



Metodologia Ágil - Framework Scrum na Gestão de Projetos de Software

Agile Methodology - Scrum Framework in Software Project Management

CAMARA, Tassia Estela¹; ALMEIDA, Gustavo².

tassiamcamara@gmail.com¹; gustavo.mc Almeida@gmail.com²

¹Analista de Sistemas - Gerente de Projetos MJV Tecnologia e Inovação, RJ.

²Engenheiro Civil - Consultor Sênior TOTVS, Rio de Janeiro, RJ

Informações do Artigo

Palavras-chave:

Scrum

Metodologia Ágil

Framework

Keyword:

Scrum

Agile Methodology

Framework

Resumo:

Este artigo buscou através de uma análise bibliográfica, em diversos autores e dentro de um estudo exploratório e descritivo, apresentar o framework Scrum e os benefícios da adoção de processos e técnicas que conduzem a metodologia Ágil. Metodologia esta que se encontra cada vez mais presente nas empresas, por possuir uma estrutura otimizada, permitindo adaptações na condução dos projetos. O Scrum se apresenta como uma importante estrutura para o desenvolvimento, sustentação e entrega de produtos em âmbito complexo, permitindo subdividir as tarefas em metas a serem desenvolvidas dentro das sprints. Em um mercado bastante competitivo as empresas buscam facilitar os métodos de criação e construção de suas soluções e escolhem formas de realizar entregas de um produto ou serviço atingindo a qualidade exigida com um tempo reduzido. Desta forma o framework Scrum se apresenta como uma saída eficaz para a gestão dos projetos de desenvolvimento de software.

Abstract

This article sought, through a bibliographic analysis of various authors and within an exploratory and descriptive study, to present the Scrum framework and the benefits of adopting processes and techniques that lead to Agile methodology. This methodology is increasingly present in companies due to its optimized structure, allowing adaptations in project management. Scrum presents itself as an important framework for the development, maintenance, and delivery of products in a complex environment, allowing tasks to be subdivided into goals to be developed within sprints. In a highly competitive market, companies seek to facilitate the methods of creating and building their solutions and choose ways to deliver a product or service that meets the required quality in a reduced time. Thus, the Scrum framework presents itself as an effective solution for managing software development projects.

1. Introdução

Muitas empresas de Tecnologia da Informação estão tentando ganhar agilidade no desenvolvimento de *software* mantendo a

consistência de um bom gerenciamento, e quem pode culpá-las(?) Equipes de sucesso que aplicam a metodologia ágil estão produzindo *software* de maior qualidade que

atendem melhor às necessidades do usuário, com maior rapidez e a um custo menor do que as equipes tradicionais. [1]

Vivenciamos uma fase riquíssima na quantidade e qualidade de admissões dos métodos ágeis em diversas organizações. Hoje é possível observar que informações sobre agilidade são discutidas nos mais diferentes graus no íntimo delas. E um dos maiores incentivadores desse movimento de admissão é o *scrum*. O *scrum* motivou os agilistas do mundo inteiro que obtivessem ferramentas para solucionar os problemas gerenciais e econômicos das empresas [2].

Dentro deste cenário, o objetivo deste artigo é apresentar esse *framework scrum* de processo ágil utilizado para gerenciar e controlar o desenvolvimento de projeto de *software* através de suas usabilidades.

2. O Manifesto Ágil

Ao se tratar do Scrum, deve-se antes abordar um assunto importante que é o manifesto ágil.

Em 2000 houve uma reunião da liderança da comunidade XP, *extreme programming*, com o propósito de abrir uma discussão sobre as práticas da metodologia tradicional. Foi observado e entendido pelos participantes que os métodos ágeis, hoje conhecidos como *Scrum*, XP, FDD (*Feature-driven development*) entre outros, caminhavam no sentido contrário dos métodos tradicionais, pois possuíam uma abordagem mais sutil e menos engessada. Como consequência dessa discussão concluiu-se que o uso do XP atenderia melhor como um método específico e que possuía algo em comum com os métodos ágeis. Posteriormente, Robert Cecil Martin, conhecido popularmente na área de desenvolvimento e gerenciamento de projetos como o Tio BOB, criou um encontro de métodos ágeis que aconteceria no ano seguinte. [3]

