UNIVERSIDADE DE SOROCABA

PRÓ-REITORIA DE PÓS-GRADUAÇÃO, PESQUISA, EXTENSÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM PROCESSOS TECNOLÓGICOS E AMBIENTAIS

Fernando Luís dos Santos

METODOLOGIAS ÁGEIS (SCRUM): UM ESTUDO SOBRE OS REFLEXOS NO TEMPO E CUSTO DA GESTÃO DE PROJETOS NO DESENVOLVIMENTO DE SOFTWARE

Sorocaba/SP
2019

Fernando Luís dos Santos

METODOLOGIAS ÁGEIS (SCRUM): UM ESTUDO SOBRE OS REFLEXOS NO TEMPO E CUSTO DA GESTÃO DE PROJETOS NO DESENVOLVIMENTO DE SOFTWARE

Dissertação apresentada à Banca Examinadora do Programa de Pós-Graduação em Processos Tecnológicos e Ambientais da Universidade de Sorocaba, como exigência para obtenção do título de Mestre em Processos Tecnológicos.

Orientador: Prof. Dr. Rogério Augusto Profeta

Sorocaba/SP
2019

Fernando Luís dos Santos

METODOLOGIAS ÁGEIS (SCRUM): UM ESTUDO SOBRE OS REFLEXOS NO TEMPO E CUSTO DA GESTÃO DE PROJETOS NO DESENVOLVIMENTO DE SOFTWARE.

Dissertação apresentada à Banca Examinadora do Programa de Pós-Graduação em Processos Tecnológicos e Ambientais da Universidade de Sorocaba, como exigência para obtenção do título de Mestre em Processos Tecnológicos.

Orientador: Prof. Dr. Rogério Augusto Profeta

BANCA EXAMINADORA

_____

Prof. Dr. Rogério Augusto Profeta
Universidade de Sorocaba - UNISO

_____

Prof. Dr. Lauro Carvalho de Oliveira
Faculdade de tecnologia de Sorocaba - CPS

_____

Prof. Dr. Thomaz Augusto Guisard Restivo
Universidade de Sorocaba - UNISO

Santos, Fernando Luís dos

S236m Metodologias ágeis (Scrum) : um estudo sobre os reflexos no tempo e custo da gestão de projetos no desenvolvimento de software / Fernando Luís dos Santos. – 2019.

126 f. : il.

Orientador: Prof. Dr. Rogério Augusto Profeta

Dissertação (Mestrado em Processos Tecnológicos e Ambientais) – Universidade de Sorocaba, Sorocaba, SP, 2019.

1. Scrum – Desenvolvimento de software. 2. Desenvolvimento ágil de software. 3. Software – Desenvolvimento. 4. Administração de projetos. I. Profeta, Rogério Augusto, orient. II. Universidade de Sorocaba. III. Título.

AGRADECIMENTOS

A presente dissertação de mestrado não poderia chegar em sua conclusão sem o apoio de várias pessoas.

Em primeiro lugar, não posso de deixar de agradecer a Luane que fez parte das diversas madrugadas e pesquisas ao meu lado. Participou de vários altos e baixos, e sempre teve uma palavra motivadora, até porque, em todo tempo mostrou-se acreditar em meu trabalho.

Desejo igualmente agradecer ao meu orientador Rogério Augusto Profeta, principalmente por toda paciência, empenho e nas dicas que fizeram total sentido prático ao meu trabalho. Muito obrigado pelo apoio e correções que elevaram o meu conhecimento.

Agradeço a todos os meus colegas de mestrado em Processos Tecnológicos e Ambientais, cujo o apoio e amizade tivemos em todo momento.

Não poderia deixar de agradecer aos meus grandes amigos do Santander que estiveram do meu lado com uma força indiscutível.

E por fim, agradecer a minha família e amigos pelo apoio incondicional que me deram, em especial aos meus pais e irmãos. E claro, não poderia esquecer de agradecer ao Couty e a Ily.

.

RESUMO

Scrum é um *framework* composto de diversos artefatos, papéis e reuniões utilizado na gestão de projetos no desenvolvimento de *software*, em busca de entregáveis em menor tempo e com um maior valor agregado, além de uma aproximação com os clientes, redução de custos, baixo risco na implantação, visibilidade do grupo e bons resultados financeiros para empresa. O Scrum propõe manter a equipe existente, sem grandes investimentos em estrutura organizacional, mas com uma mudança de cultura e um alto envolvimento em todos os níveis da organização. Propõe, ainda, a modificação do método tradicional, utilizado em diversas organizações, denominado *Waterfall* que está fundamentado em ciclos de entregas. Há uma extensa bibliografia sobre os resultados na utilização do *Scrum* e outros métodos apresentados no *Manifesto Agile.* Este estudo traz uma série de perguntas diretas, buscando avaliar as promessas descritas na bibliografia, verificando sua eficácia no resultado apresentado por pessoas que trabalharam com o *Scrum* e o *Waterfall*. Faz-se uma comparação direta, avaliando os benefícios e as complicações de cada método de trabalho. Apresentam-se, ainda, dados concretos para implantação do *Scrum* em qualquer organização, com uma explicação dos principais conceitos, fases para sua implementação, principais características, dificuldades e resultados esperados. A gestão de projetos no desenvolvimento de *software* é um ponto estratégico das empresas, gerando pressão na cobrança para que sejam eficazes e assertivas as entregas, além do alto nível de necessidade em responder às mudanças. Para organizações e pessoas dispostas a entender, fomentar e aplicar novos valores, apresentam-se informações práticas e já testadas em diversas organizações com sucesso.

Palavras-chave: *Scrum*. Ágil. Projetos. Cascata. Tempo.

ABSTRACT

Scrum is a framework composed of several artifacts, roles and meetings used in project management in software development, in search of deliverables in less time and with a higher added value, in addition to a customer approach, cost reduction, low risk in the deployment, group visibility and good financial results for the company. The Scrum proposes to maintain the existing team, without major investments in organizational structure, but with a change of culture and a high involvement in all levels of the organization. It also proposes the modification of the traditional method, used in several organizations, called Waterfall, which is based on delivery cycles. There is an extensive bibliography on the results in using Scrum and other methods presented in the Agile Manifesto. This study brings a series of direct questions, trying to evaluate the promises described by that bibliography, verifying their effectiveness in the result presented by people who worked with Scrum and Waterfall. A direct comparison is made, evaluating the benefits and complications of each method of work. It also presents concrete data for Scrum implementation in any organization, with an explanation of the main concepts, phases for its implementation, main characteristics, difficulties and expected results. Project management in software development is a strategic point of the companies, generating pressure on the collection so that deliveries are effective and assertive, as well as the high level of need to respond to changes. For organizations and people willing to understand, foster and apply new values, we present practical information already tested in several successful organizations.

Keywords: *Scrum*. Agile. Projects. Waterfall. Time.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE SIGLAS E TERMOS EM INGLÊS

Agile	Ágil
Dev team	Equipe de desenvolvimento - um subconjunto da equipe Ágil
Framework	Estrutura de suporte
KPI	Key Performance Indicator
LNU	Linguagem Necessária ao Usuário
MCSW	Máquina Construída por meio de Software
MPU	Máquina Possuída pelo Usuário
MVP	Minimum Viable Product (Produto mínimo viável)
P.O.	Product owner (Proprietário do produto)
PLoP	Padrões de Linguagem de design de Programação
Product backlog	Acúmulo de produtos incompletos / lista de itens por fazer
ROI	Return on Investiment
SM	Scrum Master
Spike	Um método de teste de produto originário da Extreme Programming
Sprint	Período de tempo definido durante o qual o trabalho específico deve ser concluído e preparado para revisão
Sprint backlog	Os itens do Sprint Backlog são tarefas por fazer, extraídas do Product Backlog
Squad	Uma estrutura organizacional multifuncional de um conjunto de equipes
Stakeholders	Partes interessadas
Stories	Histórias - itens do Product Backlog que representam parte do produto a ser implementado. As histórias devem conter uma descrição detalhada do que deve ser implementado.

SUMÁRIO

1 INTRODUÇÃO

A exigência de mercado está cada vez mais agressiva no que se refere à cobrança de velocidade na entrega de seus produtos e serviços. Isso faz com que exista a necessidade de se dar ao cliente suporte para que confie na capacidade da empresa em atender à demanda.

No caso de empresas ligadas ao desenvolvimento e à produção de *software*s, isso não é diferente. A empresa há que estar sempre um passo à frente, e a confiança conquistada é uma das vantagens competitivas. É essencial que empresas assumam o compromisso de trazer formas cada vez mais eficientes de implantação de seus projetos tecnológicos, acompanhando a rapidez do desenvolvimento tecnológico e as consequentes mudanças nos pedidos dos clientes. Eficácia, adaptabilidade, qualidade e competividade são elementos chaves.

Este trabalho busca fundamentar o processo de desenvolvimento de *software* e apresentar as vantagens do *Scrum* como um método que atende a demanda atual com bons resultados. (LEFFINGWELL; SMITS, 2005)

Fernandes (2003) indica que "*software* é um produto do trabalho humano cada vez mais presente na sociedade" e esclarece alguns termos fundamentais para o entendimento sobre a diferenciação entre os processos de desenvolvimento, produção e utilização de *software*s:

> HIERARQUIAS DE MÁQUINAS E USUÁRIOS
> Quem usa um *software* em geral é chamado de usuário. O *software* não é, de fato, uma máquina, mas sim uma descrição de máquina. Ou seja, *software* é um artefato virtual, incapaz de realizar trabalho a menos que exista uma máquina que carregue e interprete as instruções e informações contidas no mesmo, o que resulta na construção de outra máquina, de ordem superior, com a qual interage o usuário. Em outras palavras, na análise de qualquer sistema de computação estaremos sempre falando de duas máquinas. Uma máquina, de ordem **n**, é a máquina possuída pelo usuário (MPU) antes da carga e interpretação das instruções e informações contidas no *software*. A outra máquina, de ordem **n+1**, é a máquina com a qual o usuário interage, e que surge quando a máquina de ordem **n** faz a interpretação do *software*. Da combinação dinâmica entre a MPU e o *software* surge a máquina de ordem **n+1**, à qual dar-se-á o nome de máquina construída por meio de *software* (MCSW). O usuário de uma máquina pode ser humano ou máquina, esta última eventualmente atuando como intermediária na relação entre a MCSW e outro humano (usuário final). (p.1)

Sobre o desenvolvedor de *software* e sua relação com cliente e usuário, Fernandes (2003) explica:

> Para poder adquirir o *software* satisfatório, o cliente precisa compreender: quais os problema e necessidades com as quais o usuário convive e qual a MPU. Definido o contexto da solução, usualmente chamado domínio da aplicação, o cliente planeja uma solução para o mesmo através da definição das propriedades de uma máquina necessária para satisfazer ao usuário. Como o cliente não constrói diretamente o *software*, o plano de solução é expresso através de uma definição de linguagem necessária ao usuário (LNU), com gramática (sintaxe) e lógica (semântica) bem definidas. A LNU, verbalizada pela MCSW, será capaz de se comunicar com o usuário, permitindo-lhe expressar tarefas a executar, e obter resultados adequados. Outro aspecto que o cliente considera é que o problema e as necessidades do usuário existem no mundo real, e, portanto, apresentam-se em um contexto de tempo, espaço e recursos limitados. Sendo assim, também faz parte de uma solução satisfatória de aquisição de *software*: a seleção de um desenvolvedor de *software* capaz de criar um plano de construção da MCSW, que seja carregável e interpretável por MPU; e a adoção de um conjunto de condições que permitam que o *software* esteja disponível para o usuário no momento em que este necessitar, e dentro de uma relação de custo-benefício satisfatória para o cliente. [...] O desenvolvedor de *software* é um agente coletivo, responsável por criar um plano de construção de máquina (*software*) que esteja dentro das condições estabelecidas pelo cliente. *Software* é, portanto, uma meta-máquina. (p.1)

Dentro do processo de desenvolvimento de *software*s encontram-se diferentes métodos*,* sendo que o *Scrum* é um *framework* (estrutura de suporte) utilizado para criação de produtos de alta complexidade. Utilizado desde 1990, já provou sua eficácia em gerenciamento de projetos. Trata-se de uma metodologia que fornece aos clientes um *software* com mais velocidade nas entregas e mais qualidade.

Um outro método frequentemente observado é o *Waterfall*, que segue uma linha de projeto linear, com um planejamento longo, visando a entrega do produto no fim do prazo. O que se observa na utilização desse método é que, devido à dinâmica do negócio e exigência do mercado, muitas vezes ao chegar o momento de entrega, o cenário já mudou e o produto final precisa ser alterado, gerando insatisfação do cliente. Capodieci (2014) relata que a tendência de mudança está relacionada à redução de custo e tempo dos projetos e à satisfação do cliente.

Em fevereiro de 2001, dezessete praticantes de *software* de mentalidade independente escreveram o que se chamou “Manifesto Agile” (Manifesto Ágil). Embora os participantes não concordassem plenamente, encontraram consenso em

torno de quatro valores principais. Sem desprezar elementos e ferramentas tradicionais de desenvolvimento de *software*, o grupo estabeleceu uma escala de valores, na qual a flexibilidade e a colaboração são mais relevantes do que a rigidez de processos e planejamento clássicos. O Manifesto Ágil apresenta o que se supõe ser uma forma eficaz e com bons resultados para construção de *software*. Dois pontos que destacamos são: 1. a aprendizagem é uma parte importante do desenvolvimento de *software*; 2. equipes altamente produtivas são compostas por indivíduos que tomam a iniciativa e a responsabilidade de resolver os problemas à medida que surgem (ELSSAMADISY, 2007). Ruping (2003) aponta valores importantes, como: indivíduos e interações sobre processos e ferramentas; software de trabalho sobre documentação abrangente; colaboração do cliente em relação à negociação de contratos; resposta a mudanças ao se seguir um plano.

Cohn (2005) salienta a importância do planejamento, e indica que é necessário questionar-se: O que devemos construir e quando? Quão grande é? Quanto posso ter? E destaca: o planejamento deve ser descentralizado com a participação de toda a equipe, e o compromisso deve ser mantido por todos.

O Ágil tem como referências alguns métodos na gestão de projetos tecnológicos descritos em diversos livros, como:

- *XP extreme programming*; (BECK, 2000)
- *Lean Development* (POPPENDIECK; POPPENDIECK, 2003);
- *Scrum*

Neste trabalho será desenvolvido com mais detalhamento o método *Scrum*.

Já se trabalha com desenvolvimento de processos tecnológicos há mais de 70 anos, utilizando como referência os processos de engenharia, porém, nos últimos 25 anos tem-se estudado novas formas de trabalho em linha com o Manifesto Ágil. (COCKBURN, 2001)

Agilistas[1] como Sutherland e Schwaber (2007) trazem uma forma de gerir projetos com mudanças simples, com resultados positivos. Mas não promete que será facilmente dominada, será preciso trilhar os caminhos do *framework* Scrum.

Schwaber (2004) define o *framework* como desconcertante e paradoxal para o gerenciamento de projetos, isso porque ele tem uma estrutura em que mantém todo

[1] Agilistas: praticantes de desenvolvimento Ágil de software.

o projeto visível, apesar de sua simplicidade ser muitas vezes enganosa. O autor aponta que estão acostumados com os processos detalhados passo a passo, e reforça um item importante no *Scrum*, que é o MVP (Produto Mínimo Viável) que tem como objetivo entregas incrementais que geram valor para o cliente.

Alguns fatores que impulsionam o *Scrum* são: satisfação do cliente, flexibilidade frente a mudanças, colaboração, velocidade, confiança, eliminação de desperdícios, foco, melhora contínua, equipes (*teams*) auto organizadas.

Hron e Obwegeser (2018) explicam que no desenvolvimento de *software*, o Ágil tinha inicialmente sido fechado em equipes pequenas e especialistas. Com o crescimento na aceitação dos métodos ágeis, eles foram reconfigurados em várias formas, sempre partindo da forma ou conceito original.

Uma prova de que o *Scrum* tem cumprido com suas promessas é a expansão de sua utilização por parte de pequenas, médias e grandes empresas ao redor do mundo. Um míssil, quando apontado para direção correta, o *Scrum* faz com que organizações inteiras o utilizem fazendo uso de suas vantagens. Desenvolvedores antes preocupados com a formação de equipes que pudessem proporcionar um ambiente estimulante, agora encontram profissionais em todo mundo. (SUTHERLAND; SCHWABER, 2007)

De acordo com Adams (2006) as diferenças clássicas entre *Scrum* e abordagem tradicional, são:

1. Diferença no gerenciamento:
    - → *Scrum* é baseado em funções e objetivo - foca no Product owner (PO), o que os usuários avaliam como essencial para o sucesso. Apresenta um gerenciamento descentralizado.
    - → A abordagem tradicional é baseada em tarefa e custo - foca em provar ao cliente que os recursos estão sendo consumidos conforme o plano inicial. Apresenta um gerenciamento centralizado.
2. Público de alcance inicial:
    - → No *Scrum* observa-se um suporte de baixo para cima de usuários, clientes e desenvolvedores para utilização de metodologias Ágil;
    - → Na abordagem tradicional observa-se uma pressão de acionistas e investidores para que se mantenha o uso dos métodos tradicionais.

Na metodologia Waterfall a construção da tecnologia é feita em fases e segue uma sequência na qual o início da fase seguinte depende da finalização da fase anterior. Esse método é amplamente utilizado nas empresas com premissas de origens do desenvolvimento industrial: linearidade, determinismo, especialização, foco na execução e previsibilidade. Para qualquer empresa é um desafio trocar esse método, mas existem grandes empresas fazendo essas mudanças. (KORHONEN, 2009)

Waterfall segue o ciclo de vida que inclui especificações, desenvolvimento, teste e implementação. Além de ter-se tornado tradicional nas empresas, seus controles trazem uma certa tranquilidade para as organizações, ponto crucial com o qual devemos ter cautela na mudança de método. Uma falha nesse aspecto pode significar o fracasso da implantação do *Scrum*. O que é necessário alterar é a baixa comunicação entre as etapas e as decisões centralizadas do Waterfall. Já a proposta do *Scrum* é que o aprendizado aconteça durante o projeto e que suas decisões/ informações sejam compartilhadas entre as áreas/ pessoas. Além disso, pode-se mudar controles já existentes, aos quais se estavam acostumados, e eliminar atividades de coordenação entre o controlador e controlado, tornando o processo cada vez mais informal, mais objetivo e com interações diárias com a equipe. (MAHADEVAN, 2015)

Algo exagerado no Waterfall normalmente é a documentação. Elssamadisy (2007) demonstra a relação entre quantidade e utilidade, indicando que quanto mais dados na documentação, menos ela é utilizada, uma vez que identificar informações importantes torna-se complicado, em resumo, a qualidade é mais importante do que a quantidade.

Neste trabalho também é reforçada a relação entre *Scrum* e Kanban[2], possibilitando melhor visibilidade do andamento do projeto, das atividades em andamento, atividades encerradas e do tempo de espera. São ferramentas eficazes que fornecem dados de restrições e diretrizes. O Kanban em seu painel tem

[2] O kanban, nasceu inspirado no sistema de organização e funcionamento dos supermercados americanos, ao recolocar mercadorias nas prateleiras a partir do momento em que elas eram vendidas. O termo significa “tabuleiro”, e o sistema propõe a utilização de cartões (os famosos post-its ou até hoje em dia cards em uma visualização online) em um quadro para indicar e acompanhar, de maneira visual, prática e utilizando poucos recursos, o andamento dos fluxos de produção nas empresas.

basicamente as informações *To do, Ongoing* e *Done* (Por fazer, Em andamento e Feito). (KNIBERG, 2009)

O Kanban está totalmente em linha com Ágil, principalmente porque sua ideia também é a de criar somente as atividades necessárias, ou seja, definimos somente o que formos codificar, testar ou implantar. (LADAS, 2008)

Nas pesquisas realizadas sobre este tema, foi identificada uma boa prática para incluir a metodologia *Scrum* nos primeiros ciclos de trabalho: a referência de uma pessoa que possa colaborar no processo inicial indicando a técnica, oferecendo conselhos e dicas práticas sobre a metodologia, e a identificação e utilização do maior potencial das equipes. Essa referência, conhecida como o “Agile coach” (treinador Ágil) será de alguém que busque compreender a situação de cada equipe e que traga as respostas de que a equipeprecisa sobre o sistema Ágil. (DAVIES; SEDLEY, 2009)

Na avaliação dos resultados da pesquisa realizada com pessoas que já trabalharam nessas metodologias, são trazidas informações importantes quanto aos principais benefícios e barreiras do *Scrum* e *Waterfall*, além de comentários apresentados por funcionários de uma das três maiores Instituições Financeiras privadas do Brasil, esclarecendo seus principais resultados e decepções.

Entendendo as necessidades das empresas e a grande exigência do mercado, este estudo está estruturado de modo a entender os dois métodos de trabalho e apresentar os principais benefícios do *Scrum* em comparação ao Waterfall. São apresentados, ainda, os principais passos de como implantar o Ágil em uma empresa.

2 GESTÃO DE PROJETOS

A gestão de projetos das empresas é um dos pontos estratégicos para o aquecimento do ritmo das mudanças e inovações, porém o índice de sucesso nas implantações tem sido baixo. Essa constatação traz a necessidade de se realizar alguns questionamentos sobre as abordagens, ferramentas e métodos na gestão de projetos. Existe uma literatura que direciona o leitor/pesquisador para a utilização de técnicas e ferramentas com alto planejamento, além de controles com objetivo de fundamentar e padronizar a gestão de projeto, levando a formas de acompanhamento de projetos tradicionais. Contudo, encontram-se em discussão métodos mais flexíveis e fácies de serem adaptados. (MARQUES JUNIOR; PLONSKI, 2011)

Segundo Cleland (1994), nenhuma empresa pode fugir das mudanças que a tecnologia poderia trazer em alto ritmo. Com alterações nas tecnologias de produtos, serviços e no fluxo de informações, ressaltam prazos cada vez menores e a informação em contínua melhoria e quantidade.

Sabendo-se da importância dos projetos nas empresas e do alto percentual de casos em que as expectativas não são alcançadas, torna-se cada vez mais difícil confiar no cumprimento de prazo, custo e escopo que atendam o negócio e respondam à motivação do cliente em sua plenitude. Marques Junior e Plonski (2011) apresentam como resultado de pesquisas um total de apenas 28% de projetos tecnológicos que obtiveram êxito, e 103% de projetos de construção com atrasos médios. Neste momento, percebe-se a necessidade de se ser mais assertivo em um mercado competitivo. Torna-se imperativa a revisão da literatura focada no planejamento inicial e controle de forma, que limitam as ações nos projetos. Sabe-se que para o sucesso dessa abordagem é preciso que o ambiente esteja estável e controlável, porém, essa não é a realidade no mercado atual, uma vez que esse se caracteriza pela oscilação e eventualidade. (MARQUES JUNIOR; PLONSKI, 2011)

Na gestão de projetos tecnológicos de desenvolvimento de software existe a abordagem adaptativa mostrando-se uma opção, que busca adaptar-se ao ambiente, tarefas e objetivos. Neste momento é importante saber quais conceitos são mais importantes para conduzir um projeto no mercado atual.

Principalmente pela alta demanda de inovações, um método que não proporcione um ambiente adaptativo tende a inibir as iniciativas. Para Cleland (1994),

os setores de projetos estão totalmente ligados às mudanças das empresas, item esse fundamental para a sobrevivência e o crescimento da organização.

Ainda assim, é importante reconhecer itens básicos no acompanhamento de um projeto, como: tempo, custo, qualidade e pessoas.

O custo é um dos itens principais para aceitação do projeto. Trata-se de um ponto chave para o sucesso do projeto e andamento do negócio, tendo-se sempre que estar atento aos cuidados da provisão controlada. É necessário definir bem os custos no início do projeto, na fase de planejamento; ter os objetivos claros e definir bem os critérios de aceitação e os riscos. Levantar e analisar os custos de um projeto está acima de lançar dados na contabilidade (BASAK, 2006).

Para Coutinho e Oliveira (2017) os custos podem ser relacionados em diretos e indiretos. Os diretos estão atrelados às pessoas, materiais, bens e serviços aplicados no projeto. Já os indiretos são: privilégios adicionais; despesas gerais, pessoas que não estão atreladas diretamente ao projeto; aluguel, serviços de telefonia; administração e outros. Além dos custos previsíveis é importante também prever as mudanças de cenários, e um dos maiores motivos de mudanças está relacionado aos diversos *stakeholders* (as partes interessadas), de modo que no cenário atual é importante a gestão de mudanças e riscos de um projeto.

Os autores alertam para necessidade de atenção aos erros no mapeamento dos pedidos enviados pelos clientes e dos riscos que impactam diretamente em alterações no projeto, tanto no tempo como no custo, levando à possibilidade de um produto com qualidade inferior para o cliente. É de grande importância que pessoas que tenham o domínio dos custos acompanhar o projeto desde o seu início, evitando equívocos e contribuindo para uma visão assertiva do futuro.

