesta etapa que o software é preparado para ser disponibilizado para os usuários finais. A implantação pode ser realizada de diferentes maneiras, dependendo da complexidade do software e dos requisitos do projeto. Isso pode incluir implantações em nuvem, instalações locais, atualizações automáticas, entre outros métodos. Após a implantação bem-sucedida, o software fica disponível para os usuários finais (SOMMERVILLE, 2011; PRESSMAN & MAXIM, 2016).

2.1.3. Modelos de processos de desenvolvimento de software

Os modelos de processo de desenvolvimento de software desempenham um papel fundamental na gestão e organização do ciclo de vida do desenvolvimento de software (SOMMERVILLE, 2011). Esses modelos são estruturas conceituais que fornecem diretrizes, práticas e regras para planejar, executar e controlar as atividades de desenvolvimento de software.

Estes modelos evoluíram significativamente ao longo das últimas décadas em resposta às mudanças nas necessidades da indústria de software e às lições aprendidas com projetos anteriores. Historicamente, os modelos de desenvolvimento de software têm sido categorizados em duas abordagens principais: modelos tradicionais e modelos ágeis.

Dentre os modelos tradicionais, são listados alguns exemplos a seguir:

- **Modelo em Cascata (Waterfall):** esse é um dos primeiros modelos de processos de desenvolvimento de software e segue uma abordagem sequencial e linear e ele divide o desenvolvimento de software em fases distintas, como análise, design, implementação, teste e manutenção, com cada fase dependendo do sucesso da fase anterior (ROYCE, 1970).
- **Modelo em Espiral:** esse modelo introduz a ideia de iterações e ciclos de desenvolvimento repetitivos (BOEHM, 1988) e ele enfatiza a avaliação de riscos e a adaptação contínua ao longo do ciclo de vida do projeto.

Dentro do contexto dos modelos ágeis de desenvolvimento de software, há diversas metodologias, frameworks e ferramentas. Mas há dois que destacam-se pelo volume de uso no mercado atual e pelos resultados produzidos:

- **Scrum:** um dos frameworks ágeis mais populares, que enfatiza a colaboração, a comunicação constante, a entrega iterativa de incrementos de software e o desenvolvimento dividido em ciclos de trabalho curtos e focados, as sprints (SCHWABER & SUTHERLAND, 2020).
- **Kanban:** é uma abordagem baseada em fluxo que se concentra na visualização e gerenciamento do fluxo de trabalho e permite a adaptação contínua às mudanças nas prioridades do projeto (ANDERSON, 2010).

A escolha do modelo de processos de desenvolvimento de software adequado depende de vários fatores, incluindo a natureza do projeto, os requisitos do cliente, a equipe de desenvolvimento e os recursos disponíveis. A compreensão desses modelos e a capacidade de escolher e adaptar o mais apropriado são habilidades essenciais para profissionais de software e pesquisadores no campo da engenharia de software.

A gestão de projetos de desenvolvimento de software desempenha um papel crucial na entrega bem-sucedida de produtos de software (SOMMERVILLE, 2011). Ela envolve o planejamento, a execução, o monitoramento e o controle de todas as atividades do projeto para garantir que os objetivos sejam alcançados dentro do escopo, prazo, orçamento e qualidade definidos (PMI, 2017).

A complexidade inerente ao desenvolvimento de software, juntamente com a necessidade de atender a requisitos em constante evolução, torna a gestão de projetos essencial (PRESSMAN, 2016). Sem uma gestão eficaz, os projetos de software podem enfrentar desafios como atrasos, estouro de orçamento e resultados insatisfatórios.

2.2. FERRAMENTAS E FRAMEWORKS ÁGEIS

As metodologias ágeis têm ganhado cada vez mais destaque no contexto relacionado ao desenvolvimento de softwares, sendo cada vez mais utilizadas em cenários que requerem rápida adaptação às mudanças e entrega de valor contínua e no menor tempo possível. Nesse sentido, as metodologias tradicionais, tais como modelo em cascata, acabam sendo substituídas por modelos ágeis. Os métodos tradicionais são bem aplicados em situações onde há pouca ou nenhuma alteração nos requisitos do sistema. Mas isso não ocorre em projetos de desenvolvimento de softwares, onde na maioria das vezes há ambientes dinâmicos, equipes pequenas e prazos curtos e que exigem maior adaptabilidade e alta capacidade de resposta a novos requisitos.

As metodologias ágeis consistem em métodos voltados para atender às necessidades do mercado por processos mais ágeis e com menor ciclo de desenvolvimento. Elas adquiriram maior destaque em 2001, por meio “Manifesto Ágil”. Ele foi concebido por dezessete especialistas em desenvolvimento de software e fundamentou as bases comuns aos diferentes métodos ágeis existentes, dentre eles o Scrum, Extreme Programming (XP) e outros (SOARES, 2004). Dessa forma, os princípios fundamentais do “Manifesto Ágil” podem ser descritos da seguinte maneira (BEEDLE et. al., 2001):

1. Indivíduos e interações ao invés de processos e ferramentas;
2. Software operante ao invés de documentações completas;
3. Colaboração do cliente ao invés de negociações contratuais;
4. Responder a mudanças ao invés de seguir um planejamento.

O Extreme Programming (XP) é um exemplo de abordagem ágil aplicável ao processo de desenvolvimento de software. Ele se concentra na excelência técnica, comunicação, trabalho em equipe e na melhoria contínua do processo de desenvolvimento. Seus fundamentos incluem:

1. Desenvolvimento baseado na comunicação, feedback, simplicidade, coragem e respeito;
2. Aprimorar o processo de desenvolvimento por meio de práticas comprovadamente úteis;
3. Tradução dos valores em prática;
4. Comunidade que compartilha esses valores e práticas.

O XP promove práticas como desenvolvimento orientado a testes e integração contínua, valorizando a comunicação eficaz entre os membros da equipe e o cliente (BECK & ANDRES, 2004).

O Lean Software Development, por seu lado, representa uma aplicação dos princípios e práticas de Lean no contexto de desenvolvimento de software. Ele busca eliminar desperdícios e focar no que agrega valor ao cliente, dando ênfase na entrega contínua, redução do tempo de ciclo e o aprendizado constante por meio de ciclos curtos (BESSA & ARTHAUD, 2018). Alguns dos princípios Lean aplicados ao desenvolvimento de sistemas consistem em eliminar os desperdícios, amplificar o aprendizado, decidir o mais tarde possível, entregar o mais rápido possível, conceder autonomia à equipe, construir com integridade e visualizar o todo.

Há uma grande quantidade de ferramentas disponíveis no mercado para apoiar a implementação de diferentes frameworks e de práticas ágeis. Diversas delas oferecem diferentes visões para a gestão das atividades dos projetos e isso tipicamente inclui visões em forma de lista, kanban e cronograma. Mas infelizmente são poucas que oferecem suporte ao Framework Scrum.

Uma ferramenta muito popular é o Trello. Ele permite o gerenciamento de projetos por meio de listas de atividades, quadros e cartões. Também conta com outros recursos visuais que auxiliam no acompanhamento dos fluxos de atividades e permite organizar projetos de diversos níveis de complexidade de forma simples e intuitiva, promovendo a organização e colaboração entre os membros da equipe (TRELLO, 2023). Mas esse é um exemplo de ferramenta que não suporta o Framework Scrum.

O Jira é outro bom exemplo de software muito popular para o gerenciamento de projetos utilizando modelos ágeis e que permite realizar o planejamento de um projeto e monitorar as atividades de forma intuitiva. Ele reúne as informações pertinentes ao projeto e permite visualizar o que está sendo realizado com transparência para todas as partes interessadas, que conseguem acompanhar

facilmente tudo o que está sendo executado (ATLASSIAN, 2023). Uma vantagem do Jira sobre o Trello é exatamente o suporte que oferece ao Framework Scrum.

O Bitrix24 é outro exemplo de ferramenta ágil que oferece suporte ao Framework Scrum. Ela oferece diversos recursos para a colaboração de times e para a gestão de projetos e tarefas, tais como: listas de tarefas, quadros kanban, cronograma, gráfico de gantt, calendário e outros. No que se refere ao Scrum, ela oferece suporte a todos os papéis, artefatos e eventos previstos no framework (BITRIX24, 2023).

