CENTRO UNIVERSITÁRIO UNIDADE DE ENSINO SUPERIOR DOM BOSCO
CURSO ENGENHARIA DE SOFTWARE

CAIO VICTOR SAMPAIO SANTIAGO

ESTUDO COMPARATIVO DAS METODOLOGIAS ÁGEIS E PMBOK PARA O DESENVOLVIMENTO DE SOFTWARE

São Luís
2023

CAIO VICTOR SAMPAIO SANTIAGO

ESTUDO COMPARATIVO DAS METODOLOGIAS ÁGEIS E PMBOK PARA O DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada ao Curso de Engenharia de software do Centro Universitário Unidade de Ensino Superior Dom Bosco como requisito parcial para obtenção do grau de Bacharel em Engenharia de Software.
Orientador: Prof. Esp Daniel Herrera de Oliveira Lemos.

São Luís

2023

Dados Internacionais de Catalogação na Publicação (CIP)

Centro Universitário - UNDB / Biblioteca

Santiago, Caio Victor Sampaio

Estudo comparativo das metodologias ágeis e PMBOK para o desenvolvimento de software. / Caio Victor Sampaio Santiago. __ São Luís, 2023.

52 f.

Orientador: Prof. Esp. Daniel Herrera de Oliveira Lemos.

Monografia (Graduação em Engenharia de Software) – Curso de Engenharia de Software - Centro Universitário Unidade de Ensino Superior Dom Bosco – UNDB, 2023.

1. Metodologias ágeis. 2. *Project Management Body of Knowledge* (PMBOK). 3. Desenvolvimento de software. I. Título.

CDU 004.4

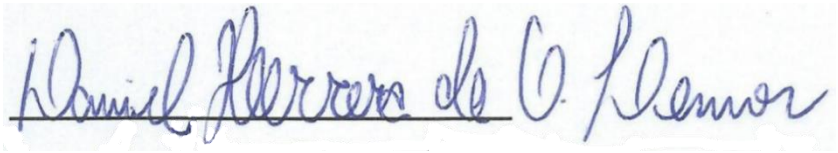
CAIO VICTOR SAMPAIO SANTIAGO

ESTUDO COMPARATIVO DAS METODOLOGIAS ÁGEIS E PMBOK PARA O DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada ao Curso de Engenharia de Software do Centro Universitário Unidade de Ensino Superior Dom Bosco como requisito parcial para obtenção do grau de Bacharel em Engenharia de Software.

Aprovada em: _____/____/_____.

BANCA EXAMINADORA:



Prof. Esp. Daniel Herrera de Oliveira Lemos

Especialista em Ciência de Dados

Centro Universitário Unidade de Ensino Superior Dom Bosco (UNDB)

_____

Prof. Me. Allisson Jorge

Mestre em Inteligência Artificial

Centro Universitário Unidade de Ensino Superior Dom Bosco (UNDB)

_____

Prof. Esp. Alysson Marquezelli

Especialista em Liderança e Gestão Empresarial

Centro Universitário Unidade de Ensino Superior Dom Bosco (UNDB)

Dedico a minha mãe, meu pai, minha avó e meu avô.

AGRADECIMENTOS

Agradeço primeiramente a meus pais Rogerio e Wania, por sempre acreditarem e confiarem no meu potencial, sem eles nada disso seria possível. Minha avó Francinete, que desde que eu nasci foi uma segunda mãe para mim, sempre presente e tentando ensinar tudo de melhor desse mundo. Meus colegas de turma que me acompanharam em todo esse período da faculdade e sempre acrescentando de forma positiva no desenvolvimento pessoal e profissional. Meus professores, e coordenador Rodrigo, que sempre esteve presente para me guiar nessa caminhada. E aos meus melhores amigos, que sempre estiveram ali pra me apoiar, escutar cada desabafo e mesmo assim não desistirem de mim em momentos difíceis, sempre presentes em auxílios de assuntos onde eu tinha dificuldade e achava que não iria conseguir.

“A ciência de hoje é a tecnologia de amanhã.”
(Teller,2020).

RESUMO

No contexto atual, as empresas de desenvolvimento de *software* enfrentam desafios constantes para se adequarem às rápidas evoluções tecnológicas. Diante dessa realidade, surge a busca por metodologias e ferramentas que possibilitem acompanhar a velocidade das mudanças do mercado, tornando-as mais competitivas. Portanto, devido essa necessidade as empresas têm encontrado nas metodologias ágeis, tradicionais e no guia PMBOK uma forma de contornar esses desafios, contudo, é fundamental entender os impactos positivos e negativos de cada abordagem e suas particularidades no âmbito de desenvolvimento. Através de pesquisa bibliográfica, coleta de dados e questionários, o presente trabalho irá compreender os conceitos das metodologias tradicionais, ágeis e o guia PMBOK, suas vantagens e desvantagens e analisar os pontos de interseção do PMBOK e Scrum referente a sua efetividade no processo de desenvolvimento de *software.*

Palavras-chave: Framework. Scrum. Metodologia. Guia. PMBOK.

ABSTRACT

In the current context, software development companies face constant challenges to adapt to rapid technological advancements. In light of this reality, there is a growing search for methodologies and tools that enable them to keep up with the pace of market changes, making them more competitive. Therefore, due to this necessity, companies have found in agile methodologies, traditional approaches, and the PMBOK guide a way to overcome these challenges. However, it is essential to understand the positive and negative impacts of each approach and their specificities in the realm of software development. Through literature review, data collection, and questionnaires, this study aims to comprehend the concepts of traditional and agile methodologies, as well as the PMBOK guide, their advantages and disadvantages, and analyze the points of intersection between PMBOK and Scrum in terms of their effectiveness in the software development process.

Keywords: Framework. Scrum. Methodology. Guide. PMBOK.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DSDM	Dynamic System Developer Model
PMBOK	Project Management Body of Knowledge
PO	*Product Owner*
WIP	*Work in Progress*
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

No mundo moderno, as empresas de desenvolvimento de *software* enfrentam uma pressão crescente para se adaptarem rapidamente às mudanças nas áreas de negócios dos seus clientes. Como resultado, muitas dessas empresas estão procurando novas metodologias de desenvolvimento que lhes permitam acompanhar a velocidade das mudanças do mercado e serem mais competitivas (DE SOUSA,2017).

De acordo com Kerzner (2017), a metodologia tradicional, também conhecida como *Waterfall*, é caracterizada por uma abordagem sequencial de desenvolvimento, em que cada fase do projeto é concluída antes de passar para a próxima fase. Contudo, essa abordagem vem sendo mais adequada para projetos com requisitos bem definidos e estáveis, mas vem sendo criticada por sua inflexibilidade em lidar com mudanças e requisitos em evolução.

Nesse contexto, as metodologias ágeis, como Scrum, têm sido amplamente adotadas, uma vez que sua ênfase na entrega rápida de produtos é altamente valorizada pelos clientes. No entanto, essas metodologias também apresentam algumas fraquezas.

Para superar essas fraquezas, o estudo comparativo das metodologias ágeis com outras metodologias de gestão de projetos, como *Project Management Body of Knowledge* (PMBOK), pode ser extremamente útil. Cruz (2013) o PMBOK é um padrão internacionalmente reconhecido para a gestão de projetos e pode ajudar a preencher as lacunas deixadas pelas metodologias ágeis.

Ao analisar essas metodologias de forma comparativa, é possível identificar os pontos fortes e fracos de cada uma delas e escolher a abordagem mais adequada para cada projeto (SOUSA,2017). Portanto, é de suma importância que as empresas de desenvolvimento de *software* considerem o estudo comparativo das metodologias ágeis juntamente com as de gestão de projetos para melhorar sua eficiência e competitividade no mercado.

O presente trabalho irá compreender o conceito dos métodos ágeis, tradicionais e do guia PMBOK por meio de pesquisas descritivas, com abordagem qualitativa em levantamento bibliográfico, artigos científico, livros e como essas abordagens podem impactar positivamente a gestão de projetos de *software*, além de

ressaltar os pontos positivos e negativos por meio de uma análise comparativa entre as principais características entre as abordagens.

1.1 Justificativa

O desenvolvimento de *software* é uma atividade complexa que necessita de cuidados e estruturas. Nesse contexto, a metodologia ágil e o PMBOK são frequentemente utilizados como ferramentas para gerenciamento de *softwares* Pressman (2016). Devido à pressão crescente para se adaptar ao mercado atual, surge a implementação de ferramentas que auxiliam essa adaptação, nesse sentido, os métodos ágeis então cada vez mais presentes como ferramenta para gestão de software. Contudo, ainda existe muita resistência das empresas quando se trata em utilizar essas práticas nos projetos atuais.

1.2 Objetivo Geral

O objetivo do presente trabalho é um comparativo entre métodos ágeis e tradicionais, guia PMBOK sobre essas ferramentas para gestão de projetos, visando identificar as principais diferenças, vantagens e desvantagens de cada abordagem e demonstrar a necessidade da utilização dessas ferramentas no desenvolvimento de *software*.

1.3 Objetivos Detalhados

a) Contextualizar o PMBOK e descrever as metodologias ágeis e tradicionais;
b) Detalhar o processo de desenvolvimento de *software* e suas etapas básicas;
c) Analisar os pontos de interseção do PMBOK e das metodologias ágeis e sua efetividade no processo de desenvolvimento de *software*.

3 REFERENCIAL TEÓRICO

3.1 Desenvolvimento de Software

O desenvolvimento de *software* é uma atividade fundamental em diversas áreas de negócios. De acordo com Pressman e Maxim (2016) o desenvolvimento de *software* surgiu por meio de uma necessidade de gerenciar e organizar o processo de criação de programas de computadores. Dessa forma, com o aumento dessas demandas por soluções que envolvessem *softwares*, houve a necessidade de determinar processos mais sistemáticos para o desenvolvimento de programas. A partir da década de 1970, iniciou-se a criação de novas metodologias para o desenvolvimento de *software*, como modelo cascata e desenvolvimento incremental, dando origem a metodologias ágeis.

3.2 Metodologias Tradicionais

No início da década de 70 todas as atividades referentes a desenvolvimento de *software* eram executadas de maneira caótica, sem estrutura e sem nenhum tipo de planejamento. Esse método resultava em produtos finais de baixa qualidade, pois não existia nenhuma documentação que os desenvolvedores pudessem utilizar para entender o passo a passo de cada processo de desenvolvimento (SEMEDO, 2012). Além disso, os produtos geralmente eram entregues fora do prazo e não atendiam às expectativas dos clientes e muito menos conseguiam entregar o que era solicitado.