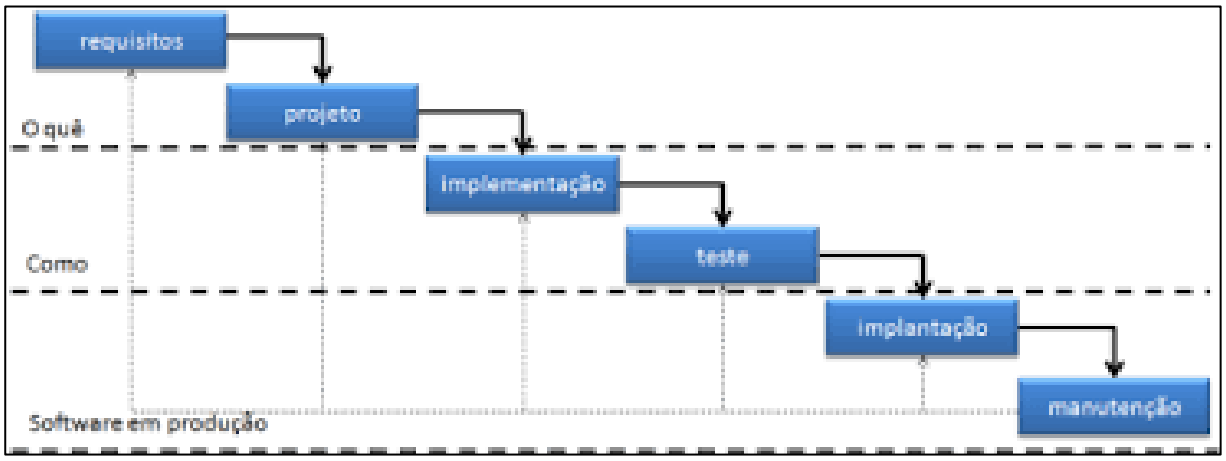
Os custos podem, ainda, se definidos em duas formas: custo contábil – em que são tratadas as normas legais e técnicas e custos gerenciais – que estão relacionados com o desenvolvimento da gestão financeira, tomada decisão e resultados do projeto. O lucro será maior quando trabalhado o custo mínimo. (POMPERMAYER; LIMA, 2003)

2.1 Métodos tradicionais

A forma tradicional de desenvolvimento de *software* é com o uso do método *Waterfall*, ou cascata, e este está presente em empresas grandes, médias e pequenas. O desenvolvimento de um projeto ou *software* se dá em fases, ou seja, segue uma linha em que realizamos o requerimento, o projeto, a implantação, a verificação, a manutenção e a entrega ao cliente. Existem algumas variantes na forma de se trabalhar com esse método, porém, normalmente se inicia com uma definição de requisitos bem detalhada, na qual a entrega final é pensada, projetada e documentada. Com as atividades fechadas para se iniciar o projeto, normalmente *Waterfall* é acompanhado por Microsoft Project. Assim, a equipe terá o levantamento do tempo que o projeto levará para ser executado, levando-se em conta as atividades por pessoa. Quando todos os itens envolvidos estiverem revisados, o projeto é enviado para aprovação, em seguida, a equipe inicia a construção. Cada pessoa finaliza sua etapa de trabalho e envia para o seguinte especialista na linha de produção, até que se chegue à garantia da qualidade do projeto. Avalia-se então, se o que está sendo entregue é o que realmente foi projetado. (SUTHERLAND; SCHWABER, 2007)

Nessa metodologia são valorizados processos, ferramentas, documentação, contratos, a sequência de um plano e organização. Seu ponto forte apoia-se na capacidade de raciocínio e planejamento dos desenvolvedores do *software*, anterior à execução. O ponto fraco, por outro lado, encontra-se exatamente nos mesmo desenvolvedores, uma vez que toda a organização exigida e a habilidade de seguir metodicamente o plano não são características facilmente encontradas entre elementos humanos – ainda mais especialmente entre aqueles envolvidos com a criatividade. Pela metodologia, espera-se que as ideias surjam apenas no início do plano, porém, sabe-se que as ótimas ideias florescem espontaneamente no decorrer do desenvolvimento projeto, de modo que a metodologia pode bloquear o processo de inovação. (SUTHERLAND; SCHWABER, 2007)

Quadro 1 - Fases da metodologia Waterfall



Fonte: Elaboração própria (2018).

O ciclo de vida do desenvolvimento de *software* compreende as seguintes etapas: requisitos, projeto, implementação, teste, implantação e manutenção, dando resposta às perguntas base – “o quê” e “como”. (QUADRO 1). Nesse ciclo, é necessária atenção especial à comunicação e ao *feedback*, ou devolutiva, entre as etapas. Trata-se de um ciclo baseado em um modelo tradicional de desenvolvimento de *software* que precisa de um design inicial (BOEHM, 1988).

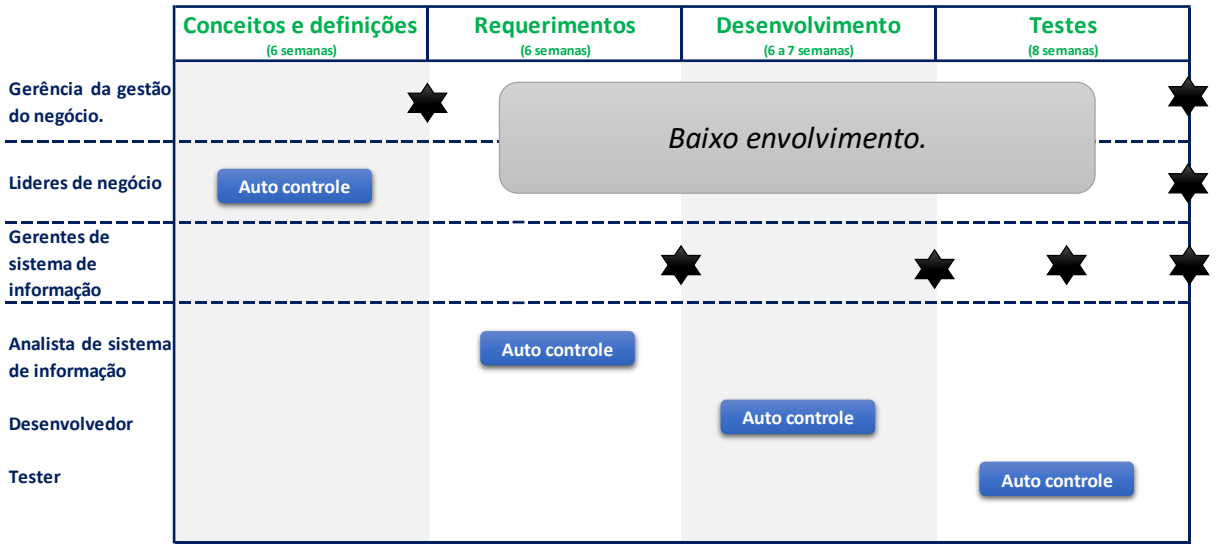
Essa abordagem dá especial atenção ao modo como o documento é escrito e à ênfase dada aos pontos chave, uma vez que o material adequadamente documentado poderá contar com maior credibilidade e, melhor aceitação. Além disso, tem-se mais facilidade para verificar se e quando uma determinada etapa foi concluída. Observa-se, contudo, que nesse tipo de abordagem os documentos são extensos e com frequência não são lidos na íntegra. Outro aspecto a ser avaliado é a dificuldade que pessoas têm em planejar e programar o futuro. A falta de clareza de ideias e incapacidade de previsão do que pode acontecer (os imprevistos), como por exemplo, as mudanças de curso no governo, na empresa e mesmo no desenvolvimento tecnológico – que podem ocorrer -, podem ter um impacto sobre o custo final, elevando-o significativamente. (SUTHERLAND; SCHWABER, 2007)

Waterfall esteve em evidência nos anos 70 na indústrias aeroespaciais e manufaturas altamente estruturadas. Para esse segmento, as mudanças ocorridas após a fase de implementação são de alto custo. Consolidando-se no segmento industrial, esse método tornou-se predominante também para o desenvolvimento de

software em grandes organizações. Nesta abordagem, observam-se outras características, como: a responsabilidade normalmente fica para a função de Sistemas de Informação; a interação entre todos os envolvidos tem a duração de algumas semanas; o envolvimento do cliente e o *feedback* ocorrem no momento final da entrega. (MAHADEVAN, 2015)

O controle dentro do método *Waterfall* é uma peça chave usada nesse método para a visualização dos percentuais de avanço. O controlador define os controles junto com as etapas e procedimentos necessários. O controle exige metas e não métodos (de como se atingiu determinado status ou evolução, por exemplo). é importante frisar que os controles do Waterfall não devem ser utilizados no Scrum, uma vez que faz referências a planos e processos concretos. Trata-se de uma abordagem focada na busca de requisitos e orçamentos para a construção do *software*. No final do desenvolvimento, o produto é entregue conforme a definição inicial do projeto. (MAHADEVAN, 2015). Trata-se de um método tradicional cujo padrão é recorrente e que faz uso de fotos/status de cada etapa.

Quadro 2 - Controles e planos do método Waterfall



Fonte: Adaptado de MAHADEVAN, L. Running on Hybrid: Control Changes when Introducing an Agile Methodology in a Traditional “Waterfall” System Development Environment. CAIS, 2015, p. 85.

No Quadro 2 é possível observar alguns aspectos do controle (MAHADEVAN (2015).

- Pontos de controle: o primeiro controle é o da gerência da equipe de negócio, - busca garantir que os requisitos cheguem às mãos da equipe de sistema de informação. Após essa etapa, a gerência novamente será envolvida na finalização dos os testes e na sinalização do *GO/ NO GO* do projeto.
- Autocontrole: o líder de negócio depende de autocontrole para entregar os requisitos do negócio. Isso deveria ajudar satisfatoriamente para lidar de forma razoável com suas responsabilidades do projeto. Porém, existem aspectos que exigem atenção, como algum eventual desvio no trabalho sendo realizado e as etapas que envolvem os analistas, desenvolvedores e *tester*.
- Envolvimento: a equipe de negócio tem mínimo envolvimento nas fases de requerimentos, desenvolvimento e testes. Essa ausência de participação total pode resultar em comunicação insuficiente ou ineficiente e pouca visibilidade do andamento do projeto.

No Waterfall, a flexibilidade de mudanças no meio do caminho, a visualização do *software* em funcionamento durante o processo de construção e a interação com o cliente são relativamente baixos.

Adzic (2009) faz uso da brincadeira do “telefone sem fio” para ilustrar como a falha na comunicação pode resultar em prejuízo, gerando mais custos, insatisfação e desconfiança. No telefone sem fio, uma criança sussurra uma frase no ouvido da próxima criança na fila, em seguida, essa passa para a próxima, e assim sucessivamente. No final, a última criança da linha dirá em voz alta o que ouviu da penúltima da linha. Na maioria das vezes, o que se ouve nada tem a ver com a frase original. Enquanto brincadeira de crianças, é algo divertido, porém, quando se trata de resolver problemas reais, a comunicação clara e objetiva é essencial.

Empresas de grande porte consideram necessário trabalhar com diferentes métodos em paralelo no desenvolvimento de *software*, *Waterfall* e *Scrum* são dois métodos que convivem simultaneamente (HARRIS, 2009). Em abordagem híbrida existe uma especificação detalhada no início, as avaliações de entregas ficam

somente no final de uma *Sprint*[3], o controle é divido em mecanismos mais estruturados e alguns emergentes. Em um estudo de caso, foram obtidos resultados positivos em uma equipe que seguia os princípios do Ágil, porém, não os adotaram puramente. MAHADEVAN, 2015)

2.2 Agile/Ágil

2.3 Manifesto Ágil

O Manifesto Ágil define valores e princípios que descrevem diversos processos ágeis. De 11 a 13 de fevereiro de 2001, um grupo de dezessete representantes de desenvolvimento de *software* se reuniu em Aspem (CO, EUA), buscando suprir a necessidade de uma alternativa aos processos de desenvolvimento de *software* pesados e orientados por documentação. Autodenominados "The Agile Alliance" (a Aliança dos Agilistas), o grupo de pensadores independentes, e às vezes rivais, chegou a um acordo, apresentado o se chamou o "Manifesto for Agile *Software* Development", ou Manifesto Agile. (HIGHSMITH, 2001). O termo "Agile" faz referência de forma genérica aos métodos ágeis. No Brasil, o termo foi traduzido literalmente, e (Manifesto) Ágil tornou-se de uso comum na literatura e no cotidiano do profissionais do ramo.

As principais metodologias davam destaque a: trabalho em equipe técnica e *product owners* (donos do negócio); comunicação "*face-to-face*"; entregáveis com qualidade em produção que geram valor agregado, equipes auto organizáveis, produtos que possibilitam tratamentos contínuos e evolução dos pedidos.

De acordo com Tabaka (2006), o Manifesto Ágil traz uma formulação marcante, caracterizada por sua simplicidade e também pelas alterações de prioridades, o que possibilita trabalhos que valorizam mais indivíduos e interações, *software* em funcionamento, colaboração com o cliente e resposta a mudanças. Essas características diferem das metodologias tradicionais que valorizam processos e

[3] *Scrum* é uma metodologia Ágil para gestão e planejamento de projetos de software. No *Scrum*, os projetos são divididos em ciclos (tipicamente mensais) chamados de *Sprints*. O Sprint representa um Time Box [caixa de tempo] dentro do qual um conjunto de atividades deve ser executado. Metodologias ágeis de desenvolvimento de software são iterativas, ou seja, o trabalho é dividido em iterações que, no caso do *Scrum*, são chamadas de *Sprints*. (https://www.desenvolvimentoagil.com.br/*scrum*/)

ferramentas, documentação abrangente, negociação de contratos e o seguimento de um plano (QUADRO 3).

Quadro 3 - Valores da balança para o Manifesto Ágil

Fonte: Elaborado pelo autor 2018

Fonte: Elaboração própria (2018).

Em linha com seu conjunto de regras simples pode-se observar claramente, no método ágil, uma declaração para a comunicação e a colaboração (QUADRO 3). Quando se fala de indivíduos e ferramentas e processos, a interação torna mais eficazes os processos tecnológicos, sendo que uma parte considerável do trabalho de um desenvolvedor está atrelado a negociações. Algumas das competências que facilitam o a interação são: o respeito à opinião de terceiros, envio de respostas/ status sem ataque a terceiros, a ausência de ações e palavras que desmotivem a equipe, como a criação de situações que indiquem um futuro duvidoso, e a capacidade de se chegar em uma conclusão em grupo, sem que se tente estabelecer verdades absolutas. (TABAKA, 2006)

São também fundamentais a colaboração durante todo o contrato e as negociações. A equipe de projeto e o cliente precisam definir um relacionamento desde o início do projeto. É importante que o cliente e a equipe de desenvolvimento saibam trabalhar com a emissão e recepção de informações sem receio de retaliações. Com isso, cria-se um ambiente em que se pode negociar, discutir,

debater, convergir e, consequentemente, chegar a uma melhor solução. Com o poder da colaboração, o cliente torna-se parte da equipe e a produtividade aumenta na abordagem de desenvolvimento de *software*, e acaba se tornando adaptativa. (TABAKA, 2006)

O Ágil tem as práticas mais bem-sucedidas no desenvolvimento de *software*. Refletindo sobre essas elas chega-se a denominadores comuns: ciclos para respostas a mudanças, rápida aprendizagem a partir de observação de erros, e colocação na prática dessa aprendizagem. (ELSSAMADISY, 2007) Lembrando que *frameworks* ou desenvolvimento Ágil normalmente têm uma adaptação a ser realizada, principalmente devido às circunstâncias fornecidas pelo ambiente. (HRON; OBWEGESER, 2018)

Assim, tem-se que duas questões são fundamentais para se seguir com o Ágil: primeiro, o aprendizado é uma parte importante e segundo, as equipes com alta produtividade são aquelas formadas por pessoas com iniciativas e responsabilidade para resolver os problemas que surgem. (ELSSAMADISY, 2007)

A maioria das pessoas que iniciam no Ágil acabam se preocupando mais com processo e ferramentas, porém, deveriam atentar-se mais aos indivíduos e interações.

Quando se pode dizer que Ágil teve sucesso? Segundo Elssamadisy (2007), deve-se voltar aos objetivos e verificar os resultados obtidos, como a redução nos erros ou falhas, o tempo de entrega comparado com o mercado, a melhora na produtividade, a apresentação do *software* com maior valor agregado para o negócio. Essas seriam medidas básicas para avaliar se a implantação está sendo satisfatória.

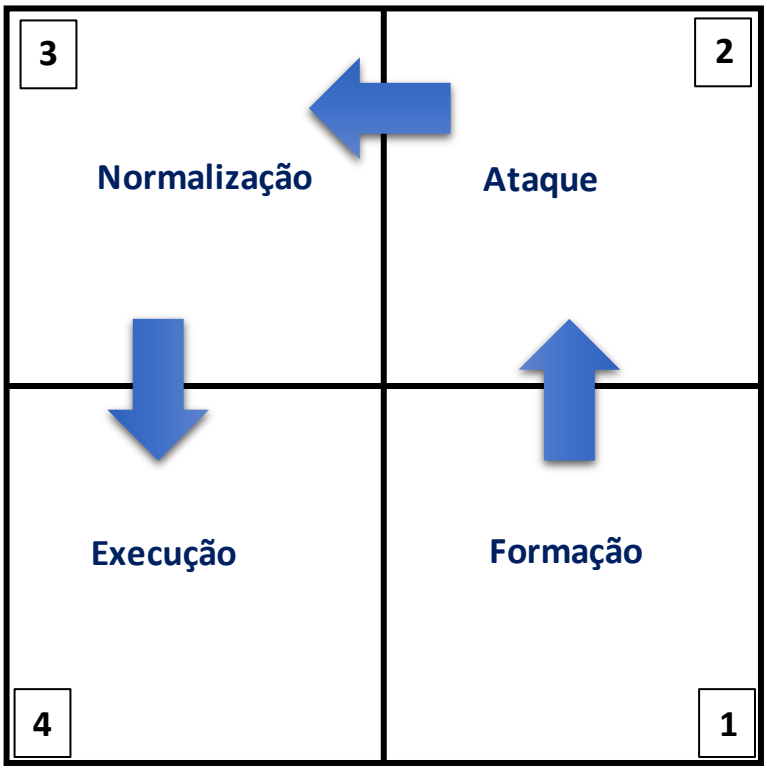
O valor de negócio é um pilar importante para o cliente, por isso, devemos ter claras suas expectativas. Elssamadisy (2007) abordou itens importantes na geração de valor para o negócio:

- A redução de tempo de mercado: entrega do *software*, ainda que não completamente finalizado – mas em frações, traz um valor significativo para o cliente, devido à possibilidade de esse poder começar a utilizá-lo. Isso ocorre principalmente nas entregas de sub funcionalidades, que permitem ao cliente viabilizar uso e ganho.
- Aumento de qualidade: relacionado a questões de minimização de defeitos, usabilidade e escalabilidade. Práticas que ajudem os clientes com uma resposta rápida às mudanças, como por exemplo, uma nova

legislação em que é necessário modificar o *software* para o seu cumprimento. O Ágil responde diretamente a mudanças, trazendo qualidade em uma alteração solicitada e a possibilidade de diversos processos de testes.

- Maior visibilidade: o Ágil entrega uma visibilidade para o cliente que pode dirigir o projeto, gerenciar riscos e suas expectativas. Essa visibilidade evita que o cliente tenha de enfrentar surpresas desagradáveis, como entregas de funcionalidade não esperadas, o que gera falta de confiança, ou seja, práticas que aumentem a visibilidade do desenvolvimento do *software* aumentam o ciclo de confiança e a relação com o cliente.
- Redução de custos: os métodos Ágil são mais rápidos, mais baratos e melhores. Sua rapidez supera o tempo de mercado, oferecendo melhor qualidade e valor de mercado. Para que seja mais barato, tem-se que trabalhar a plataforma de custos do desenvolvimento de *software*, com atenção especial a alguns itens, como: horas/ homem e a manutenção do sistema.

Quadro 4 - Fases de maturidade de uma equipe



Fonte: Adaptado de KANER, S. Facilitator's Guide to Participatory Decision-Making. San Francisco: Jossey-Bass, 2007. p. 30.

Kaner (2007) propõem que todas equipes passam por uma série de fases em sua criação (QUADRO 4)

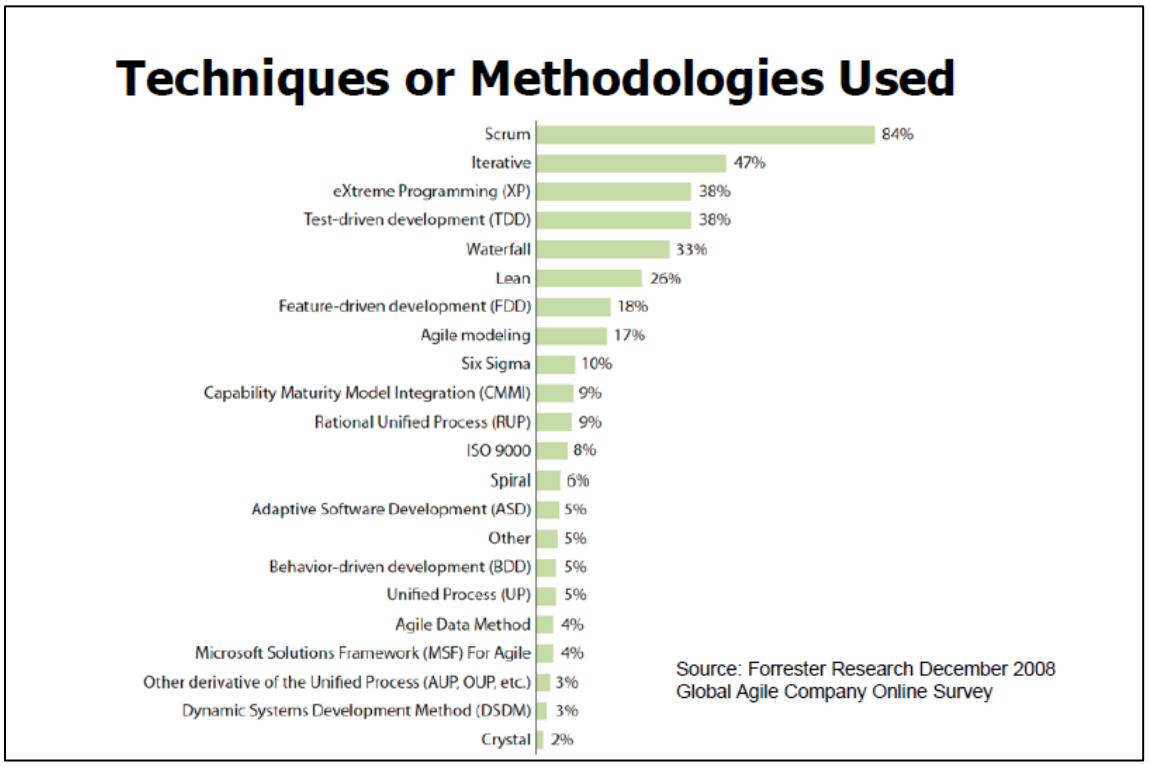
1. Formação: nessa fase a equipe está apenas aprendendo como todos trabalham e avaliando quem vai assumir cada papel. Neste momento, é preciso um líder facilitador para orientá-los e manter a equipe em curso.
2. Ataque: dá-se início a disputas por poder e controle, podendo resultar em desentendimentos, desconfiança e a formação de pequenos grupos fechados. A presença de um líder para gestão de conflitos é essencial.
3. Normalização: as equipes nesta fase estão mais harmonizadas para entrar em acordos comuns e apresentam capacidade de seguir as decisões. A confiança amadurece entre membros da equipe e esses conseguem dividir tarefas, inclusive em pequenos grupos. As divergências são resolvidas de forma rápida e o líder atua cada vez menos, entrando apenas como facilitador.

4. Execução: a equipe agora pode focar no desempenho e resultados, já tendo uma cultura saudável e auto-gestão, trilhando seu caminho em busca da alta performance.

No Ágil não se pode considerar a equipe como apenas um grupo de trabalho. Para Tabaka (2006), uma equipe são poucas pessoas com habilidades que se completam em busca de um objetivo e se sente mutualmente responsáveis. O Ágil transforma grupos de indivíduos em equipes que promovem comunicação, simplicidade, coragem, *feedback* e respeito, entre outras características. O *Scrum* facilita a auto-organização e o trabalho compartilhado, visando experiências de um presente e futuro com sucesso.

O Ágil tem um reconhecimento mundial como a melhor forma de desenvolvimento de *software*. (SUTHERLAND; SCHWABER, 2007)

Figura 1 - Técnicas ou metodologias mais utilizadas



Fonte: Estudo das metodologias mais utilizadas no Ágil. Disponível em: http://jeffsutherland.org/scrum/PracticalRoadmapScrumMicrosoft6Aug2009.pdf. Acesso em: 12 jan. 2019.

No Ágil existem algumas metodologias utilizadas, e um estudo realizado pela *Global Agile Company Online Survey* demonstra o seu uso, tornando visível o destaque da metodologia *SCRUM*. (FIGURA 1)

Embora a metodologia *Scrum* tenha um grande destaque em seu uso, existem outras com grandes referências no mercado, como o *Lean Software Development* e o *XP Extreme Programming.*

2.3.1 Lean *Software* Development

Lean *Software* Development tem como origem o Sistema Toyota de Produção.