Em 2001, aconteceu o encontro em um resort na cidade de *Utah*, houve uma grande discussão e um consenso sobre como seriam

as metodologias mais adequadas para o desenvolvimento de *software* e então foi criado o que denomina-se hoje de manifesto ágil. O manifesto ágil possui quatro valores e doze princípios que direcionam a aplicação da metodologia ágil.[3]

2.1 Valores do Manifesto ágil

Os valores que suportam a metodologia ágil, são eles:

- i. maior importância para a interação da equipe de desenvolvimento de *software* e sua boa comunicação à utilização de ferramentas e processos;
- ii. o maior valor para o cliente é receber o produto funcionando e atendendo ao seu negócio, documentação é importante desde que se tenha o necessário e que agregue valor;
- iii. um projeto sendo construído de forma colaborativa com o cliente faz com que todos estejam alinhados e caminhando juntos para o sucesso do projeto;
- iiii. ser adaptativo, receber *feedback* e responder a mudança de forma coerente faz com que o fluxo de desenvolvimento de projeto com nível de incerteza funcione mais adequadamente do que se fosse guiado por um plano engessado. [3]

2.2 O Manifesto Ágil e seus princípios

O manifesto ágil também possui 12 princípios que norteiam a metodologia ágil, são eles:

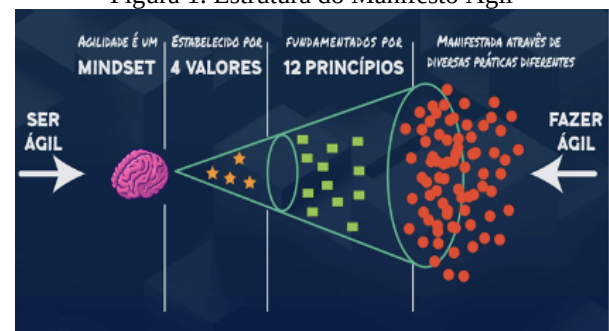
- a) o primeiro princípio ágil consiste na garantia do contentamento do cliente através da priorização de entregas incrementais e de valor agregado ao projeto de *software*;
- b) ainda priorizando a satisfação do cliente, o processo ágil aceita alteração de requisitos, caso seja necessário, com o intuito de atingir o objetivo de negócio do cliente mesmo após a entrega de uma iteração.

- c) o processo ágil possui uma flexibilidade para que cada etapa ofereça ao cliente a possibilidade de atualizar o seu produto de forma que mantenha a sua vantagem competitiva no mercado;
- d) o *Modus operandi* (modo de operação) do Ágil determina a realização de pequenas entregas incrementais, mas sem perder a qualidade do seu produto entregue;
- e) a colaboração é uma peça importante na engrenagem do processo ágil, pois com a participação de todo o time em cada etapa, desde o cliente com a sua percepção sobre o negócio ao time que desenvolve, é possível chegar ao produto ideal;
- f) poder de autogestão, subsídio de ferramentas e ambientes adequados ao desenvolvimento do projeto, são primordiais para a motivação do time ágil;
- g) comunicações otimizadas são mais eficazes na transmissão de conhecimento de um projeto comparada a documentação, porém, os registros ainda são necessários se e somente se agregarem valor;
- h) na evolução de um projeto ágil é considerado um entregável um *software* que atingiu o seu pleno funcionamento e não atividades finalizadas;
- i) a construção de um ambiente favorável ao projeto, com organização, planejamento e envolvimento de todos os *stakeholders* é fundamental para um acompanhamento do projeto de forma contínua e sustentável;
- j) a constante realização do refinamento e a verificação do alinhamento de requisitos técnicos e de negócios, provê uma solução de valor sem grandes mudanças na fase de sua entrega final;
- k) simplificação no trabalho desenvolvido pelo time focando mais no desenvolvimento do produto e menos em

artefatos do projeto, ganhasse na produtividade;

- l) times ágeis possuem em seu core a auto organização, são perfeitamente capazes de trabalharem realizando as suas divisões de tarefas sem a necessidade da interferência do *Scrum Master*;
- m) na finalização de cada iteração a equipe realiza uma avaliação para que possam identificar o que deve ser mantido e o que deve ser melhorado, afinal, um dos objetivos é a melhoria contínua.[4]