2.3. O FRAMEWORK SCRUM

O Scrum representa uma alternativa ao modelo de gerenciamento de projetos conhecido como *waterfall* (ou cascata). Esse modelo caracteriza-se pelo esforço de planejamento com o propósito de elaborar planos detalhados do projeto em termos de escopo, cronograma, aquisições, custos, recursos, riscos, qualidade, stakeholders e comunicações. Um dos problemas nesse modelo é a demora para se avançar para a fase de execução do projeto, uma vez que isso tende a ocorrer somente após ter sido concluído o seu planejamento. E uma das limitações mais importantes diz respeito à dificuldade de se construir um plano detalhado e assertivo para o projeto quando o mesmo se insere em um ambiente de alta incerteza e volatilidade.

O Scrum é um framework que foi estruturado para “ajudar pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos” (SCHWABER & SUTHERLAND, 2020, p. 4). Essa definição evidencia o objetivo central do Scrum: a geração de valor. Além disso, ela também esclarece que o Scrum é uma boa alternativa para tratar de problemas complexos e que ele faz isso através do desenvolvimento de soluções utilizando uma abordagem adaptativa, que contrasta com a abordagem preditiva.

2.3.1. Fundamentos do Scrum

Segundo SCHWABER & SUTHERLAND (2020, p. 4), o Scrum baseia-se no empirismo e no *lean thinking*. O empirismo indica que o conhecimento provém da experiência e que as decisões devem se basear na observação. O *lean thinking*, por

sua vez, ressalta a importância de foco na redução de desperdícios e em concentrar-se no que é essencial.

O Scrum prescreve uma abordagem iterativa e incremental (SCHWABER & SUTHERLAND, 2020, p. 4) e isso contribui tanto para otimizar a previsibilidade quanto para controlar os riscos. Uma abordagem iterativa pressupõe contato frequente com os clientes com o intuito de apresentar a solução que está sendo construída e coletar feedbacks que possam orientar e direcionar o esforço subsequente. E o aspecto incremental dessa abordagem sugere a liberação de incrementos utilizáveis da solução para os clientes e que isso deve ser feito o mais cedo possível.

Para desenvolver soluções, o Scrum envolve grupos de pessoas que possuam as habilidades e conhecimentos necessários para realizar o trabalho requerido ou que possam adquiri-las conforme isso for exigido. O que importa é que essas pessoas, coletivamente, possuam as habilidades exigidas. E esse trabalho é desenvolvido combinando quatro eventos formais que estão contidos dentro de um evento agrupador ou contêiner, a Sprint.

Segundo SCHWABER & SUTHERLAND (2020, p. 4), são os eventos do Scrum que implementam os seus três pilares: a transparência, a inspeção e a adaptação. E são esses pilares que representam as bases de sustentação do Scrum.

Um dos três pilares do Scrum é a transparência. Esse pilar estabelece que o trabalho realizado deve ser visível tanto para quem o está executando quanto para quem o recebe (SCHWABER & SUTHERLAND, 2020, p. 5). A importância desse pilar se dá em função de que decisões são tomadas com base no estado percebido dos artefatos formais do Scrum e um baixo nível de transparência sobre eles pode levar a decisões que aumentam os riscos e diminuem o valor produzido.

Outro pilar do Scrum é a inspeção e ela é habilitada pela transparência. Realizar inspeções frequentes e diligentes é fundamental para analisar o progresso em direção à meta e para detectar e avaliar variações ou problemas (SCHWABER & SUTHERLAND, 2020, p. 5). Os cinco eventos do Scrum ajudam a manter a cadência que é requerida para as inspeções.

O último pilar do Scrum é a adaptação e ela é habilitada pelas inspeções. A adaptação é essencial para que os ajustes requeridos sejam realizados sempre que for produzido um resultado inaceitável (SCHWABER & SUTHERLAND, 2020, p. 5). Nesses casos, os ajustes devem ser realizados o mais rápido possível para minimizar os impactos e prevenir novos desvios.

O sucesso no uso do Scrum depende de as pessoas viverem, efetivamente, cinco valores: compromisso, foco, abertura, respeito e coragem (SCHWABER & SUTHERLAND, 2020, p. 5). Todos devem assumir o compromisso em atingir os objetivos traçados e a suportarem uns aos outros. O foco deve estar concentrado em realizar o melhor trabalho possível em direção à meta. Todos devem estar abertos em relação ao trabalho e aos desafios e devem se respeitar como pessoas capazes e independentes. Além disso, devem ter coragem para fazer a coisa certa e para trabalhar com problemas difíceis.

Os valores do Scrum orientam o trabalho, ações e comportamentos e não devem ser diminuídos ou minimizados. Eles são reforçados pelos eventos e pelos artefatos, fazem os pilares empíricos ganharem vida e constroem a confiança.

2.3.2. Time Scrum

Um Time Scrum (*Scrum Team*) representa a unidade fundamental do Scrum e é composto por um pequeno grupo de pessoas que exercem os seguintes papéis: um Scrum Master, um Product Owner e Developers (SCHWABER & SUTHERLAND, 2020, p. 6). É importante ressaltar que não há hierarquia ou sub-times em um Time Scrum porque trata-se de uma unidade coesa de profissionais que atuam juntos para atingir a meta do produto.

Um Time Scrum deve ser composto por 10 pessoas ou menos. Esse número é prescrito pelo Scrum porque times menores tendem a se comunicar melhor e serem mais produtivos. A ideia é que o Time Scrum seja grande o suficiente para conseguir produzir um incremento significativo em uma Sprint e ser pequeno o suficiente para desempenhar suas atividades com elevada agilidade.

Os Developers são os membros do Time Scrum que se dedicam a criar qualquer aspecto de um incremento utilizável do produto (SCHWABER & SUTHERLAND, 2020, p. 6). Para isso, eles devem possuir as habilidades necessárias e estas variam em conformidade com o domínio do trabalho a ser realizado. De qualquer forma, eles possuem responsabilidades típicas, tais como: criar o plano para a Sprint (o Sprint Backlog), desenvolver o incremento em conformidade com a Definição de Pronto, adaptar continuamente o plano para continuar progredindo em direção à meta e se responsabilizarem mutuamente como profissionais.

O Product Owner é o responsável por maximizar o valor do produto resultante do trabalho do Scrum Team (SCHWABER & SUTHERLAND, 2020, p. 7). Ele é o responsável pelo gerenciamento do Product Backlog e isso implica em: desenvolver e comunicar a meta do produto, criar e comunicar os itens do Product Backlog, ordenar os itens do Product Backlog e garantir que o Product Backlog seja transparente e compreensível.

O Scrum Master atua tanto junto ao Time Scrum quanto aos demais stakeholders da organização e sua missão é ajudar todos no entendimento da teoria e prática do Scrum (SCHWABER & SUTHERLAND, 2020, p. 7). Ele deve liderar todo o processo de aplicação do Scrum e contribuir para melhorar a aplicação de suas práticas. Ele serve ao Time Scrum treinando os seus membros, ajudando a concentrarem-se na produção de incrementos de alto valor e em conformidade com a Definição de Pronto, atuando na remoção de impedimentos e garantindo que os eventos ocorram, sejam positivos e mantenham-se no *timebox* especificado. Ele serve ao Product Owner ajudando-o a encontrar técnicas para a definição da meta do produto e para o gerenciamento do Product Backlog, a estabelecer um planejamento empírico e facilitando a colaboração com stakeholders. Ele também serve a organização treinando e orientando as pessoas na adoção do Scrum, planejando e aconselhando implementações do Scrum, ajudando stakeholders a compreenderem e aplicarem uma abordagem empírica e removendo barreiras entre eles e o Time Scrum.

2.3.3. Eventos Scrum

Há cinco eventos previstos no framework Scrum: Sprint, Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective. A Sprint representa um container dentro do qual ocorrem os outros quatro eventos (SCHWABER & SUTHERLAND, 2020, p. 8). Cada um desses eventos representa uma oportunidade para o Time Scrum inspecionar e adaptar os artefatos do Scrum. É importante destacar que esses eventos são concebidos exatamente para permitir a transparência que é necessária para o bom funcionamento do Scrum e se eles não forem conduzidos conforme são prescritos podem ser perdidas oportunidades valiosas para a inspeção e adaptação dos artefatos. Ademais, sugere-se que esses eventos sejam realizados sempre no mesmo horário e local como forma de reduzir a complexidade, criar regularidade e reduzir a necessidade de reuniões não definidas no Scrum.