Ao analisar os níveis de produtividade da Toyota logo após a II Guerra Mundial, Taiichi Ohno percebeu que a diferença de desempenho entre a Toyota e os principais fabricantes norte-americanos de um fator dez não poderia ser resultado exclusivo de uma deficiência na força de trabalho japonesa e concluiu que o fator significativo era desperdício "*Muda*". Levando em conta as ideias de Kiichiro Toyoda (fundador da Toytoa Motor Corporation) sobre a produção *Just in Time*, e fazendo diversas experiências, gradualmente, ao longo dos anos de 1945 a meados da década de 1970, Ohno construiu um conjunto coerente de princípios e práticas que veio a ser conhecido como “Sistema Toyota de Produção”, um conceito que abordava aspectos "além da produção em grande escala”

Os três princípios básicos do Sistema Toyota de Produção são:

- Produza componentes a tempo para o seu uso (produção *'Just in Time'*)
- Construa qualidade em todas as partes do processo ("*Jidoka*")
- Crie um processo contínuo (o "fluxo de valor")

Tendo os resultados de Ohno como fundamentação do método *Lean*, tem-se claro que qualquer coisa que não gere valor para o cliente ou fazer algo que não é necessário naquele momento é desperdício. A primeira coisa a ser observada na implantação do *Lean Software Development* são os desperdícios. O segundo passo é identificar e eliminar as maiores fontes de desperdício. Em seguida, deve-se eliminar as outras fontes de desperdício. E, por fim, é necessário refazer esse processo. A cada vez que o processo é feito, observa-se que algumas coisas não são tão imprescindíveis quanto se pensava. (POPPENDIECK; POPPENDIECK, 2003)

Essa atitude requer mudanças na cultura e hábitos da organização, e quando realizada percebe-se melhorias significativas e sustentáveis, além disto, é importante entender a diferença entre desenvolvimento de *software* e um processo de produção. Desenvolvimento de *software* seria a criação de uma receita enquanto a produção seria o seguir da receita. *Lean* reconhece as necessidades do desenvolvimento de *software* para que o produto alcance um equilíbrio da função, utilidade, confiança, economia e o meios para que os sistemas trabalhem juntos. (POPPENDIECK; POPPENDIECK, 2003)

Nesse método, existe a preocupação de se assegurar que as decisões gerem planos e execuções, visando buscar mudanças com tolerâncias. Existe, porém, uma ressalva, é melhor alterar uma decisão já tomada do que arriscar dar prosseguimento a um plano falho. Outros conceitos que o método busca são a entrega rápida visando a satisfação do cliente, a redução de riscos, flexibilidade empresarial e flexibilidade na tomada de decisão.

O *Lean* expande os fundamentos Ágil com várias ferramentas, entre elas, estão os temas de desperdício, custo, motivação, repetições, liderança, sincronização, perícia e testes. (POPPENDIECK; POPPENDIEK, 2003),

2.3.2 XP Extreme Programming.

Segundo Beck (2000), o *XP* Extreme Programming é uma maneira leve, eficiente, de baixo risco e flexível para o desenvolvimento de *software*. Seus valores são comunicação, *feedback* imediato, simplicidade, mudança incremental[4] e trabalho de qualidade.

O XP faz duas promessas:

- Para os programadores: garante que vão trabalhar em itens importantes e não terão que atuar sozinhos. Além disso, poderão tomar decisões para as quais estão qualificados, sem a burocracia existente em metodologias tradicionais.

[4] Como o próprio nome sugere, a mudança incremental está relacionada a acrescentar, somar ou agregar algo, normalmente alguma melhoria nos processos ou na estrutura organizacional. Um exemplo disso é a implementação de novos sistemas, controle de ponto, implantação de novas rotinas, novos projetos, entre outras coisas. (MARQUES, 2017)

- Para os clientes: promete gerar valor agregado em semanas e de forma continua, além de reconhecer a necessidade de mudanças em tempo de projeto, possibilitando alterações sem alterações exageradas nos custos. (BECK, 2000)

XP é uma considerada uma disciplina de desenvolvimento de *software* devido à obrigatoriedade de alguns itens, como por exemplo, a escrita clara dos testes, para que todos possam entendê-los. No entanto, as ideias do XP não são novas e a maioria foi comprovado ao longo do tempo, até por isto, tem um sentido conservador. A grande inovação do XP está na busca de três itens: a colocação dessas práticas em um único local; a garantia de que as práticas se apoiem em seu limite; e garantia máxima de que estão sendo praticadas.

Projetos de desenvolvimento de *software* aprsentam diversos riscos e o XP atua da seguinte forma (BECK, 2000):

- Correções ou Mudanças: o XP pede ciclos curto de no máximo quatro semanas, e nesse período devem acontecer todas interações necessárias e deve-se manter prioridades de implantação dos recursos para resolver o problema ou avaliar uma alteração de negócio.
- Cancelamento do projeto: o XP solicita que o cliente passe a definição de pequenas porções que gerem valor e, com isto, elimina-se a possibilidade de cancelamento total do projeto em produção.
- Sistema com problema: o XP exige que um conjunto de testes seja executado várias vezes em cada iteração.
- Falta de recurso: devido ao XP insistir em executar somente as atividades com maior prioridade, tende-se a realizar atividades que realmente são necessárias ao projeto, elinando assim, custos adicionais e esforço desnecessário. Isso apoia diretamente os projetos que se iniciam com orçamento limitado.
- *Turnover*: o XP solicita que os programadores estimem a quantidade de trabalho que vão realizar, e que aceitem a quantidade de esforço imposta. Com o *feedback*, vão lhes passando informações para que possam melhorar suas estimativas e sempre com respeito às contribuições do time. (BECK, 2000)

O *XP Extreme Programming* está relacionado à pergunta “Como você programaria se tivesse tempo suficiente?”. Alguns tipos possíveis de respostas seriam: “Fazendo diversos testes”, “Obtendo manutenção sistêmica, com algum aprendizado”, “Falaria com colegas”... Beck (2000), por sua vez, indica que não se pode esquecer de que se trata de negócio e que se deve estar atento ao que é suficiente é primordial. Além disto, existe uma flexibilidade nas alterações priorizadas e o cliente sempre será bem-vindo para alterar a funcionalidade, colocando outra em seu lugar. Contudo, esse procedimento exige um novo ciclo de alinhamento e pode desestabilizar o planejamento inicial.

O XP muda o paradigma, eliminando o medo da mudança, uma vez que nesse método o errar é feito com um baixo custo. Diferente do método tradicional em que mudanças tardias geram alto custo, o XP sugere que as mudanças devem ser constantes e que não se deve temê-las, principalmente se suas práticas estão sendo seguidas.

3 SCRUM

Os métodos ágeis nascem de uma crença em uma abordagem fundamentada na realidade das pessoas para realizar mais produtividade. Ao invés de investir tempo em documentações extensas, foca nas entregas rápidas e com valor agregado. Busca pequenas equipes multidisciplinares e não grandes estruturas organizacionais. Quando se utiliza o Ágil tem-se um reconhecimento do que é feito e entregue de início. Um dos métodos mais rápidos e de grande reconhecimento é o *Scrum*, e várias equipes relatam mudanças importantes ou transformações completas, tanto na produtividade como na moral da equipe. *Scrum* é simples ainda que de grande eficácia. (SUTHERLAND; SCHWABER, 2007)

Scrum é um *framework* utilizado para criação de produtos de alta complexidade. Para a implantação desse modelo é necessário entender a definição de papéis, eventos, artefatos e as regras que mantêm o *Scrum*. Este é um método leve e simples de entender, porém é complicado de se dominar. (SCHWABER, 2009)

Utilizado desde 1990 deixa clara sua eficácia em gerenciamento de projetos, como nos exemplos apresentados por Sutherland (2014) em diversas instituições, como o FBI e grandes empresas tecnológicas e financeiras.

O *Scrum* é uma metodologia que fornece para os clientes *software* com mais velocidade nas entregas e mais qualidade. Já está sendo utilizado no desenvolvimento de *software* por diversas empresas de grande porte, como Yahoo e Google. Principalmente pela produtividade das equipes de desenvolvimento interna e terceirizadas, redução de custos, e aumento na lucratividade, como é o caso em projetos da Telecom e Google Adwoeds. (SUTHERLAND; SCHWABER, 2007)

O primeiro desenvolvimento de *software* utilizando *Scrum* ocorreu em 1993. Jeff Sutherland[5] foi o engenheiro chefe da equipe e definiu os principais papeis. Com isto contratou o primeiro Product Owner (PO) e *Scrum* Master (SM), que desenvolveram o primeiro *product backlog* (acúmulo de produtos incompletos / lista

[5] Jeff Sutherland é o inventor e co-criador do *Scrum*. Ele começou o primeiro *Scrum* na Easel Corporation em 1993 e trabalhou com Ken Schwaber para emergir o *Scrum* como um processo formal na OOPSLA '95. Juntos, eles ampliaram e aprimoraram o *Scrum* em muitas empresas de software e organizações de TI e ajudaram a escrever o Manifesto Ágil.
Jeff é CEO da *Scrum* Inc., presidente da *Scrum* Foundation e coach Agile da OpenView Venture Partners, que executa todas as suas operações internas com a *Scrum*, além de mais de 30 empresas do portfólio.

de itens por fazer) e o *Sprint* backlog (lista de tarefas)[6]. Em 1995 Jeff Sutherland juntou-se a Ken Schwaber para formalizar o processo de desenvolvimento do *Scrum*. (SUTHERLAND; SCHWABER, 2007)

Scrum passou pelo processo de padrões de linguagem de design de programação (PLoP). Mike Beeble trabalhou intensamente para fazer do *Scrum* um padrão organizacional e demostrou que o método foi criado para os seguintes pontos: trazer energia; foco; clareza; transparência na implantação; velocidade; regularizar metas individuais e da empresa; criar valor agregado; melhorar a comunicação; o desenvolvimento individual e a qualidade de vida. (ELSSAMADISY, 2007)

Figura 2 - Origem do nome *Scrum*

Fonte: SCHWABER, K. Guia do *Scrum*. *Scrum* Alliance, 2009. p10.

Scrum foi inspirado em um termo do rugby (FIGURA 2) e

> [...] é usado como uma metáfora para refletir o grau de cooperação necessário para ter sucesso.

[6] Os itens do Sprint Backlog são extraídos do Product Backlog, pelo time, com base nas prioridades definidas pelo Product Owner e a percepção da equipe sobre o tempo que será necessário para completar as várias funcionalidades. Cabe à equipe determinar a quantidade de itens do Product Backlog que serão trazidos para o Sprint Backlog, já que é ela quem irá se comprometer a implementá-los.

> Um time de rugby pontua quando a bola avança pela linha do gol e é tocada no chão. A maneira como a bola se move é (principalmente) sendo carregada e sendo passada ou passada lateralmente / para trás. Se o portador da bola for atingido, ele deve soltar a bola e o jogo não para.
> Como segurar a bola deixa você muito vulnerável a ser enfrentado pelos garotos do time adversário, é imperativo manter a bola em movimento entre os companheiros do time. E isso significa que toda a equipe é necessária para pontuar. (HARPER, 2012)

Scrum tem origem inglesa *scrimmage* que significa qualquer luta em pequenas porções. (TAKEUCHI, 1986)

Em que o *scrum* no rugby e no desenvolvimento de *software*s coincidem? No jogo, os atacantes de ambas as equipes se amontoam juntos usando seu tamanho, força e capacidade de trabalhar juntos para pegar a bola, no desenvolvimento de *software*, espera-se essa união, força, motivação, coragem, com um objetivo comum – portanto, o grau de cooperação é o que ambos têm em comum.

O *scrum* visa o controle de processos empíricos, entregando um comportamento de integração, com maior previsibilidade e controle dos riscos.

Segundo Schwaber (2009) método é sustentado por três grandes pilares:

1. Transparência - deixa visíveis os temas que podem influenciar o resultado. Não são apenas transparentes, mas conhecidos por todos, disponíveis para quando há um problema a ser solucionado.
2. Inspeção - os vários aspectos devem sofrer inspeções com regularidade para manter as variações aceitáveis para o projeto, mas reforça também o cuidado com o excesso de inspeção.
3. Adaptação - se na inspeção for detectado um desvio nos resultados esperados, deverá haver ajuste no processo ou material da melhor maneira possível.

No *Scrum* existem as Reuniões (*cerimonies*) como pontos de inspeção e adaptação; na Daily[7] é possível inspecionar se a equipe está em direção à meta e realizar as adaptações para o dia seguinte. Além dessa, há outras três reuniões:

[7] Daily *Scrum* Meetings - No *Scrum*, em cada dia de um Sprint, a equipe realiza uma reunião chamada "*scrum* diário". As reuniões são normalmente realizadas no mesmo local e na mesma hora todos os dias. Idealmente, uma reunião diária de *scrum* é realizada pela manhã, pois ajuda a definir o contexto para o trabalho daquele dia. Essas reuniões *scrum* são estritamente limitadas em 15 minutos. Isso mantém a discussão rápida, mas relevante.

Planning, *Review* e *Retrospective*, que apoiam a inspeções e adaptações necessárias na *Sprint*[8]. (SCHWABER, 2009)

Scrum é um processo Ágil que permite manter o foco na entrega de maior valor ao negócio e no menor tempo possível, permitindo a rápida e contínua inspeção do *software* em produção. No entanto, a agilidade de implantação do *Scrum* também está ligada ao grau de modificações exigido pela empresa, à efetividade da liderança e à urgência em se ter resultados no dia a dia.

Buscar um processo Ágil "para muitas pessoas é uma reação à burocracia das metodologias fundamentadas. É uma tentativa de um compromisso entre nenhum processo e processo demais, porém é apenas o suficiente para atingir uma meta razoável". (BECK, 2000)

As necessidades do negócio é que determinam as prioridades do desenvolvimento de um sistema. As equipes se auto organizam para definir a melhor maneira de entregar as funcionalidades de maior prioridade. Entre uma e cinco semanas todos podem ver o *software* real com qualidade em produção, decidindo se o mesmo deve ser liberado ou se deve continuar a ser aprimorado por mais um "*Sprint*". Desenvolver um *software* é uma tarefa difícil, pelos processos organizacionais e pela burocracia, que conduzem a impedimentos e desperdício. A equipe deve sempre estar atenta aos impedimentos para eliminá-los o mais rápido possível. (LEFFINGWELL; SMITS, 2005).

No processo Ágil é importante que seus líderes procurem um ambiente de autogerenciamento, visando a melhora contínua, aprendizado, cooperação e comunicação, evitando ou trabalhando rapidamente os impeditivos que afetam o desempenho do "Squad"[9]. É importante, ainda que se tenha a crença de que essa equipe auto organizada, auto gerenciada e multifuncional terá mais entregas e com qualidade. (2005).

Segundo Leffingwell e Smits (2005), para a implantação do *Scrum* é importante seguir alguns passos, sendo eles: identificar precursores e padrinhos, iniciar com

---

[8] Sprint é um período de tempo definido durante o qual o trabalho específico deve ser concluído e preparado para revisão.

[9] O modelo "Squad" (de esquadrão) é uma estrutura organizacional multifuncional de um conjunto de equipes, cada uma com a missão de resolver desafios específicos de produtos. Cada Squad é composto de colaboradores individuais de diferentes disciplinas e liderados por um colaborador individual.

pequenos projetos e pensar sobre os pontos positivos e fracos. Além disso, a implantação pode seguir fases:

- "Avaliação e Preparação" – fase em que devemos observar se a organização está apta para responder às demandas Ágil, como por exemplo, gestores abertos a mudanças e comunicação sobre a metodologia para empresa. Na preparação temos treinamentos específicos de Ágil, como por exemplo, introdução ao Ágil/*Scrum* e outros relacionados aos papeis de Product Owner e *Scrum* Master.
- "Projeto Piloto" – fase em que escolhe um projeto piloto para que se tenha essa experiência na empresa, tornando possível a visualização dos benefícios em um projeto tecnológico. Nesse piloto, o *Scrum* Master e a gerência devem estar focados nos impedimentos e devem criar ações imediatas ou inclui-los no Product Backlog.
- "Expansão Organizacional" do *Scrum* - essa terceira fase tem relação entre o piloto e a viabilidade de uma possível expansão de seus benefícios para uma parte da empresa. Tendo essa fase em curso é preciso realizar um programa de formação para todos papéis no *Scrum* (*Scrum* Master, Product Owner, Equipe de desenvolvedores e engenheiros de *software*).

  Outros pontos da expansão são:

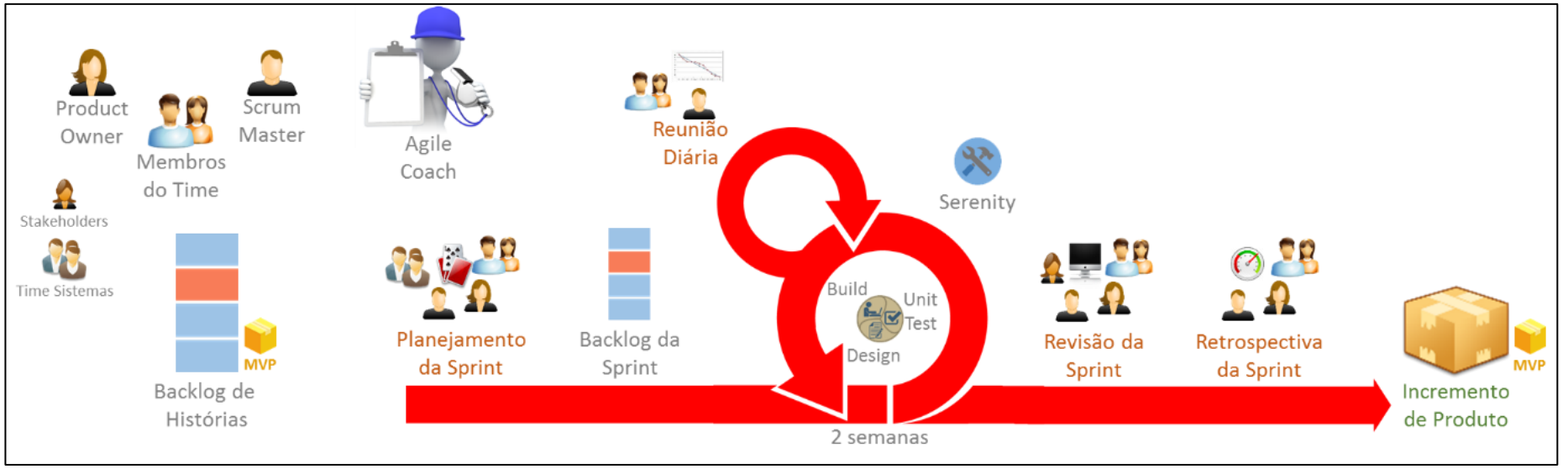
  - Criar painéis com os dados das tarefas (Tasks Board), seus *releases*, valor agregado e "KPI"[10] de acompanhamento.
  - Leitura de livros e artigos sobre o tema.
  - Ter *feedbacks* dos projetos pilotos demostrando os resultados para todos.
  - Spike[11] para os membros da equipe sobre assuntos de interesse comum entre as equipes e com duração média de 90 minutos.

[10] KPI é a sigla para o termo em inglês Key Performance Indicator, que significa Indicador-chave de Desempenho. Esse indicador é utilizado para medir o desempenho dos processos de uma empresa e, com essas informações, colaborar para que alcance seus objetivos.

[11] Um "spike" é um método de teste de produto originário da Extreme Programming que usa o programa mais simples possível para explorar soluções potenciais. Ele é usado para determinar quanto trabalho será necessário para trabalhar ou solucionar um problema de software. Normalmente, um 'teste de spike" envolve a coleta de informações adicionais ou testes. O termo é usado em abordagens ágeis de desenvolvimento de software, como *Scrum* ou *Extreme Programming.*

- “Ampliar” o uso do *Scrum* para grande parte dos projetos da empresa é a quarta fase, uma vez que já existe uma vivência com os pilotos da fase anterior, com equipes preparadas, conhecedoras do método, dos benefícios e das práticas para eliminar os empecilhos.
- “Medir, Avaliar e Ajustar” criando KPIs de controle na expansão do uso nos projetos e seu retorno é a quinta fase. Essas métricas devem estabelecer as principais entregas do *Sprint*, trazendo a visibilidade do produto final e suas funcionalidades que geram valor para o negócio ou cliente. Os indicadores podem ser divididos em KPIs de processo, onde temos um acompanhamento do Product Backlog e reuniões do *Scrum*, e também KPIs de projeto aos quais a organização está acostumada, como informações de incidentes, cobertura do *script* de teste, funcionalidades entregues, plano em longo e curto prazo, principais impactados, entre outros.
- “Consolidação” é a etapa final, em que se incluem todos os projetos da empresa, com *Scrum* implantado, as equipes iniciam uma evolução natural, com retorno para empresa e crescimento pessoal.

Figura 3 - Fluxo da metodologia *Scrum*



Fonte: Adaptada de SUTHERLAND, J; SCHWABER, K. The *Scrum* Papers: Nuts, Bolts, and Origins of an Agile Process. Washington: PatientKeeper, 2007.

Scrum tem uma estrutura de desenvolvimento de *software* realizada em ciclos, que tem o nome de *Sprint* e uma duração normal de 1 a 4 semanas (FIGURA 3). Uma vez definida sua data de início e fim de uma *Sprint*, essa não deve ser alterada, mesmo que não sejam entregues todos os requisitos. No início da *Sprint* são selecionados os itens priorizados no backlog e a equipe se compromete em entregá-los. Em seguida, todos os dias a equipe se reúne por 15 minutos para realizar a *Daily*. No fim da *Sprint*, a equipe apresenta o que foi concluído na "*Review"* (revisão) e tem um *feedback* do PO, e com este resultado vão para "Retrospectiva" identificar plano de ação nos pontos a desenvolver e multiplicar os positivos. (SUTHERLAND; SCHWABER, 2007)

No *Scrum* é necessária uma facilidade na tomada de decisão participativa. Kaner (2007) lista alguns requisitos para que isto possa acontecer, são eles: crença na sabedorias e criatividade da equipe; capacidade de ouvir aberta e ativamente; conhecer a dinâmica de trabalho em grupo; respeito às pessoas e seus pontos de vista; forte colaboração nas soluções de problemas interpessoais; flexibilidade nas resoluções de problemas; tomadas de decisões. Duas lições básicas - a primeira é que se as pessoas não participam da solução dos problemas ou não concordam com a solução apresentada, a execução será abaixo do esperado. A segunda lição é que o sucesso de uma empresa não está somente nos produtos e serviços, tecnologia ou valor de mercado, mas sim, no aproveitamento e concentração do capital intelectual de suas equipes.

Sabe-se que a condução de projetos e desenvolvimentos de *software* não são simples, e por isso, é preciso atenção. Até porque, como disse o jornalista americano H. L. Mencken (1880 - 1956) "Para todo problema complexo existe sempre uma solução simples, elegante e completamente errada". (APPELO, 2010)

Enfim, *Scrum* é uma estrutura simples de inspecionar e adaptar. Nele são necessárias algumas funções básicas, três papeis, quatro conjuntos de reuniões e três artefatos:

- Papéis: Product Owner, *Scrum* Master, Development Team;
- Reuniões: Planning, Review, Daily e Retrospective;
- Artefatos: Product Backlog, *Sprint* e Stories.

3.1 Papéis

3.1.1 Product Owner (PO)

O *product owner* (proprietário do produto) é a pessoa que tem os requisitos necessários e relacionados ao *Product Backlog*. O *product owner* é uma função de desenvolvimento de *scrum* desempenhada por uma pessoa que representa a comunidade comercial ou de usuários (os *stakeholders* ou partes interessadas) e é responsável por trabalhar com o grupo de usuários para determinar quais recursos estarão na versão do produto. Nesse papel é necessária total autonomia para decidir o que deverá ser entregue em cada *Sprint*, relacionado ao seu alto conhecimento das necessidades e funcionalidade do produto, assim como uma visão total da *Squad*. Em alguns casos o PO é o próprio cliente final ou em outros casos é o gerente do produto/ negócio em diversas empresas, o importante é ficar atentos a suas funções neste método. (SUTHERLAND; SCHWABER, 2007)

Se possível, é importante ter uma pessoa de negócio nesse papel. Segundo Sutherland e Schwaber (2007) o PO tem algumas responsabilidades:

- Construir, aperfeiçoar e manter o Backlog da equipe;
- Coordenar e gerenciar reuniões de planejamento de *Sprints*;
- Ser a maior fonte de informações sobre as prioridades do projeto;
- Indicar claramente os itens necessários do Product Backlog;
- Encomendar os itens do Product Backlog e ajudar a atingir as metas;
- Assegurar que o Product Backlog esteja visível, claro e transparente a todos;
- Otimizar o valor do trabalho entregue pela equipe de desenvolvimento;
- Assegurar a qualidade da entrega do produto.
- Ter claro o retorno sobre o investimento (ROI),
- Priorizar recursos conforme valor agregado;
- Ajustar as prioridades com no mínimo 30 dias;
- Dar o OK ou NOT OK para os resultados entregues.

O PO tem em sua meta o primeiro passo do *Scrum* que é articular a visão do produto, que acaba virando a lista de 'histórias'[12] *(stories)* do seu Product Backlog. (SUTHERLAND; SCHWABER, 2007) Ele é um tomador de decisões e faz uma relação direta com os clientes, e com isso, tem capacidade de trazer *feedbacks* diários sobre a construção realizada. Tarefas como a escrita das histórias (suas especificações) e testes de aceitação fazem parte do dia-a-dia do PO Sua localização também é importante para que ele esteja de fácil acesso quando preciso. (ADAMS, 2006)

3.1.2 *Scrum Master* (SM)

O *Scrum Master* (SM) atua como facilitador do *Daily Scrum* e é responsável por remover quaisquer obstáculos que sejam levantados pela equipe durante essas reuniões. O papel de *Scrum* Master é tipicamente exercido por um gerente de projeto ou um líder técnico, embora, em princípio possa ser qualquer pessoa do time. É a pessoa responsável pelo apoio a toda a equipe, garantindo que todos os passos do *Scrum* sejam seguidos. Sua atuação deve ser ativa quanto a qualquer impedimento que exista para a entrega do produto, assegurando o êxito da *Sprint*, conforme compromissos assumidos.