Figura 1. Estrutura do Manifesto Ágil



Fonte: Vilas Boas [4]

3. SCRUM

3.1 A criação do Scrum

Em 1993, Jeff Sutherland criou o *Scrum* e o nome é uma analogia à pesquisa realizada em 1986 pelos professores Nonaka e Takeuchi, que foi publicada na revista de negócios de Harvard. Neste estudo, os autores fizeram um comparativo entre as equipes de alto desempenho e multifuncionais com a formação “*scrum*” que existia nas equipes de *Rugby*. Esse processo foi formalizado para a indústria mundial de *software* no primeiro artigo sobre *Scrum*, escrito por Ken Schwaber em 1995 e publicado na conferência de *OOPSLA*. [5]

3.2 Definição

Segundo o Scrum Guide, o Scrum é:

Um framework que possibilita as pessoas a realizarem tratativas e gerarem soluções para problemas que tenham complexidade alta e que sejam adaptativos enquanto de forma criativa entregam produtos com alto

valor agregado. O Scrum é de fácil entendimento, porém complexo em sua dominância. É um framework usado desde o início de 1990 para gerenciar projetos com criação de produtos complexos. Ele é um framework altamente adaptável oferecendo a possibilidade de customização quanto aos processos e técnicas. A clareza sobre a sua eficácia vem da melhoria contínua dos produtos, times e ambientes de trabalho proveniente de suas práticas e técnicas. (p.3) [4]

3.3 Conceitos

O scrum é um *framework* que detém em seu fundamento um processo com entregas incrementais e iterativas. A equipe do projeto é responsável por realizar a compreensão dos requisitos do produto, assim como, viabilidade técnica, análise das habilidades dos integrantes da equipe, ferramentas, etc. Para que juntos possam gerar a melhor solução técnica para a criação do produto que não necessariamente seja a versão final, pois podem haver alterações para atender alguma necessidade do cliente ou da evolução do produto. A criatividade do processo é o fundamento responsável pela eficácia do scrum.

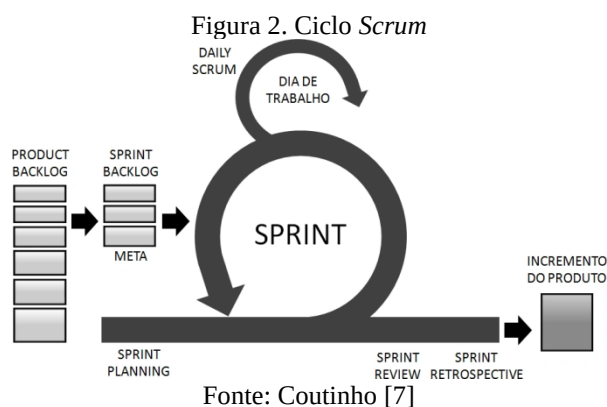
Tendo em vista facilitar a compreensão deste *framework* é importante aprendermos sobre algumas expressões que fazem parte do scrum:

3.3.1 Elementos

- a) *Product Backlog*: Lista dinâmica com todas as atividades necessárias e que devem ser realizadas para atingir a entrega do resultado final. Esse grupo de atividades deve ser organizado de forma a agregar valor a cada entregável. Ela pode ser alterada durante a execução do projeto, ou seja, novas tarefas podem surgir e as existentes podem ser excluídas ou substituídas. (p.14) [6];
- b) *Sprint*: Um período de trabalho com duração geralmente de duas semanas e ou menos de um mês, onde algum valor é gerado e adicionado a um produto que está sendo desenvolvido. Este ciclo é incremental, ou seja, uma etapa após a

outra até a entrega do produto final. (p.9) [6];

- c) *Sprint Backlog*: Relação com todos os itens que serão desenvolvidos dentro do tempo estipulado de uma *sprint*. Para esta lista evita se qualquer mudança que possa impactar de forma negativa a realização dos entregáveis deste produto. (p.16) [6];
- d) *Sprint Planning Meeting*: Momento onde o time faz o planejamento dos itens priorizados para o desenvolvimento na *sprint*. (p.10) [6];
- e) *Sprint Review Meeting*: Reunião que acontece a cada finalização da *sprint* com o objetivo de validar a entrega da equipe, verificar se os requisitos estabelecidos no planejamento foram seguidos e recolher o *feedback* do cliente e ou usuário final. (p.13) [6];
- f) *Daily Scrum*: Reunião diária com duração aproximada de 15 minutos onde os membros da equipe contam o que fizeram no dia anterior e o que planejaram para o dia de trabalho que se inicia. (p.12) [6].