A Sprint é caracterizada como um evento contêiner porque todos os outros quatro eventos ocorrem em seu interior e é aí que é realizado todo o trabalho necessário para atingir a meta do produto. Conforme SCHWABER & SUTHERLAND (2020, p. 8), a Sprint é um evento de duração fixa com duração não superior a um mês e que permite estabelecer uma cadência na entrega de valor. É definida uma meta específica para cada Sprint para que todos saibam qual é o valor que se espera entregar ao final da mesma. Quando uma Sprint é finalizada, inicia-se imediatamente a próxima Sprint.

SCHWABER & SUTHERLAND (2020, p. 9) destacam alguns pontos fundamentais a serem observados durante uma Sprint:

- O Product Backlog deve ir sendo refinado conforme isso for necessário;
- O escopo do trabalho a ser realizado pode ser renegociado com o Product Owner conforme mais é aprendido;
- Nenhuma mudança que coloque em risco a meta da Sprint deve ser feita;
- A qualidade esperada para as entregas não pode ser diminuída.

Também é importante compreender os benefícios de organizar o trabalho do Time Scrum na forma de Sprints. Segundo SCHWABER & SUTHERLAND (2020, p. 9), esse modelo permite elevar a previsibilidade e permite a inspeção e adaptação do progresso em direção à meta do produto. Considerando o tempo máximo de uma Sprint, isso ocorre ao menos uma vez por mês. Também pode-se estabelecer um padrão de Sprints mais curtas (de 3, 2 ou 1 semana) quando isso for útil para gerar ciclos de aprendizagem mais curtos e, com isso, antecipar o tratamento de riscos de custo e esforço.

Não há um padrão de ferramentas ou práticas para prever e acompanhar o progresso do trabalho em uma Sprint. Mas SCHWABER & SUTHERLAND (2020, p. 9) destacam algumas alternativas: gráficos de burn-down ou de burn-up e cumulative flows. Mas embora o uso dessas ferramentas seja comprovadamente útil, o que é fundamental é o acompanhamento empírico do trabalho.

O primeiro evento que ocorre no interior de uma Sprint é a Sprint Planning e ele é realizado de forma colaborativa pelo Time Scrum com a finalidade de definir o trabalho a ser realizado na Sprint (SCHWABER & SUTHERLAND, 2020, p. 9). Para que o resultado desse evento seja efetivo, o Product Owner deve garantir que todos estejam preparados para discutir os itens do Product Backlog considerando sua relação com a Meta do Produto. Também é permitido trazer convidados a esse evento para fornecer conselhos se o Time Scrum entender que isso contribuirá de alguma forma.

Conforme SCHWABER & SUTHERLAND (2020, p. 9), a Sprint Planning deve ser conduzida por três perguntas:

- Por que essa Sprint é valiosa?
- O que pode ser feito nessa Sprint?
- Como o trabalho escolhido será realizado?

Para ajudar a responder por que a Sprint é valiosa, o Product Owner deve propor o que pode ser feito no produto para aumentar o seu valor e utilidade. Então, todo o Time Scrum deve colaborar para a definição da Meta da Sprint para que possa comunicar por que ela é valiosa para os stakeholders (SCHWABER & SUTHERLAND, 2020, p. 9). É importante destacar que a definição da Meta da Sprint é obrigatória e deve estar estabelecida antes do final da Sprint Planning.

O segundo passo da Sprint Planning é selecionar o que será feito na Sprint e isso deve ser realizado em conjunto pelo Product Owner e Developers (SCHWABER & SUTHERLAND, 2020, p. 10). Eles devem selecionar os itens do Product Backlog para incluí-los na Sprint Atual e podem refinar esses itens durante esse processo para elevar a compreensão deles. E apesar das dificuldades naturais para o Time Scrum avaliar quantos e quais itens podem ser concluídos em uma Sprint, a sua confiança tende a aumentar conforme acumularem um histórico cada vez maior sobre o desempenho passado e conforme a Definição de Pronto estiver cada vez mais clara e consolidada.

O terceiro passo da Sprint Planning é conduzido pelos Developers e consiste em planejar o trabalho que deve ser realizado para produzir um Incremento que atenda à Definição de Pronto para cada um dos itens selecionados do Product Backlog e adicionados à Sprint atual (SCHWABER & SUTHERLAND, 2020, p. 10). Isso geralmente é feito decompondo os itens selecionados em itens menores (de um dia ou menos), mas a forma como isso é realizado fica a critério exclusivo dos Developers.

É importante destacar que a Sprint Planning tem um Timebox definido e sua duração máxima é de 8 horas para uma Sprint de um mês e esse evento geralmente é mais curto para Sprints que também sejam mais curtas (SCHWABER & SUTHERLAND, 2020, p. 10). Ao final desse evento, deve ter sido produzido o Sprint Backlog e este é composto pela Meta da Sprint, os itens do Product Backlog selecionados para a Sprint e o plano elaborado para entregá-los.

A Daily Scrum é o único evento da Sprint que é repetitivo, ou seja, ocorre mais que uma vez na mesma Sprint. Trata-se de um evento de 15 minutos a ser realizado todos os dias úteis e, para reduzir a complexidade, sugere-se que ocorra sempre no

mesmo horário e local. A finalidade deste evento é permitir a inspeção do progresso em direção à Meta da Sprint e adaptar o Sprint Backlog, conforme isso for necessário (SCHWABER & SUTHERLAND, 2020, p. 10). Esse é um evento dos Developers e só requer a presença destes. O Scrum Master e o Product Owner só participam se estiverem trabalhando ativamente nos itens da Sprint Backlog e, nesse caso, sua participação também é na condição de Developer.

A Daily Scrum contribui para melhorar a comunicação, identificar impedimentos, acelerar a tomada de decisões e reduzir a necessidade de reuniões adicionais. Segundo SCHWABER & SUTHERLAND (2020, p. 10), os Developers devem concentrar-se no progresso rumo à Meta da Sprint e nas ações para o próximo dia de trabalho porque isso fortalece o foco e o autogerenciamento.

A Sprint Review é o penúltimo evento da Sprint e ele tem um Timebox definido: sua duração máxima é de 4 horas para uma Sprint de um mês e geralmente é mais curto para Sprints que também sejam mais curtas (SCHWABER & SUTHERLAND, 2020, p. 11). A finalidade deste evento é inspecionar o resultado produzido na Sprint atual e propor adaptações para as próximas. A dinâmica deste evento consiste na apresentação conduzida pelo Time Scrum para os principais stakeholders com foco nos resultados obtidos e no progresso rumo à Meta do Produto.

Durante a Sprint Review, os stakeholders participam ativamente da revisão de tudo o que foi produzido e avaliam o que isso mudou em seu ambiente. O Time Scrum e os stakeholders colaboram para definir o que fazer a seguir e o Product Backlog pode ser ajustado para a inclusão de novos itens ou a exclusão daqueles que não façam mais sentido. O mais importante é compreender que a Sprint Review deve ser uma sessão de trabalho com muitas trocas entre os envolvidos e não deve ser limitada a uma reunião de apresentação de resultados (SCHWABER & SUTHERLAND, 2020, p. 11).

A Sprint Retrospective é o último evento da Sprint e ele tem um Timebox definido: sua duração máxima é de 3 horas para uma Sprint de um mês e geralmente é mais curto para Sprints que também sejam mais curtas (SCHWABER & SUTHERLAND, 2020, p. 11). A finalidade deste evento é planejar como aumentar a qualidade e a eficácia. A dinâmica deste evento consiste na inspeção do que foi realizado na última Sprint no que se refere a indivíduos, interações, processos, ferramentas e Definição de Pronto. As mudanças propostas que forem entendidas com

maior potencial de impacto positivo são implementadas o mais rapidamente possível e podem até já serem adicionadas ao Sprint Backlog da próxima Sprint.

2.3.4. Artefatos Scrum

Os artefatos do Scrum representam trabalho ou valor e devem ser utilizados para maximizar a transparência das informações, viabilizar a inspeção por parte de todos os envolvidos e permitir que se realize adaptações a partir de uma mesma base (SCHWABER & SUTHERLAND, 2020, p. 11). Cada artefato possui um compromisso para maximizar a transparência e o foco e permitir a medição do progresso:

- Product Backlog: o compromisso é a Meta do Produto.
- Sprint Backlog: o compromisso é a Meta da Sprint.
- Incremento: o compromisso é a Definição de Pronto.