Segundo Sutherland e Schwaber (2007) entre as responsabilidades do S. M estão:

- Garantir que a equipe esteja produtiva e funcional;
- Viabilizar a colaboração entre os papéis e funções.
- Proteger a equipe em relação a situações externas;
- Garantir que o método *Scrum* esteja sendo realizado.

Todo a equipe deve ter um *Scrum Master* para conduzi-lo. (HRON; OBWEGESER, 2018)

Nas empresas têm sido aceitos projetos ineficientes e impeditivos por muito tempo, com o *Scrum* tem-se a possibilidade de resolver esses problemas, e é papel do *Scrum Master* atuar diretamente nesses casos junto com a equipe.

---

[12] Histórias (*Stories*) – São itens do Product Backlog que representam parte do produto a ser implementado. As histórias devem conter uma descrição detalhada do que deve ser implementado.

Os impedimentos sempre devem sofrer uma atuação imediata ou priorizada em relação ao *product backlog,* mas antes é importante que se saiba identificar esses impedimentos. Normalmente, os impedimentos estão atrelados ao processo de construção, atitudes das pessoas no uso do método e seu comprometimento, engenharia de *software*, retorno sobre o investimento e temas organizacionais. (LEFFINGWELL; SMITS, 2005)

Diversas equipes acabam não eliminando seus impedimentos, o que pode resultar na necessidade de dobrar o trabalho da sua *squad*.

Tabela 1 - Exemplos de priorização de impedimentos

		Notas			
Impedimentos	**Score de priorização**	**Retorno**	**Tempo**	**Custo**	**Complexidade**
Sistema legado com problemas	450	9	8	7	5
Testes não efetuados	400	8	8	7	5
Atividades entre squad não priorizadas	333	6	5	6	7
Orçamento não aprovado	333	8	8	8	8

Fonte: Elaboração própria (2018).

Na tabela 1 observa-se a priorização dos impedimentos levando em consideração os principais atributos:

- Retorno: nesse caso, deve-se informar uma pontuação de 0 a 9 de retorno que pode ser gerada com a retirada deste impeditivo, atrelando-o ao *product backlog* do PO
- Tempo: pontuar de 0 a 9 o tempo que será gasto para retirar esse impeditivo.
- Custo: pontuar de 0 a 9 o custo financeiro para retirar o impeditivo, lembrando que neste caso pode-se colocar horas de pessoas, custas contratuais, compra de novas ferramentas, entre outros.
- Complexidade: pontuar de 0 a 9 a complexidade do impeditivo, levando em consideração as barreiras a serem enfrentadas.

O atributo “Score de priorização” (ou Pontuação de priorização) é um cálculo baseado na pontuação do Retorno dividido pela soma do Tempo, Custo e Complexidade e multiplicado por mil. Isso possibilidade uma relação direta de ganho

que esta atividade pode trazer para a empresa versus o custo. A fórmula proposta leva o nome de seu autor, SANTOS. (FIGURA 4)

Figura 4. Fórmula Santos

Fonte: Elaboração própria (2018).

A fórmula Santos pode ser considerada para avaliação também do *product backlog* do PO

3.1.3 Dev.Team – (Equipe de desenvolvimento)

> A equipe de desenvolvimento é um subconjunto da equipe Ágil. Consiste em profissionais dedicados que podem desenvolver, testar e implantar uma história, um recurso ou um componente. O Dev Team normalmente inclui desenvolvedores de *software* e testadores, engenheiros e outros especialistas dedicados necessários para completar uma fatia vertical da funcionalidade. Para consistência com a definição do *Scrum*, a equipe de Desenvolvimento exclui o Product Owner e o *Scrum* Master. Estes fazem parte da equipe Agile maior. (© *Scaled Agile, Inc.)*

O Dev. Team tem o papel de detalhar e produzir todas as tarefas para entrega do produto de cada *Sprint*. Além de levantar qualquer impeditivo que aconteça, buscar autogerenciamento, transparência e apoiar outros desenvolvedores.

O Dev Team precisa sempre procurar o sucesso da equipe e os individuais, mas não pode esquecer que deve existir uma harmonia. Nas palavras de DeMarco (1995, p.36), "Um indivíduo só pode ter sucesso na medida em que o todo prospera. E o todo só pode prosperar, na medida em que todos se empenham."

Segundo Sutherland e Schwaber (2007), algumas das características do Dev. Team são:

- Deve ter no máximo 7 e no mínimo 2 membros;
- Deve avaliar o que é possível ser realizado na *Sprint* e especificar as entregas;

- Dentro das diretrizes do projeto, tem livre arbítrio para tomar decisões para alcançar as metas;
- Auto-organização de suas atividades;
- Deve apresentar o trabalho entregue para o PO

O estilo de programação em par traz diversos conceitos, princípios e resultados que o Dev Team deve buscar. Williams (2002) explica que a programação em par se constitui de duas pessoas trabalhando em uma mesma tarefa. Embora pareça fácil, um par de pessoas implica em duas personalidades diferentes, o que exige harmonia e cooperação mútua. Os programadores estão acostumados a trabalhar sozinhos. Alguns desenvolvedores tendem a ser tímidos e provavelmente não vão assumir que não sabem algo. Existe um estudo realizado na Universidade de Cape Town, África do Sul, que revela que 42% dos desenvolvedores tendem a trabalhar com os problemas sozinhos. (COCKBURN, 2001)

A programação pareada tem um dos parceiros como condutor, que está digitando, e outro como navegador, que observa o trabalho, procurando defeitos táticos e estratégicos. Nesse esquema de trabalho, se a parceria der certo e os integrantes se dispuserem a ceder, aprender e compartilhar, o resultado traz alguns benefícios, como:

1. Qualidade: a parceria viabiliza uma produção sem defeitos;
2. Tempos: produção de códigos em menor tempo e com mais qualidade, além de um trabalho engajado;
3. Moral: trabalham mais felizes, pois há apoio mútuo e parceria
4. Confiança e trabalho em equipe: companheiros tornam-se parceiros e amigos
5. Transferência de conhecimento: dissemina conhecimento para o grupo
6. Reforço da aprendizagem: aprendem observando como seus colegas realizam as atividades e também escutando seus parceiros.

3.2 Reuniões

Chamadas *cerimonies*, em inglês, as Reuniões distribuem-se em quatro tipos que compõem um conjunto, *Planning* (planejamento)*, daily (*reuniões diárias de 15

minutos*), retrospective* (retrospectiva) e *review* (revisão). Nelas é possível manter o controle e adaptação constantes, deixando em evidência o produto desenvolvido, o processo e os impedimentos.

3.2.1 Planning

Planning é a primeira reunião. Nela, e o Product Owner (PO) apresenta as histórias a serem realizadas e a equipe de desenvolvimento detalha todas as tarefas necessárias para entregar o produto solicitado na *Sprint*, é também quando são pontuadas as histórias com objetivo de avaliar o esforço, caso não tenha sido feita antes a pontuação. (HRON; OBWEGESER, 2018).

A Planning apresenta o seguinte formato:

- Timebox: 4 horas por 2 semanas de *Sprint* ou 8 horas por 4 semanas de *Sprint*
- Participantes:
  - *Scrum* Master: quem facilita a reunião,
  - P. O.: quem esclarece os detalhes dos itens do *backlog* do produto e seus respectivos critérios de aceitação;
  - Equipe de desenvolvimento: quem define o trabalho e o esforço necessário para cumprir o compromisso do *Sprint*.

Não se deve iniciar o Planning se os itens abaixo não estiverem fechados, conforme reforça Kniberg (2007)

- Product backlog preenchido e as histórias completas em termos de informações e requisitos;
- A indicação e presença do PO responsável pelo *product backlog*;
- Os itens que o P. O. identificar como necessidade para a próxima *Sprint* devem ser colocados como prioridade alta;
- É preciso que a equipe tenha claro que o valor de negócio é somente para ajudar na priorização das histórias
- É importante o P. O. conheça todas as histórias que coloca na *Planning*, apesar de ser possível que outras pessoas enviem histórias para que ele as priorize.

As reuniões de *Planning* ocorrem sempre antes do início de cada *Sprint*. Um objetivo de um *Sprint* deve ser representado por uma descrição curta, de uma ou duas frases, especificando o que a equipe planeja alcançar durante o *Sprint*. Esse trabalho de descrição deve ser feito de modo colaborativo por toda a equipe e pelo PO

Anteriormente à reunião de *Planning,* o PO deve identificar todos os itens com o maior valor e trabalhar para que estejam prontos. O Dev Team deve buscar compreender a funcionalidade e os aspectos das histórias a serem incluídas no *Sprint*.

O Planning é um aspecto importante porque é quando ocorre o entendimento entre o desenvolvimento e o cliente (ou PO) sobre quando algo novo (e funcional) estará disponível. Também é no momento do *Planning* que é definido o tamanho do ciclo de *feedback*, que é igual ao comprimento do *Sprint*. Assim, como resultado da reunião de *Planning*, o cliente terá conhecimento de o que funciona e que receberá algo dentro de duas semanas. A equipe de desenvolvimento, por sua vez, saberá que receberá um *feedback* sobre seu trabalho e terá mais conhecimento sobre o próximo trabalho em duas semanas. Com isso se trabalha a previsibilidade. (QUICK*SCRUM*, 2019)

Ao final da reunião, a equipe e o PO concordam sobre o melhor plano a ser desenvolvido, com base no que sabem, e o trabalho é iniciado.

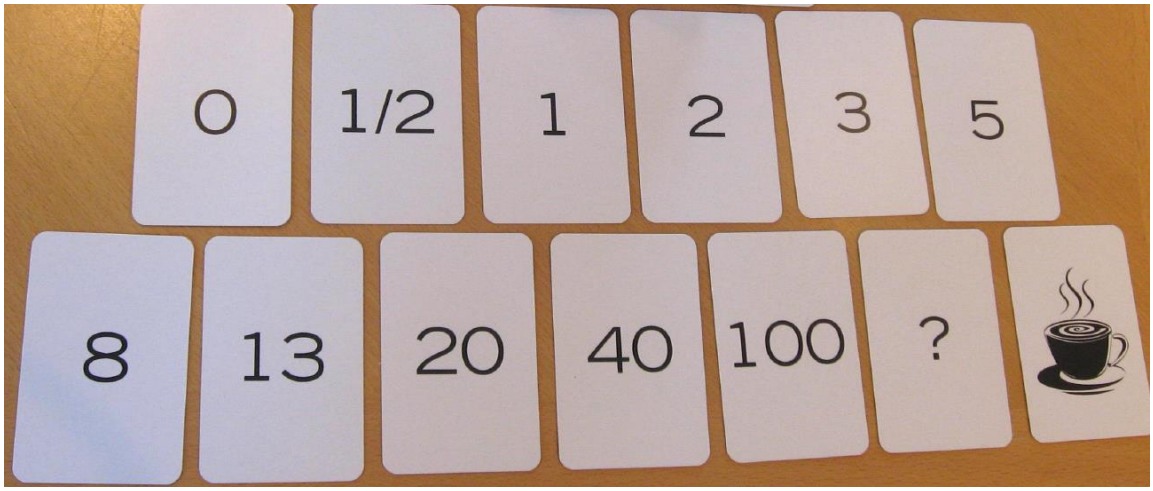
Uma estimativa de tempo para construção das histórias pode ser planejada com o Planning Poker, quando toda a equipe se envolve para avaliar o tamanho da história. Normalmente diversas pessoas participam da história executando testes, codificando, passando para produção, entre outras atividades. Quando a equipe faz a avaliação em conjunto, resulta em união, mais questionamentos e empenho. Pode acontecer de os integrantes com uma estimativa diferente para cada história e a utilização do Planning poker possibilita um entendimento comum. (KNIBERG, 2007)

> O Planning Poker é uma técnica ágil de estimativa e planejamento baseada em consenso. Para iniciar uma sessão de planejamento de pôquer, o PO ou o cliente lê uma história de usuário ágil ou descreve um recurso para os estimadores.
> Cada estimador segura um baralho do Planning Poker com valores como 0, 1, 2, 3, 5, 8, 13, 20, 40 e 100, que é a sequência que recomendamos. Os valores representam o número de pontos de história, dias ideais ou outras unidades nas quais a equipe faz a estima.
> Os estimadores discutem o recurso, fazendo perguntas ao P.O, conforme necessário. Quando o recurso foi totalmente discutido, cada

> estimador seleciona em particular um cartão para representar sua estimativa. Todas as cartas são reveladas ao mesmo tempo.
> Se todos os estimadores selecionaram o mesmo valor, isso se torna a estimativa. Caso contrário, os estimadores discutem suas estimativas. (MOUNTAIN GOAT *SOFTWARE*, 2019)

Na sequência, os estimadores devem então compartilhar suas razões, sendo que uma discussão mais aprofundada ocorre e cada estimador volta a selecionar uma carta de estimativa e todas as cartas são reveladas simultaneamente. Esse processo se repete até que se chegue a um consenso. Caso esse consenso não seja atingido, será possível que a reunião seja adiada até que informações adicionais possam ser obtidas. No total, cada integrante deve trabalhar com 13 cartas (FIGURA 5).

Figura 5 - Planning Poker



Fonte: Kniberg, H. *Scrum* and XP from the Trenches. Crisp, 2007. p. 15.

Decifrando as cartas:

“0” é uma história que será avaliada como pronta ou a demorar alguns minutos;

“?” indica que o integrante não consegue avaliar a história

 indica a solicitação de um intervalo.

Para Elssamadisy (2007) a *Planning poker* afeta indiretamente pontos importantes para Ágil quando os valores estão relacionados aos atrasos, à usabilidade esperada pelos clientes, à visibilidade e ao tempo de mercado.

Um erro em que as equipes podem incorrer no processo do *Planning poker* é criar discussões em torno do tempo. O melhor é se limitar aos pontos e deixar a precisão do tempo para as tarefas ou histórias, sem ficar preso a ferramentas. Deve-

se lembrar que o desenvolvimento do *software* é um jogo de interação e comunicação. (ELSSAMADISY, 2007)

Após a equipe ter selecionado todas as histórias, o *Scrum Master* deve liderar um planejamento para dividir as atividades em tarefas. Essas tarefas são os itens necessários para cumprir com o que foi prometido para o PO e a implantação desses recursos. (SUTHERLAND; SCHWABER, 2007)

3.2.2 Daily

Daily é a reunião diária na qual todos do grupo ficam de pé por 15 minutos em frente ao Kanban e os desenvolvedores basicamente respondem a três perguntas. O que fez ontem? O que vou fazer hoje? Quais são os impeditivos para continuar alguma atividade? (MAHADEVAN, 2015). Tem-se como objetivo adquirir uma visão geral do projeto, atender as pessoas em suas necessidades e adaptar o plano em tempo real daquele dia. O S.M. tem um papel importante e responsabilidades nessa fase, além de liderar a reunião, deve ter claras as tarefas que estão encerradas, as tarefas iniciadas, as tarefas descobertas e qualquer alteração na estimativa. (SUTHERLAND; SCHWABER, 2007)

O painel (Kaban) a ser utilizado na Daily para o acompanhamento das tarefas, pode ter diferentes formatos, porém alguns elementos são comuns a todos. No quadro 5, é apresentado um modelo. O preenchimento deve ser da esquerda para direita.

- *READY TO START* - (Pronto para começar) contém todas as histórias prioritárias para o PO, que estão aguardando um refinamento.
- *HISTORY* – (História) é o campo onde estão as histórias selecionadas na *Planning* para implantação na *Sprint*.
- *TO DO* - (A ser feito) são todas as tarefas exigidas para a implantação das histórias da *Sprint* em vigor. Neste caso, permanecem as que ainda não foram iniciadas.
- *DOING* (Sendo feito) - são as tarefas em andamento sendo realizada por um integrante da equipe. Na Daily é importante visualizar se as tarefas estão há mais de 8 horas nesta etapa do painel.

- *TESTING* (Testando) - é quando a tarefa já foi desenvolvida e o *Tester* da equipe pode avaliar se está com a qualidade e se satisfaz a expectativa do PO, buscando sempre a avaliação de pronto definida na *Planning*.
- *DONE* (Feito) deve-se acrescentar a tarefa nessa etapa quando estiver 100% finalizada.
- IMPEDIMENTOS - é o espaço no painel ao qual o *Scrum master* deve estar muito atento. Nesse espaço a equipe vai acrescentar qualquer item que esteja impedindo o desenvolvimento das tarefas ou histórias.
- RETROSPECTIVA - é o espaço para se acrescentar os itens bons ou a desenvolver identificados na *Daily* para apresentação na reunião da Retrospectiva.

Quadro 5 – Painel utilizado na Daily, para acompanhamento do fluxo da Histórias

Sprint	Objetivo: Ex.: Aprovisionar a funcionalidade X no software Y.				
N°: da Srint	Data de inicio: DD/MM			Data fim: DD/MM	
Ready to Start	**History**	**TO DO**	**DOING**	**TESTING**	**DONE**
São histórias em que o P.O. priorizou para as próximas sprints.	Histórias que estão sendo executadas para atingir o objetivo da sprint.	Todas as tarefas para garantir as histórias definidas na Planning.	Tarefas em execução no dia.	Tarefas em teste das funcionalidades.	Tarefas finalizadas.
Impedimentos	**Retrospectiva**				

Fonte: Elaboração própria (2018).

Assim tem-se um entendimento dos progressos e atividades, além da identificação dos impedimentos. Com o tempo, a equipe começa a perceber os

benefícios e passa a fornecer informações diárias para que se possa atuar em tempo hábil sobre os problemas e entrar em um ritmo produtivo, o que também fortalece a colaboração. O contato diário também possibilita maior integração entre os membros da equipe, favorecendo ainda mais especialmente as equipes iniciantes. As *Daily* também favorecem o aumento do comprometimento individual, uma vez que ninguém quer atrapalhar o andamento da equipe. Com o tempo, observa-se uma redução significativa das respostas ilegítimas por atividades não entregues. Por outro lado, para alguns, essa rotina é considerada como micro gerenciamento e, nesse caso, pode atrapalhar o andamento da equipe. (MAHADEVAN, 2015)

O local e horário das *Daily* devem ser planejados na Planning, sendo essencial para o início das atividades que cada integrante comece a trabalhar. As reuniões pela manhã são mais recomendadas, entre 9 e 10 horas, porém, o mais importante é que a equipe concorde com o horário, para que haja participação de todos.

Elssamadisy (2007) reforça a importância da interação diária devido à melhoria contínua da comunicação e do aprendizado. E nessa reunião tem-se um tempo ótimo, uma vez que é breve e objetiva, evitando desperdícios. Com as conversas diárias limitam-se as ferramentas de comunicação não direta como, documentos, e-mail, telefonemas, e outras formas que estão propensas ao erro.

Um cuidado que a equipe deve ter nesta reunião é evitar que um membro ou mesmo a própria equipe use o tempo para apresentar detalhes de suas atividades com relatos ou *status*. Isto alongaria a reunião e diminuiria a sua utilidade com a redução do foco. Caso exista algum item a ser detalhado, deve-se agendar um outro momento para debate do tema específico. Outro item para garantir o foco é ter presentes todos (e apenas) os participantes que serão diretamente afetados pelos resultados e decisões da reunião. (ELSSAMADISY, 2007)

3.2.3 Retrospective

Retrospective: nessa reunião, como o nome diz, faz-se uma retrospectiva, avaliando os principais fatos que aconteceram durante a *Sprint*, tanto os positivos como os que devem ser melhorados. A equipe, no fim da *Sprint,* busca aprendizado constante, propondo e ações necessárias para melhorias. Derby (2006) resume a

retrospectiva em cinco etapas recomentadas para toda equipe: definir contexto, coletar dados, gerar *insights*, definir o que fazer e encerramento.

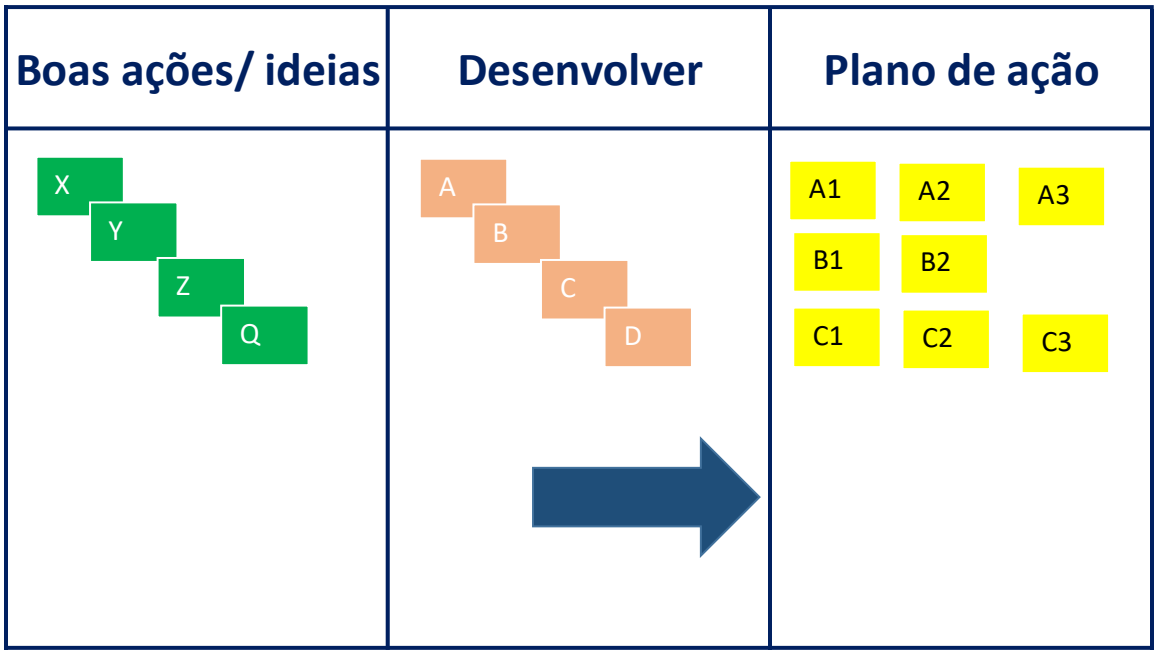
- Definir contexto: estabelece-se um objetivo e foco que contribua para o conforto da equipe em participar. Essa etapa estimula a participação de todos porque visa a aprendizagem em grupo, deixando aberto o espaço para diferentes pontos de vista.
- Coletar dados: ocorre durante as duas últimas semanas. Inicia-se sempre com dados mais óbvios, como impedimentos, decisões pontuais, novas tecnologias. Identificam-se os marcos bons e ruins em um painel com cores diferentes, provocando o início da discussão.
- Gerar *insights*: é o momento de perguntar “Por quê?”. Analisar as condições, padrões e interações ajuda a gerar *insights*. Além disso, observam-se os eventos, riscos ou resultados inadequados. Consideram-se as possibilidades analisando-as analiticamente uma a uma e a suas causa e efeitos. É importante, contudo, que se tenha uma visão abrangente dos métodos e práticas.
- Definir o que fazer: nessa etapa a equipe tem uma lista de plano de ações. Deve-se escolher as ações prioritárias para buscar as melhorias em duas *Sprints*. O PO deve ajudar a equipe na escolha das opções com as quais possam se comprometer e, consequentemente, visualizar resultados positivos. Com isto, deixa-se a equipe determinar o melhor caminho, ao invés de pré-definir o resultado.
- Encerramento: é importante não deixar que o trabalho da retrospectiva se perca, documentando e divulgando os planos de ações. Deve-se também lembrar de acompanhar com a equipe a evolução das ações levantadas na retrospectiva durante a *Sprint*. Ações que o time não identifique como possíveis de se realizar na próxima *Sprint*, não devem entrar nesse acompanhamento, mas sim, deve-se deixá-las para serem avaliadas nas próximas conversas

O mais importante sobre essa reunião é que a equipe garanta que ela aconteça. Por algumas razões a equipe pode não se ver engajada em realizar essa reunião, no entanto, é o momento em que avaliam o que podem melhorar e o que realmente é bom e deve continuar sendo executado. Neste momento tem-se a equipe aberta para

novas ideias e para adquirir planos de ação para melhorar, claro que não precisa esperar a retrospectiva para levantar uma boa ou má ação que aconteceu, porém, havendo esse espaço garante-se que todos possam ser ouvidos. (KNIBERG, 2007)

A retrospectiva pode ser realizada em um período entre 1 e 3 horas dependendo da quantidade de ações levantadas pela equipe. A reunião deve contar com a presença do PO, do S.M., e da equipe e todos devem se manifestar. Para facilitar o transcorrer da reunião, é aconselhável relacionar os itens bons e os itens a melhorar e coloca-los em um quadro à vista dos participantes.

Figura 6 - Painel para uma retrospectiva



Fonte: Elaboração própria (2018).

Na figura 6 tem-se uma demonstração simples de um painel que deve ser construído pela equipe na retrospectiva, contendo as seguintes distribuições:

- Boas ações/ ideias: visa apresentar para a equipe todas ações positivas para garantir que sejam repedidas nas próximas *Sprints*. Nesta etapa qualquer dos integrantes pode inserir no quadro um item relacionado a uma atitude de um companheiro, do grupo ou dele próprio.
- Desenvolver: apresenta ações, comportamento, atitudes, planos que impactaram positivamente o andamento da *Sprint*.

- Plano de ação: a equipe deve elencar alguns dos itens a desenvolver e listar atividades para um plano ação. Não é necessário realizar um plano para todos os itens, mas sim, selecionar no mínimo dois e ter um plano claro e o resultado deve ser acompanhado e apresentado na próxima reunião.