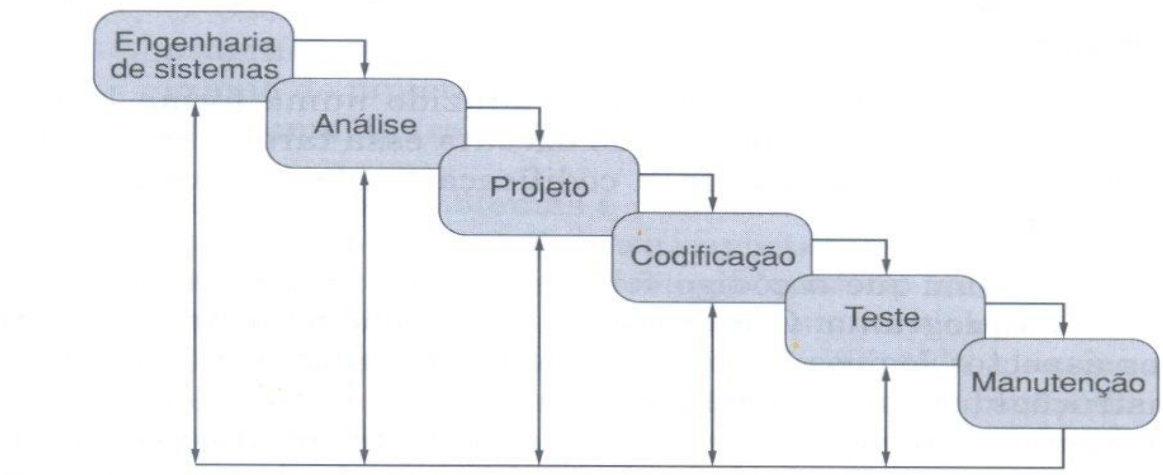
Segundo Soares (2004), nessa época o custo para fazer alterações e correções eram exorbitantes, pois a limitação de acesso a computadores era muito grande e naquela época não existiam as ferramentas de apoio ao desenvolvimento de *software*, como analisadores de códigos ou depuradores. Dessa forma, surge a necessidade de planejar e documentar antes de fazer qualquer implementação no projeto.

Para Sommerville (2011), a metodologia tradicional é baseada em engenharias clássicas, como a Civil e Naval, que preconiza a realização de atividades em sequência, tendo a necessidade de finalizar uma tarefa para dar sequência às

próximas. Essa abordagem permite que cada projeto siga um caminho bem definido e previsível.

Logo, nota-se um método muito engessado, tendo menor possibilidade de mudanças. Portanto, é possível considerar a utilização dessa metodologia em projetos com requisitos bem definidos e sem muitas mudanças durante todo o processo de desenvolvimento.

Figura 1 – O ciclo de vida da Metodologia Cascata



Fonte: Pressman (2016)

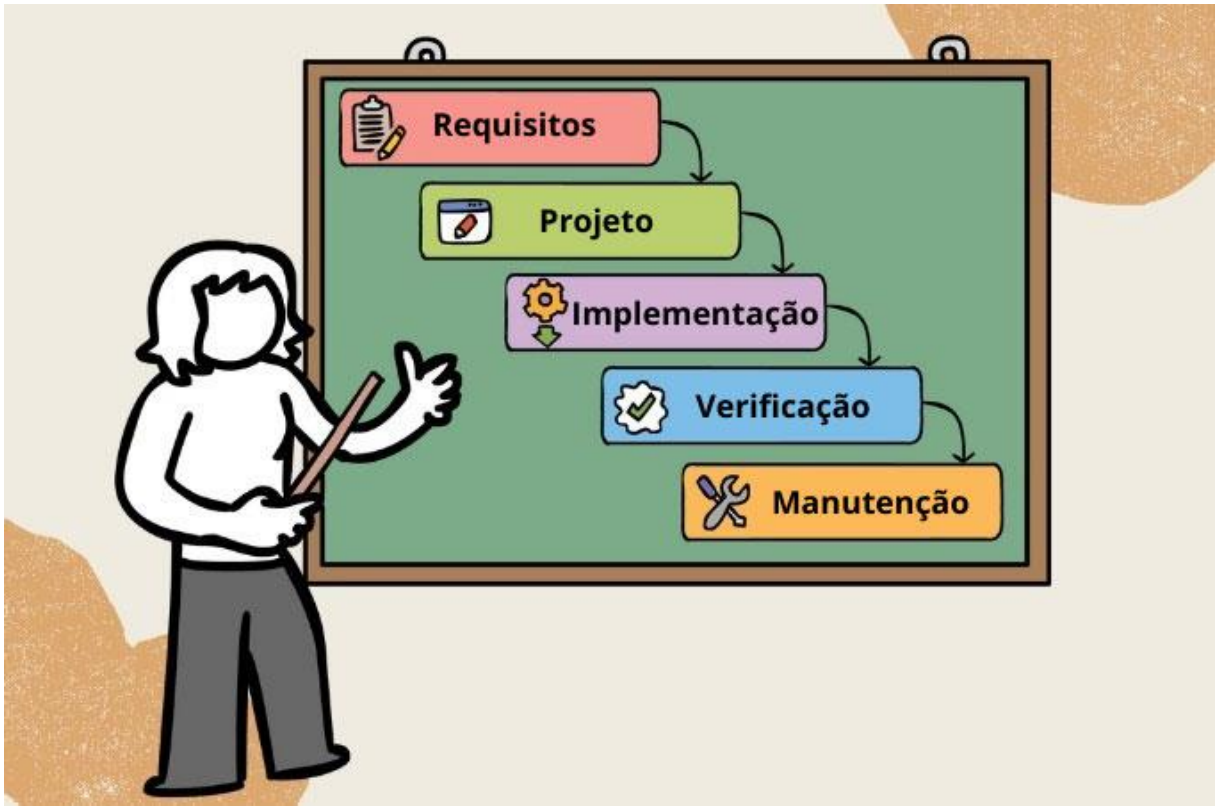
A Figura 1 demonstra a sucessão de etapas onde é necessário finalizar uma tarefa para dar início às demais.

3.2.1 Modelo em Cascata

Um dos pioneiros ao que se refere a metodologia tradicional, é o modelo em cascata, também conhecido como *Waterfall*. Essa abordagem tem como principal característica a sequência linear de fases, dando início no levantamento de requisitos até a conclusão e o processo de manutenção (Pressman, 2016).

Segundo Miguel (2003) o modelo em cascata é uma sucessão de etapas bem definidas. A Figura 2 indica as etapas do modelo em cascata.

Figura 2 – Modelo de Cascata



Fonte: Daniel Green (2022)

Seguindo essas etapas o modelo em cascata conseguiu dominar o desenvolvimento de *software* até o início da década de 90 (Soares, 2004). Contudo, Segundo Sommerville (2007) é muito desastroso especificar totalmente um *software* antes do início de sua implementação.

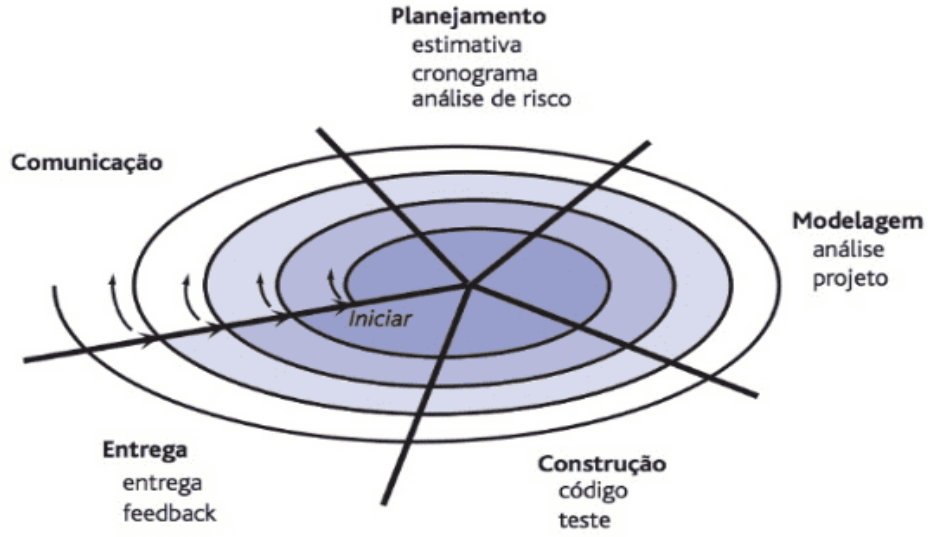
Seguindo esse pensamento, atualmente o mercado está muito aquecido e necessita de rápidas entregas e existem constantes pedidos de alterações e melhorias para se adequar cada vez mais a competitividade entre as grandes empresas. Por isso, o modelo em cascata não se destaca com tanta eficiência na implementação de desenvolvimento de *software* complexos. No entanto, sua falta de eficiência em grandes projetos resultou no surgimento de outros modelos tradicionais e ágeis (SOUZA, 2017 apud PRESSMAN, 2005).

3.2.2 Modelo Espiral

O modelo em cascata conseguiu dominar o desenvolvimento de *software* no início da década de 90, mas em 1988 surge o Modelo Espiral criado por Barry Boehm (BOE, 88). De acordo com (BARBOSA, 2006) o modelo espiral foi

desenvolvido selecionando as melhores características dos principais paradigmas clássico e prototipação, dessa forma, foi possível implementar a análise de riscos por meio do desenvolvimento de protótipos. Segundo Pressman (2011) o *software* será criado através de versões progressivas, dando início a um modelo ou protótipo e em etapas posteriores, é possível produzir atualizações e melhorias cada vez mais complexas do sistema.

Figura 3 – Modelo Espiral



Fonte: Pressman (2011)

É possível analisar as principais etapas conforme ilustra a Figura 3 – Modelo Espiral. Ainda seguindo as ideias de Pressman (2011) é possível analisar que o modelo está dividido em 5 regiões:

a) Comunicação: É necessário manter contato com o cliente para adquirir o máximo de informações importantes para o sistema;
b) Planejamento: É a etapa de estimativa de custo, de tempo de duração para concluir o projeto e determinar as equipes;
c) Modelagem: Início da criação do software;
d) Construção: Momento de testagem, instalação e avaliação do cliente;
e) Entrega: Por fim, avaliação do produto final e planejamento para futuras melhorias e novas implementações caso o cliente necessite de melhorias futuras.

Contudo, nesse modelo é de suma importância que exista a incorporação de um analista de riscos, pois é necessário que consiga reconhecer os principais riscos do projeto, logo no início do planejamento ou o projeto não vai conseguir evoluir gerando muito atraso e demandando a criação de um novo processo. (SEMEDO, 2012)

3.3 Metodologias Ágeis

É indubitável que mesmo com esse grande crescimento da metodologia tradicional na década de 90, ainda existiam muitos gargalos e o desenvolvimento de *software* era muito caótico, dessa forma, surge a reunião de um grupo de dezessete desenvolvedores de *softwares*, visando o debate sobre possíveis métodos e formas de melhorar o desenvolvimento de *software* e entrega dos produtos. Tal evento ficou conhecido como “Manifesto para o Desenvolvimento Ágil de Software”, nascendo os primeiros valores e princípios para a formação de novas abordagens mais flexíveis (BECK, 2001).