O Product Backlog é a fonte de trabalho do Time Scrum e se materializa como uma lista ordenada do que é proposto para melhorar o produto (SCHWABER & SUTHERLAND, 2020, p. 12). O seu conteúdo é mutável, uma vez que tende a haver a inclusão e exclusão de itens. Além disso, o *Product Backlog refinement* é uma atividade contínua de quebrar os itens e incluir informações adicionais para se chegar a itens menores e mais claros. O compromisso do Product Backlog é a Meta do Produto e ela descreve um estado futuro que se deseja para o produto e serve como um alvo para o Time Scrum direcionar os seus esforços.

O Sprint Backlog é um plano elaborado pelos Developers acerca do trabalho que pretendem realizar durante a Sprint para atingir a sua meta (SCHWABER & SUTHERLAND, 2020, p. 12). O seu conteúdo é mutável, uma vez que é elaborado durante a Sprint Planning e vai sendo atualizado durante toda a Sprint conforme mais vai sendo aprendido. Esse artefato deve conter detalhes suficientes para orientar a inspeção do progresso do trabalho na Daily Scrum e representa uma imagem que dá ampla visibilidade, em tempo real, ao trabalho desenvolvido pelos Developers e ele é composto pela Meta da Sprint (por que), os itens do Product Backlog selecionados para a Sprint (o que) e as ações a serem realizadas para entregar o Incremento (como).

Um incremento é uma entrega concreta e representa um avanço em direção à Meta do Produto (SCHWABER & SUTHERLAND, 2020, p. 13). Cada novo incremento que é liberado deve ser adicionado aos incrementos anteriores e deve ser verificado para garantir que funcionem juntos. Para que possa fornecer valor, é exigido que o

incremento seja utilizável. É importante ressaltar que a Sprint Review não deve ser interpretada como um marco para entregar valor e os incrementos que forem criados podem ir sendo liberados aos stakeholders ao longo da Sprint e a soma deles é apresentada na Sprint Review. Também é importante destacar que um incremento só pode ser liberado se atender à Definição de Pronto, que consiste em uma descrição formal do estado exigido para o incremento para que se considere como atendidos os critérios de qualidade exigidos para o produto. Um incremento que não atende à Definição de Pronto não pode ser apresentado na Sprint Review e deve retornar ao Product Backlog para ser selecionado novamente em uma Sprint futura.

3. PROPOSTA DE SOLUÇÃO DA SITUAÇÃO-PROBLEMA

Um estudo de caso foi selecionado com o objetivo de analisar a utilização do framework Scrum no desenvolvimento de um software e o seu impacto sobre a entrega de valor à organização. Esse projeto foi selecionado considerando a principal necessidade da empresa: que o sistema desenvolvido contemplasse as principais funcionalidades para a gestão do portfólio e do ciclo de vida dos projetos.

Para atingir o propósito do artigo, a escolha metodológica foi direcionada para uma abordagem qualitativa, utilizando o método de estudo de caso. Essa seleção visa a análise e interpretação dos “fenômenos individuais, organizacionais, sociais, políticos ou de grupo” (YIN, 2005, p. 20), permitindo a exploração das variáveis em estudo por meio de entrevistas e análise documental.

A abordagem é de caráter descritivo, pois descreve as características de uma população, característica ou experiência, promovendo uma nova perspectiva sobre uma realidade já existente (GIL, 2010).

No âmbito dos estudos de caso existem seis fontes distintas para a coleta de evidências: documentos, registros arquivados, entrevistas, observação direta, observação participante e artefatos físicos. (YIN, 2005). Dentre as fontes citadas, foi realizada a análise de documentos, observação direta, observação participante e entrevistas.

Para facilitar o entendimento de como a pesquisa foi conduzida, na Figura 1 encontra-se um fluxograma simplificado com as principais atividades realizadas e na sequência uma breve explicação dos principais pontos.

Figura 1 – Materiais e métodos do estudo.

Na etapa inicial da pesquisa, como procedimento técnico, foi conduzida uma revisão bibliográfica para identificar os aspectos relevantes a serem avaliados no objeto de estudo. Os principais temas explorados foram o processo de desenvolvimento de software, ferramentas e frameworks ágeis e o framework Scrum.

Para analisar o impacto da utilização do Scrum no desenvolvimento de software, foi selecionado um projeto que ainda iria iniciar para ser possível acompanhar todo o seu ciclo de vida, desde a estruturação inicial do time, toda a sua execução e a análise tanto da dinâmica de trabalho estabelecida quanto do resultado final para a organização. Durante a execução do projeto, o time de Scrum foi documentando todo o processo e compartilhando estes registros.

Os pesquisadores acompanharam de perto o desenvolvimento do sistema e com isso também realizaram registros relevantes para a conclusão da pesquisa. Com a finalização do projeto, foram realizadas entrevistas com o Time Scrum e com o cliente para identificar possíveis lacunas na aplicação do Scrum no desenvolvimento do projeto e delas resultaram dados quantitativos e qualitativos para avaliar como se deu a utilização do Framework Scrum e o resultado produzido para a organização. E para avaliar os resultados, foram analisados todos os documentos desenvolvidos ao longo do projeto e a entrevista realizada com os partícipes do projeto: Product Owner, Scrum Master e Developers.

4. RESULTADOS E DISCUSSÃO

Este tópico tem por finalidade apresentar e interpretar os resultados obtidos no decorrer deste estudo, oferecendo uma análise dos dados coletados e das observações realizadas. Ele inicia com uma visão geral sobre a empresa onde foi realizado o estudo de caso deste artigo. Em seguida, apresenta o estudo de caso analisado e a composição da equipe que o executou. Por fim, explora a solução que foi desenvolvida, apresenta os dados levantados e faz a análise dos resultados.

4.1. APRESENTAÇÃO DA EMPRESA

A empresa tratada como objeto de estudo desta pesquisa, denominada Alfa, busca promover a inovação, a pesquisa científica, o desenvolvimento tecnológico e a integração entre empresas, instituições de ensino e pesquisa, startups e profissionais da área de biotecnologia, agronegócio e tecnologia da informação.

Essa empresa detém um vasto espaço territorial para abrigar empresas e organizações que atuam em diferentes setores: saúde, agroindústria, tecnologia, entre outros. Ela persegue contribuições para a economia local através da promoção da pesquisa, desenvolvimento e inovação e procura atrair investimentos e talentos nessas áreas, criar empregos e fomentar a colaboração entre diferentes segmentos.

Há muitas frentes de trabalho paralelas e múltiplos objetivos a serem atingidos, tais como: a criação de uma universidade, a instalação de um hospital de alta complexidade, expansão de loteamentos residenciais e áreas comerciais. Nesse contexto, o Escritório de Gerenciamento de Projetos (EGP) ou Project Management Office (PMO) da empresa enfrenta a seguinte situação-problema: não possui um software para o gerenciamento do portfólio e nem para o acompanhamento o ciclo de vida dos projetos de maneira sistêmica e organizada.

O gerenciamento de portfólio e de projetos pressupõe seguir etapas e aplicação de boas práticas. O PMO em estudo não tem isso consolidado e é comum seguir para a execução sem cumprir minimamente o esperado nas etapas de iniciação e planejamento. Isso traz diversos prejuízos como o direcionamento errado dos esforços, retrabalho e a falta de organização.

4.2. COMPOSIÇÃO DO TIME SCRUM

Uma das iniciativas da empresa Alfa é o desenvolvimento de trainees na área da Tecnologia da Informação com um programa de captação anual. Este time passa a atuar nas empresas residentes do ecossistema e na própria empresa Alfa durante os seis primeiros meses. Dentre os selecionados para o programa no ano de 2023, 04 (quatro) trainees foram alocados para atuar em um projeto para o PMO da empresa Alfa, conforme a necessidade apresentada pelo responsável da área.

O sistema foi desenvolvido utilizando o framework Scrum. Para fins didáticos e para corroborar com a narrativa do desenvolvimento do sistema, seguem as distribuições dos papéis considerando as terminologias do framework:

- **Product Owner:** esse papel foi assumido pelo colaborador responsável pelo Programa Trainee e sua responsabilidade foi definir e priorizar o que precisava ser feito e representar os interesses das partes interessadas.
- **Scrum Master:** esse papel foi assumido de forma rotativa pelos trainees e suas responsabilidades incluíam facilitar o uso eficaz do Scrum, remover obstáculos e ajudar a equipe a melhorar continuamente (esse papel era assumido por um membro diferente do time a cada nova semana).
- **Developers:** foi composto por quatro programadores selecionados para o Programa de Trainees de 2023 e foram os responsáveis por realizar o trabalho de desenvolvimento do novo sistema para o PMO.