As equipes mais eficazes adaptam as práticas de desenvolvimento de *software* relevantes. Para adaptar é necessário *feedback* constante no desenvolvimento, mas também no comportamento de cada um e na forma como as pessoas estão trabalhando. Por essa razão, o respeito é extremamente importante nessa reunião, nas coletas das informações individuais e, também, ao gerar ideias em grupo para trabalhar em algumas ações e reconhecer outras. (ELSSAMADISY, 2007)

Há, porém, que se cuidar para evitar os erros frequentes nessas reuniões. Elssamadisy (2007) ilustra os principais, ou mais comuns, erros apresentados:

- Ter como meta vários planos de ações e não conseguir cumpri-los;
- Criar ações que não são mensuráveis ou realizáveis;
- Optar por ações que estão fora do grupo;
- Não dar andamento às ações listadas.

Adzic (2009) mostra outra questão sobre a qual devemos estar atentos. Quando reunimos de 5 a 10 pessoas podemos alcançar a chamada “sabedoria das multidões”, estado em que um grupo reunido apresenta um nível mais elevado de conhecimentos e sabedoria do que um único indivíduo. No entanto, é preciso observar que às vezes alguns elementos ou mesmo boa parte do grupo pode estar apenas concordando com tudo com o intuito de evitar conflitos.

3.2.4 Review

Para que ocorra a *Review*, ou Revisão, é necessária a presença do PO, do S.M., da equipe de desenvolvedores e, se possível, clientes, acionistas, especialistas, executivos e demais pessoas interessadas. A reunião se inicia com os desenvolvedores apresentando os resultados e com o *software* funcionando, para a visualização do PO ou/e seus *stakeholders*. Na sequência o PO conduz a reunião para demostrar o *status* do negócio e de uso da tecnologia. Por fim, definem-se quais tarefas foram concluídas e aceitas da *Sprint* em questão, no caso das histórias que

não forem aceitas, é preciso avaliar a forma adequada de redirecioná-las em uma próxima *Sprint*. (SUTHERLAND; SCHWABER, 2007)

Essa interação permite que a equipe de desenvolvedores possa apresentar para os responsáveis de negócio e clientes um protótipo, ou a entrega após os testes. Com isto, a equipe aumenta seu relacionamento com gerentes. Neste momento, a equipe de negócio consegue expor sua avaliação das entregas realizadas. (MAHADEVAN, 2015)

Itens que não forem aceitos na *Review* devem virar insumo para serem discutidos na Retrospectiva e, se for o caso, incluir na próxima sprint. (SUTHERLAND; SCHWABER, 2007)

Um dos problemas encontrados nas organizações que buscam o Ágil é que as pessoas não estão acostumas a visualizar as entregas em subconjuntos, e por isso, acabam não conseguindo dar um *feedback* claro sobre o resultado entregue. Caso isto aconteça, uma sugestão é realizar perguntas curtas aos participantes da reunião (P.O., Dev. Team e *Stakeholders*), como: Até que ponto nossa demonstração atendeu as suas expectativas? Com que facilidade o *dev team* tem realizado o desenvolvimento do *software*? Existem requisitos que você gostaria de discutir?

É necessário cuidado com essa abordagem para não tirar o foco da reunião que é a demonstração do que foi realizado, porém, ela pode ajudar os interessados a apresentarem seus *feedbacks* que podem virar insumo para o sucesso das próximas *Sprints*. (ADAMS, 2006)

3.3 Artefatos

3.3.1 Product Backlog

Product Backlog é um artefato sob responsabilidade do PO. Trata-se de uma lista de todos os entregáveis/ requisitos/ histórias da *squad*, relacionados ao produto final, a seus clientes e usuários e seus benefícios. Esse artefato deve deixar claras as suas prioridades, mantendo uma relação positiva com os *stakeholders*. É um documento dinâmico que é priorizado conforme as necessidades atuais do cliente, sendo que o maior valor agregado deve estar em primeiro lugar. (HRON; OBWEGESER, 2018)

Segundo Kniberg (2007), é preciso ter o *product backlog* como o coração do *Scrum*. No decorrer da elaboração é preciso buscar constantemente as metas de negócio e quando se tem muitas histórias técnicas, o PO deve buscar o ganho oculto[13], e priorizar conforme esse levantamento.

O *Product Backlog* pode ser atualizado durante o projeto com a visão clara de tudo o que precisa ser desenvolvido. O conteúdo normalmente tem tamanho distinto e exige níveis diferentes de esforços, e será na *Planning* que essa história será quebrada em porções menores. Um dos mitos do *Scrum* é que ele impede de a equipe ter uma especificação detalhada, porém, na verdade, o *product backlog* tem que estar com o material suficiente para que a equipe possa estimar e construir o software, e cabe ao PO definir o quanto mais de detalhe será colocado no p*roduct backlog*. (SUTHERLAND; SCHWABER, 2007)

3.3.2 *Sprint*

Sprint é o período em que acontece todas as reuniões, artefatos e com todos os papéis envolvidos para garantir a entrega do produto. É normalmente realizada entre intervalos de três a seis semanas. Em todas as *Sprints* é necessário que haja entregas de funcionalidade que gerem valor para o negócio ou para o cliente, e nesse período deve-se identificar, priorizar e atuar sobre os impedimentos. (LEFFINGWELL; SMITS, 2005)

S*prints* mais curtas permitem entregáveis mais ágeis, além de facilitar a mudança de direção quando necessário, além propiciar de *feedbacks* constantes, um alto fluxo de entregas e melhoria continua. No entanto, se a equipe é madura, as *Sprints* com longo prazo oferecem algumas vantagens, principalmente por propiciar mais tempo para a entrega das histórias, menor quantidade de reuniões de planejamento e mais tempo para avaliar os problemas. (KNIBERG, 2007)

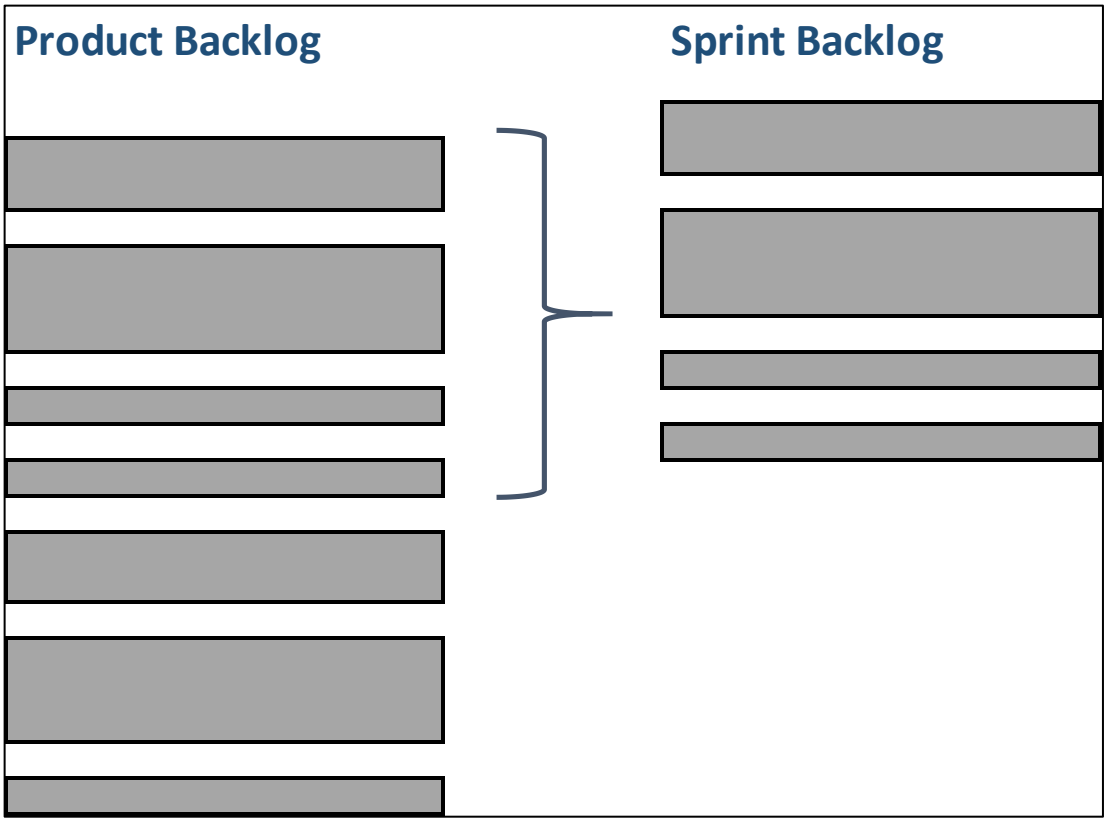
> Quando você está começando no *Scrum*, é razoável experimentar o comprimento do *Sprint* para descobrir o que funciona melhor para o seu contexto naquele momento. Há uma ressalva: *Sprints* mais curtos (uma a duas semanas) ajudam a revelar problemas / impedimentos mais rapidamente. Às vezes isso é desconfortável; e as equipes consideram *Sprints* mais longos novamente para evitar lidar com

13

> esses problemas. Esta não é uma abordagem do tipo *Scrum*; O *Scrum* é destinado a trazer os problemas que temos para o primeiro plano. *Sprints* de uma ou duas semanas são curtos, enquanto os de três a quatro semanas são longos. (LEVISON, 2013)

Definir a meta da *Sprint* sempre é algo que pode trazer ansiedade e preocupação para a equipe. A definição de metas e a organização das histórias a serem apresentadas podem facilitar a quebra dessa tensão. A seleção das histórias feita durante a *Planning* é fundamental para o andamento da *Sprint*. Essas histórias migram do *Product Backlog* para a *Sprint*. (FIGURA 7)

Figura 7 - Ilustração de seleção de histórias do Product Backlog



Fonte: Elaboração própria (2018).

Na figura 7 temos uma representação das histórias como retângulos cujo tamanho demonstra o esforço, sendo a primeira história de maior importância para o PO.

3.3.3 Histórias

Histórias são as funcionalidades do produto, descritas em *post-its* (na maioria das vezes) em um nível que a equipe de desenvolvedores possa entender e quebrá-las em tarefas, além de sempre estar atrelado a um benefício e seu beneficiário.

Conforme exemplo no Quadro 6, Kniberg (2007) explica que para compor uma história são necessários os seguintes itens:

Quadro 6 - Dados mínimos para uma história

ID	Nome	Importância	Estimativa inicial	Descrição	Solicitante	Notas
1	Garantias reais	R$ 1.000.000,00	5	Quando o contrato da pessoa física estiver com um cadastro de garantias, é necessário verificar se a garantia é real (imóveis, máquinas, entre outros) para gravar esse dado para melhor precificar o produto para o cliente.	Product Owner	É necessário gravar em todas as bases enviadas para o cliente, porém neste momento não se alteraram os relatórios.
2	Cartão de crédito	R$ 50.000,00	8	Quando houver algum atraso da fatura de cartão de crédito, enviar um alerta para o cliente via SMS.	Cliente	Não precisa gravar em bases interna.
3	Conta corrente	R$ 3.000.000,00	8	Caso tenha um saque ou transferência acima de R$ 10.000,00 enviar uma notificação para a equipe de prevenção de fraudes.	Parceiros: Equipe de Fraudes	Nenhuma.

Fonte: Elaboração própria (2018).

- ID: uma identificação única da história;
- Nome: um nome curto da história, sendo o suficiente para diferenciar umas das outras;
- Importância: a definição do valor agregado desta história, cuja medida pode variar, como por exemplo, valor financeiro de retorno, redução de tempo, entre outros;

- Estimativa inicial: a equipe avalia o esforço de uma história para entrega final, nesse caso, pode-se utilizar o *Planning poker* baseado na sequência numérica de Fibonacci[14];
- Como demostrar: uma descrição do *status* da história e suas funcionalidades para entrega, contendo as regras, telas, testes, entre outros;
- Solicitante: quem solicitou o pedido – podendo ser o próprio PO, um cliente ou um *stakeholder*;
- Notas: algum esclarecimento que facilite o desenvolvimento ou entendimento da história, devendo ser breve.

Há que se ressaltar que as histórias não podem ser muito grandes e nem muito pequenas. Com pontuações de 0,5 a *squad* pode cair no erro de micro gerenciamento, porém, histórias com 40 pontos tendem a ser parcialmente completas. É aconselhável quebrar em menores para um melhor planejamento da *Sprint*, uma média ideal seria ter história com 5 a 15 pontos e uma *Sprint* com 70 pontos totais. (KNIBERG, 2007):

Além das histórias que o PO apresenta, a equipe e o *Scrum Master* devem levantar a necessidade de histórias técnicas, isto porque, muitas vezes é necessária a instalação de um hardware, algumas integrações sistêmicas, entre outros, que podem afetar diretamente o produto final.

- Devemos reforçar a importância em dois pontos na elaboração de uma história:Descrição do requisito: Necessário dar uma atenção especial aos requisitos solicitados, deixando-os de forma clara para qualquer leitor. Uma definição mal escrita pode trazer prejuízos para o cliente e sua empresa.

> [...] um jogo online de poker no Reino Unido teve um de seus requisitos mal escrito, o que os desenvolvedores realizar um arredondamento dos valores para baixo em alguns casos. Isto gerou vantagens para alguns jogadores e a empresa teve prejuízos de milhares de dólares e uma redução drástica na confiança de seus clientes. Em algumas conversas com os desenvolvedores, todos alegaram que não tinham uma definição clara, e por isto, realizavam esta alteração..(ADZIC, 2009)

[14] Sequência de Fibonacci é a sequência numérica proposta pelo matemático Leonardo Pisa (1170 – 1250, Itália), mais conhecido como Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

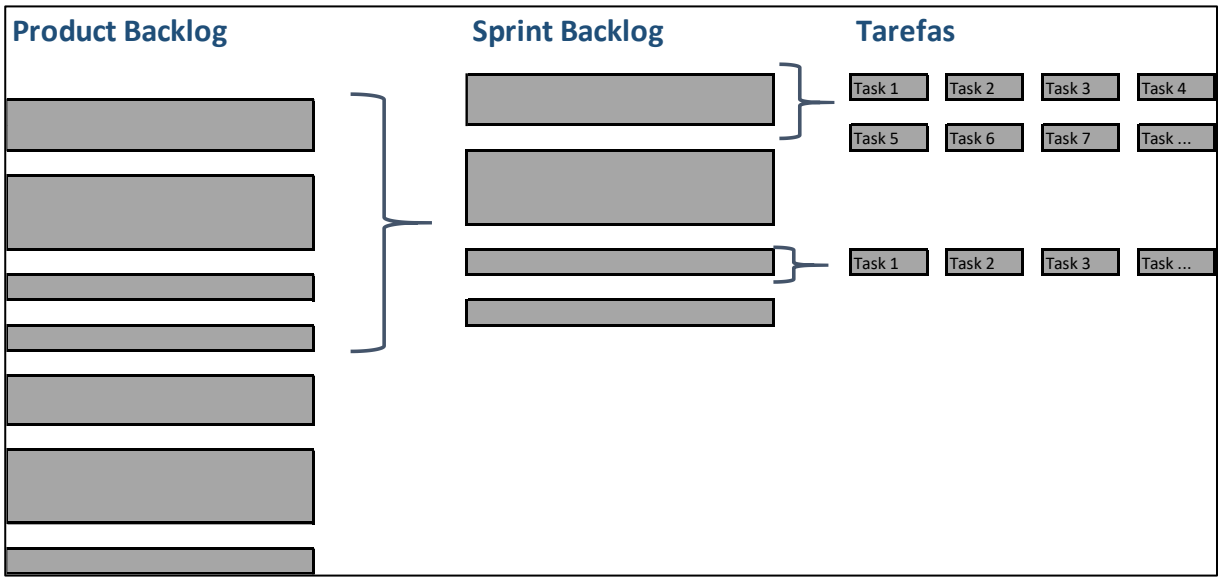
- Debate sobre o requisito: A equipe deve desafiar os requisitos questionando o P.O. sobre a definição apresentada, procurando lacunas, e incoerências. Antecipando os problemas que devam acontecer no desenvolvimento do *software,* e que isto, não chegue em produção na utilização do cliente.

3.3.4 Tarefas

São todas as atividades necessárias para entrega do produto, no seu menor nível, escrita pelos desenvolvedores e com apoio do *Scrum Master* são chamadas tarefas. O ideal seria ter essas tarefas em uma granularidade possível de ser executada em até 8 horas, sendo o acompanhamento diário é mais eficaz.

Há uma diferença entre tarefa e história, ou seja, história, já definida no item 3.3.3, é de responsabilidade do PO, já as tarefas são a abertura das histórias realizadas pela equipe e, em relação a elas, o PO não deve ter preocupação. Nas equipes novas existe uma resistência em abrir em tarefas das histórias, porém isto não deve acontecer pelos seguintes motivos: a abertura das tarefas deixa as histórias mais claras; revela trabalhos adicionais; e as reuniões diárias ficam mais eficientes. (KNIBERG, 2007)

Figura 8 - Ilustração de seleção de tarefas do *Sprint* Backlog

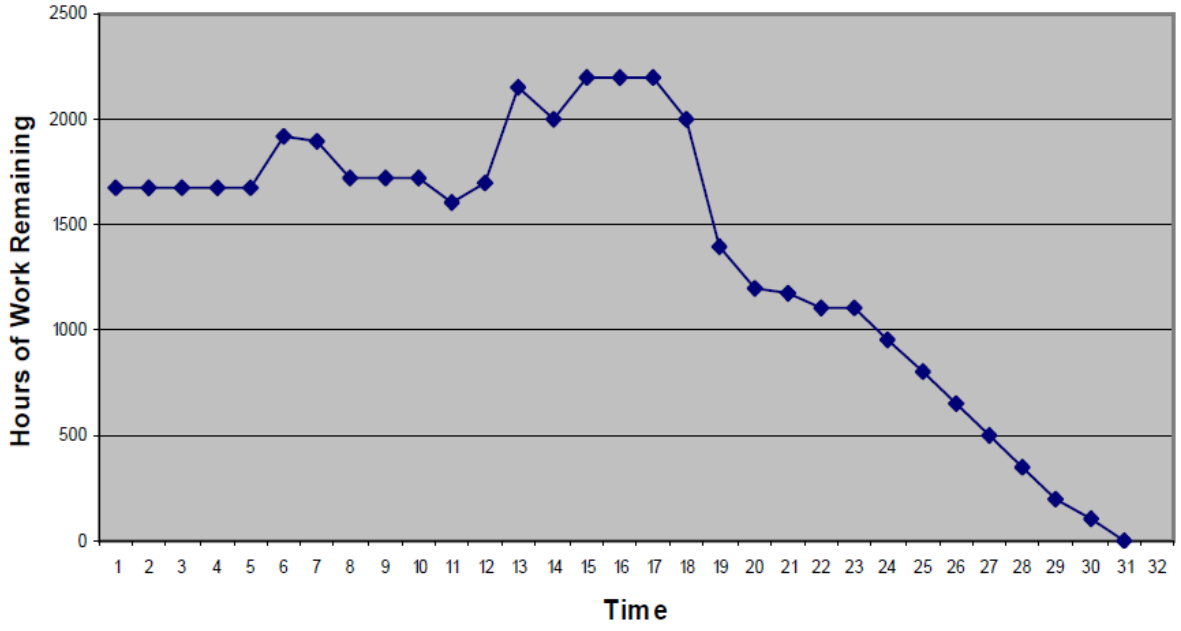


Fonte: Elaboração própria (2018).

Na figura 8 observa-se uma abertura das histórias em tarefas, algo que deve ser feito em todas as histórias para todas as *Sprints*. Com isto, a equipe poderá se orientar e cada integrante pegar uma tarefa para executar no dia. É comum acontecer de integrantes finalizarem algumas tarefas e dizer que não sabem o que fazer no dia, por isto, nas reuniões diárias (*Daily),* o desenvolvedor pode adquirir uma nova tarefa ou ajudar outro integrante da equipe que esteja com uma atividade parada.

Para uma *Sprint* com sucesso é importante um controle das atividades identificadas no início, para isso, tem-se o *Burndown Chart, onde* a estimativa total de trabalho é apresentada de forma acumulativa. O *Scrum Master* recalcula a métrica conforme as atividades são concluídas.

Figura 9 - *Burndown Chart*



Fonte: SUTHERLAND, J; SCHWABER, K. The Scrum Papers: Nuts, Bolts, and Origins of an Agile Process. Washington: PatientKeeper, 2007, .p.18

Quando o valor acumulativo for zero significa que se tem a entrega total dos itens e uma *Sprint* com sucesso (FIGURA 9). A equipe tem uma visão clara da evolução do trabalho no decorrer dos dias. Os defeitos ou as tarefas que foram identificados no decorrer da *Sprint* podem ser contabilizados, e por isto, a linha acumulativa deve crescer e ser maior que o volume de horas identificado na *Planning*. É possível visualizar alguns casos na FIGURA 9 entre os dias 7, e 17. Em resumo,

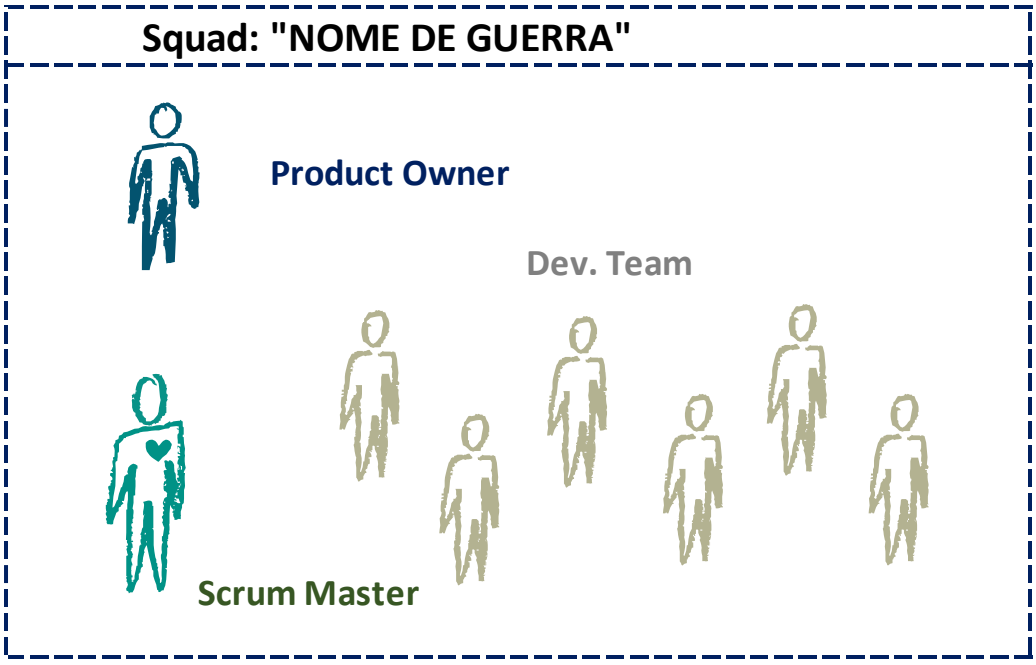
este *chart* (gráfico) é uma ferramenta para orientar a equipe na conclusão das tarefas e finalizar a *Sprint* com sucesso. (SUTHERLAND; SCHWABER, 2007)

3.3.5 Squad

O *Squad,* cuja correspondência em português é ‘esquadrão’, é basicamente composto pelo espaço físico e as pessoas envolvidas desempenhando seus papéis. A *squad* contempla um *Product Owner*, um *Scrum Master* e de seis a oito desenvolvedores (*Dev. Team*). Abaixo de seis desenvolvedores tem-se com uma equipe pequena para entrega do produto final e acima de oito desenvolvedores tem-se muitas pessoas para fazer a gestão, o que pode resultar em redução da produção individual.

A *squad* deve ficar em um local fixo durante a *Sprint* para conduzir as reuniões diárias e as atividades planejadas inicialmente. Além disto, a equipe deve permanecer integral durante todo o *sprint*. Cada Squad recebe um “nome de guerra’, geralmente um nome divertido que pode ou não fazer referência à empresa em que trabalha e/ou ao projeto em andamento.

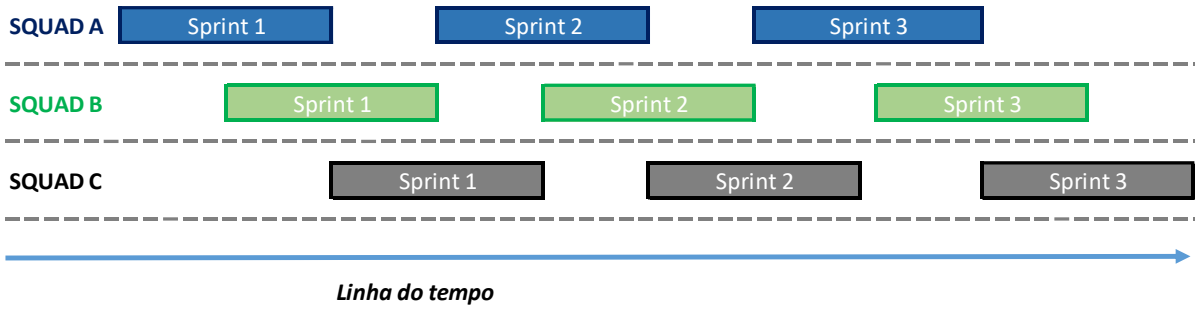
Figura 10 - Integrantes da squad



Fonte: Elaboração própria (2018).