3.3.2 Papéis e Responsabilidades

No *Scrum* as responsabilidades em um projeto estão divididas entre os três papéis essenciais:

- i. *Product Owner* (Proprietário do Produto):

No planejamento do Produto o *Product Owner* é o detentor do conhecimento de negócio do produto, ele trabalha com os

Stakeholders chave da empresa (partes interessadas-grupo de pessoas que sofrem impacto direto e indireto do projeto) para conseguir vislumbrar o produto a ser construído. É o *Product Owner* quem decide se um item atende a todas as expectativas. É o responsável pela validação dos critérios de aprovação durante a execução da Sprint. (p.6) [6];

ii. *Scrum Master* (Mestre *Scrum*): É o líder servo do time, zelador dos processos *Scrum* e defensor do time contra interferências externas. A garantia que todo o time esteja funcional e produtivo, prover visibilidade e comunicação constante além de remover impedimentos relatados pelo time nas reuniões diárias, são algumas de suas atribuições. (p.7) [6];

iii. *Development Team* (Equipe de Desenvolvimento): Time auto organizável responsável por prover o direcionamento técnico no projeto, gerar as estimativas e identificar dependências técnicas do *backlog*. Os membros da equipe são responsáveis pelo sucesso do projeto e por cada entrega realizada. (p.6) [6].

3.4 Técnicas do *Scrum*

O *Scrum* fornece técnicas específicas para o desenvolvimento do projeto, e estabelece um agrupamento de regras e práticas que devem ser seguidas para a condução e sucesso do projeto.

3.4.1 Geração do *Product Backlog*

No início de um novo projeto o *Product Owner* juntamente com os *Stakeholders* (partes interessadas) realiza uma reunião para o entendimento das expectativas e reais necessidades do cliente. É após essa reunião que o *Product Owner* possui os requisitos funcionais e não funcionais necessários para a geração do *product Backlog*.

O *Product Backlog* geralmente não é completo no início do projeto, ele é criado inicialmente com requisitos mais claros e se expande conforme se adquire mais informações sobre o produto final.

Também, as informações que devem constar em um *Product Backlog*, são:

ID: Identificação única com auto incremento criado para o controle das histórias;

Nome: Descrição curta da história, mas suficiente para que o time entenda sobre o que está sendo falado;

Prioridade: Geralmente dividido entre baixo, médio e alto. Ajuda a identificar quais atividades são mais importantes para o desenvolvimento;

Estimativa Inicial: Tempo estipulado pelo time para a execução de cada atividade;

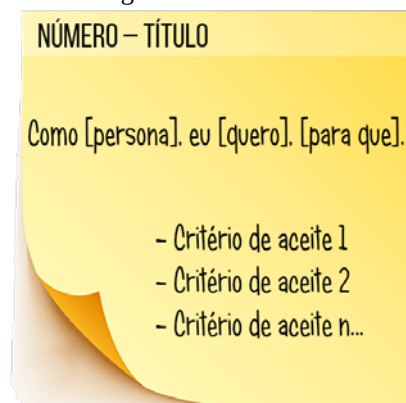
Observação: Campo livre para descrever informações importantes sobre a história ou tarefas.

3.4.2 *User Stories*

Com a criação e definição do *Product Backlog*, o *Product Owner* e o cliente alinham quais são os requisitos prioritários e quais possuem mais valor ao negócio. Feito isso, cada requisito é selecionado e então o *Product Owner* cria as *User stories* (estória do usuário), que são textos simples, curtos, criados sob o ponto de vista do usuário e que auxilia a equipe de desenvolvimento no entendimento para a implementação do sistema e estimativa das atividades. Geralmente são escritas respeitando três direcionamentos importantes, são eles:

- O Ator - pessoa que tem ação no sistema, quem irá usá-lo;
- Ação - o que o ator deseja fazer no sistema, é através da ação que ele atinge o seu objetivo no sistema;
- Funcionalidade - é o resultado gerado pela ação do ator. [8]

Figura 3. User Stories



Fonte: Cardoso [8]

Para a criação de uma boa história, é necessário também seguir um conceito chamado INVEST:

Independente: História sem dependências e que pode ser implementada sem interferir nas demais;

Negociável: A história deve ser escrita de forma que possibilite a negociação do time sobre a sua complexidade na implementação;

Valiosa: A história deve possuir um item que seja valioso para o produto e que este valor seja claro para o time e ao cliente.