Esses papéis foram distribuídos visando promover a transparência, inspeção e adaptação contínua durante o desenvolvimento do produto demandado. A solicitação e especificação da necessidade foi apresentada pelo responsável pelo PMO da empresa Alfa e consistia no desenvolvimento de um sistema de workflow. Espera-se que o time Scrum realizasse o desenvolvimento desse sistema com as principais funcionalidades requeridas para a gestão do portfólio e do ciclo de vida dos projetos.

4.3. APRESENTAÇÃO DO ESTUDO DE CASO

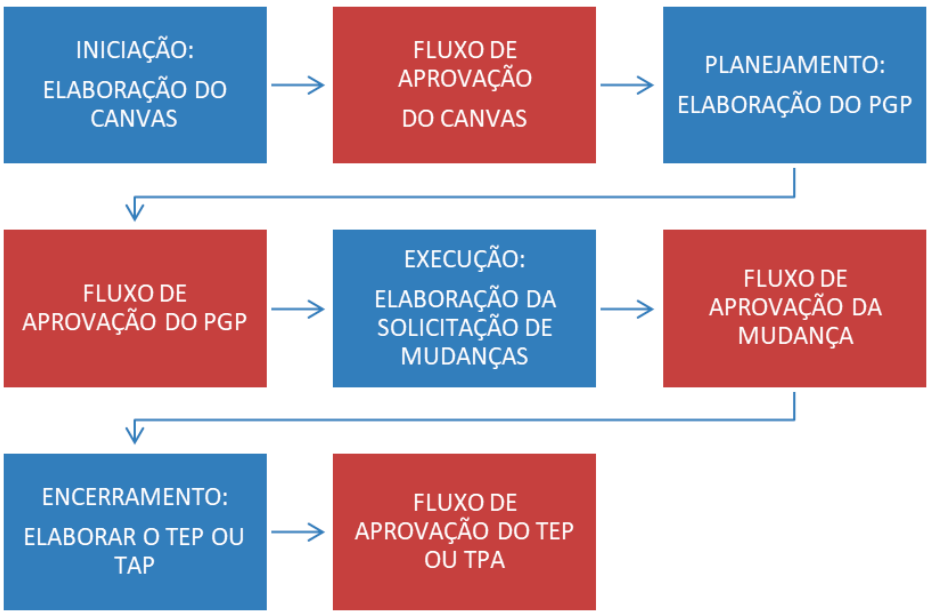
O PMO da empresa Alfa ainda está em processo de implantação de padrões e de consolidação de boas práticas de gestão de projetos. O sistema proposto foi concebido para ser aderente ao processo padronizado que está sendo implementado e possibilitar a gestão de projetos da empresa de maneira sistêmica e organizada.

O software proposto centra-se na aprovação dos documentos produzidos no processo de gerenciamento de projetos. O PMO utilizava pastas compartilhadas e planilhas eletrônicas para a gestão dos projetos. Esse novo sistema foi concebido para trazer maior padronização, organização e confiabilidade às informações. A presente seção irá detalhar tanto o processo atual de gerenciamento de projetos na empresa Alfa quanto o software que foi proposto para sistematizar o workflow de aprovações dos documentos ligados a ele.

4.3.1. Processo atual

O processo escolhido para ser sistematizado com o apoio do time Scrum foi o workflow de aprovações dos documentos que são produzidos nas quatro fases do gerenciamento de projetos no PMO da empresa Alfa. Esses documentos consistem em planilhas elaboradas com o uso do Microsoft Excel seguindo *templates* padronizados e que são compartilhadas em um servidor de arquivos. As aprovações desses documentos são realizadas via troca de mensagens por e-mail com as pessoas que devem apreciá-los e às quais esses documentos são anexados. Esse é um processo bastante moroso e com rastreabilidade precária. A Figura 2 ilustra o processo que o sistema deveria atender.

Figura 2 – Workflow de aprovações do PMO da empresa Alfa.



Através desse fluxograma, é possível perceber que esse processo está dividido em quatro fases distintas: iniciação, planejamento, execução e encerramento. Essas são as fases típicas de um modelo de gerenciamento de projetos inspirado no modelo *waterfall* (cascata). É esse o modelo que serviu como inspiração primária para a elaboração da metodologia empregada no PMO da empresa Alfa. Também é possível perceber que há documentos que são produzidos em cada fase e que esses documentos precisam passar por um processo de aprovação.

A seguir, serão apresentadas informações adicionais sobre esse processo para destacar os principais marcos, responsabilidades e entregas em cada etapa. Isso permitirá entender melhor o processo e esclarecer as principais lacunas e necessidades que deveriam ser sanadas com a implantação do sistema de workflow no PMO da empresa Alfa.

Esse processo é iniciado com um setor qualquer da empresa Alfa abrindo uma nova demanda para o PMO. A abertura da demanda é realizada por aquele que assume o papel de proponente do projeto e o fluxo que precisa ser cumprido para finalizar essa etapa preliminar envolve os seguintes passos:

1. **Formalização da demanda:** o solicitante abre uma demanda para o PMO via sistema de chamados para propor um novo projeto.
2. **Aprovação da demanda**: dentro do próprio sistema de chamados o PMO avalia o pedido e, se for aprovado, a adiciona no controle do portfólio de projetos (uma planilha do Excel) com o status de “iniciação” e designa um Gestor de Projetos para auxiliar o proponente do projeto a elaborar o Project Canvas.

Tendo sido cumprida a etapa anterior para formalização e aprovação de demanda aberta para o PMO, passa-se para a fase de iniciação do projeto. Nessa fase, há dois passos importantes a serem dados:

1. **Elaboração do Project Canvas:** com o auxílio do Gestor designado, o proponente do projeto deve elaborar o Project Canvas utilizando um *template* padrão em planilha do Excel fornecido pelo PMO.
2. **Aprovação do Project Canvas:** a aprovação desse documento é realizada via e-mail na seguinte ordem: PMO e Patrocinador. O PMO avalia a estrutura do documento e se os campos foram preenchidos com as informações pertinentes ao seu propósito e o patrocinador avalia as informações e aprova o início da fase de planejamento do projeto.

Tendo sido cumprida a fase de iniciação, passa-se para a fase de planejamento do projeto. Nessa fase, há dois passos importantes a serem dados para a construção e aprovação do Plano de Gerenciamento do Projeto (PGP):

1. **Elaboração do PGP:** o PMO designa um Gestor de Projetos para conduzir a construção do Plano de Gerenciamento do Projeto (PGP) e normalmente é o mesmo que apoiou o desenvolvimento do Project Canvas (não sendo obrigatório) e é ele que avalia quais documentos devem compor o PGP para atender às necessidade de detalhamento das seguintes áreas de conhecimento do projeto: partes interessadas, escopo, tempo, aquisições, custos, qualidade, recursos, comunicação e riscos. Todos esses documentos são elaborados a partir de *templates* padronizados disponibilizados pelo PMO na forma de planilhas do Excel.
2. **Aprovação do PGP:** a aprovação desse documento é realizada via e-mail na seguinte ordem:
    a. **PMO:** avalia a estrutura do PGP e confere se os documentos que o compõem foram elaborados em conformidade com os padrões estabelecidos e com as informações pertinentes ao seu propósito.
    b. **Proponente do Projeto:** avalia o conjunto de documentos que compõem o PGP para assegurar que contemplam de forma clara e completa o que está sendo proposto.
    c. **Patrocinador:** analisa o conjunto de documentos que compõem o PGP, avalia os benefícios esperados para o negócio, o orçamento proposto e aprova o início da execução do projeto.

Durante a fase de execução do projeto podem ocorrer solicitações de mudanças e a formalização da conclusão de pacotes previstos no escopo do projeto. Para estas atividades existem dois documentos: a Solicitação de Mudança (SM) e o Termo de Aceite de Entrega (TAE). A SM é um documento empregado para obter a permissão formal para alterar os planos que compõem o PGP para responder às mudanças que impactaram o projeto. O TAE é um documento que formaliza uma entrega parcial do escopo acordado e que atendeu os requisitos. O fluxo de aprovação desses dois documentos ocorre via e-mail e segue o mesmo fluxo de aprovação do PGP, mas cabe ao Gestor do Projeto identificar outros stakeholders e áreas técnicas que devam ser adicionados.