3.3.1 Manifesto Ágil

Segundo Melo et al. (2018) o Manifesto Ágil foi um documento criado em 2001 por um grupo de desenvolvedores de software que estavam insatisfeitos com as metodologias tradicionais devido aos seus processos muito engessados e por causa das entregas catastróficas. A partir disso, o manifesto estabelece a valorização de indivíduos e interações, entregas de softwares completos de acordo com o requerimento do cliente e respostas a mudanças.

Seguindo esses princípios estabelecidos nesse período, foi possível desenvolver e aprimorar algumas metodologias ágeis como Scrum, XP (*Extreme Programming*), *Kanban*, entre diversas outras, mas cada uma possui suas particularidades e foco em diferentes aspectos do desenvolvimento de *software* Beck et. al (2001).

Para Beck et. al (2001) é importante entender os dozes principais conceitos que norteiam a ideia de metodologia ágil, são eles:

a) Um dos principais problemas da metodologia tradicional foi a insatisfação dos clientes, logo é preciso manter o cliente satisfeito por meio de entregas frequentes e contínuas de softwares funcionais;
b) É necessário aceitar mudanças nos requisitos mesmo em estágios avançados de desenvolvimento, devido ao mercado competitivo e aquecido;
c) Entregas de softwares funcionais, em períodos de curto prazo;
d) A colaboração entre membros da equipe de desenvolvimento e o cliente deve perdurar todo o processo de desenvolvimento do projeto;
e) Manter um bom clima organizacional é um ponto chave para motivar e auxiliar o foco no desenvolvimento;
f) A principal forma de transmitir informação é por meio de comunicação clara face a face;
g) Para continuar o desenvolvimento é necessário ter o software funcional como primeiro estágio;
h) Os processos ágeis auxiliam no desenvolvimento sustentável, permitindo que os patrocinadores, desenvolvedores e usuários consigam manter um ritmo constante;
i) Atenção contínua à qualidade técnica e ao design do software;
j) Simplicidade é essencial para maximizar o tempo para possíveis prazos apertados;
k) Uma equipe qualificada é onde todos contribuem para o sucesso do projeto, por meio de auto-organização e gestão participativa;
l) Reflexão e diálogos constantes são um dos principais pilares quando se trata sobre o processo de desenvolvimento e ações para ser mais eficiente.

3.3.2 As Metodologias Ágeis

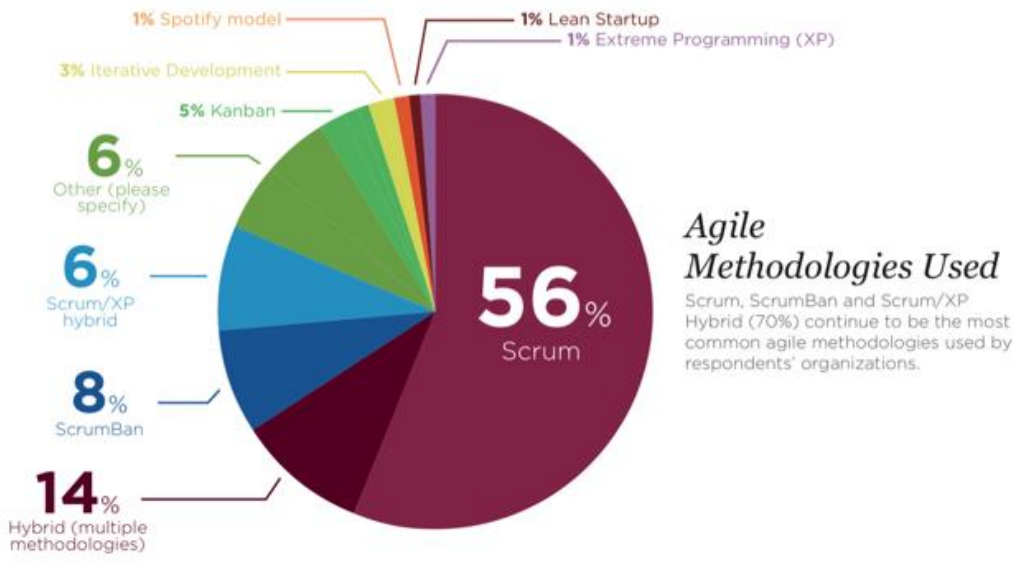
Anteriormente, foi compreendido o surgimento da metodologia tradicional e uma breve contextualização sobre alguns conceitos que norteiam a metodologia ágil. Nesse capitulo descreve o princípio de funcionamento de cada metodologia ágil e um aprofundamento no mesmo, para que se permita a comparação e validação dos pontos fortes e fracos.

Para Pressman (2005) uma vez que existem diversas metodologias ágeis, é necessário entender quais as que vem se destacando cada vez mais no cenário empresarial, sendo assim, há sete principais tipos de metodologias ágeis que mais se destacam atualmente, são elas:

- *Kanban*
- *Scrum*
- *Scrumban*
- *Extreme Programing (XP)*
- *Lean*
- *Scrum & XP*
- *DSDM*

Segundo o relatório sobre estado do *Agile (State of Agile Report)* tem produzido relatórios ao longo dos anos mostrando quais são as principais metodologias ágeis utilizadas pelas empresas.

Figura 4 – Principais Metodologias Ágeis



Fonte – Agile (2017)

A Figura 4 exibe, resumidamente, as metodologias mais utilizadas nos anos de 2020. É possível analisar com facilidade que o Scrum vem sendo a principal metodologia ágil utilizada nos desenvolvimentos de projetos com um total de 56%. Contudo, é de suma importância entender que cada método tem suas peculiaridades

e é necessário compreender essas 7 metodologias para alcançar a melhor implementação no desenvolvimento de *software*.

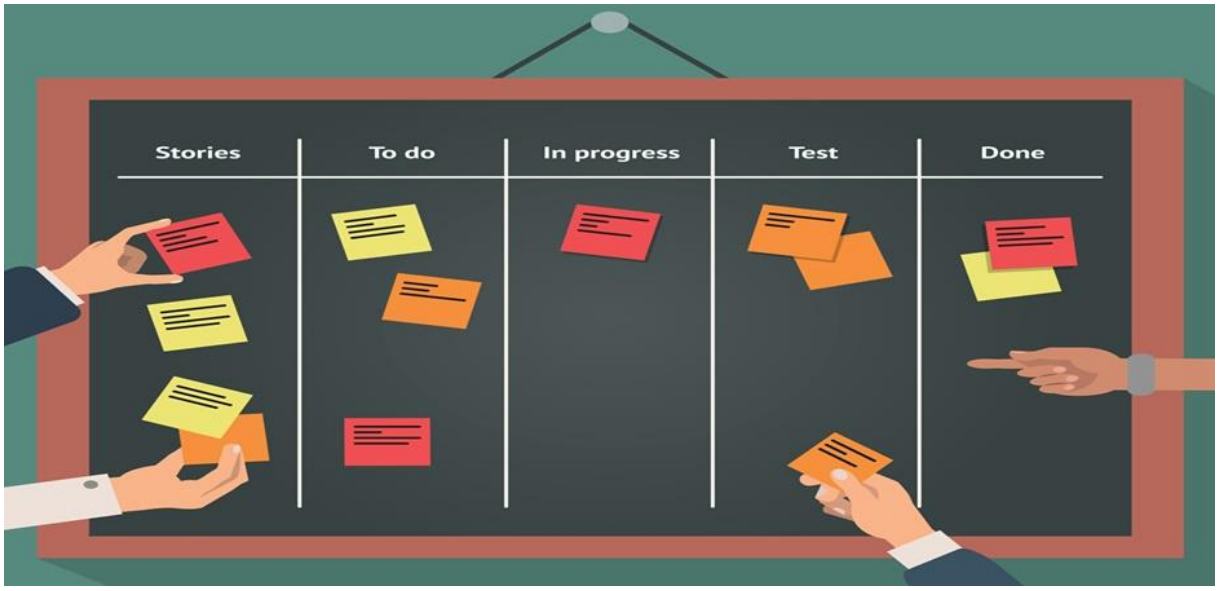
3.3.3 Kanban

O método Kanban surge na década de 40, mais propriamente na Toyota, quando o Taiichi Ohno, percebe alguns gargalos no processo produtivo da fábrica (TEIXEIRA,2013).

De acordo com Liker (2004), o Kanban foi desenvolvido como uma forma de gerenciar e controlar fluxos de produções, com o objetivo de maximizar a eficiência e minimizar o desperdícios de recursos, dessa forma, o Kanban utiliza uma abordagem em que a produção é controlada de acordo com a necessidade de demanda real do cliente, isso significa que em vez de aumentar a produção para ter uma quantidade maior no estoque, o sistema funciona por meio de cartões Kanban para sinalizar o momento em que os materiais são necessários em cada estágio do processo de produção, sendo assim, é possível garantir que cada processo na linha de produção receba as peças necessárias e que não falte nenhum material para criação dos produtos.

Segundo Junior (2019) com o surgimento do Kanban, nasce a filosofia *Just in Time* – (JIT) que consistia em determinar tudo o que deve ser produzido sem gerar estoque, além de manter as informações relacionadas ao produto, como por exemplo: nome do cliente, onde o produto deve ser armazenado e como deve ser transportado.

Figura 5 – Método Kanban



Fonte: (10 Kanban Board Examples, 2017)

A figura 5 mostra o conceito de" *Work in Progress* - WIP", que compreende delimitar o número máximo de itens em cada fase (REINERTSEN E BELLINSON, 2014). A partir disso, é possível compreender alguns princípios básicos que norteiam esse método, são eles:

- Começar com o que existe, pois o método não obriga a implementação de vários conjuntos de processos;
- Aceitar e incentivar mudanças contínuas, é necessário que a organização entre em acordo para poder manter as mudanças;
- Respeitar o processo atual, é indubitável que é necessário manter processos que estão funcionando bem. Entretanto, é indispensável fazer uma análise e procurar possíveis pontos fracos para serem melhorados no método de trabalho;
- Incentivar a liderança, o incentivo de atos de lideranças são fundamentais em todos os níveis da organização;
- Gerir o fluxo, gerenciar o fluxo através de um modelo visual é indispensável, pois não adianta ter fluxos bem definidos sem ter um gerenciamento sobre eles, deve ser rápido e suave para minimizar o risco de custos e atrasos;
- Regras bem definidas, é de suma importância que as regras estejam bem definidas para o entendimento de toda equipe, caso contrário é impossível fazer melhorias e entender quais são os pontos fracos do processo.