Estimável: A história deve ser escrita com um nível de detalhamento que permita ao time estimá-la.

Pequena: A história precisa ser pequena para que possa ser implementada em uma única *sprint*.

Testável: A história precisa ser testada para a garantia do funcionamento e a execução do teste deve ser de fácil entendimento para o time.

Aplicando o INVEST o time de desenvolvimento terá histórias de qualidade, mas às vezes existirá a necessidade em dividi-las em Temas e épicos, que em suas definições são:

Temas: É o agrupamento de histórias de usuários que possuem informações comuns. Muitas delas estão direcionadas a seguir um único caminho, porém, não são necessariamente entregues juntas. É

importante dizer que um tema pode obter vários épicos ou várias histórias de usuário.

Épicos: São grandes histórias que não podem ser concluídas em uma única *sprint*. As histórias que contemplam o épico podem ser concluídas de forma independente, porém só agregam valor ao negócio quando o épico é finalizado. [12]

Visando obter histórias condizentes com os requisitos do produto, são criados os critérios de aceitação que são, listas de critérios estabelecidos e validados pelo *Product Owner* com o intuito de garantir a qualidade e assertividade das funcionalidades desenvolvidas. Esses critérios são adicionados as histórias e são divididos em dois tipos:

- i. *Definition of ready* (definição de preparado): Conjunto de critérios que cada item do *backlog* deve obedecer para que ele esteja com todas as informações descritas e preparadas para a implementação; [12]
- ii. *Definition of done* (definição de pronto): Conjunto de critérios que valida cada item desenvolvido do *backlog* da *sprint*. Somente após essa validação que o item é considerado concluído para a entrega. [13]

3.4.3 Estimativas

Com as histórias definidas, o time chega à fase de estimativa para o desenvolvimento da atividade e para a geração desta existe um método chamado *Planning Poker* (planejamento por cartas), muito utilizado pelas equipes ágeis. [10]

O *Planning Poker* é um método em formato de jogo onde o time mede as histórias pela sua referência de grandeza, como, P, M, G.

Para iniciar o jogo é necessário definir os *Story Points*, as unidades de medida. Segundo Mike Cohn:

Um story point é uma junção da quantidade de esforço envolvido no desenvolvimento de uma feature, a complexidade desse desenvolvimento é o risco contido nele. [10]

Definido os valores para cada pontuação, o jogo pode ser iniciado. Cada integrante do

time terá em suas mãos um baralho de cartas numeradas, as mais comuns começam com a sequência Fibonacci (0,1,2,3,5,8,13,20,40,100) que representarão os pontos. O Product Owner apresenta cada requisito e cada membro do time escolhe uma carta com o valor de acordo com o seu entendimento sobre o tamanho do requisito.

Caso não haja consenso sobre o tamanho do requisito, é solicitado ao integrante que mostrou o menor valor e o de maior valor que expliquem os seus motivos e é aberta uma discussão sobre o requisito, então uma nova rodada é feita até que se haja uma unanimidade. O processo se repete até que todos os requisitos sejam avaliados. [9]

Figura 4. Planning Poker



Fonte: Bonfim [9]

3.4.3 Quadro de tarefas

Com as atividades estimadas, chega o momento da *Sprint Planning Meeting*, o time planeja as atividades priorizadas para o ciclo e inicia a fase de desenvolvimento do produto.

Para garantir o gerenciamento do desenvolvimento do produto é comumente usado um quadro de tarefas, que pode ser *online* ou físico, ele geralmente é dividido em 3 colunas, que são: a fazer, em execução e feito. O time usa *post its* para descrever cada tarefa que será executada na *sprint* e estas são coladas no quadro. As tarefas são movidas a cada atualização de *status* que ocorre nas reuniões de *Daily Scrum*. Ao final de cada *sprint* é esperado que todas as tarefas estejam na coluna de feito.