Na figura 10 observa-se a montagem de uma *squad*. Este formato atende uma *squad* apenas, porém, dependendo do empregável, é possível se ter mais ou menos integrantes no *Dev.team*. Porém, não é recomendável que se varie com um número maior que dois integrantes para mais ou para menos. Supondo que o produto final tenha a necessidade de mais pessoas, o que deve ser feito é uma nova *squad* e as equipes devem trabalhar em comunidades. (FIGURA 11)

Figura 11 - Ilustração de *Sprints* e *Squad*s trabalhando em comunidade

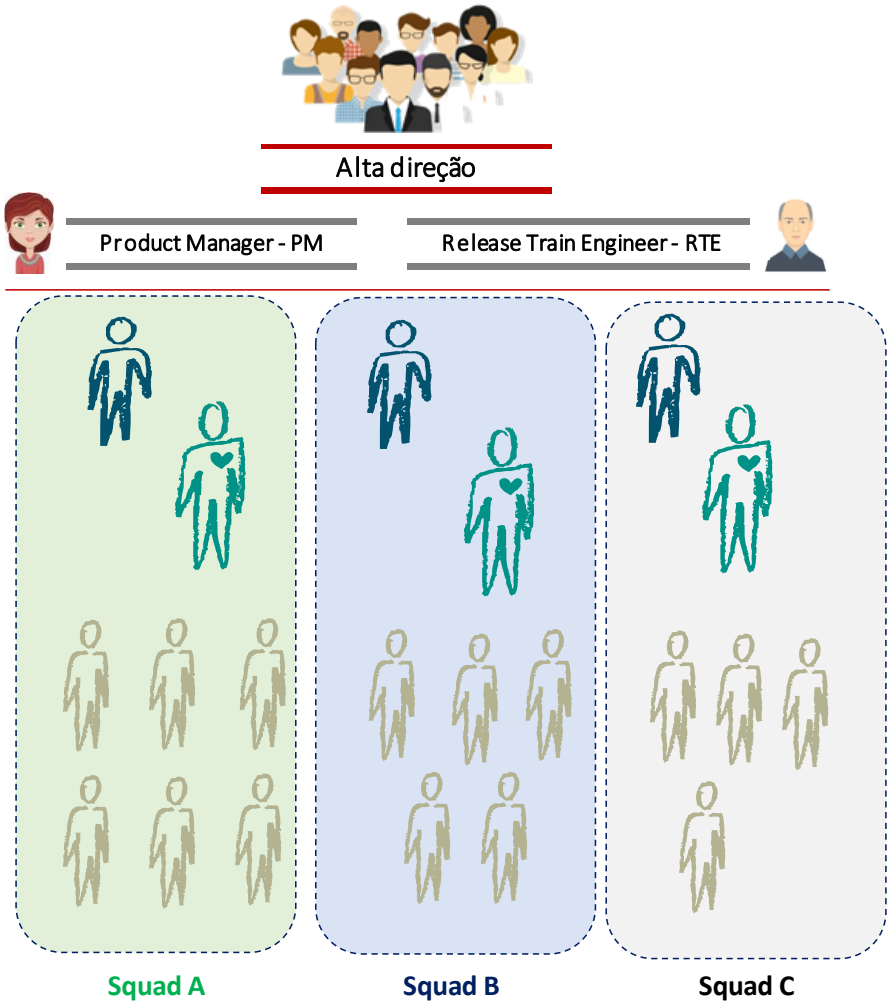


Fonte: Elaboração própria (2018).

Para que todas as *squads* se mantenham no ritmo e com o objetivo correto é importante criar o papel do Gerente de Produto, ou *Product Manager* (PM) que é o responsável pelo PO no apoio à construção do *Product backlog* e à organização das priorizações de todos as *squads*. Além disso, outro papel é o do Engenheiro de Treinamento de Liberação, ou *Release Train Engineer* (RTE) que será o responsável dos *Scrum Master* com a missão de tirar grandes impeditivos que precisem ser levados para alta direção ou direcionamento da equipe.

Com essa estrutura (FIGURA 12), todos podem trabalhar para o mesmo objetivo. Nela, o *Product Manager* está diretamente ligado aos *Product Owner*, representados de azul e o RTE ligado ao *Scrum Master,* representado em verde, direcionando a equipe, em no tom de cinza. Com isto, dá-se início ao trabalho de Ágil em escala. A equipe compartilha suas vivências e dificuldades sendo enfrentadas. Essas conversas são ricas para o desenvolvimento da equipe e velocidade na solução de problemas.

Figura 12 – Ágil escalado



Fonte: Elaboração própria (2018).

Em uma abordagem em escala é necessário ter um Orientador Ágil, ou *Agile coach,* para orientar as equipes ajudando-as a encontrar e aproveitar as oportunidades. Existem equipes que começam a colher frutos desde o início e dobram com ou triplicam sua produtividade nas *Sprints* seguintes. O conhecimento do *Agile coach* vai ajudar o PO, o S*crum* master e a equipe a alcançar sua melhor *performance*. Dentro do ciclo do Ágil existem diversos momentos em o Coach pode ajudar, são eles: formação da equipe e estimativa na realização das retrospectivas, na área de testes, na definição de pronto, no tamanho das histórias e nas consultas sobre as melhores práticas do *Scrum*. (DAVIES; SADLEY, 2009)

Segundo Davies e Sadley, a grande arte do Agile coach está em atrelar o contexto da empresa, princípios, valores e as boas práticas do *Scrum* fazendo com

que estes mundos se conversem. Um dos grandes problemas enfrentados está na adesão do *Scrum* pela empresa e no entendimento de seus benefícios. Embora não seja possível prescrever uma fórmula para o resultado de *coaching* (trabalho de orientação do *Coach*), em razão de os projetos, empresas, pessoas serem diferentes em cada vivência, não é possível falar sobre coaching sem realmente conhecer as boas práticas, princípios e os valores de Ágil e *Scrum*. O *Agile coach*, em resumo, deve ter claras as ferramentas disponíveis, conhecendo para que servem e como devem ser utilizadas.

Para pode estar no comando do seu próprio destino, cada equipe precisa sempre buscar sua auto-organização, trabalhando para atingir suas metas. Esse tipo de equipe demandará menos gestão externa ou direção, resultando em melhor adaptação às mudanças, mais produtividade e melhor desempenho na resolução de problemas, ou seja, é uma equipe preparada para dar a resposta adequada às questões que surgem. (ELSSAMADISY, 2007)

3.4 Kaban

Os painéis (*Kaban*) a seguir demonstram alguns modelos utilizados ao longo do processo do Scrum para o acompanhamento do fluxo do trabalho.

Quadro 7 - Visão geral dos valores do projeto

Product Vision Board

Vision statement: Ex.: Disponibilizar o produto X para equipe de negócio e estimação dos parâmetros "XXX".				
Target group	Needs	Product	Benefits	Value
São basicamente todos os stakeholders.	As necessidades dos clientes com a entrega do produto.	As principais funcionalidade em sua maior granularidade.	Os benefícios esperados.	Os valores a serem alcançados.

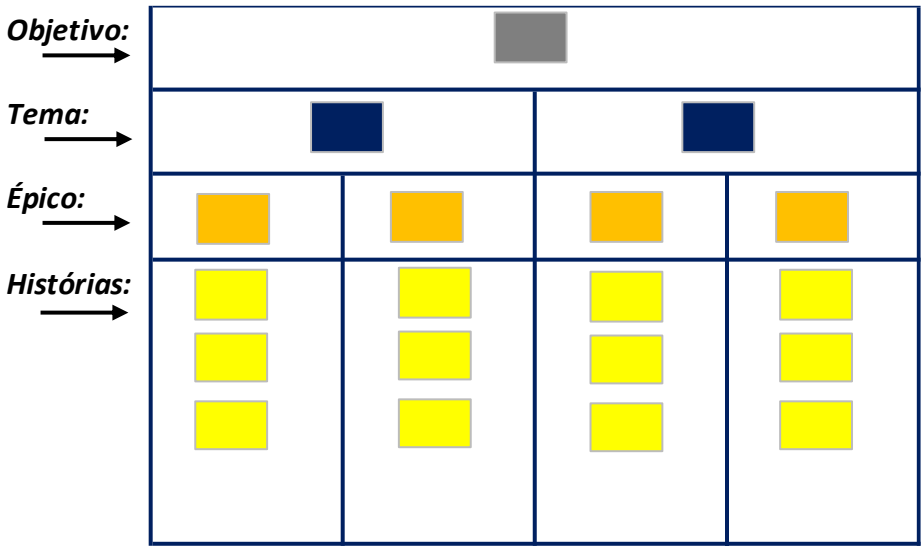
Fonte: Elaboração própria (2018).

O *Product Vision Board* (Quadro de visão do produto) apresenta a visão da *squad* para que todos os elementos da equipe conheçam o valor do produto em que estão trabalhando. (QUADRO 7)

Esse quadro deve ser elaborado antes da primeira *Sprint* em um *pre-game*, do qual participam os clientes, o P.O, o S*crum master*, o *Agile coach* e a *Dev. Team*. No final da montagem todos têm claras as seguintes informações:

- *VISION STATEMENT*: Uma breve visão do ideal da *squad*, contendo os principais valores para criação dessa *squad*.
- *TARGET GROUP*: É uma lista de todos os *stakeholders* (partes interessadas) para a *squad*, como por exemplo, uma equipe interna, acionistas, clientes externos, entre outros. Isto para a equipe conhecer a quem realmente estão atendendo e reconhecerem a importância do seu trabalho.
- *NEEDS*: São as necessidades que motivaram que o *target group* a demandar esse produto. Especifica por que o *target group* vai usar e pagar pelo produto.
- *PRODUCT*: São as grandes funcionalidades do produto esperadas em uma visão de alto nível. Sendo assim, o *product backlog* deve estar associado a esses grandes produtos ou entregáveis.
- *BENEFITS*: São os benefícios esperados com as estregas da equipe, como por exemplo, redução no tempo de atendimento, mais assertividade no retorno ao acionista, atendimento regulatório, entre outros.
- *VALUE*: Nesse item podemos colocar o valor esperado, como por exemplo, aumento de lucro, redução de despesa, entre outros.

Quadro 8 - Story Mapping: a visão do Product Backlog e suas histórias



Fonte: Elaboração própria (2018).

No *Story mapping* (mapeamento da história), temos uma visão geral do *product backlog* desde um alto nível até as histórias. (QUADRO 8). A importância desse quadro está relacionada principalmente à seleção de histórias feitas pelo PO. Essas histórias por sua vez devem sempre estar relacionadas com objetivo principal da *squad*, e com isto, toda a equipe tem uma visibilidade clara da relação das suas atividades diárias com produto maior. Esse quadro tem quatro separações básicas, que são:

- OBJETIVO: Esse item está relacionado diretamente com objetivo em alto nível, sendo por exemplo, reduzir gastos ou aumentar o lucro. Lembrando que deve estar totalmente relacionado com o *VISION STATEMENT*.
- TEMA: É uma abertura para facilitar o entendimento e organização dos épicos[15], neste caso, colocamos termos conhecidos pela equipe e pelo

[15] Épicos: "são *User Stories* que representam itens grandes demais ou sem detalhes suficientes para serem desenvolvidos individualmente. Basicamente, este item representa uma grande *story* que dará origem a uma carga maior de desenvolvimento ou, ainda, representar a ideia do projeto construído como um todo, criando a partir dele outras *User Stories*. Grandes histórias de usuários são normalmente referidas como épicos em todo projeto *Scrum*. Não há limite mágico no qual chamamos uma *story* particular de épica. Significa apenas 'grande *user story*'". Disponível em: https://sitecampus.com.br/user-story-epico-e-tema-qual-diferenca/. Acesso em 23 jan. 2019.

PO. Podemos colocar como exemplos, gastos indiretos, gastos diretos, custos com cobranças, entre outros.

- Épico: Este item está relacionado diretamente com as grandes entregas e funcionalidades.
- HISTÓRIAS: Aqui fica o planejamento das histórias pelo PO.

Esses quadros são de grande importância para dar visibilidade para todos da equipe e principalmente para executivos sobre o andamento do processo, para se estar alinhado com a visão do negócio, ter uma apresentação desde alto nível até o mais detalhado. Traz uma segurança na condução da *squad* e demostra maturidade quando se tem todos os quadros bem preenchidos e de conhecimento de toda a equipe.

3.5 *Scrum* Executivo

Um item importante na utilização do *Scrum* é sua apresentação aos executivos, com a finalidade de mostrar o valor agregado que esse método leva para a empresa. Um executivo, por exemplo, na presença de uma equipe em trabalho, pode questionar o fato de essa estar jogando cartas no período de trabalho. Há que se ter tato ao explicar que estão realizando a *Planning poker,* processo que estimula a comunicação e influencia opiniões, além de potencializar a força das reuniões diárias que alinham a equipe, com as metas de curto prazo.

É necessário cuidado no trato com as reclamações dos executivos no método Ágil. Essas reclamações podem estar relacionadas a: mudança de requisitos devido ao plano estratégico da empresa; equipe multidisciplinar da qual um gerente se torna parte; a distância de quem cria o plano estratégico e quem o executa. Sobre esse último item, sabe-se que existe uma visão de que normalmente a estratégia certa é essencial para o sucesso, porém em sua maioria a má execução das grandes estratégias pode levar ao fracasso. (NORTON; KAPLAN, 2001)

Uma tática que ainda pode ser explorada é implantar o scrum no dia a dia dos executivos, levando para eles uma adequação da *Planning poker*,; a Daily, que pode ser substituída por um cafezinho diário com seus diretores ou gerentes; a criação de um Kanban na sala do CIO (*Chief Information Officer*), a criação de um *Executive*

Product Vision para guiar a equipe executiva. É um tema ainda aberto para exploração e adequação ao dia a dia dos executivos.

3.5.1 O papel do executivo

Leffingwell e Smits (2005) reforçam o papel dos diretores e CIOs de serem os *Scrum* Master da mudança organizacional, tendo a responsabilidade de levar para a equipe as práticas do *Scrum. Realizando* uma proteção para as equipes focarem nas entregas de cada *Sprint*, eliminando qualquer impedimento para o sucesso da entrega e aplicação do método Ágil. Isto é possível quando o executivo busca a comunicação, mudanças e remoção de impedimentos.

Não se deve ter na empresa uma cultura de penalização de pessoas, falta de aceitação de erros ou de qualquer tipo de repressão. Essa cultura tende a levar ao insucesso. O ideal é que trabalhem juntos na identificação e eliminação de obstáculos e na busca por soluções. Envolvimento é uma palavra chave para o sucesso, independente do tamanho da empresa e sua cultura particular.

Schwaber (2004), Appelo (2010) e Leffingwell e Smits (2005) apontam a necessidade de a gerência estar atenta aos ganhos que a empresa obtém com a mudança para o *Scrum*, e que a relação com equipe-gerência deve ser baseada em comunicação direta e constante. Os autores dão ênfase à necessidade de os líderes serem facilitadores e mentores, indicando que o dar apoio e poder à equipe gera entre todos os envolvidos maior segurança e confiança, agregando valor à empresa e aumentando o destaque da liderança.

3.5.2 Escalando *Scrum*

Escalabilidade é chave para um modelo de gestão funcionar acompanhando o crescimento do negócio e dos projetos. Segundo Leffingwell e Smits (2005), com o sucesso de implantação do *Scrum* será necessário implantar uma escalabilidade com diversos desenvolvedores, *Scrum Master* e *Product Owner*, isto porque haverá projetos maiores do que um *squad* pode assumir sozinho e será necessário trabalhar em comunidades, existindo alguns desafios:

1. Escalando a empresa: inicialmente é necessário separar as equipes (*squads*) em comunidades por características da entrega final, como por

exemplo, produto, sistema, serviço, segmento, temas. Com isto, várias *squads* podem trabalhar em paralelo com funcionalidades diferentes e buscando o mesmo *release* (lançamento). Lembrando que quando incluímos novas *squads* para um projeto maior, entende-se que toda a arquitetura já foi pré-definida para estabilidade e foco nas funcionalidades.

2. Infraestrutura e ferramentas corporativas: A equipe normalmente vai querer organizar seus artefatos, e todos os temas de infraestrutura, que são pré-requisitos para início da criação das funcionalidades, devem estar fechados. Por isto, neste item é importante identificar as oportunidades de melhorias constantes no gerenciamento do *backlog*, relatórios de impedimentos, evolução da infraestrutura e ferramentas corporativas.
3. Coordenar diversas *squads*: surgem diversas questões quando se trabalha em equipes grandes, por isso, é preciso haver um acompanhamento diário ou mensal, planejamento de *Sprint*, baseado no *release* fechado para todos, fazendo-se um *Scrum* dos *Scrum* Masters, ou RTE para acompanhar e direcionar a evolução das equipes, além de tirar os grandes impedimentos.

A escalabilidade do *scrum* pode ser explorada ainda de forma mais ampla e oferece um rico espaço para aumentar o conhecimento em Ágil e gestão de grandes projetos.

4 MÉTODO DA PESQUISA

Avaliando a literatura de maior referência em metodologias ágeis e *Scrum* de Sutherland (2014) e tendo em vista a comparação entre *Scrum* e Waterfall, levantou-se um número de questionamentos que podem ser utilizados para desenvolver a compreensão do tema proposto e a elaboração estratégica desta pesquisa.

O método utilizado é uma pesquisa quantitativa, com comportamento qualitativo, método esse já consagrado no mundo acadêmico.

A pesquisa qualitativa se evidencia desde os anos 1960, enfrentando a observação humanista e trazendo diversas técnicas eficazes para as diferentes pesquisas.

A utilização da pesquisa quantitativa e qualitativa tem princípios na investigação, organização e coleta de dados, e busca de esclarecimento do tema ou problema proposto. Sendo assim, após a coleta é possível se organizar a informação em variáveis com características comuns e se estabelecer um relacionamento entre elas, para que se prossiga à análise de conteúdo, buscando a compreensão dos dados levantados. Vale ressaltar, que o melhor método para se ter uma compreensão perfeita do objeto da pesquisa é vivenciar a experiência. (CARVALHO, 2007)

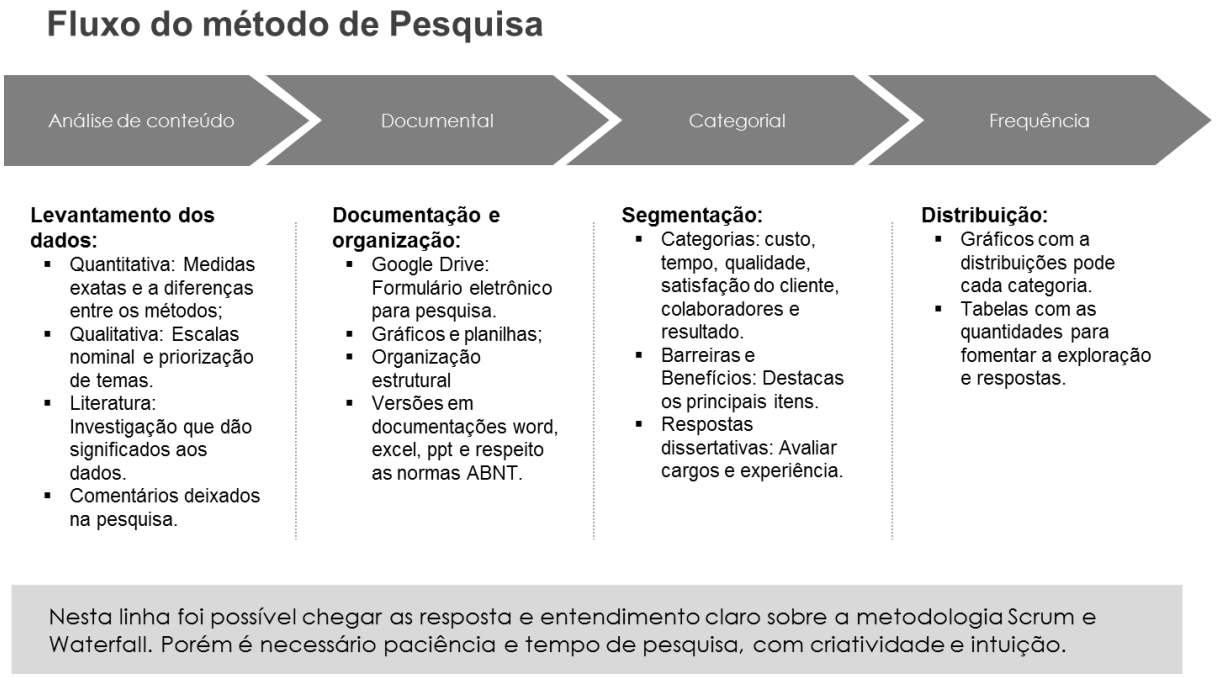
> Inicialmente, pode-se dizer que análise de conteúdo é uma técnica refinada, que exige muita dedicação, paciência e tempo do pesquisador, o qual tem de se valer da intuição, imaginação e criatividade, principalmente na definição de categorias de análise. Para tanto, disciplina, perseverança e rigor são essenciais (FREITAS, CUNHA JR.; MOSCAROLA, 1997).

De acordo com Mantellini (2009), a análise de conteúdo, documental, categorial e frequência, são passo fundamentais para se estruturar uma pesquisa quantitativa e qualitativa após a coleta dos dados.

- Análise do conteúdo: além de realizar a interpretação após a coleta dos dados, conteúdo desenvolve-se por meio de técnicas mais ou menos refinadas quanto ao rigor objetivo ou subjetivo. Aplica-se a linguagem verbal (oral ou escrita) e também a imagens, desenhos, pinturas, cartazes, mídias e toda comunicação não verbal, como gestos, posturas...

- Documental: realiza análise de linguagem verbal expressa por meios da escrita, seguindo fundamentos científicos sistemáticos e objetivos, buscando a inferência de conhecimentos relativos ao documento em análise.
- Categorial: considera todo o texto na análise, passando-o por uma classificação e quantificação, segundo a frequência de presença ou ausência de itens de sentido. É um método de gavetas ou de rubricas significativas que permitem a classificação dos elementos de significação constitutivos da mensagem (BARDIN, 1997). Busca direcionar os princípios do estudo em categorias que apõem o leitor na resposta do problema ou tema.
- Frequência: trata da distribuição estatística recorrente, levando o pesquisador a explicar ou discutir o entendimento que as variedades podem fornecer.

Figura 13 - Fluxo do método de pesquisa para a metodologia Scrum



Fonte: Elaboração própria (2018).

O fluxo do método de pesquisa (Figura 13) retrata a sequência básica realização desta pesquisa.

4.1 Grupo da pesquisa

Este estudo visa realizar a pesquisa com um grupo de pessoas que trabalham ou trabalharam com as metodologias tradicionais (Waterfall) e ágeis (*Scrum*), para que sejam efetivos na comparação. Foi selecionando um grupo diversificado de cargos, desde estagiários até gerentes de projetos (TABELA 2), o que poderá trazer uma visão de toda hierarquia de um projeto, desde as funções operacionais até a gestão de pessoas e projetos.

O grupo selecionado conta com pessoas com idade entre 20 e 60 anos, visando a observação das pessoas com mais experiência e outras com menos tempo no mercado de trabalho, perfazendo um total de 36 pessoas que responderam o questionário identificado na da Tabela 2.

A pesquisa foi realizada em uma instituição financeira que estava trocando sua metodologia de trabalho para *Scrum* na gestão de seus projetos e processos tecnológicos.

Tabela 2 - Quantidade de participantes da pesquisa

Cargo		Tempo de experiência		QTD
Codigo	**Descrição**	**Mínimo**	**Máximo**	**Pessoas**
ES	Estágiario	0	2	4
ANL	Analista	2	5	22
COR	Cordenador	2	10	6
GER	Gerente/ Superintendente	2	21	4
Total		0	21	36

Fonte: Elaboração própria (2018).

Na Tabela 2 tem-se algumas características que devemos explorar de imediato: descrição do cargo, tempo de experiência e quantidade de pessoas.

- Descrição do cargo: atividade profissional atual do respondente. Este dado será importante para delimitar características da visão operacional e da visão gerencial.
- Tempo de experiência: reforça a competência do grupo para realizar a pesquisa e torná-la válida, além claro, de possibilidades de riqueza de comentários.

- Excludentes: é possível identificar quatro pessoas que não têm ou tiveram experiência com um dos métodos da pesquisa (*Scrum* ou Waterfall), por este motivo não serão consideradas suas respostas.
- Total: quantidade total de pessoas que responderam o questionário.

A pesquisa foi realizada via um formulário eletrônico utilizando o “google drive”, com um link automático para acesso.

4.2 Elementos da pesquisa

A pesquisa está baseada em um questionário (Figuras 14 – 18) que visa indicar os principais dados que afetam a gestão de projetos e a comparação entre a metodologia Waterfall e *Scrum*. Com isto, foi dividida em seções:

- 1ª e 2ª seção - Excludentes: Foram propostas duas perguntas iniciais diretas. A primeira, se o entrevistado já trabalhou com a metodologia *Scrum* e a segunda, se já trabalhou com a Waterfall.