Um projeto pode ser encerrado de duas maneiras e para ambos também existe um *template* padrão em planilha do Excel que é fornecido pelo PMO: o Termo de encerramento do Projeto (TEP) e o Termo de Projeto Abortado (TPA). O TEP deve ser emitido e aprovado para projetos finalizados e com as entregas concluídas conforme o acordado. Já o TPA deve ser emitido e aprovado quando decide-se interromper a execução de um projeto sem que se tenha atingido seu objetivo. A aprovação desses dois documentos ocorre via e-mail e segue o mesmo fluxo de aprovação do PGP, mas cabe ao Gestor identificar outros stakeholders e áreas técnicas que devam ser adicionadas.

4.3.2. Desenvolvimento do sistema proposto

O PMO da empresa Alfa propôs o desenvolvimento de um sistema de workflow para realizar a gestão das aprovações de documentos contemplados no ciclo de vida dos projetos. Os principais benefícios esperados com a construção e implantação desse sistema eram: a redução de tempo total requerido para a aprovação de cada documento, maior controle e rastreabilidade sobre alterações realizadas e aprovadas e consolidação do armazenamento desses documentos em uma plataforma única e dentro de uma sistemática padronizada para permitir o resgate imediato e seguro dos mesmos.

Não foi necessário terceirizar o desenvolvimento do sistema, pois ele foi desenvolvido internamente pelo time de trainees e isso diminuiu significativamente o custo do projeto. O desenvolvimento do sistema ocorreu com a utilização do framework Scrum, tendo sido designado um Product Owner e o time foi composto por quatro Developers. O Scrum Master não era fixo: a cada sprint, um Developer assumia esse papel até que todos o tivessem exercido e, então, esse ciclo de revezamento era reiniciado. Dessa forma, cada Developer assumiu o papel de Scrum Master em paralelo com as atividades de desenvolvimento algumas vezes ao longo do processo de desenvolvimento do sistema.

Os ciclos de desenvolvimento ocorreram em sprints de uma ou duas semanas, a depender das tarefas que precisavam ser realizadas durante o período: quanto maior o risco, menor o tempo de duração da sprint. Foram treze sprints ao longo dos seis meses de desenvolvimento, que era o tempo previsto para a conclusão do programa de trainees.

Durante as sprints 01 a 08, o time de Developers não sofreu alterações. Mas a partir da Sprint 09, esse time foi reduzido para metade do número inicial. Isso ocorreu por conta de demandas internas da empresa Alfa e de decisão tomada pela diretoria em realocar dois Developers para outras atividades da organização. Em função disso, nas cinco últimas sprints, o Time Scrum pôde contar apenas com dois dos quatro Developers que iniciaram o projeto.

4.4. APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Um questionário foi elaborado e aplicado para avaliar a aderência das práticas adotadas pelo Time Scrum no desenvolvimento do sistema proposto com o que é prescrito no Framework Scrum. Essas perguntas foram encaminhadas para o Time Scrum: os quatro Developers e o Product Owner. Das 28 perguntas desse questionário, 21 permitiam como resposta apenas “Sim” ou “Não”. As respostas a essas perguntas objetivas foram tabeladas para permitir uma análise quantitativa. As outras sete perguntas abertas foram incluídas para permitir uma análise qualitativa do resultado do estudo de caso. A Tabela 1 apresenta todas as perguntas e inclui o percentual de respostas “Sim” e “Não” para cada uma das perguntas objetivas.

Tabela 1 – Formulário de pesquisa e resultados de questões objetivas.

PERGUNTA	SIM	NÃO
1. O Time Scrum teve um **Product Owner** claramente designado?	100%	-
2. O **Product Owner** foi efetivo na definição e comunicação da Meta do Produto?	100%	-
3. O **Product Owner** foi efetivo na criação, priorização e comunicação dos itens do Product Backlog e em manter esse artefato acessível ao Time Scrum e outros stakeholders?	100%	-
4. O Time Scrum teve um **Scrum Master** claramente designado?	100%	-
5. O **Scrum Master** apoiava o Product Owner, os Developers e stakeholders da organização para ajudá-los na aplicação das práticas do Scrum?	100%	-
6. O **Scrum Master** apoiava a equipe na remoção de	100%	-

obstáculos?		
7. Houve acúmulo de **papéis** por parte de membros do **Time Scrum** (ex.: Scrum Master e Developer)? Por que? Houve prejuízos decorrentes disso?	Questão Aberta	
8. Houve **troca** ou **redução** de membros do **Time Scrum**? Quais? Por que? Houve prejuízos decorrentes disso?	Questão Aberta	
9. Há uma cultura de **transparência**, **inspeção** e **adaptação** sendo promovida em toda a organização?	100%	-
10. Havia **transparência** no trabalho que estava sendo realizado tanto para o Time Scrum quanto para os clientes e demais stakeholders?	100%	-
11. As **inspeções** realizadas na Daily Scrum, Sprint Review e Sprint Retrospective eram produtivas no sentido de levantar melhorias?	100%	-
12. As melhorias levantadas através das **inspeções** eram implementadas subsequentemente para **adaptar** o produto ou os processos de trabalho com ganhos para o Time Scrum?	100%	-
13. O **Backlog do Produto** era visível e atualizado e refletia as prioridades do cliente?	100%	-
14. A duração de uma Sprint foi fixada em quanto tempo e se manteve o mesmo para todas as Sprints?	Questão Aberta	
15. Qual era a duração da Sprint Planning?	Questão Aberta	
16. Na **Sprint Planning,** o Time Scrum sempre conseguia estabelecer uma Meta da Sprint para comunicar o valor a ser entregue ao término da Sprint?	100%	-
17. O Time Scrum utilizava a **Sprint Planning** para selecionar os itens do Product Backlog a serem desenvolvidos na Sprint?	100%	-
18. O Time Scrum utilizava a **Sprint Planning** para planejar o trabalho a ser realizado para produzir um Incremento que atendesse à **Definição de Pronto** para cada um dos itens selecionados do Product Backlog?	100%	-
19. A **Definição de Pronto** descrevia claramente o estado formal para um Incremento atender aos critérios de qualidade exigidos para o produto e isso era conhecido pelo Time Scrum e pelo cliente?	100%	-
20. As **Daily Scrum** eram realizadas regularmente no mesmo horário e local e todos os membros do Time Scrum participavam ativamente?	100%	-

21. Qual era a duração da Sprint Review?	Questão Aberta	
22. A **Sprint Review** era realizada ao final de cada Sprint para inspecionar o trabalho realizado e obter feedback do cliente ou stakeholders?	100%	-
23. Haviam sugestões de melhoria nas **Sprint Reviews** e eram implementadas com ganhos efetivos para os processos e para o produto na Sprint seguinte?	100%	-
24. Qual era a duração da Sprint Retrospective?	Questão Aberta	
25. A **Sprint Retrospective** era realizada ao final de cada Sprint com foco na análise do Time Scrum sobre como aumentar a qualidade e a eficácia?	80%	20%
26. As sugestões feitas nas **Sprint Retrospectives** eram implementadas com ganhos efetivos para os processos e para o produto na Sprint seguinte?	80%	20%
27. O Time Scrum conseguia entregar **Incrementos** de software funcionais no final de cada Sprint?	100%	-
28. Quantas Sprints e quanto tempo foi necessário para a liberação das primeiras funcionalidades do software para o cliente em ambiente de produção?	Questão Aberta	

Observa-se que o Time Scrum entende que teve papéis definidos e devidamente designados. Em relação ao papel do **Product Owner**, os pesquisados foram unânimes em afirmar que o mesmo estava claramente designado e que foi efetivo na definição e comunicação da Meta do Produto, na criação, priorização e comunicação dos itens do Product Backlog e em manter esse artefato acessível ao Time Scrum e outros stakeholders.

Os pesquisados também foram unânimes ao afirmar que um **Scrum Master** claramente designado. Também afirmaram que o Scrum Master apoiava o Product Owner, os Developers e stakeholders da organização para ajudá-los na aplicação das práticas do Scrum e apoiava a equipe na remoção de obstáculos. Mas é importante salientar que houve acúmulo de papéis com um Developer assumindo o papel de Scrum Master e que esse papel era atribuído a um Developer distinto a cada nova Sprint. O posicionamento dos entrevistados a esse respeito foi de que não houve prejuízos decorrentes dessa prática, uma vez que essa rotatividade foi devidamente planejada e pensada com a finalidade proporcionar a oportunidade dos Developers vivenciarem esse outro papel.