Com o quadro de tarefas é possível de forma rápida e objetiva, obter a transparência sobre o trabalho realizado diariamente pela equipe de desenvolvimento e oferece a inspeção necessária para se ter sucesso nas entregas de cada *sprint* e ao final do projeto.

Figura 5. Quadro de tarefas



Fonte: Bonfim [9]

3.4.4 Métricas

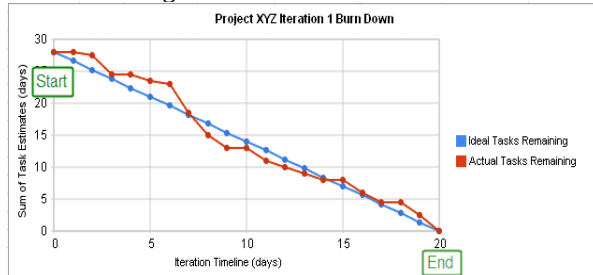
As métricas possuem a finalidade de mensurar cada estágio do ciclo de vida de um projeto, proporcionando assim a compreensão sobre o desempenho da equipe de desenvolvimento e qualidade do produto desenvolvido.

É através das métricas que a equipe possui insumos para realizar as tomadas de decisões necessárias e manter a melhoria contínua da equipe.

Existem diversas métricas, mas serão abordadas as mais utilizadas:

a. Burndown da Sprint: É um indicador que auxilia na gestão da Sprint Backlog e tem como responsável a equipe de desenvolvimento. Na Sprint Planning, o time decide quantas horas ou pontos possuem como capacidade de desenvolvimento dentro do período de tempo estipulado para a Sprint. Com a capacidade de desenvolvimento definida e quantidade de horas ou pontos de trabalho, o time possui o entendimento sobre o que pode ser entregue diariamente, esse valor é subtraído da iteração, e o resultado é o valor previsto.

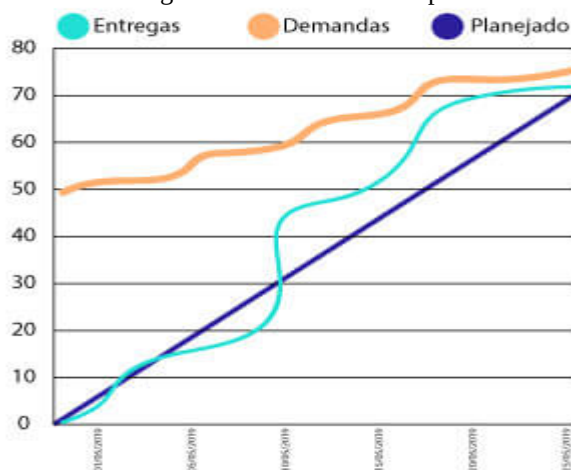
Figura 6. Gráfico de Burndown



Fonte: AMcom[14]

b. Burnup Projeto: Diferentemente do burndown da Sprint que demonstra o quanto a equipe entregou de escopo por sprint, este indicador demonstra o quanto o time realizou de entrega sobre o escopo total do projeto. Neste gráfico são consideradas duas variáveis: o prazo planejado (eixo X), entregas realizadas (eixo Y).

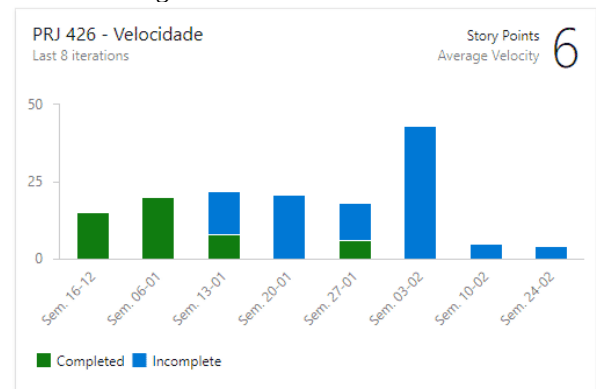
Figura 7. Gráfico de Burnup



Fonte: AMcom[14]

c. Velocidade da equipe: Esta métrica proporciona ao *Scrum Master* conhecer o ritmo do seu time. Para isto, precisa ser levado em consideração algumas unidades de medidas, que podem ser as horas de desenvolvimento que a equipe consegue suportar, quantos pontos das *User Stories* o time consegue entregar ou itens entregues em cada sprint. Essa medição é mais adequada e assertiva se for realizada após a terceira *sprint*. [14]