3.3.4 *Extreme Programming* (XP)

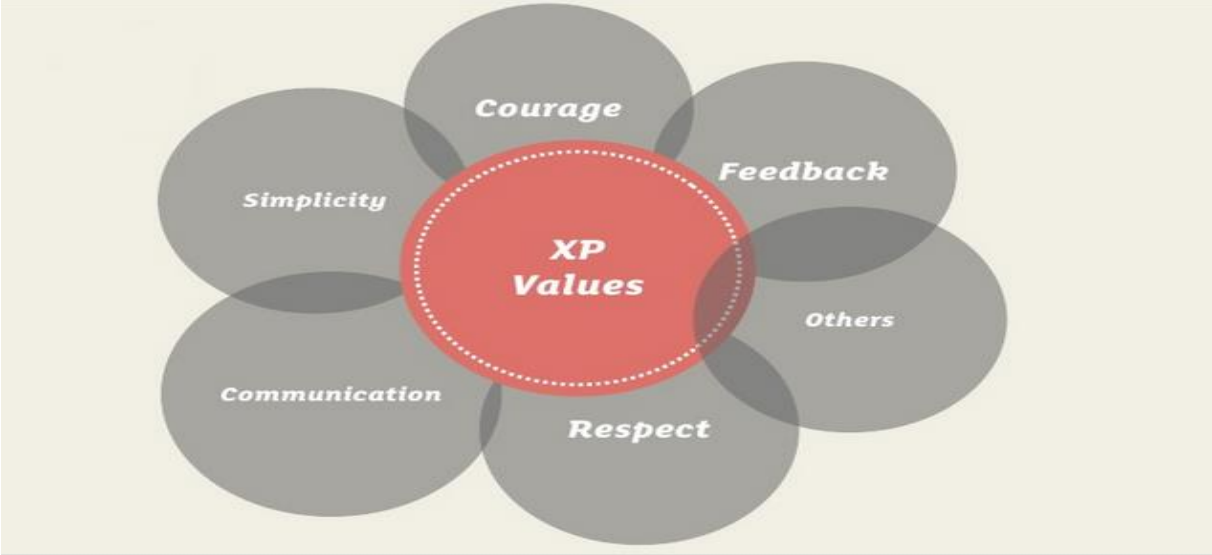
Extreme Programming (XP) é um uma metodologia ágil para desenvolver *softwares* com padrões elevados, onde existe uma melhor performance para equipes pequenas e médias que desenvolvem *software* baseado em requisitos vagos e que estão em constante mudança (Anand e Dinakaran, 2016). Sendo assim, a XP se destaca em 3 princípios, são eles: feedback constante; abordagem incremental e comunicação entre as pessoas é reforçada.

Como mencionado por DE SOUSA (2017) o XP utiliza a regra 80 – 20, em que 80% dos benefícios são obtidos com apenas 20% do esforço. Tem como objetivo principal alcançar a eficiência e efetividade, por meio de equipes pequenas e com

seus papéis bem definidos. Além do que, são seguidos valores, princípios e práticas básicas durante todo o processo de desenvolvimento.

Como mencionado anteriormente o XP consiste em uma prática que possui alguns pilares como princípios e valores que visam a construção de um software de qualidade por meio de comunicação, simplicidade, feedback, coragem e respeito.

Figura 6 – XP Valores



Fonte: (Semedo,2012)

Mediante o que foi observado na imagem: Comunicação é referente ao aspecto fundamental quando se trata de qualidade do produto, isso é de suma importância devido a necessidade de comunicação dentro da equipe para não existir ruídos que possam levantar especulações ou dúvidas sobre o projeto. Portanto, quanto melhor e mais clara for a comunicação, melhor serão os resultados e soluções implementadas dentro do produto.

Simplicidade é um fator chave quando uma equipe se depara com um grande projeto, pois é necessário se perguntar qual a forma mais simples de entregar e solucionar essa problemática? Ou seja, a simplicidade auxilia na confiança do trabalho que está sendo realizado; Feedback é um ponto muito importante quando se lida com pessoas, ou seja, é fundamental conseguir receber a opinião real do cliente, seja ela positiva ou negativo, pois é uma forma de melhorar o projeto em questões técnicas e relações harmônicas.

Já a coragem é um valor crucial, uma vez que é necessário ter coragem para dar continuidade na prática de outros valores, reconhecer que as vezes é necessário solicitar uma ajuda quando não se consegue bater as metas diárias ou

alcançar os objetivos esperados; é importante simplificar e comunicar de forma clara e coesa qualquer atraso ou incapacidade de atender os prazos acordados pelo cliente; Respeito é outro valor fundamental para o funcionamento de todos os valores do XP. Sem respeito, não existe comunicação, e sem comunicação o feedback se torna ineficaz. Logo, faltar com respeito significa destruir a equipe.

Na segunda edição do livro de Beck (2000) foram adicionados alguns princípios ou práticas que auxiliam o funcionamento dos valores são eles;

- Planejamentos: é uma prática que visa estabelecer a comunicação necessária entre equipe e cliente para conseguir determinar o planejamento do projeto, definindo funcionalidades, prazos e receber o feedback cliente;
- Entregas frequentes: é uma forma de validar a construção do projeto por meio de pequenas entregas e amostras para o cliente, dessa forma, é possível reduzir o risco de falhas e aumentar a flexibilidade da equipe em relação a possíveis mudanças do projeto;
- Metáfora: ela cria um modelo de alto nível para o sistema que está sendo desenvolvido, auxiliando a entender como cada parte se relaciona e como elas trabalham juntas;
- Projeto simples: é uma prática que visa criar um sistema que seja fácil de compreender, manter e evoluir. É necessário que exista algumas revisões do que se encontra, já que o projeto vai de desenvolver cada vez mais;
- Testes: essa etapa permite a melhoria constante do produto, pois ela visa garantir que o software funcione corretamente, essa prática envolve a criação de testes automatizados que podem ser executados continuamente, enquanto o cliente é responsável pelos testes funcionais;
- Refatoração: consiste na atualização constante do código, visando a melhoria e possível mudança para linguagens mais recentes e que performe melhor;
- Programação em pares: consiste em ter dois programadores trabalhando juntos, visando minimizar a chance de erros ou falha;
- Propriedade coletiva: permite que toda equipe tenha acesso ao código, sendo possível fazer modificações sem ter autorização, contudo é de suma importância notificar toda equipe sobre as mudanças que possam ser necessárias;

- Integração contínua: permite que a equipe coloque o código em um repositório central, isso ajuda a garantir que o software esteja sempre em estado funcional e pronto para ser entregue;
- Semana de quarenta horas: as horas extras não são recomendáveis. É essencial que a equipe tenha descanso para a produtividade se manter alta, pois um programador cansado tem diminuição no seu raciocínio, e por consequência, aumenta a chance de erros;
- Cliente presente: para ter um feedback mais rápido e preciso, é necessário que o cliente esteja presente e que faça parte da equipe de desenvolvimento, visando esclarecer dúvidas e isso ajuda a garantir que o software vai atender todas as necessidades do mesmo e reduz a possibilidade de retrabalho;
- Padrões de desenvolvimento: é importante que todos os membros da equipe sigam um mesmo padrão de codificação, para que o código possa ser facilmente compreendido e modificado por outros membros do projeto. Não existe necessidade de seguir um padrão específico, desde que haja um padrão e que a equipe o adote voluntariamente como seu.

3.3.5 *Lean*

A metodologia *Lean Manufacturing* (Lean) é outra metodologia que surgiu no pós 2 Guerra, no Japão pela Toyota. Nesta época a indústria japonesa passava por diversas crises e problemas devido essas guerras e possuía uma produtividade muito baixa junto com uma escassez de recursos, dificultando muito na implantação de produção em massa.

De acordo com Poppendieck (2014) o desenvolvimento de software Lean é uma metodologia que possui foco no desenvolvimento de pequenos lotes, mas que entreguem produtos e serviços de extrema qualidade, visando sempre na satisfação dos clientes através da melhor utilização dos recursos e sempre visando a redução de desperdícios de recursos. Para Fadel (2010) é necessário seguir 7 princípios para atingir a máxima qualidade na entrega rápida e com baixo custo são eles:

- Eliminar o desperdício: o desperdício em si acontece em diversos pontos de vista, dentre elas, dinheiro, tempo e recursos. Logo, a existência de funcionalidades incompletas gera desperdício em esforços e não adicionam valor ao software, códigos

incompletos geram uma dificuldade na integração e tiram o foco a respeito da intenção inicial do código.

- Amplificar o aprendizado: é de suma importância que as experiências negativas vivenciadas pela equipe devem ser extraídas para que se tenha aprendizado e sejam fonte de conhecimento e uma forma de amadurecimento da equipe e dos processos;
- Adiar comprometimentos e manter flexibilidade: é possível adiar algumas decisões para que o grupo como um todo decida como proceder determinadas tarefas com mais conhecimento sobre o projeto e dando uma liberdade para flexibilizar algumas mudanças do planejamento para aprender e atingir as metas;
- Entregar rápido: ciclos rápidos no desenvolvimento de software pode ser uma estratégia interessante quanto a velocidade do desenvolvimento atende a necessidade do cliente, permitindo adiar as tomadas de decisões para que consiga ter um maior conhecimento sobre o projeto para futuras mudanças e adaptações;
- Tornar a equipe responsável: a equipe de desenvolvimento necessita ser responsável na confecção do produto, logo o conhecimento dos detalhes técnicos é indispensável na definição de processos e tomadas de decisão. O *Lean* utiliza técnicas de produções puxadas para que os trabalhadores consigam identificar as *tasks* que precisam ser realizadas;
- Construir Integridade: a integridade de um software permite compreender que ele possui uma arquitetura coerente e que atende os requisitos do cliente, dessa forma, é muito importante utilizar uma arquitetura adequada e que exista testes automatizados visando analisar a integridade do produto para que seja possível reduzir o nível de problemas operacionais.
- Visualizar o Todo: permite a identificação de gargalos, desperdícios e atrasos por meio de uma visualização em diversas áreas do projeto. Isso fica atrelado com a construção de integridade devido a análise e testagem de alguns processos que podem receber otimizações ou eliminação de atividades para fornecer um produto ou serviço de qualidade para o cliente.

3.3.6 *Dynamic Systems Development Method* (DSDM)

A metodologia de desenvolvimento de sistemas dinâmicos é uma metodologia que surge com a proposta de ajudar projetos que possuem o prazo de entrega limitado. Pressman (2005) defende que para conseguir a satisfação do cliente é necessário utilizar a prototipagem incremental, esse processo tem como princípio 20% será baseado no tempo, enquanto 80% é correspondente à porcentagem que deve ser entregue.

A DSDM Consortium (AgileBusiness, 2017) é um grupo de empresas mundiais que mantém o DSDM e definem o método por meio do ciclo de vida. Entretanto, para dar início ao ciclo de vida é precedido duas atividades adicionais:

- Estudo de viabilidade – tem como função estabelecer e verificar se o projeto é viável tecnicamente ou não.
- Estudo do negócio – analisa e compreende os requisitos funcionais da aplicação, visando agregar valor de negócio e a definição da arquitetura do mesmo.

Após um estudo muito criterioso envolvendo os requisitos funcionais e definindo que a aplicação é viável para a utilização desse framework, surge a implementação dos três ciclos de vida são:

- Interação dos modelos funcionais – é apresentado um conjunto de protótipos que apresentam funcionalidade para o cliente, nessa primeira etapa focada na obtenção de feedback dos usuários conforme as testagens do protótipo;
- Fase da implementação – é necessário efetuar uma revisão de funcionalidades, a fim de garantir o funcionamento do projeto, pois é a fase de entrega ao cliente da solução final ou parcial;
- Fase do pós-projeto – Ocorre a validação do cumprimento dos objetivos e verifica se foram obtidos benefícios para o negócio. Durante essa etapa, é assegurado a manutenção e aprimoramento contínuo do sistema, seguindo as normas do DSDM.