Figura 14. Pessoas que conhecem *Scrum*

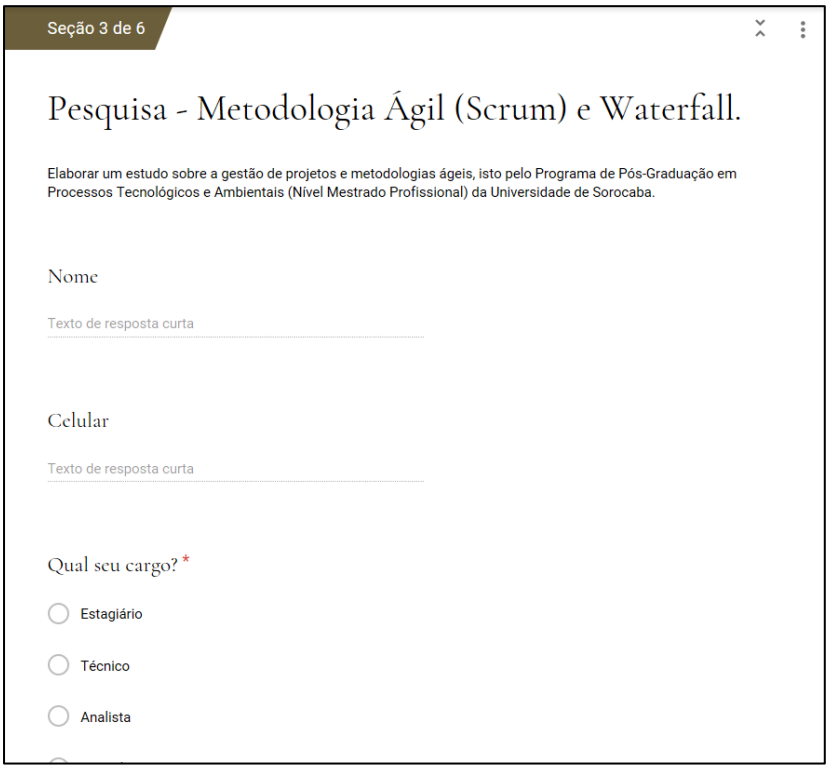
Fonte: Elaboração própria (2018).

Figura 15. Pessoas que conhecem Waterfall

Fonte: Elaboração própria (2018).

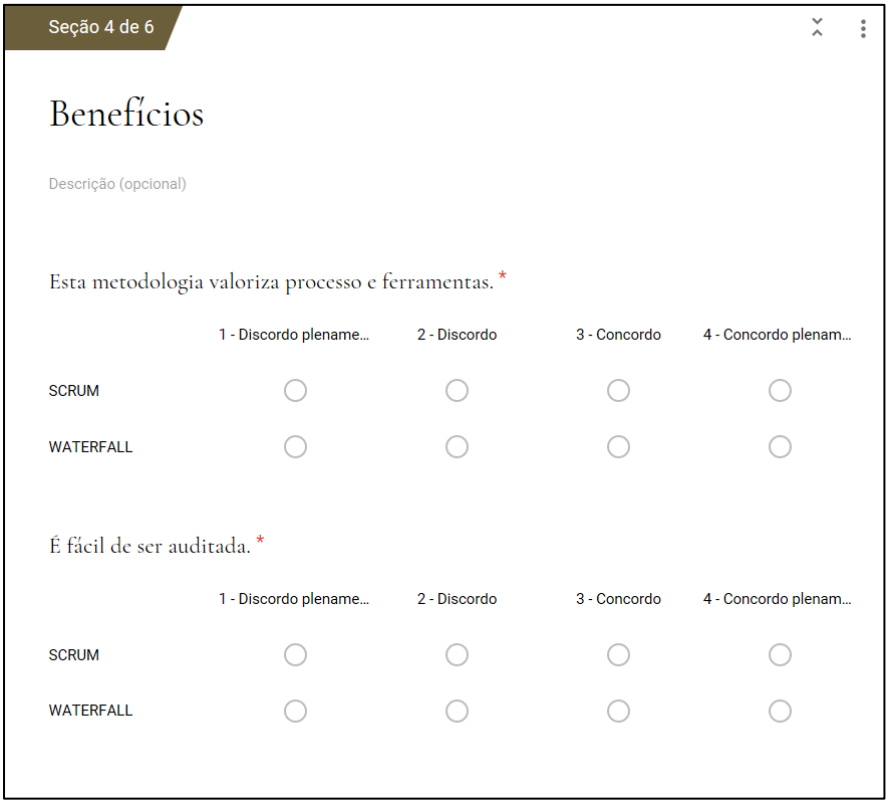
- 3ª seção - Histórico profissional: nessa etapa foram preenchidos os dados históricos de cada profissional, como: nome, cargo, formação, tempo de experiência. Com essa informação torna-se factível criar novas categorias e/ou visões, como: profissionais novos e com mais experiências; ou cargos de gestão e operacionais.
- 4ª e 5ª seções -- Avaliação dos benefícios e barreiras entre *Scrum* e Waterfall. As perguntas foram elaboradas de modo s provocar o respondente a fazer uma comparação direta entre *SCRUM* e Waterfall, sendo enumerados diversos benefícios e barreiras de um projeto tecnológico. Além das afirmações já existentes no questionário, foi adicionado um quadro vazio para que os respondentes pudessem acrescentar e pontuar quaisquer barreiras ou benefícios.
- 6ª seção - Questionário dissertativo: Foi elaborada uma questão dissertativa para trazer a percepção individual sobre o tema: “É possível reduzir custos e tempo em projetos tecnológicos utilizando a metodologia *SCRUM* no lugar da WATERFALL? ”.

Figura 16. Dados cadastrais



Fonte: Elaboração própria (2018).

Figura 17. Benefícios



Fonte: Elaboração própria (2018).

Figura 18. Barreiras

Fonte: Elaboração própria (2018).

4.3 Lógica da pesquisa

O insumo da pesquisa traz diversas aberturas e distribuições possíveis para avaliação dos resultados, porém apenas algumas serão focadas.

a) Avaliação direta entre *Scrum* e Waterfall, tanto para benefícios quanto para barreiras;
b) Avaliação pelo histórico profissional;
c) Avaliação dos submodelos da pesquisa;
d) Distribuição individual de cada item da pesquisa;
e) Entendimento da avaliação descritiva/ subjetiva de cada entrevistado.

Na avaliação foram utilizados os seguintes parâmetros.: “4 - Concordo plenamente”, ” 3 - Concordo”, “2 - Discordo” “1 - Discordo Plenamente”.

4.3.1 Avaliação direta

Para que seja eficaz a resposta ao tema desta tese é necessária uma comparação direta entre as metodologias *SCRUM* e *Waterfall*. Cada respondente apresentará terá uma afirmação de sobre benefícios e barreiras de um projeto. Neste momento, foram apresentadas as duas metodologias uma ao lado da outra para motivar uma comparação.

Há vários itens a serem explorados nos quais uma metodologia impacta mais que a outra, de forma positiva ou negativa, na gestão de projetos tecnológicos.

4.3.2 Dados cadastrais

Nos dados cadastrais é possível categorizar a visão dos respondentes de forma que se possa enriquecer a pesquisa – as categorias são: a visão por tempo de experiência, cargo ou idade, tornando possível a verificação referente à percepção das pessoas com menos tempo de experiência divergir ou não daquelas com mais tempo.

4.3.3 Submodelos da pesquisa

A pesquisa foi pensada em submodelos dentro das categorias de Barreiras e Benefícios.

No questionário temos afirmações relacionadas a um grupo específico, sendo eles: Custo, Tempo, Qualidade, satisfação do cliente, colaboradores e resultado. A seguir é demostrada essa separação (QUADRO 9), embora no questionário as afirmações tenham sido apresentadas aleatoriamente.

No Quadro 9 é possível visualizar todas as categorias abordada na pesquisa, com um total de 31 afirmações, sendo 20 itens na seção de benefícios e 11 na seção das barreiras.

A ordem em que foram apresentados os temas no formulário está referenciada entre parênteses. A intenção foi realmente trazer as questões de forma aleatória entre as subcategorias para forçar o leitor a fazer a análise individual e não ser influenciado para dar uma nota avaliando sua questão anterior. A preocupação está principalmente

na possibilidade de o respondente dar uma nota olhando a subcategoria e não os itens de forma individual.

Quadro 9 - Categorias levantadas na pesquisa

Categoria	Benefícios	Barreiras
Custo	É possível negociação do contrato (6)	Entrega do projeto com custos adcionais (31)
	Elimina desperdício (9)	Investimento em estrutura administrativa (26)
	Reduz os custos de projetos (17)	
Tempo	Responde as mudanças (7)	Entrega do projeto fora do prazo (28)
	Metodologia conhecida no mercado de trabalho (8)	
	Tem velocidade na entrega (11)	
Qualidade:	Valoriza processo e ferramenta (1)	Não temos trabalho excessivo (25)
	Fácil de ser auditado (2)	Não tem documentação excessiva (21)
	Documentação é importante (4)	
	Tem qualidade nas entregas (13)	
Satisfação do cliente	Possível visualizar o software em funcionamento (3)	É possível mudar funcionalidades (26)
	Existe colaboração com o cliente (5)	
	Os resultados atendem os pedidos do cliente (20)	
Colaboradores:	Apoia o desenvolvimento das pessoas (10)	Visão Compartilhada do andamento do projeto (22)
	Existe visibilidade do grupo (14)	Tem comunicação (23)
	Baixa mudança de cultura na empresa (15)	Micro gerenciamento (24)
		Baixo turn-over (29)
Resultado:	Baixo risco na implantação (12)	Não temos planejamento extenso (30)
	A empresa fica mais competitiva (16)	
	Busca gerar valor para empresa (19)	
	Facilita o contato com o cliente (18)	

Fonte: Elaboração própria (2018).

As categorias apresentadas não estão disponíveis para o leitor/respondente, sendo uma referência apenas para conotação qualitativa e quantitativa, além claro, de estarem diretamente relacionadas com a intenção de resposta ao tema desta dissertação. A categorias são:

- Custo: neste item temos afirmações claras sobre grandes preocupações de custos em todos projetos tecnológicos, como desperdício, negociação do contrato, custos adicionais e infraestrutura.
- Tempo: item complexo e relativo em qualquer projeto, por isto, foram apresentadas afirmações relacionadas com a velocidade da entrega, resposta a mudança e prazo.

- Qualidade: mensurar a visão de qualidade pode não ser tarefa fácil, uma vez que cada pessoa tem uma experiência e ideia individual em relação a esse item. Foi então apresentada uma pergunta direta sobre qualidade e o foco ficou na pontuação. Para agregar valor, os itens apresentados abrangem documentação, excesso de trabalho, ferramentas e processos.
- Satisfação do cliente: qualquer projeto deveria buscar constantemente a satisfação do cliente e estar preparado para novas solicitações. Nessa categoria são abordadas mudança de pedidos, colaboração e entrega final.
- Colaboradores: Os desafios no desenvolvimento de *software* estão cada vez maiores, e uma equipe engajada é essencial para as entregas. Para avaliar este item vamos avaliar a visibilidade do grupo, desenvolvimento, comunicação e gestão de pessoas.
- Resultados: neste item são esperados principalmente, valor agregado, baixos riscos, competitividade e planejamento.

4.3.4 Distribuição individual

Além da análise por categorias, a distribuição dos questionamentos individuais é explorada nesta pesquisa, avaliando-se assim todas as distribuições de cada questão de forma individual.

4.3.5 Pergunta descritiva

Além das avaliações estatísticas que serão possíveis conforme colocado, também será levada em conta a resposta ao questionamento descritivo. Nesse item, será possível levantar as percepções e dados subjetivos e avaliar a experiência dos respondentes. Tem-se em mente que as pessoas normalmente levam para esse tipo de respostas os itens que lhes são mais representativos, e com esse material é possível enriquecer a análise de resultados e conclusão.

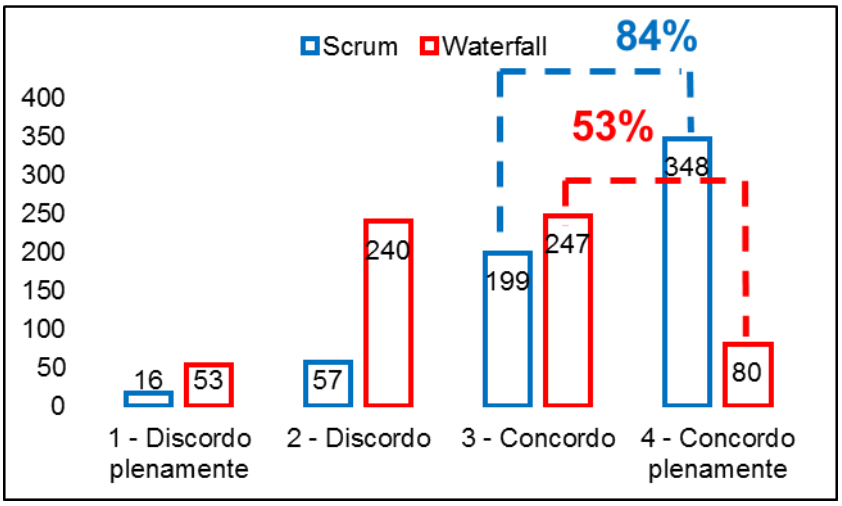
5 RESULTADOS E DISCUSSÕES

Neste tópico vamos iniciar uma apresentação de todo o resultado obtido com a pesquisa realizada. Em primeiro, temos o resultado consolidado de todas as barreiras e benefícios. Na sequência uma apresentação por subcategorias, custo, tempo, qualidade, satisfação do cliente, colaboradores e resultado. Em cada subcategoria teremos abertura dos resultados de cada questão que seja importante para consolidação e entendimento do que está sendo apresentado. Por fim, um consolidado geral de todos os grupos de informações.

5.1 Benefícios

Para identificar se o *Scrum* quando comparado com o Waterfall oferece mais benefícios, foram realizadas 20 questões relacionadas a benefícios. O intuito era colocar o respondente “face a face” com uma comparação direta entres os dois métodos e assim conseguirmos perceber se ele acredita que realmente o *Scrum* traz mais benefícios que o Waterfall. (GRÁFICO 1)

Gráfico 1 - Benefícios



Fonte: Elaboração própria (2018).

Observa-se que 84% concordam que o *Scrum* tem benefícios, o que representa a soma de 348 do item 4-concordo plenamente e 199 do item 3-concordo, dividido pelo total, que é a soma de todas as barras do *Scrum* (em azul). (QUADRO 1)

A mesma lógica foi utilizada para Waterfall e e foi obtida uma *performance* de 53%, demostrando assim um nível mais baixo de satisfação com o método.

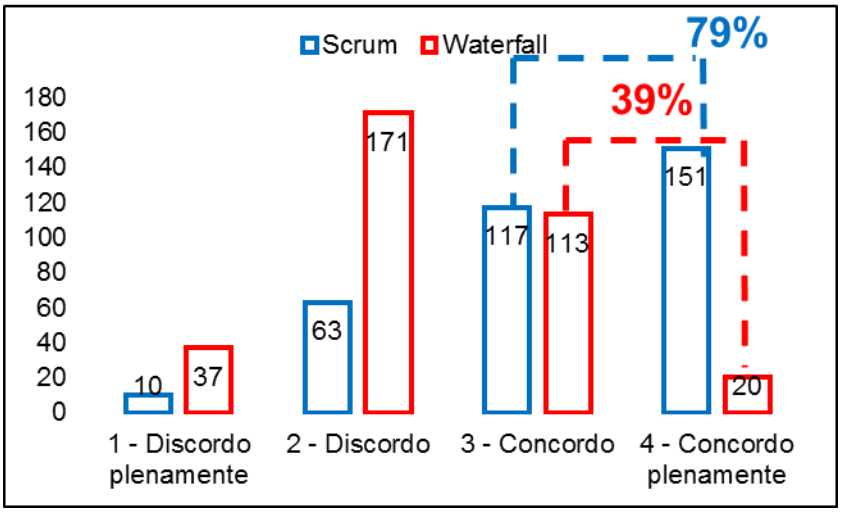
Observam-se ainda 240 respostas ao item 2 - discordo, o que reforça a percepção de insatisfação maior na utilização do método tradicional (na cor vermelha).

Resta um ponto que exige atenção para o *Scrum*, mesmo trazendo mais benéficos ainda há o que ser trabalhado no método. O gráfico 1 apresenta 57 respostas que discordam e 16 que discordam plenamente. Serão detalhadas as perguntas para melhor visualização dos pontos de atuação e explicação dos benefícios.

5.2 Barreiras

Para identificar qual método reduz as barreiras enfrentadas no dia-a-dia em um projeto de desenvolvimento de *software* foram realizadas 11 perguntas com este foco ou agrupamento. O gráfico 2 traz o resultado.

Gráfico 2 - Barreiras



Fonte: Elaboração própria (2018).

No gráfico 2 tem-se uma relação entre item “4 – concordo plenamente” e o 2 - discordo. Quando observado o método *Scrum* vê-se uma aceitação de 151 respostas no item 4 – concordo plenamente, no entanto, Waterfall tem apenas 20. Quando se examina o item 2 – discordo, tem-se as barras invertidas.

Nessa visualização há indícios de que o prometido pela metodologia *Scrum* de ser um método simples, transparente e com alta interação é concreto, sendo que o método traz ótimos resultados, principalmente quando comparado como o *Waterfall*.

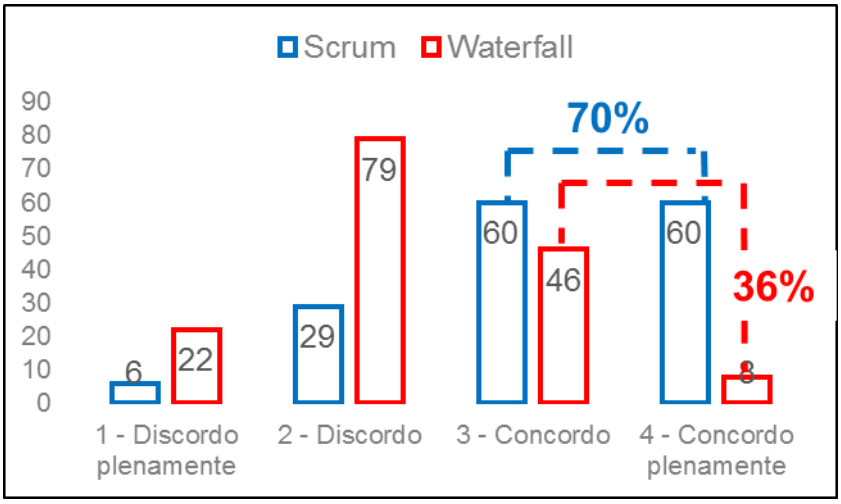
Os principais pontos positivos que elevam esse gráfico a favor do *Scrum* são: prazo de entrega do projeto, possibilidade de mudança das funcionalidades e o gerenciamento. Os itens serão detalhados a seguir.

5.3 Diagnóstico por submodelo

5.3.1 Custo

Neste item existe uma grande expectativa devido ao fato de ter uma relação direta com o tema desta dissertação. Diversas referências bibliográficas trazem no Scrum uma esperança em redução de custo de forma significativa. Principalmente quando o método é comparado com métodos tradicionais. No Gráfico 3, há cinco perguntas específicas sobre a experiência dos respondentes em relação à redução de custo.

Gráfico 3 - Reduz Custos



Fonte: Elaboração própria (2018).

Observa-se a representação sobre o custo em relação às duas metodologias de gestão de projetos. Segundo Capodieci (2014), *Scrum* indica que uma das tendências é a redução de custos quando o método e colocado em prática.

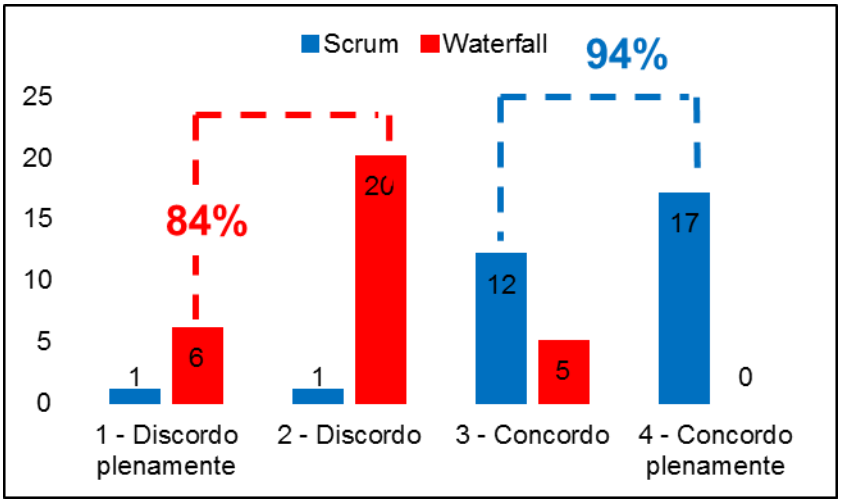
No gráfico 3 observa-se uma característica clara de que grande parte das respostas favorecem o *Scrum* como um método de redução de custos em projetos. São 70 % das respostas que indicam uma redução de custos na utilização do *Scrum.* No entanto, quando avaliado o método *Waterfall* observam-se 36% das respostas indicando a de redução de custos.

Em projetos de desenvolvimento de *software* é preciso estar preparado para os custos imprevisíveis, gestão de mudanças e atuação nos riscos de projetos. (COUTINHO; OLIVEIRA, 2017) O método *Scrum* entrega para os usuários uma forma mais simples de trabalhar essas dificuldades e, por isto, observa-se uma *performance* positiva significativa entre os respondentes.

Para esclarecer melhor a percepção dos respondentes são abertas algumas questões individuais sobre Custos.

O primeiro item a ser detalhado é: “elimina desperdício”. Essa foi uma afirmação apresentada aos respondentes para avaliar o *Scrum* e o *Waterfall*.

Gráfico 4 - Elimina desperdício



Fonte: Elaboração própria (2018).

Observa-se uma tendência significativa mostrando que o método *Scrum* elimina desperdícios. São 17 pessoas que concordam plenamente, 12 concordam e apenas 2 discordam que *Scrum* elimina desperdício, representando 94% das respostas. (Gráfico 4)

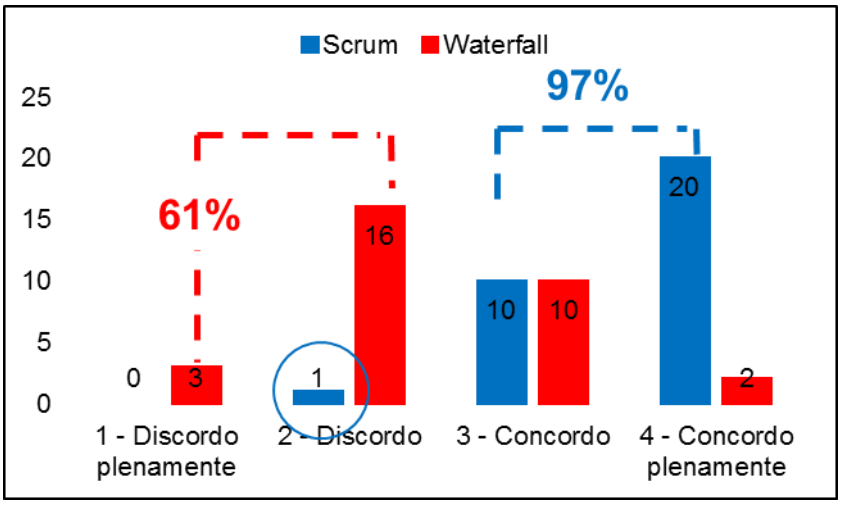
A tendência deste gráfico confirma o relato de Sutherland e Schwaber (2007) que o método *Scrum* tem o cuidado de não realizar atividades desnecessárias, como

por exemplo documentação em excesso e, ainda, de se adaptar ao futuro e mudanças que podem vir, ao invés de gastar recursos em planejamentos detalhados que não serão utilizados.

Waterfall apresenta um quadro complicado em que 84% - que é a soma de 20 do item “discordo” e 6 do item “discordo plenamente”, sobre o total que 31 respondentes indicam que este método não elimina desperdícios.

Outra questão com um demonstrativo significativo foi “é possível a negociação do contrato”.

Gráfico 5 - É possível negociação de contrato?



Fonte: Elaboração própria(2018).

No gráfico 5 é possível visualizar uma insatisfação de diversos clientes e gerentes de negócio de qualquer projeto de desenvolvimento de *software*.

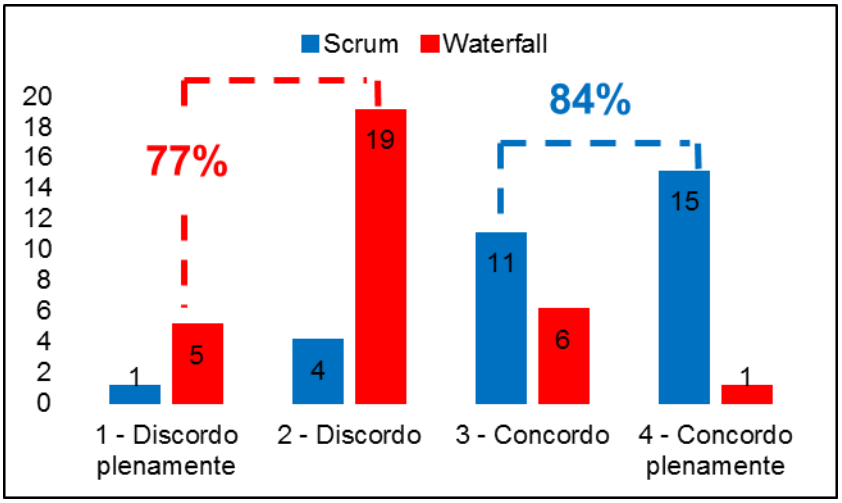
A negociação inicial no *Waterfall* é “sagrada” e não tem alterações no decorrer do caminho. Pode haver uma mudança de cenário no mercado ou apenas uma falha na comunicação inicial do projeto e não haverá alteração no escopo. Para quaisquer alterações necessárias serão incluídos custos adicionais devido a alteração do escopo de trabalho. Por isso, uma insatisfação representada em 19 pessoas que discordam ou discordam plenamente de que a metodologia *Waterfall* tem flexibilidade na negociação do contrato.