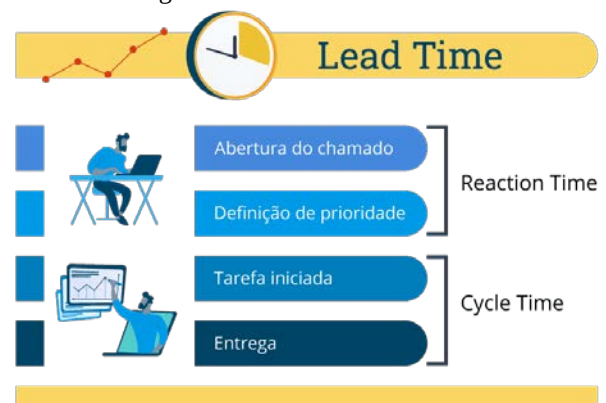
Figura 8. Gráfico de Velocidade



Fonte: AMcom[14]

d. *Lead Time*, *Cycle Time* e *Reaction Time*: A *Lead Time* representa o tempo total desde a criação da tarefa até a sua conclusão. O *Lead Time*, pode ser dividida em duas sub métricas, o *Cycle Time*, que é a representação do tempo que uma tarefa está em desenvolvimento até a sua conclusão e o *Reaction Time* que representa o tempo que uma atividade foi criada no *backlog* até o início de seu desenvolvimento. [15]

Figura 9. Gráfico de Velocidade



Fonte: Carolina [16]

4. Considerações finais

A contribuição principal desta pesquisa é mostrar através do estudo do tema os benefícios que a metodologia ágil, especificamente o framework Scrum, trouxe para os projetos de desenvolvimento de software, principalmente os projetos que possuem incertezas sobre os requisitos do produto, sofrem diversas mudanças e possuem equipes pequenas.

Toda metodologia que venha a oferecer resultados mais satisfatórios num ambiente

cada vez mais competitivo e otimizado é válido de ser empregado, então observou-se que para atender a uma necessidade latente do mercado de gerarmos produtos ou serviços de forma mais rápida, contínua e assertiva, seria necessário o uso da metodologia ágil. Desta forma o *Scrum*, se firmou como um dos *frameworks* mais usados para a gestão dos projetos de desenvolvimento de *software*.

Conceitualmente o *scrum* visa tornar mais simples os projetos, onde o foco se torna as demandas do negócio e o que é relevante diretamente para o cliente. Isso é viabilizado através do uso de curtos ciclos de desenvolvimento e revisões ao fim de cada um deles, garantindo não somente *feedbacks* constantes, mas uma maior integração entre os membros da equipe. E dentro deste ponto o *scrum* apresenta outro de seus principais fatores, uma capacidade de melhoria constante dentro do que é necessário.

Para tanto este *framework* cria ganhos não somente quanto ao uso do tempo, o que gera avanços financeiros óbvios para a empresa e para o cliente, mas também a possibilidade de um processo de criação e desenvolvimento constante em todos os projetos de forma a acrescentar à equipe conhecimentos e formas inovadoras de abordar novos projetos que deverão ser úteis em ocasiões futuras.

Gerencialmente o Scrum se torna uma ferramenta ideal para a constante e cada vez mais acelerada mudança da demanda e do mercado quando se adapta constantemente as questões que são improváveis de serem previstas no início do projeto, mantendo os objetivos macros iniciais fixos, mas alterando o seu escopo conforme a identificação da necessidade e sua forma de execução para o dinamismo e construção do projeto.

Constata-se então que a utilização do Scrum é altamente recomendada, por proporcionar um ambiente de trabalho mais favorável à criatividade, fomentando a colaboração entre a equipe, por sua flexibilidade e poder de adaptação na construção do produto, realização de entregas de valor ao cliente a cada iteração finalizada,

e a garantia de processos internos mais ágeis e com transparência fazendo com que as empresas alcancem seus objetivos e vantagens competitivas frente aos seus concorrentes.