3.4 *Scrum*

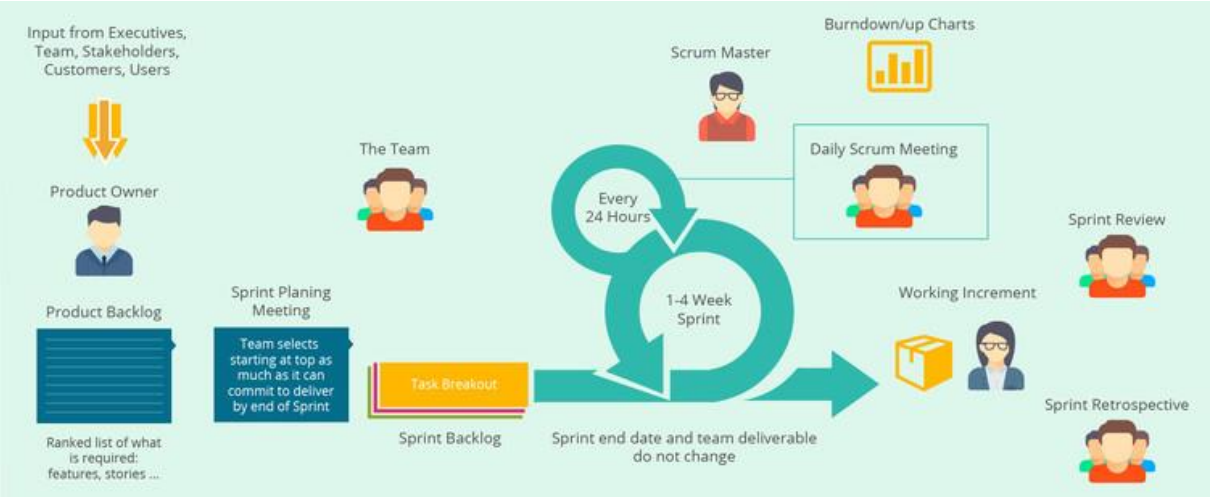
O Scrum é um *framework* ágil que que surgiu nos anos 80, Hirotaka Takeuchi e Nonaka Ikujiro construíram uma estratégia para desenvolvimentos de produtos que consistia em uma equipe trabalhar como um só, com a finalidade de

entregar um produto de extrema qualidade em um curto período de tempo. Essa ferramenta é amplamente utilizada na gestão de projetos, em especial no desenvolvimento de software, devido aos seus princípios iterativos e incrementais, possibilitando que equipes autônomas contribuam efetivamente para isso (SCHWABER E SUTHERLAND, 2013).

Segundo Sutherland (2013) a palavra *Scrum* tem sua origem no jogo de *rugby*, onde é utilizada para descrever uma regra de reinício do jogo em que os jogadores se unem e empurram a equipe adversária na tentativa de ganhar posse de bola. Essa analogia é aplicada no contexto do *Scrum*, onde, ao iniciar um *sprint*, a equipe de desenvolvimento e o cliente trabalham em conjunto para alcançar o objetivo proposto.

O *Scrum*, representado na Figura 7, demonstra o funcionamento do seu desenvolvimento, onde é realizado ciclos chamados de *sprints*, que ocorrem consecutivamente, sem pausas. A cada *sprint*, a equipe se compromete com o cliente na implementação de requisitos específicos, os quais não podem ser alterados ao longo do ciclo. Diariamente, a equipe se reúne para relatar o progresso realizado e discutir as futuras etapas a fim de alcançar os objetivos do *sprint*. Ao final do ciclo, é feita a entrega de um produto testado e demonstrado ao cliente, obtendo-se feedback que pode ser incorporado na próxima iteração, de acordo com (LEI ET AL, 2017).

Figura 7 – Framework Scrum



Fonte: (Sutherland,2013)

3.5 Papéis do Scrum

No Scrum existem três papéis: *Product Owner* (PO), *Team* e *Scrum Master.* Esses papéis são bem definidos e cada papel tem propósitos e tarefas diferentes, possibilitando que cada pessoa consiga desenvolver o trabalho de forma autônoma. As equipes nesse contexto são concebidas para serem flexíveis, criativas e produtivas.

3.5.1 *Product Owner* (PO)

O Product Owner, ou (“dono do produto”), desempenha um papel fundamental no processo, sendo o responsável por gerir o *Product Backlog*. O mesmo é responsável por definir, comunicar, tomar decisões estratégicas e garantir que as necessidades e expectativas dos clientes e usuários sejam atendidas de forma eficaz. Além disso, o PO atua como ponto focal e exerce um papel de liderança no que diz a respeito ao produto, logo é ele que toma as decisões (BEZERRA,2020).

3.5.2 *Team*

A equipe de desenvolvimento é o grupo de pessoas que executam o trabalho de desenvolvimento do software, possibilitando a criação e realização das entregas do incremento ao final determinado em cada *sprint*. Essas equipes, são conhecidas pela sua autonomia e pela capacidade de gerenciar seu próprio trabalho, dessa forma, não é necessário a definição de títulos ou funções específicas para às pessoas, pois, a responsabilidade pertence a equipe como um todo, e isso permite com que cada membro contribua de acordo com suas habilidades e conhecimentos, visando sempre na execução e entrega no que foi comprometido em cada *sprint*. É importante ressaltar que o *Scrum Master* não é contabilizado como membro dentro da equipe de desenvolvimento (SCHWABER E SUTHERLAND, 2013).

3.5.3 *Scrum Master*

Para Schwaber (2013) a principal função do Scrum Master é desempenhar um papel de facilitador para garantir que os princípios do Scrum estão sendo colocados em prática, ou seja, ele é uma forma de comunicação entre o PO e a equipe de desenvolvimento. Dessa forma, como ele é responsável por garantir que as regras

e princípios do Scrum sejam atendidos, ele também consegue identificar e remover ou auxiliar na solução de possíveis obstáculos que estejam prejudicando o desenvolvimento do projeto ou na produtividade da equipe. Contudo, é necessário ressaltar que o *Scrum Master* não possui o papel de liderança sobre a equipe de Scrum (Araujo,2017).

3.6 Eventos

No Scrum é necessário estabelecer alguns eventos que ocorrem com regularidade. Esses eventos compartilham os princípios ágeis, precisando ocorrer dentro de um período pré-definido e com duração específica. Cada evento é sinônimo de reuniões de alinhamento e inspeção sobre o projeto.

3.6.1 *Sprint*

De acordo com Pereira (2007), as sprints são ciclos de desenvolvimento do produto, que possuem uma duração de duas a quatro semanas e que deve gerar algum valor para o produto do cliente, dessa forma, é de suma importância que as sprints não passem da data acordada ou extrapole o período de um mês, pois sprints muito longas podem gerar muitos problemas referente ao planejamento e desenvolvimento do projeto, podendo aumentar drasticamente o risco de erros no projeto. Durante esse período ocorrem outros eventos para manter e observar o desenvolvimento do projeto, são eles: *sprint planning*, reuniões diárias (*Daily Scrum)*, revisão da sprint (*Sprint Review)* e reunião de retrospectiva (*Sprint Retrospective)*.

3.6.2 *Sprint Planning*

Esse é evento é responsável e tem como objetivo desenvolver o plano de trabalho a ser executado durante o *Sprint*. A sua duração pode variar entre uma a oito horas, para um *sprint* de um mês. Nesse momento, é fundamental a presença do Scrum master para dirigir a reunião e garantir que ocorra esse evento durante o prazo limite.

3.6.3 *Daily Scrum*

São reuniões curtas e diárias, com o *timebox* de 15 minutos, que tem como função reunir toda a equipe para discutir o que foi produzido no dia, rever o progresso feito desde a última *Daily Scrum* e alinhar as próximas atividades com o time de desenvolvimento nas próximas 24 horas. Para Araujo (2017), durante essa reunião, é importante que o Scrum Master ou membros da equipe de desenvolvimento esclareçam as seguintes questões:

- Quais tarefas foram feitas ontem?
- Quais tarefas não foram feitas hoje e qual o impedimento delas?
- Quais as tarefas que serão feitas amanhã?

3.6.4 *Sprint Review*

Essa reunião ocorre no final do *sprint* para demonstração do trabalho realizado para todos os integrantes do projeto. Esse evento tem como finalidade a obtenção de *feedback* sobre o incremento produzido e a adaptação do Product Backlog caso necessário. Esse evento possui uma particularidade, pois é o único evento na qual existe a participação de pessoas externas do projeto, dessa forma, o PO pode e deve convidar o cliente ou patrocinadores para colaborarem com mais *feedbacks* (DE SOUSA,2017).

3.6.5 *Sprint Restrospective*

A retrospectiva da sprint ocorre logo após a *sprint review* e antes do próximo *sprint planning*. Durante essa reunião, a equipe avalia seu próprio desempenho, reconhecendo os aspectos positivos e negativos, a fim de criar um plano que possibilite melhorias para a próxima sprint. É importante ressaltar que nessa retrospectiva o foco é concentrado no processo de trabalho da equipe, ou seja, trata-se de uma reunião de inspeção que permite a discussão e identificação de obstáculos que foram mapeados e estão sendo resolvidos por meio de novas abordagens.

3.7 Artefatos

0

Continuando no pensamento de Schwaber e Sutherland (2013) artefatos representam itens indispensáveis no projeto, pois, permitem construir uma visão geral do projeto, fornecendo transparência e oportunidades para inspeção e adaptação, ou seja, devido ao nível de detalhes que os artefatos trazem para o projeto é possível fazer implementações e adicionar procedimentos que agreguem valor aos requisitos.

3.7.1 *Product Backlog*

No Scrum, o product backlog é uma lista priorizada de todas as funcionalidades, requisitos, tecnologias, melhorias e correções que deverão ser desenvolvidas pela equipe de desenvolvimento para a conclusão do projeto. Contudo, esse artefato não é concluído, devido a sua evolução e constante mudanças a fim de entregar um produto polido. Diante do exposto, o PO é responsável pelo gerenciamento do *Product Backlog*, pois nessa etapa existe a necessidade de seguir uma lista de prioridades determinada pelo o mesmo (ARAUJO,2017).

3.7.2 *Sprint Backlog*

A *Sprint Backlog* é um conjunto de tarefas específicas e detalhadas vinda do *Product Backlog* onde a equipe de desenvolvimento precisa realizar ou estipular uma previsão de entrega durante a *Sprint*. Por fim, o objetivo dessa etapa é conseguir concluir todas as tarefas do *Sprint Backlog*, tendo como resultado um incremento funcional e possivelmente entregável. Entretanto, caso todos os objetivos não sejam concluídos é possível fazer uma devolução ao *Product Backlog* para fazer mudanças ou apenas priorizar as tarefas não concluídas em futuras *Sprints* (Schwaber e Sutherland, 2013).