Muito embora não exista uma proibição referente ao acúmulo de papéis e que a prática em equipes pequenas seja relativamente comum, é importante ressaltar que desempenhar mais de um papel simultaneamente pode trazer sobrecarga ao membro da equipe que está exercendo mais de uma função. Outro aspecto importante diz respeito à rotatividade no desempenho da função de Scrum Master. Uma vez que esse papel foi assumido por vários membros no decorrer do desenvolvimento, é válido supor que a produtividade do time pode ter sofrido impactos por precisar absorver essa mudança a cada Sprint. Também é importante destacar que o número de Developers passou de quatro para dois a partir da nona Sprint e que, a partir daí, um deles precisava acumular os papéis de Developer e de Scrum Master.

No tocante aos três pilares fundamentais do Scrum, **transparência**, **inspeção** e **adaptação**, todos os pesquisados afirmaram que esses pilares estão presentes e são promovidos em toda a organização, que as inspeções realizadas na Daily Scrum, Sprint Review e Sprint Retrospective eram produtivas no sentido de levantar melhorias e que as melhorias levantadas através das inspeções eram implementadas subsequentemente para adaptar o produto ou os processos de trabalho com ganhos para o Time Scrum.

Destaque-se que os pilares do Scrum são fundamentais no decorrer da evolução do projeto na medida em que a **transparência** promove a clareza do que está sendo desenvolvido, **inspecionar** permite verificar constantemente as melhorias e ajustes que precisam ser incorporados ao desenvolvimento e **adaptar** fecha o ciclo ao tornar o time capaz de promover as mudanças necessárias. Dessa forma, entende-se que os membros do Time Scrum mantinham clareza de entendimento acerca da Meta do Produto e da Meta da Sprint, das atividades requeridas para se atingi-las, que a cada Sprint eram feitas as inspeções necessárias para garantir que a entrega estivesse em conformidade com a meta e com a Definição de Pronto e que melhorias de processo eram incorporadas à Sprint subsequente.

No que se refere ao **Backlog do Produto**, os integrantes afirmaram com unanimidade que o mesmo manteve-se visível e atualizado e refletia as prioridades do cliente. Mas é importante registrar o fato de que o modelo de documento adotado pelo Product Owner era uma planilha bastante limitada, que a coluna de prioridade sequer era preenchida e que esse documento não permanecia em local acessível para o cliente e que suas atualizações não eram compartilhadas regularmente com ele.

Com relação à **Sprint Planning**, a equipe informou que o prazo médio da reunião era de duas a três horas, uma média de tempo condizente com o que é preconizado com sprints de uma a duas semanas. A equipe utilizava esse momento para estabelecer e pontuar quais atividades seriam desenvolvidas no decorrer da sprint, estabelecendo uma meta clara sobre qual seria a entrega de valor ao término da Sprint.

Os pesquisados também foram unânimes ao afirmar que a **Definição de Pronto** descrevia claramente o estado formal para um Incremento atender aos critérios de qualidade exigidos para o produto e isso era conhecido pelo Time Scrum e pelo cliente. Ressalte-se que aqui há uma objeção importante por parte da percepção do cliente que será apresentada ainda nesta seção.

Também houve unanimidade entre os pesquisados ao afirmarem que as **Daily Scrum** eram realizadas regularmente no mesmo horário e local e todos os membros do Time Scrum participavam ativamente. Até onde pôde ser verificado, esse é um dos eventos sobre o qual não há qualquer ressalva a ser registrada.

Todos os pesquisados também afirmaram que a **Sprint Review** era realizada ao final de cada Sprint para inspecionar o trabalho realizado e obter feedback do cliente ou stakeholders e que haviam sugestões de melhoria que eram apresentadas e eram implementadas com ganhos efetivos para os processos e para o produto na Sprint seguinte. De fato, esse evento teve a sua realização regular mantida e sua duração fixa era de uma hora, mesmo no caso de Sprint de duas semanas. Também há ressalvas quanto à sua efetividade por parte do cliente e isso será explorado na sequência.

Houve uma divergência entre os pesquisados no caso da **Sprint Retrospective**: 80% responderam que esse evento era realizado ao final de cada Sprint com foco na análise do Time Scrum sobre como aumentar a qualidade e a eficácia e que as sugestões feitas eram implementadas com ganhos efetivos para os processos e para o produto na Sprint seguinte. Mas 20% respondeu que isso não ocorria: que esse evento não era realizado e/ou que não produziu ganhos efetivos.

Os pesquisados foram unânimes ao responder que o Time Scrum conseguia entregar Incrementos de software funcionais no final de cada Sprint. Mas aqui há uma divergência frontal com a percepção do cliente e isso também será explorado na sequência desta seção. Mas é importante destacar que as primeiras funcionalidades só foram liberadas em ambiente de testes na quinta Sprint (48 dias após o início do projeto) e que as funcionalidades só foram liberadas em ambiente de produção após a conclusão da 13ª e última Sprint (quatro meses após o início do projeto).

Além das percepções do Time Scrum que foram capturadas através do formulário de pesquisa que foi preenchido e cujos dados foram apresentados e analisados, também foram capturadas as percepções do cliente. Este foi representado pelo Supervisor, por uma Analista de Projetos e pelo time de oito Gestores de Projetos do PMO. O Supervisor foi quem trabalhou junto com o Product Owner para a concepção inicial do projeto e o estabelecimento da Meta do Produto e tornou-se seu ponto de contato ao longo de todo o trabalho para priorização do Product Backlog e para a revisão das funcionalidades desenvolvidas durante a Sprint Review. A Analista de Projetos também passou a ser um ponto de contato direto do Product Owner para direcionamento do trabalho dos Developers e também participava da Sprint Review para revisar as funcionalidades concluídas. Os Gestores de Projetos foram envolvidos mais fortemente quando passou a haver a liberação de funcionalidades em ambiente de testes e o seu papel era avaliar a sua aderência às necessidades do cliente, registrar falhas localizadas e sugerir melhorias.

Aqui é importante tecer uma análise crítica e um pouco mais detalhada sob a perspectiva do cliente com ênfase naqueles pontos em que houve baixa aderência entre as práticas adotadas pelo Time Scrum e o que é prescrito do guia do Framework Scrum. Essa análise será apresentada a seguir, dividindo-a em quatro blocos distintos: Pilares, Time Scrum, Eventos e Artefatos.

No que se refere aos pilares do Scrum, observou-se que houve baixa efetividade em sua aplicação prática. O poder da transparência, inspeção e adaptação ficaram prejudicados especialmente pelo limitado envolvimento do cliente. Exemplo disso foi a baixa efetividade do Product Owner para dar visibilidade do Product Backlog ao cliente, já que esse documento não ficava em local acessível para ele e tampouco a sua atualização era compartilhada regularmente. Isso limitava a capacidade de inspeção deste documento e, consequentemente, de contribuir para a sua adaptação e melhoria. Além disso, o contato direto do cliente com o Product Owner, com o Scrum Master e com os Developers foi demasiadamente concentrado e limitado à Sprint Review e isso se mostrou insuficiente para todas as trocas que dependem do envolvimento e engajamento do cliente. Esse evento costumava ser realizado na forma de uma reunião de uma hora ao final de cada Sprint e era só nesse momento que o cliente tinha o primeiro contato com as funcionalidades desenvolvidas pelo Time Scrum naquela Sprint. Seria mais produtivo se ele fosse envolvido ao longo da Sprint para ter

a oportunidade de inspecionar os avanços e contribuir para a adaptação e melhoria do que estava sendo desenvolvido.

No que se refere ao Time Scrum, foi possível observar o exercício deficitário dos papéis de Product Owner e de Scrum Master e uma volatilidade significativa na composição do time de Developers. O Product Owner falhou especialmente no envolvimento e interação com o cliente e na gestão do Product Backlog, conforme já descrito. E o exercício do papel do Scrum Master ficou bastante prejudicado porque esse papel sofreu uma alta rotatividade e envolveu acúmulo de funções: a cada Sprint esse papel era exercido por um Developer distinto e isso trouxe prejuízos para o seu desempenho tanto em função da troca contínua quando por ser exercido por membro do Time Scrum que acumulava as funções de Developer. Além disso, o time de Developers também foi afetado pela decisão da organização que priorizou outras demandas e realocou dois dos quatro Developers ao final da oitava Sprint e reduziu pela metade a capacidade deles nas últimas cinco Sprints.