5. Referências

- [1] COHN, Mike. *Desenvolvimento de software com Scrum - Aplicando métodos ágeis com sucesso*, p.25, 2009 Disponível em: https://books.google.com.br/books?id=gbgpDwAAQBAJ&lpg=PP1&ots=LKcEmqKYd-&dq=m%C3%A9todo%20%C3%A1gil%20scrum&lr=lang_pt&hl=pt-BR&pg=PR2#v=onepage&q&f=false. Acesso em: 11out.2021
- [2] SABBAGH. Rafael. *Scrum: Gestão ágil para projetos de sucesso*, p.11, 2014. Disponível em: https://books.google.com.br/books?id=pG-CCwAAQBAJ&lpg=PT7&ots=ETOWoRDEz8&dq=scrum%20%C3%A1gil&lr=lang_pt&hl=pt-BR&pg=PT11#v=onepage&q&f=false. Acesso em 11 out. 2021
- [3] BRASILEIRO, Roberto. *Manifesto Ágil*. Disponível em: <https://www.metodoagil.com/manifesto-agil/> Acesso em 11out.2021
- [4] VILAS BOAS, Carol. *Manifesto Agil: 20 anos*. Disponível em: <https://www.zup.com.br/blog/manifesto-agil> Acesso em 11out.2021
- [5] EMANOELE, Alícia. *A história da origem do Scrum e a sua relação com o Rugby*. Disponível em: <https://www.voitto.com.br/blog/artigo/surgimento-do-scrum> Acesso em 25dez.2021
- [6] SCHWABER, Ken; SUTHERLAND, Jeff. *Scrum Guide*. 2017. Disponível em: <https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf> Acesso em 25dez.2021.

- [7] COUTINHO, Thiago. *Sprint Scrum: o que é e como funciona?* 2018 em: <https://www.voitto.com.br/blog/artigo/sprint-scrum> Acesso em 25dez.2021.
- [8] CARDOSO, Bruno. *User Story - História de usuário.* <https://odonodoproduto.com/user-story-historia-de-usuario/>. Acesso em 25dez.2021.
- [9] BONFIM, Marcio. *Práticas e artefatos comumente utilizados com Scrum.* Disponível em: <https://www.devmedia.com.br/praticas-e-artefatos-comumente-utilizados-com-scrum/27911> Acesso em 25dez.2021.
- [10] BRASILEIRO, Roberto. *Planning Poker: A melhor maneira de estimar qualquer atividade.* Disponível em: <https://www.metodoagil.com/planning-poker/> Acesso em 25dez.2021.
- [11] BERNARDO, Kleber. *Estória de usuário. Você saberia contar?* Disponível em: <https://www.culturaagil.com.br/estoria-de-usuario-voce-saberia-contar/> Acesso em 29 dez.2021.
- [12] VENTURA, Plínio. *Definition of Ready (DoR) – A importância do item Ready no Scrum.* Disponível em: <https://www.ateomomento.com.br/definicion-of-ready-dor-a-importancia-do-item-ready-no-scrum/> Acesso em 29 dez.2021.
- [13] VENTURA, Plínio. *Definition of Done (DOD) – A importância do item pronto no Scrum.* Disponível em <https://www.ateomomento.com.br/definicion-of-done-dod-a-importancia-do-item-done-no-scrum/> Acesso em 29 dez.2021.
- [14] AMCOM. *Qual a importância de mensurar projetos ágeis? Métricas e indicadores para implementar no seu time!* <https://www.amcom.com.br/importancia-de-mensurar-projetos-ageis-metricas-e-indicadores/> Acesso em 08 jan.2022
- [15] MINETTO, Elton. *O que é Lead time, Cycle Time e Reaction Time?* <https://imasters.com.br/agile/o-que-e-lead-time-cycle-time-e-reaction-time> Acesso em 08 jan.2022
- [16] CAROLINA. *Como reduzir o lead time dos processos e aumentar a produtividade na sua empresa* <https://blog.runrun.it/lead-time/> Acesso em 08 jan.2022