3.7.3 *Increment*

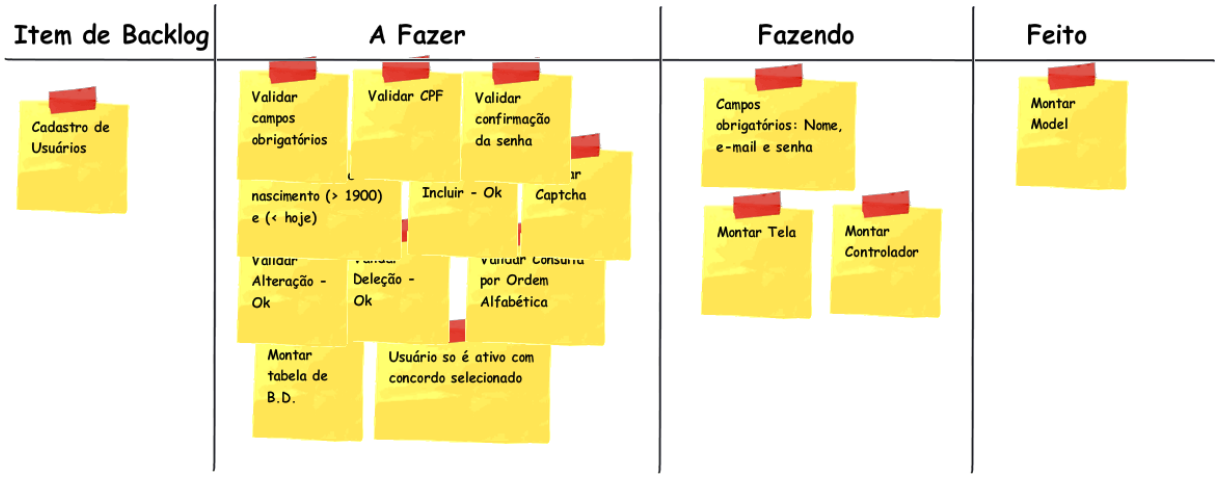
Por fim, essa etapa é quando ocorre a união de todos os itens concluídos nas *sprints,* tendo criado um incremento funcional e completo contendo todos os itens presentes desde o Product Backlog até a definição de concluído.

3.8 Scrumban

O termo Scrumban foi primeiramente mencionado em 2008 em um artigo publicado por Corey Ladas. Apesar de ser uma metodologia nova, ela é considerada uma metodologia híbrida, devido a combinação de benefícios expostos no Scrum e Kanban como possibilitar utilizar a abordagem versátil de gerencialmente de fluxo de trabalho vinda do Scrum e a flexibilidade do Kanban. Diante o exposto, essa metodologia híbrida traz novos benefícios quando se trata de auxiliar a lidar com as constantes mudanças de requisitos vindo dos clientes, sem sobrecarregar o processo (REIS 2021). O scrumban tem como práticas:

- Fluxo de valor – é uma prática herdada do Kanban que é muito importante devido a necessidade de ter uma visão clara e transparente do trabalho em andamento, compreender o que cada membro da equipe está desenvolvendo e a progressão de cada tarefa, juntamente da identificação de possíveis gargalos ou atrasos. Nessa etapa é comum utilizar um quadro representado por cartões podendo movimentar entre as colunas para atualizar o processo das etapas, geralmente incluem A fazer (*TO DO*), em andamento (*Doing*) e concluído (*Done*) como pode ser visto na Figura 8.

Figura 8 – Quadro Scrumban

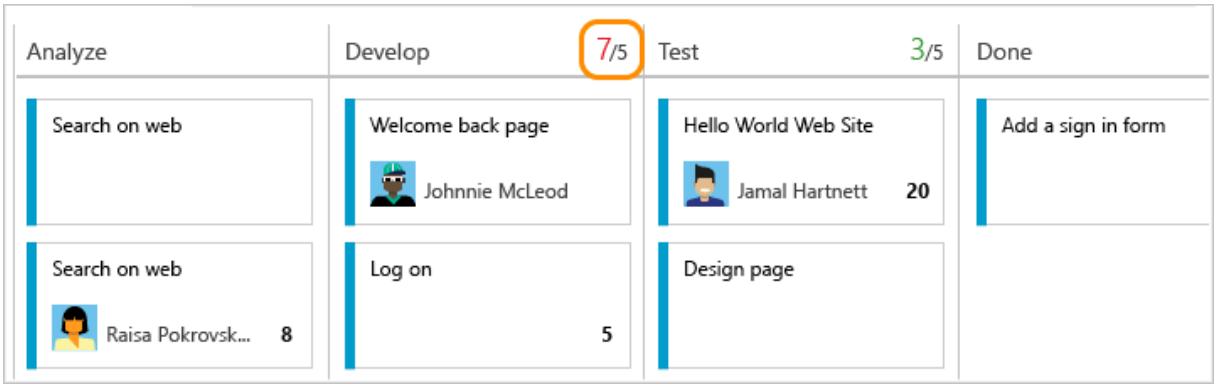


Fonte: (Stoica,2013)

- *Pull Work* – é o princípio que permite às equipes decidirem qual trabalho abordar conforme as necessidades do cliente. Isso possibilita a colaboração e organização da equipe, pois quando é concluída uma tarefa a equipe tem liberdade para iniciar outra *task* sem necessidade de pedido para o líder.

- WIP - No scrumban cada elemento deve possuir um limite de trabalho em progresso, ou seja, é criado um estudo com finalidade de prever uma mediana da quantidade de itens do backlog que aquela equipe vai conseguir entregar, podendo definir o limite total para futuras *sprints*. Como o WIP é utilizado no kaban, o mesmo sugere que seja aplicado no fluxo de trabalho como mostra a figura 9.

Figura 9 – WIP no Scrumban



Fonte: (Microsoft,2023)

3.9 Scrum & XP

Scrum com XP é mais uma metodologia híbrida que busca trazer o ponto forte de duas metodologias ágeis de forma unificada, ou seja, nessa metodologia reúne a prática de gestão do Scrum com algumas práticas de engenharia do XP. Por se tratar de uma metodologia híbrida é muito comum que as empresas consigam se moldar utilizando essas ferramentas de forma única, por conta disso, na literatura pesquisada, entre março de 2023 e junho de 2023, não foi possível encontrar muitas referências acerca do mesmo.

Segundo Mushtaq e Qureshi (2012) esse modelo é iniciado por um "*sprint zero*", onde o produto é dividido em subprocessos, durante essa *sprint*, são criados alguns prazos e priorizações de processos, em seguida, é utilizado a abordagem do Scrum. Entretanto, na fase de design a equipe se reúne para concluir os itens da *sprint* seguindo os princípios do XP, fazendo casos de testes e desenvolvendo diagramas de classes.

Nessa metodologia é fundamental seguir as práticas de programação em par, testes e refatoração, pois todos os integrantes do desenvolvimento possuem direito

para fazer alterações no código. Já a fase de testagem é presente durante o início do projeto, durante e no final do projeto.

3.10 PMBOK – Project Management Body of Knowledge

De acordo com Machado (2022) o projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado único. Sua caracterização ocorre devido a necessidade de ter um objetivo definido, um prazo determinado e recursos alocados para execução. Entretanto, um projeto é diferente das operações contínuas de uma organização, pois é temporário e possui escopo definido.

O guia PMBOK traz um conjunto de grupos de processos, boas práticas e diretrizes aplicadas e aceitas ao gerenciamento de projetos e é administrada pelo *Project Management Institute* (PMI). O PMI foi uma organização pioneira no que se refere a certificação em gerenciamento de projetos, dessa forma, o mesmo pontua o PMBOK como um guia muito funcional na maioria dos projetos. Contudo, isso não significa que ele é o mais correto ou que apenas essas práticas funcionam no gerenciamento de projetos, mas ele agrega muito na redução de problemas devido aos seus ensinamentos (CRUZ,2014).

Por se tratar de um guia e não uma metodologia faz com que vários profissionais e iniciantes na área interpretem de forma errada, uma vez que é entendido que guia é algo extenso e muitas vezes pesado ou pouco flexível quando se trata de projetos que precisam ser ágeis e dinâmicos. Porém, isso não é verdade, pois o PMBOK apresenta um conjunto de boas práticas que agrega muito valor quando atrelada a uma metodologia, a partir disso, é de suma importância saber administrar qual metodologia é mais apropriada para tornar essa união flexível e não engessada. Sob esse ponto de vista, é fundamental entender os grupos de processos.

3.10.1 Grupo de Processos

O guia PMBOK é conhecido por sugerir “o que deve ser feito”, mas não define “como deve ser feito” devido ao entendimento que cada projeto é único, mesmo possuindo características semelhantes, devido essa ausência de detalhar o que deve ser feito destaca-se um ponto negativo no guia, em vista disso, o mesmo sugere cinco

grupo de processos, que devem ser conhecidos e compreendidos para auxiliar na implementação com o Scrum, são eles:

- Grupo de processos de iniciação: esse grupo de processos é constituído por todos os processos no início de um novo projeto ou fase de um projeto, onde o objetivo principal é a definir o início do projeto e a definição do gesto do projeto.
- Grupo de processos de planejamento: nessa fase do processo, o projeto é cuidadosamente planejado, levando em consideração todos os objetivos para os quais o projeto foi criado. Para isso, são criados documentos que listem os requisitos, custos e possíveis riscos.
- Grupo de processos de execução: o trabalho que foi planejado no processo anterior é colocado em prática.
- Grupo de processos de monitoramento e controle: é um processo que está presente em todo o projeto, pois é realizado o acompanhamento e monitoramento do progresso acerca do desenvolvimento do projeto, sendo possível identificar mudanças e antecipar problemas.
- Grupo de processos de encerramento: é responsável por concluir todas as atividades no decorrer de uma fase ou do projeto, é necessário ter aceitação do cliente. Nesse processo pode ser feito uma revisão para futuras mudanças e atualizações e é de suma importância que toda a documentação seja arquivada.

3.10.2 Áreas de conhecimento

O guia PMBOK (2021) agrupa 47 processos que são distribuídos em 10 áreas de conhecimento que devem ser estruturadas e aplicadas no gerenciamento de um projeto, como escopo, custos, qualidade e recursos humanos. Devido a sua importância, serão apresentados de forma breve as dez áreas de conhecimento do guia PMBOK na visão de (CRUZ,2014):

- Gerenciamento da Integração do projeto: ocorre a integração de todos os processos de todos os grupos de gerenciamento e suas atividades. Esse é um momento fundamental para consolidar, articular e integrar um plano geral, visando o gerenciamento e execução do projeto para atender os requisitos.