No que se refere aos eventos do Scrum, foi possível observar deficiências no planejamento e execução da Sprint, Sprint Planning, Sprint Review e Sprint Retrospective. A duração da Sprint foi alterada por diversas vezes com impacto sobre a previsibilidade e sustentabilidade do ritmo de trabalho do time. A Sprint Planning não era precedida por um alinhamento mais sólido do Product Owner com o cliente para tratar da priorização dos itens do Product Backlog e/ou para a definição da Meta da Sprint: esse alinhamento era realizado apenas esporadicamente e de forma demasiadamente rápida e superficial. A Sprint Review tinha o tempo limitado de uma hora mesmo para em Sprints de duração de duas semanas e era frequente o cliente ter o primeiro contato com as funcionalidades desenvolvidas somente nessa ocasião e haver diversas dificuldades técnicas e erros que prejudicava a apresentação das mesmas. E apesar de não participar da Sprint Retrospective, a percepção do cliente é que esse evento teve eficácia limitada em função de que pedidos realizados ao Time Scrum não foram atendidos e um exemplo disso foi a solicitação feita pelo cliente ao final da Sprint 1 e reforçada posteriormente em outras ocasiões para que as funcionalidades desenvolvidas fossem disponibilizadas para a equipe do PMO em um ambiente de testes e o Time Scrum só a atendeu muito tardiamente (na Sprint 10).

No que se refere aos artefatos do Scrum, foi possível constatar que houve a adoção de um modelo deficitário de Product Backlog e de Sprint Backlog e que sua atualização e compartilhamento também apresentaram deficiências. Além disso,

também foi observada uma baixa consistência na Definição de Pronto para os Incrementos.

O Product Backlog era uma planilha contendo apenas as seguintes colunas: Tipo da Tarefa (Melhoria, Bug, Ajuste), Descrição da Tarefa, Sprint e Prioridade (Alta, Média, Baixa). O primeiro problema aqui está na própria estrutura do documento: não está organizado por Estória de Usuário e sim por tarefa e isso o caracteriza como um Backlog da Sprint e não um Backlog do Produto e a coluna prioridade sequer era preenchida. Além disso, não havia a necessária transparência em relação a esse documento já que não ficava em local acessível para o cliente e tampouco era atualizado e compartilhado regularmente.

Também cabe aqui uma análise crítica acerca da Definição de Pronto sob a perspectiva do cliente. Durante a primeira Sprint Review, o cliente solicitou esclarecimentos ao Product Owner acerca da Definição de Pronto porque ela não havia sido estabelecida. Então, firmaram acordo para considerar uma funcionalidade do sistema concluída somente depois de testada pela equipe do PMO e já liberada em ambiente de produção. Mas apesar disso, os relatórios de avanço elaborados pelo Product Owner incluíam um gráfico de Burn Down e registravam essas funcionalidades como concluídas sem que tivessem sido testadas e disponibilizadas. Além disso, a liberação das primeiras funcionalidades em ambiente de testes para a equipe do PMO só ocorreu durante a Sprint 5 e a liberação das primeiras funcionalidades em ambiente de produção só ocorreu após a Sprint 13.

Tendo em vista tudo o que foi exposto até aqui, fica evidente as diferenças significativas de percepções captadas através da pesquisa com o Time Scrum e das observações feitas pelo próprio cliente. A visão do Time Scrum é de que o trabalho foi conduzido com uma aderência extremamente alta ao Framework Scrum e que os resultados produzidos foram satisfatórios. Já a percepção do cliente é de que houveram desvios significativos das práticas adotadas comparativamente ao que está prescrito no Framework Scrum. Além disso, ficou evidente a alta frustração do cliente com a demora significativa para a liberação de funcionalidades em ambiente de produção e que, para ele, representava a entrega efetiva de valor.

5. CONSIDERAÇÕES FINAIS

Ao escolher o framework Scrum para o desenvolvimento do sistema proposto para o PMO da empresa Alfa, uma das principais expectativas é que ele contribuísse para a entrega antecipada de valor. Isso era ainda mais importante em função de que o tempo de desenvolvimento se limitava aos seis meses do programa de trainees. Esperava-se que a aplicação desse framework contribuísse para uma dinâmica de trabalho mais transparente, com foco nas entregas de maior valor e que houvesse a liberação de funcionalidades para uso efetivo o mais cedo possível.

Pode-se afirmar que a adoção do framework Scrum neste projeto não produziu o efeito esperado. O principal fato constatado que sustenta essa afirmação é a liberação de funcionalidades para uso efetivo em ambiente de produção para a equipe do PMO da empresa Alfa somente ocorreu na Sprint 5 e isso significa que a entrega de valor para o cliente só ocorreu ao final de um período de 4 meses.

O presente estudo também permite relacionar o resultado aquém do esperado à baixa aderência das práticas adotadas pelo Time Scrum com o que é prescrito no guia do framework Scrum. Dentre os pontos em que essa aderência apresentou-se prejudicada, foi possível identificar os seguintes:

- **Pilares:** aplicação limitada dos pilares do Scrum, especialmente pela baixa transparência e interação entre Product Owner e cliente.
- **Time Scrum:** exercício deficitário dos papéis de Product Owner e de Scrum Master e volatilidade na composição do time de Developers.
- **Eventos:** deficiências no planejamento e execução da Sprint, Sprint Planning, Sprint Review e Sprint Retrospective.
- **Artefatos:** modelo e uso deficitário do Product Backlog, Sprint Backlog e baixa consistência na Definição de Pronto para os Incrementos.

Ao realizar a apresentação dos dados coletados e observações realizadas, identificaram-se algumas questões que merecem um aprofundamento na análise e que podem ser temas para estudos futuros. Essas questões são listadas a seguir:

- Fatores críticos para antecipar entrega de valor ao cliente com Scrum.
- Papel do Product Owner e Scrum Master na relação com o cliente.
- Engajamento do cliente em eventos e construção do Product Backlog.
- Importância da Definição de Pronto no contexto do Scrum.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, D. J. **Kanban: Successful Evolutionary Change for Your Technology Business**. 1ed. Blue Hole Press, 2010.

ATLASSIAN. **O que é o Jira Software?.** Disponível em: <https://www.atlassian.com/br/software/jira/guides/getting-started/introduction#what-is-jira-software>. Acesso em: 12/09/2023.

BITRIX24. **Ferramentas.** Disponível em: <http://bitrix24.com.br/tools>. Acesso em: 10/12/2023.

BECK, K.; ANDRES, C. **Extreme Programming Explained: Embrace Change.** 2. ed. Boston: Addison-Wesley Professional, 2004.

BEEDLE, M. *et al*. **Manifesto para Desenvolvimento Ágil de Software.** Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 27 ago. 2023.

BESSA, T.; ARTHAUD, D. D. B. Metodologias ágeis para o desenvolvimento de softwares. **Revista Ciência e Sustentabilidade**, v. 4, n. 2, p. 173-213, jul./dez. 2018**.**

BOEHM, B. W. A spiral model of software development and enhancement. **Computer**, v. 21, n. 5, p. 61-72, 1988.

GALDINO, P. R. **Análise da aplicação de Design Thinking para o levantamento de requisitos: estudos de casos de projetos de sistemas de saúde**. 2021. Trabalho de Conclusão de Curso (TCC) - Curso de Engenharia de Software, Pontifícia Universidade Católica, Belo Horizonte, 2021.

PMI - Project Management Institute. A Guide to the Project Management Body of Knowledge: PMBOK Guide. 6. ed. **Project Management Institute**. 2017.

PRESSMAN, R.; MAXIN, B. **Engenharia de Software: uma abordagem profissional**. 8 ed. Porto Alegre: Bookman, 2016.

ROYCE, W W. Managing the development of large software systems: concepts and techniques. In: **Proceedings of the 9th international conference on Software Engineering**. 1970. p. 328-338.

SCHWABER, Ken; SUTHERLAND, Jeff. **O guia do Scrum.** 2020. Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf>. Acesso em: 27/08/2023.

SOARES, M. S. Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. **INFOCOMP Journal of Computer Science**, Universidade Presidente Antônio Carlos, 1-3, 2004**.**

SOMMERVILLE, I. **Engenharia de Software**. 9 ed. São Paulo: Pearson Education do Brasil, 2011.

SOUZA, R. S. **Testes de software para garantir a qualidade**. 2016. 40f. Trabalho de Conclusão de Curso – Faculdade de Tecnologia de Americana, Americana, 2016.

TRELLO. **Noções básicas.** Disponível em: <https://trello.com/tour>. Acesso em: 12/09/2023.

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