- Gerenciamento do escopo do projeto: é uma área necessária para garantir que o projeto inclua todo o trabalho necessário a desenvolver e controlar o que está e o que não está incluído no projeto para garantir que tudo seja cumprido conforme o planejado e podendo evitar trabalhos desnecessários que possam gerar futuros custos e atrasos.
- Gerenciamento do tempo do projeto: é fundamental para gerenciar o término e a entrega das fases do projeto ou entrega, sendo a maioria de planejamento e somente um de controle.
- Gerenciamento dos custos do projeto: é uma atividade que visa gerir e controlar o custo do projeto e orçamentos de modo que seja possível concluir o projeto dentro da estimativa do orçamento aprovado.
- Gerenciamento da qualidade do projeto: tem como finalidade assegurar que o projeto satisfaça todas as necessidades que ele foi originado e realizado, por esse motivo é fundamental a definição das políticas de qualidade, objetivos, requisitos.
- Gerenciamento dos recursos humanos do projeto: é uma atividade que organiza e gerencia as equipes do projeto. Diante isso, é possível motivar e gerenciar cada perfil de pessoa para melhorar o desempenho das equipes.
- Gerenciamento das comunicações do projeto: abrange os processos relacionados a criação, armazenamento, coleta, distribuição e recebimento de informações para as partes interessadas, nessa etapa o gerente de projetos investe a maior parte do seu tempo com comunicação, pois se acredita que comunicação é a chave entre as pessoas.
- Gerenciamento dos riscos do projeto: utiliza as atividades de planejamento, identificação, análise, planejamento de respostas e controle de riscos com finalidade de identificar todos os impactos negativos, procurando solucionar e eliminar tais eventos, fazendo com que aumente a probabilidade de efeitos positivos.
- Gerenciamento das aquisições do projeto: tem como finalidade gerir todo os produtos, serviços ou obtenção de resultados externos a equipe de projeto e definição de melhores fornecedores a gestão de contratos.

- Gerenciamento das partes interessadas do projeto (*stakeholders)*: é responsável por manter diálogo contínuo com os *stakeholders* do projeto, pois a identificação de todas as pessoas pode atingir o sucesso do projeto.

4 Análise comparativa: metodologia tradicional x ágil

Ao pesquisar dados que pudessem aprimorar este projeto, procurei por informações relevantes e úteis que pudessem acrescentar valor ao trabalho em questão. Os dados abaixo são do relatório de 2015 do Chaos Report, lançado pelo Standish Group (Hastie e Wojewoda,2015), com base em um estudo que analisou mais de 50 mil projetos, foi observado que os projetos que adotaram metodologias ágeis apresentaram uma taxa de sucesso superior. Curiosamente, essa tendência também é aplicada em projetos de grande escala, contrariando a suposta desvantagem das metodologias ágeis nesse contexto. Os resultados detalhados podem ser encontrados na Tabela 1.

Tabela 1 – Taxa de sucesso entre metodologia tradicional e ágil

	Abordagem	Sucesso	Desafiador	Falho
Todos	Ágil	39%	52%	9%
	Tradicional	11%	60%	29%
Projetos Grandes	Ágil	18%	59%	23%
	Tradicional	3%	55%	42%
Projetos Médios	Ágil	27%	62%	11%
	Tradicional	7%	68%	25%
Projetos Pequenos	Ágil	58%	38%	4%
	Tradicional	44%	45%	11%

Fonte: Hastie e Wojewoda (2015)

4.1 Comparação das Metodologias Ágeis

Para Araujo (2017) as metodologias convergem em muitas das suas práticas, uma vez que todas foram construídas com base nas mesmas premissas e finalidades vindas pelo Manifesto Ágil. Com base na análise realizada por autores que dominam suas metodologias, surge a comparação e levantamento de pontos fortes e fracos, de forma breve, na Tabela 2.

Tabela 2 – Comparação entre as metodologias abordadas

Metodologia	Pontos positivos	Pontos Negativos
Scrum	Flexibilidade para adaptação a mudanças de escopo; Interação entre equipe; Ciclo de entrega curtos.	Pouca documentação é criada; Falta de planejamento do escopo; Ausência de técnicas práticas.
Extreme Programming	Integração diária do código produzido; Equipes pequenas; Refatoração.	Indicado para equipes pequenas; Não é indicado para sistemas complexos, pois a análise não é detalhada;
Kanban	WIP (*Work in Progress)*; Adaptação rápida a mudanças; Identificação de gargalos e problemas.	Requer maturidade da equipe; Pode levar a priorização equivocada; Desenhado para manutenção.
Scrumban	WIP (*Work in Progress*); Visualização do fluxo do trabalho; Abordagem colaborativa.	Complexidade de implementação; Risco de sobrecarga de trabalho; Requer maturidade da equipe.
Scrum & XP	Refatoração; Desenvolvimento orientado a testes; Ciclo de entrega curtos.	Ausência de documentação; Indicada para equipe pequenas.
DSDM	Equipes autónomas; Utilização de protótipos.	Necessita de requisitos estáveis; Complexidade e curva de aprendizado inicialmente acentuada.

Fonte: Sutherland (2013), Savoine (2009), Teixeira (2013), De Sousa (2017).

4.2 PMBOK X Scrum

O PMBOK é um muito poderoso quando atrelada alguma metodologia ágil, como foi descrito em capítulos anteriores. Dessa forma, será feita uma análise comparativa entre o Scrum e o PMBOK na Tabela 3. Ademais, o Scrum é escolhido

para essa comparação devido a sua popularização no mercado de trabalho atual de acordo com a Figura 4.

Tabela 3 – Comparação PMBOK e Scrum

	Pontos positivos	Pontos negativos
PMBOK	Documentação do projeto bem controlada; Atividades bem definidas; Desenvolver a equipe do projeto.	Pouca flexibilidade do escopo; Responsabilidade centralizada; Alto custo em documentação e controle.
Scrum	Definição do gerenciamento do projeto; Interação entre equipe; Ciclo de entrega curto.	Pouca documentação é criada; Falta de planejamento do escopo;

Fonte: Araujo (2017)

De acordo com a Tabela 3 e capítulos anteriores, é possível notar características semelhantes e alguns pontos distintos, ou seja, o Scrum tem como um dos principais pontos negativos a falta de documentação e é possível perceber que esse aspecto no Guia PMBOK é algo muito bem explorado e desenvolvido. Outro fator negativo é que no PMBOK a centralização de responsabilidade é muito evidente, sendo o gerente de projetos responsável pelo planejamento, monitoramento e controle do projeto, já o Scrum, ele prioriza a interação entre equipe e a divisão de responsabilidade é estabelecida por toda a equipe, possibilitando que cada pessoa consiga desenvolver o trabalho de forma autônoma.

Desse modo, pode-se perceber que cada abordagem possui seu diferencial e especificidade para cada projeto. O PMBOK é mais adequado quando se trata de projetos com escopo bem definido e requisitos menos complexo. Já o Scrum é recomendado para projetos mais complexos e com escopo flexível. Como dito anteriormente, mesmo com suas especificidades é possível utilizar essas duas abordagens juntas para se completarem.

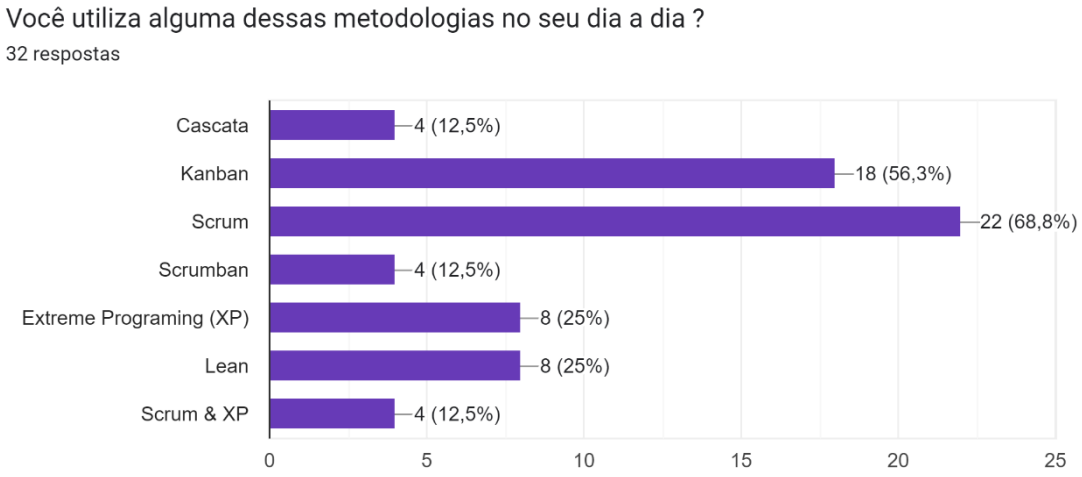
5 Resultados e Discussão

Para agregar embasamento prático e atual à problemática abordada neste trabalho, foi construído um questionário com o objetivo de coletar informações junto a profissionais que atuam na área e possuem experiência. Optou-se pelo uso do Google *Forms* (Formulário) devido a sua praticidade, disponibilidade e a criação de gráficos que possam ilustrar o assunto abordado.

5.1 Discussão

Nessa pesquisa foi possível coletar 32 respostas, permitindo a realização de uma análise dos resultados por meio de gráficos comparativos. A primeira pergunta abordou o uso das metodologias no dia a dia.

Gráfico 1 – Utilização de metodologias no dia a dia

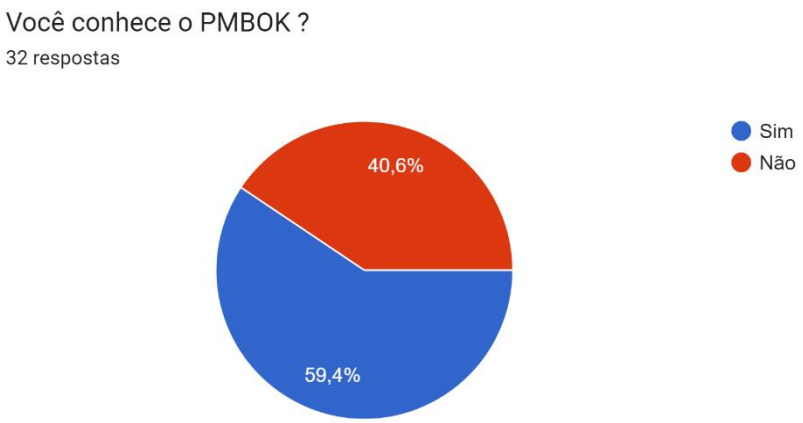


Fonte – Pesquisa realizada

O Gráfico 1 demonstra que 68,8% dos entrevistados utilizam o Scrum como metodologia principal no dia a dia e em segundo lugar o Kanban, por meio desse gráfico é possível de forma prática e embasada salientar o motivo da comparação entre Scrum x PMBOK como abordado anteriormente.

A segunda pergunta é referente ao conhecimento sobre o guia PMBOK, tendo o intuito de analisar se o público é familiarizado com esse guia tão importante no quesito gestão de projeto.

Gráfico 2 – Familiaridade com PMBOK

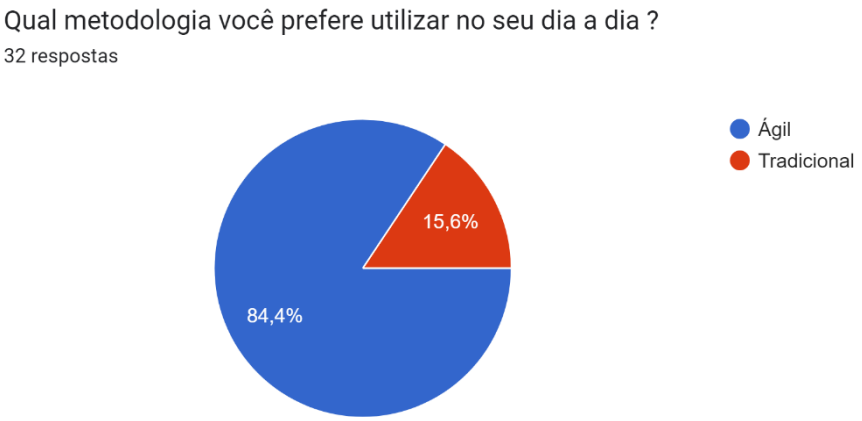


Fonte – Pesquisa realizada

De acordo com o Gráfico 2 é possível analisar que 59,4% das pessoas conhecem o PMBOK, sendo um valor já esperado quando se trata de uma ferramenta muito positiva. Porém 40,6% não conhecem esse guia, podendo analisar que é uma porcentagem muito alta deixando de conhecer uma ferramenta que pode auxiliar ainda mais na produtividade referente a entrega de produtos, sabendo que o mercado de trabalho tem como foco produtividade e entrega de produtos e serviços de qualidade, como discorrido no presente trabalho.

A terceira e última pergunta é referente a preferência dos participantes sobre qual metodologia abordar no dia a dia.

Gráfico 3 – Preferência de metodologia



Fonte – Pesquisa realizada

Como apresentado no Gráfico 3, o número de pessoas que prefere utilizar a metodologia ágil no dia a dia é de 84,4% tendo um valor muito discrepante e esperado, contudo quando se compara os números do Gráfico 2 e o Gráfico 3 é possível perceber que no Gráfico 3 existe uma discrepância acentuada, ou seja, se mais pessoas tivessem o conhecimento necessário sobre o PMBOK é possível que os participantes consigam melhorar a sua produtividade e gerenciamento de projetos no dia a dia.

6 Conclusão

Ao longo dos anos, o desenvolvimento de *software* tem evoluído juntamente com as metodologias que são utilizadas para a criação do mesmo. As metodologias tradicionais foram pioneiras quando o assunto era desenvolvimento de *software*, mas a forma como era desenvolvida refletia na qualidade final do produto, ou seja, eram executadas de maneira caótica, sem estrutura e sem nenhum tipo de planejamento. Devido a esse e outros fatores, o custo para fazer alterações e correções nos projetos eram exorbitantes.

Devido a essas problemáticas e a constante necessidade de se adequar as tecnologias que não param de evoluir e se manter de igual para igual em um mercado super aquecido, surge o manifesto ágil como forma de documentar e agrupar ideias de profissionais da área com a visão de criar técnicas e métodos voltados a otimizar o trabalho e escapar dessas problemáticas que a metodologia tradicional enfrentava. Em vista disso, o ponto de partida inicial para fugir desses problemas foi a metodologia ágil, devido a sua familiaridade com trabalho em equipe, princípios e práticas.

Contudo, mesmo as metodologias ágeis sendo muito conhecidas e utilizadas no mercado de trabalho, algumas pessoas as desconhecem ou possuem certo preconceito devido já estar familiarizado com a metodologia tradicional, ou seja, para uma empresa que já possui muitos anos utilizando a metodologia tradicional como base do seu negócio, pode acabar sendo muito custoso se adaptar para processos ágeis, mesmo tendo resultados positivos ao longo prazo.

Sabe-se que a metodologia ágil mesmo tendo todos seus pontos positivos ela também possui pontos negativos, os resultados da pesquisa demonstraram que é de suma importância compreender cada metodologia ágil e tradicional para analisar seus pontos fortes e fracos visando adaptação para um projeto, como constatado no trabalho, o método ágil mais utilizado é o Scrum e como mencionado ele possui diversos pontos fracos, como a definição do escopo e falta de documentação, surge o guia PMBOK para ultrapassar essas dificuldades.

Diante o exposto, o guia PMBOK apresenta uma abordagem mais estruturada e voltada para projetos com escopo bem definido e requisitos estáveis, portanto, não existe uma abordagem única que seja a melhor em todas as situações como foi destacado nas pesquisas comparativas. A escolha entre metodologia ágil, tradicional e PMBOK devem ser baseadas nas características e requisitos específicos

de cada projeto, sendo de extrema relevância considerar alguns fatores chave, como: natureza do projeto, cultura organizacional, maturidade da equipe e tempo do projeto.

Por fim, este estudo comparativo das metodologias ágeis e PMBOK oferece uma base de conhecimento e reflexão para profissionais e organizações que necessitam compreender a importância de cada método e ferramenta envolvendo no desenvolvimento de software, para auxiliar na tomada de decisões estratégicas e na escolha de uma abordagem adequada para concluir seus objetivos.

REFERÊNCIAS

ANAND, R. Vijay; DINAKARAN, M. - Popular Agile Methods in Software Development:
Review and Analysis. International Journal of Applied Engineering Research. 11:5 (2016) 3433–3437.

ARAUJO, Lucas Soares de. Gerenciamento de projetos de software com PMBOK e SCRUM: um estudo e análise comparativa do método tradicional e ágil. 2017.

BARBOSA, Anderson Luiz. Analise comparativa de metodologias para o gerenciamento de projetos de desenvolvimento de software. 2006. 135p. Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Eletrica e de Computação, Campinas, SP. Disponível em: https://hdl.handle.net/20.500.12733/1603424. Acesso em: 27 mar. 2023.

BECK, K. et al. Manifesto for Agile Software Development. Disponível em: http://agilemanifesto.org/. Acesso em: 04 mar. 2023.

BECK, Kent - Embracing change with extreme programming. Computer. 32:10 (1999) 70–77.
- Extreme programming explained: embrace change. [S.l.]: addison-wesley professional, 2000

BEZERRA, J. H. D. S. UTILIZAÇÃO DA METODOLOGIA SCRUM NO ENSINO DE GERENCIAMENTO DE PROJETOS DE SOFTWARE. South American Journal, UFAC, Rio Branco, v. 7, n. 1, 2020.

Cruz, Fábio. Scrum e PMBOK unidos no Gerenciamento de Projetos. Brasil, Brasport.

DE SOUSA, João Carlos Azevedo. **Estudo comparativo das metodologias ágeis e PMBOK**. 2017. Tese de Doutorado. Instituto Politecnico de Viseu (Portugal).

DOS SANTOS SOARES, Michel. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. **Revista Eletrônica de Sistemas de Informação**, v. 3, n. 1, 2004.

FADEL, Aline Cristine; SILVEIRA, H. da M. Metodologias ágeis no contexto de desenvolvimento de software: XP, Scrum e Lean. **Monografia do Curso de Mestrado FT-027-Gestão de Projetos e Qualidade da Faculdade de Tecnologia–UNICAMP**, v. 98, p. 101, 2010.

HASTIE, Shane; WOJEWODA, Stéphane - Standish group 2015 chaos report-q&a with Jennifer Lynch [Em linha], atual. 2015. [Consult. 3 jun. 2023].

JUNIOR, Wilton Antonio Machado et al. Controle de estoque: gestão de processos utilizando a ferramenta Kanban com o suporte da metodologia ágil Scrum. Research, Society and Development, v. 8, n. 1, p. e2381531, 2019.

Kerzner, H. (2017). Project Management Metrics, KPIs, and Dashboards: A Guide to Measuring and Monitoring Project Performance. John Wiley & Sons.

LEI, Howard et al. - A statistical analysis of the effects of Scrum and Kanban on software development projects. Robotics and Computer-Integrated Manufacturing. 43:2017) 59– 67.

LIKER, Jeffrey K. The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer. McGraw Hill Professional, 2004.

MACHADO, EDUARDO ERUSTES MAFRA; DE OLIVEIRA, CRISTINA CORRÊA. RELAÇÃO ENTRE AS BOAS PRÁTICAS DESCRITAS NO PMBOK E AS METODOLOGIAS ÁGEIS PARA O DESENVOLVIMENTO DE SOFTWARES. **Revista Científica e-Locução**, v. 1, n. 21, p. 27-27, 2022.

MELO, L. M. M. et al. Agile software development: A systematic literature review. Journal of Systems and Software, v. 144, p. 306-325, 2018.

MIGUEL, António - Gestão de Projetos de Software. FCA-Editora Informática, Ed. 2003)

PEREIRA, Paulo; TORREÃO, Paula; MARÇAL, Ana Sofia. Entendendo Scrum para gerenciar projetos de forma ágil. **Mundo PM**, v. 1, p. 3-11, 2007.

POPPENDIECK, M.: Lean Software Development. 29th International Conference on Software Engineering. IEEE. 2014.

PRESSMAN, R. S. Software Engineering: A Practitioner's Approach. McGraw Hill, 2016.

PRESSMAN, R. S.; MAXIM, B. R. Engenharia de software: uma abordagem professional. 8ª ed. AMGH, Porto Alegre, 2016.

REINERTSEN, Donald; BELLINSON, Tom - The principles of product development flow:generation lean product development. [S.l.] : Celeritas Publishing–2008Додаток Б Додаток В Додаток, 2014

REIS, Augusto Albuquerque. Scrumban-metodologia híbrida com scrum e kanban para desenvolvimento de software. 2021.

SAVOINE, Márcia et al. Análise de Gerenciamento de Projeto de Software Utilizando Metodologia Ágil XP e Scrum: Um Estudo de Caso Prático. **XI Encontro de Estudantes de Informática do Tocantins**, p. 93-102, 2009.

SCHWABER, Ken; SUTHERLAND, Jeff - Guia do Scrum. Julho de. 2013).

SEMEDO, Maria João Moreno et al. Ganhos de produtividade e de sucesso de Metodologias Ágeis VS Metodologias em Cascata no desenvolvimento de projectos de software. 2012.

SOMMERVILLE, Ian. Software engineering (ed.). America: Pearson Education Inc, 2011.

SOMMERVILLE, I. - Software EngineeringInternational computer science series Software engineering. [Em linha]. [S.l.] : Addison-Wesley, 2007 Disponível em WWW:. ISBN 978-0-321- 31379-9.

STOICA, Marian; MIRCEA, Marinela; GHILIC-MICU, Bogdan - Software development: Agile vs. traditional. Informatica Economica. 17:4 (2013) 64.

TEIXEIRA, Bruno Afonso - Agile and traditional project management: bridge between two worlds to manage IT projects PhD Thesis