Porém, existe tendência contrária no gráfico quando observado o *Scrum*, ou seja, a promessa das referências bibliográficas de que esse é um método flexível é cumprida. Observam-se 98% das pessoas concordando que é possível negociar o

contrato utilizando o *Scrum* e o que era um custo ou uma frustação para o cliente se transforma em receita.

Por fim, em relação aos custos, é apresentada a questão “reduz os custos de projetos”.

Gráfico 6.Reduz os custos dos projetos



Fonte: Elaboração própria (2018).

O resultado dessa colocação no Gráfico 6 é excelente para o tema proposto desta dissertação. Em que se tem 84% dos respondentes concordando que o método *Scrum* reduz custos de projetos.

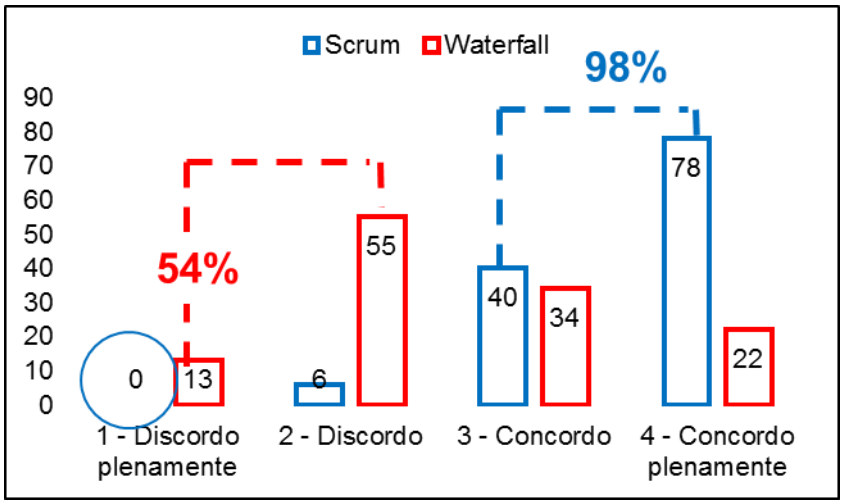
Além disto, outra informação importante é que os respondentes poderiam seguir a mesma avaliação para o método *Waterfall*, deixando o gráfico com a mesma avaliação positiva. Mas foi totalmente o contrário e 77% discordam que *Waterfall* reduz os custos de projetos.

Resultado que apresenta uma informação significativa na comparação entre os dois métodos, e ajuda na decisão de uma organização sobre qual utilizar. Principalmente quando o tema é relacionado diretamente aos custos da empresa.

5.3.2 Tempo

Tempo é algo discutido em todos os projetos e não seria diferente nos resultados desta dissertação. Por essa razão, serão analisadas todas as questões que dão insumo para resposta à questão tempo, para *Scrum* e *Waterfall*.

Gráfico 7- Redução de tempo



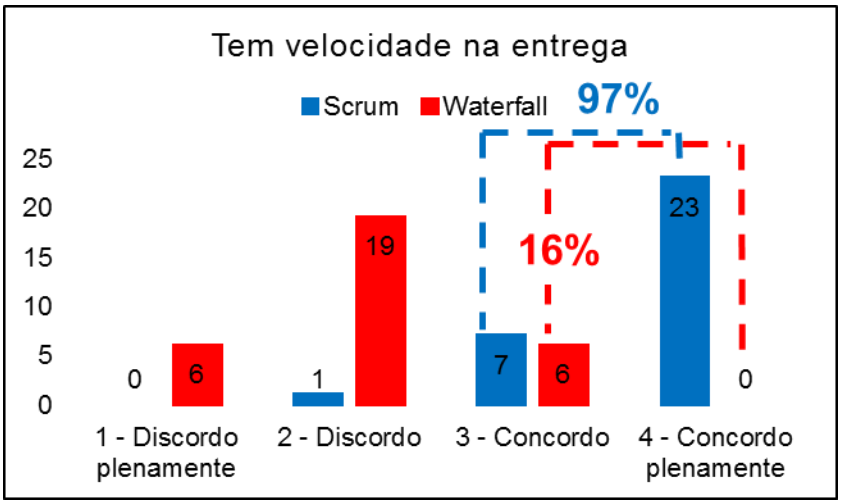
Fonte: Elaboração própria (2018).

Segundo Elssamadisy (2007), na utilização de métodos Ágil tem-se uma redução de tempo nos projetos de desenvolvimento de *software*. O Gráfico 7 traz a mesma percepção dos respondentes, sendo que 98% das respostas indicam que *Scrum* reduz o tempo de projeto: 40 das respostas concordam e 78 concordam plenamente. Kniberg (2007), Elssamadisy (2007), Sutherland e Schwaber (2007) descrevem diversas formas e dão dicas para redução de tempo de projetos, parte delas já descritas nesta dissertação. E com esse resultado torna-se visível que essas dicas e técnicas funcionam e, sim, haverá redução de tempo. Ainda mais se a organização estiver utilizando métodos tradicionais antes da mudança para o Scrum.

Serão detalhadas as questões sobre tempo e esclarecidos os principais pontos.

O primeiro item a ser detalhado é “Tem velocidade na entrega”. Esta foi uma afirmação apresentada aos respondentes que deveriam avaliar entre o *Scrum* e *Waterfall*.

Gráfico 8. Tem velocidade na entrega



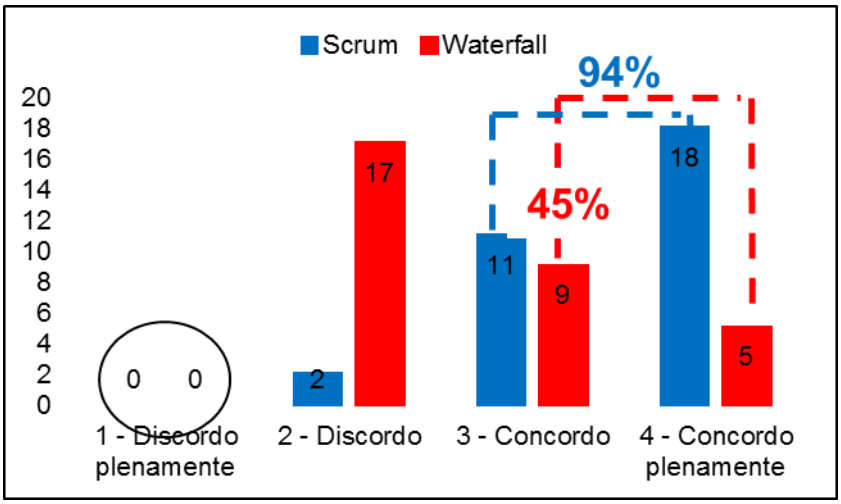
Fonte: Elaboração própria (2018).

No gráfico 8 observa-se um dado muito importante. Nenhum dos respondentes concorda plenamente que o *Waterfall* tem velocidade na entrega, o que reafirma que é um processo lento quando se fala em desenvolvimento tecnológico. Este é um mundo dinâmico e precisa ter velocidade ou sua empresa será “engolida” pelo mercado. Porém, há uma esperança para essas organizações e é possível visualizá-la no gráfico: 97% dos entrevistados concordam que *Scrum* tem velocidade na entrega. E isto deve estar relacionado diretamente com a forma de trabalho do *Scrum*, que Sutherland e Schwaber (2007) explicam, detalhando como funciona o processo de uma *Sprint*. Tem-se entregas que geram valor para o negócio em pouco tempo, mesmo que sejam em pequenas porções.

Para continuar com um entendimento claro desta subcategoria, a avaliação do resultado da questão “Entrega do projeto no prazo”.

Segundo Cleland (1994), os prazos estão cada vez mais curtos para empresas que querem seguir crescendo, e a tecnologia é um dos pilares. Estudos realizados por Marques Junior e Plonski mostram altos índices de projetos de tecnologias não entregue ou entregue fora do prazo. Por isto, avaliar o Gráfico 9 é tão importante. Como um primeiro indicador, pode-se afirmar que 94% concordam que o *Scrum* entrega os projetos no prazo. Enquanto 45 % acreditam que o método W*aterfall* entrega no prazo.

Gráfico 9. Entrega do projeto no prazo

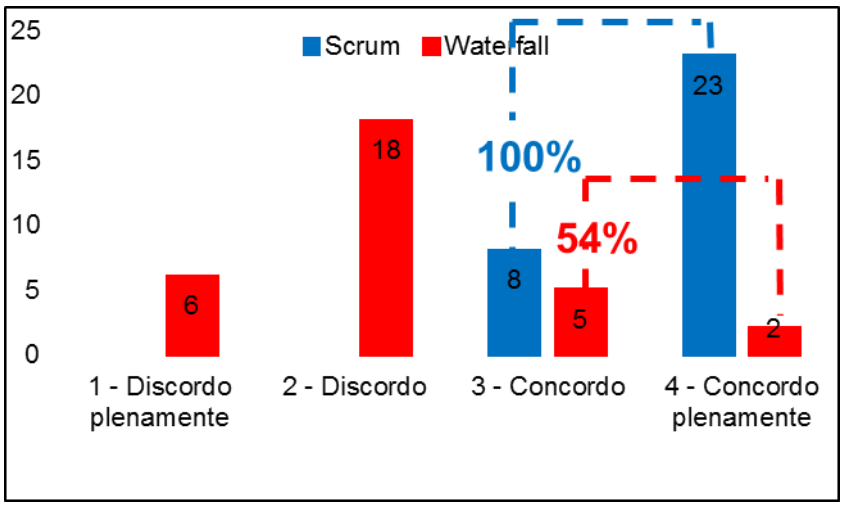


Fonte: Elaboração própria (2018).

Outro dado bem visível é que nenhum dos respondentes discordam plenamente. Um indicador que traz uma reflexão relativa às entregas do *Waterfall* ocorrerem, ainda que parcialmente, mas fica a indagação sobre a entrega da totalidade do projeto.

A questão seguinte que é “responde a mudanças” pode ajudar a responder esta dúvida.

Gráfico 10 - Responde as mudanças

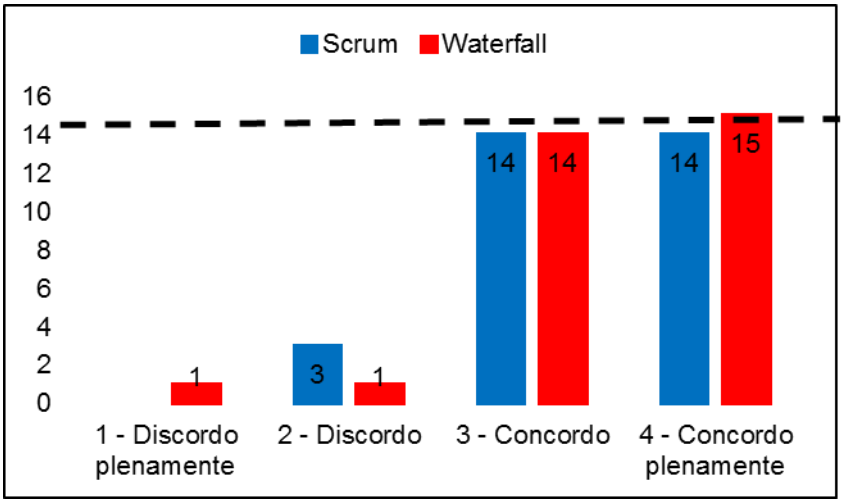


Fonte: Elaboração própria (2018).

De início ve-se no gráfico 10 um desenho muito positivo para *Scrum* e bem desafiador para *Waterfall*. Cleland (1994) reforça que nenhuma organização pode fugir das mudanças em seu desenvolvimento de *software* ou quaisquer projetos tecnológicos. Sabendo disso, o *Manifesto Agile,* segundo Tabaka (2006), diz que o método utilizado tem as respostas que o cliente espera e o mercado está exigindo. No gráfico observa-se 100% de afirmações sobre o *Scrum* responder às mudanças. Mais um ponto positivo da pesquisa, isto porque, é uma das grandes promessas do *Scrum* declarada por Schwaber (2004; 2009). No *Waterfall* já se tem claro que o processo está relacionado às definições iniciais e qualquer alteração não será acatada, por isto, apenas sete pessoas concordaram que esse método responde a mudanças.

Por fim, nesta subcategoria é colocar a questão “Metodologia é conhecida no mercado de trabalho”. A razão dessa afirmação colocada para os respondentes é que visualizem a percepção de tempo e contratação de pessoas no mercado de trabalho que conhecem esses métodos.

Gráfico 11 - Metodologia conhecida no mercado de trabalho



Fonte: Elaboração própria (2018).

No gráfico 11 é possível visualizar uma igualdade entre *Waterfall* e *Scrum*, ou seja, os dois métodos são conhecidos no mercado de trabalho. Porém, também se observa uma leve tendência a afirmar que o *Scrum* não é conhecido e isso deve estar relacionado ao fato de que o método *Scrum* é utilizado desde 1993, segundo Sutherland e Schwaber (2007). Já o *Waterfall* é utilizado desde os anos 1970, de acordo com Mahadevan (2015). No futuro seria interessante rever esse gráfico e

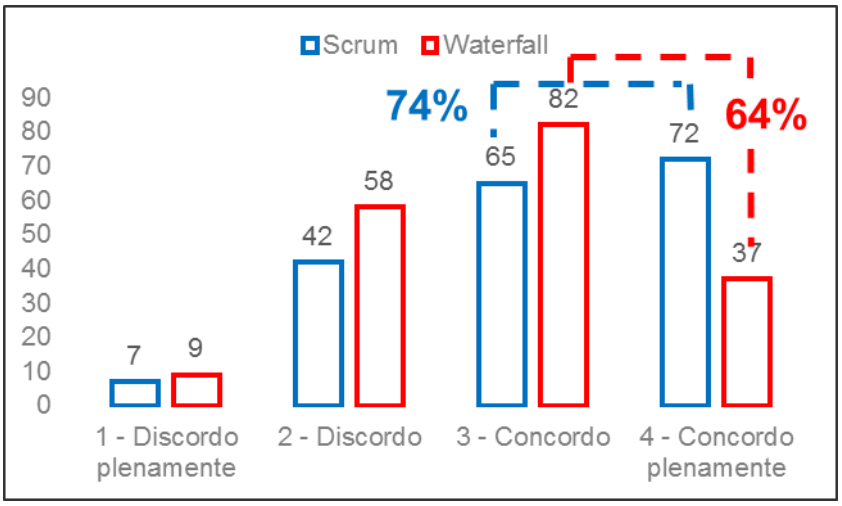
avaliar se houve uma mudança devido à exigência cada vez maior de que se conheça o método *Scrum* e, cada vez menos o *Waterfall*.

5.3.3 Qualidade

Qualidade é uma demanda para todos os projetos, e cada um tem sua categoria ou a forma esperada. Isto depende do produto, do cliente, da situação do mercado ou da situação da empresa. No entanto, temos um consenso comum que é entregar o que foi solicitado pelo cliente.

Além disto, em projetos tecnológicos tem-se temas de auditoria e documentação, que direcionam a qualidade do desenvolvimento de um software. Para ajudar, é avaliado um gráfico consolidado por uma subcategoria chamada Qualidade.

Gráfico 12. Qualidade na entrega

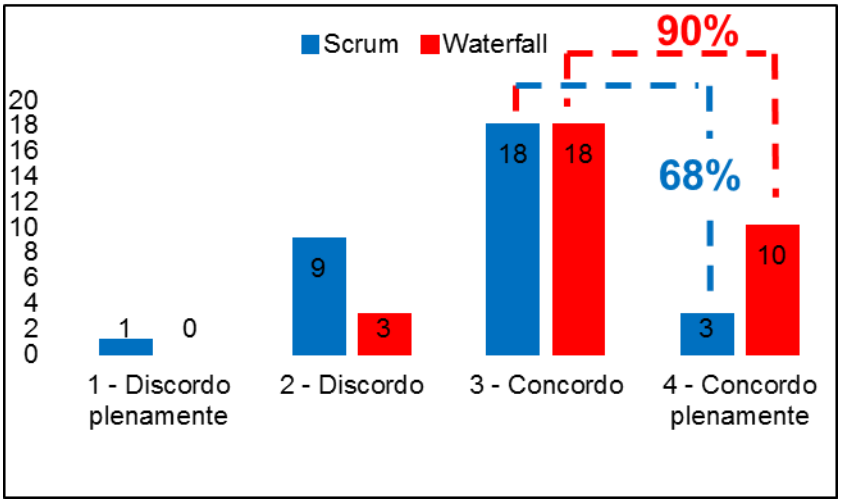


Fonte: Elaboração própria (2018).

No *Scrum*, são prometidas diversas reduções quanto ao custo e tempo, porém isto poderia impactar a qualidade das entregas, porque seria uma decepção da mesma forma para o cliente final. No gráfico 12, tem-se 74 % de aceitação que o *Scrum* entrega com qualidade, estando um pouco inferior em relação ao *Waterfall*, com 64%. Apesar de uma acentuação importante no ‘concordo plenamente’. O gráfico mostra que as duas metodologias buscam as entregas com qualidade, principalmente quando diz respeito à documentação e processos auditáveis. Para entender melhor, segue uma validação detalhada.

Para iniciar tem-se a questão “É fácil de ser auditada”. Muitos processos tecnológicos precisam passar por auditoria para ter o “selo” de qualidade deste setor que normalmente responde diretamente para alta gestão da organização.

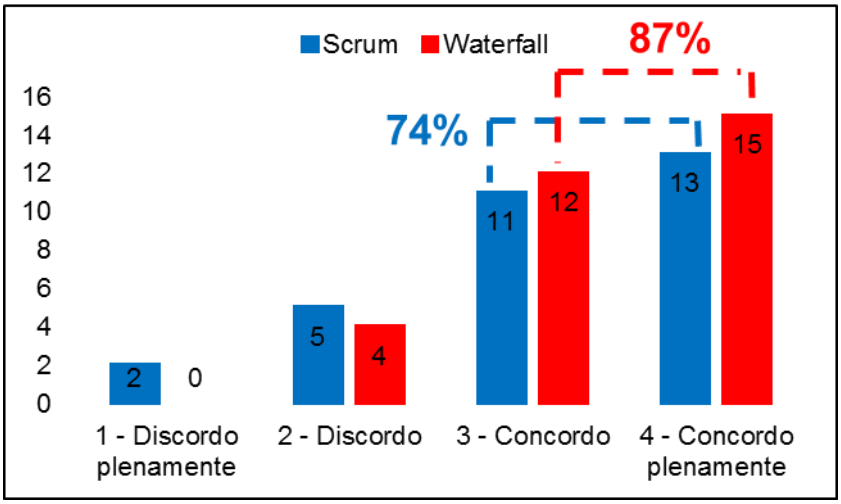
Gráfico 13. É fácil de ser auditado



Fonte: Elaboração própria (2018).

De acordo com o gráfico 13, há uma relação superior da *Waterfall* que tem 90% de aceitação versus *Scrum* que tem 68%. Uma característica clara do projeto em método *Waterfall*, devido ao fato de esse valorizar e utilizar grande parte do seu tempo de projeto em documentações. Segundo Adams (2006), este método foca em provar para seus consumidores o que foi realizado o planejamento inicial, tornando assim um processo preparado para ser auditado. Já o *Scrum* foca no cliente e na entrega conforme ele espera, porém, ainda se tem um mito de que o *Scrum* não precisa de documentação, o que é verificado no gráfico seguinte.

Gráfico 14. É importante documentação



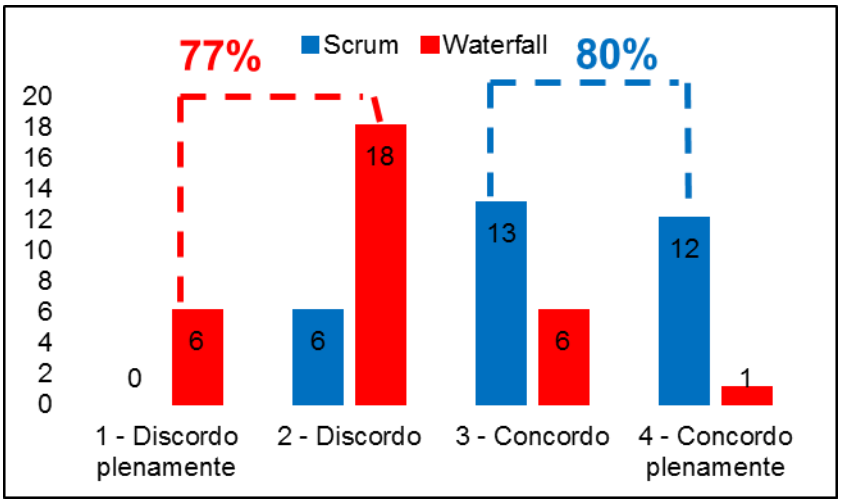
Fonte: Elaboração própria (2018).

Fica claro neste gráfico (14) que o mito persiste na visão das pessoas que usam o *Scrum*. Segundo Sutherland e Schwaber (2007), o *Scrum* não impede uma documentação detalhada, o que o método afirma é que deve ser realizada somente a documentação suficiente para atender o projeto e o cliente, ou seja, se é necessária uma documentação mais detalhada deve-se colocar no *product backlog*. Devido a esse mito que ainda persiste e continua impactando a visão de utilização do *Scrum, no* gráfico, observa-se que no *Scrum* 74% das pessoas das pessoas têm a percepção de que a documentação é importante, contra 87% do *Waterfall*, além claro de nenhuma pessoa colocar o *Waterfall* na classificação de 'discordo plenamente'.

Para continuar o tema tem-se a questão seguinte "não tem documentação excessiva" nos projetos.

Sabe-se que dependendo da necessidade do projeto é necessária uma documentação, mas segundo Elssamadisy (2007) a documentação na metodologia Waterfall é exagerada. Observando o gráfico 15, os respondentes também acreditam que é produzida uma documentação excessiva. Colocando uma lupa na quantidade de discordo observa-se que 77 % das pessoas que acreditam que existe documentação em excesso, ou seja, tem-se 24 pessoas de 31 respondentes que confirmam a teoria do Elssamadisy (2007)

Gráfico 15. Não tem documentação excessiva

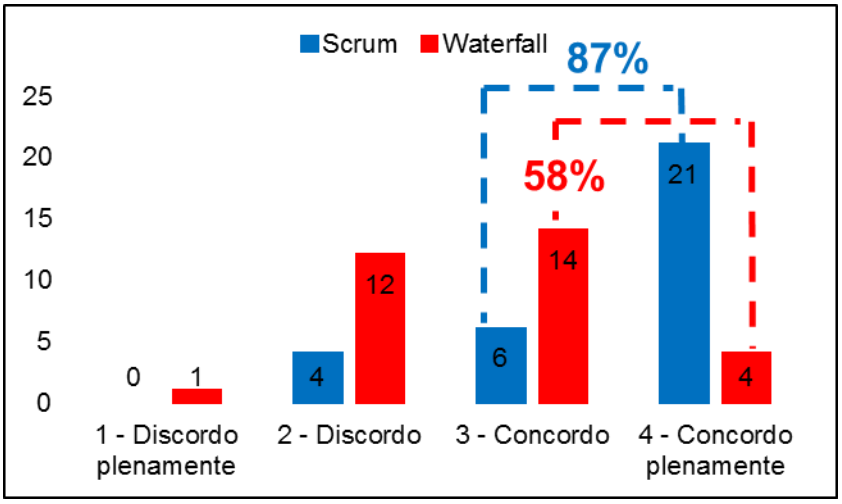


Fonte: Elaboração própria (2018).

Vale ressaltar, que apesar do número do gráfico (15) ser positivo para o método *Scrum*, não ter excesso não significa anular a documentação, mas sim, fazer o necessário.

Por fim, uma das promessas do *Scrum* é fazer todas as reduções e manter a qualidade, com o intuído de ter essa resposta de uma forma objetiva a questão seguinte é “tem qualidade na entrega”.

Gráfico 16. Tem qualidade na entrega



Fonte: Elaboração própria (2018).

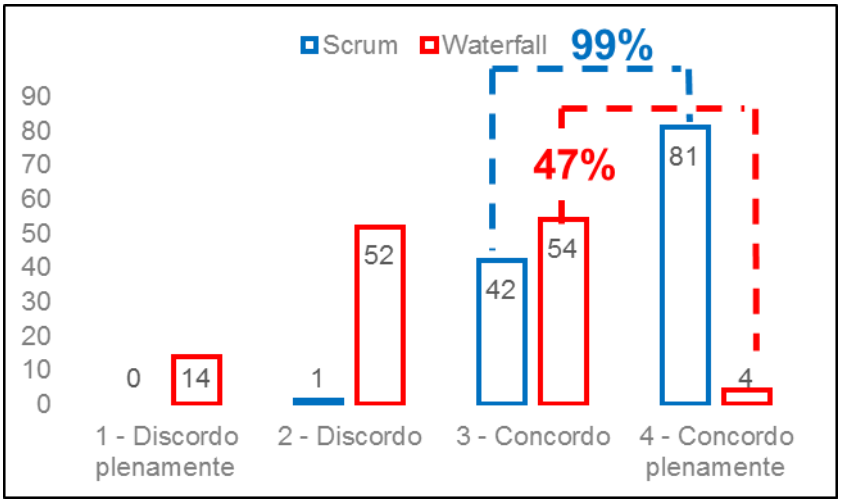
Esta é uma preocupação de todas as organizações ao implantar o *Scrum*. “Será que teremos a mesma qualidade?” No gráfico 16 a resposta é bem conclusiva. Tem-se 87% dos respondentes indicando que concordam que o *Scrum* tem qualidade na entrega, sendo que em sua grande maioria concordam plenamente e nenhuma pessoa discorda plenamente. Já o método *Waterfall* tem uma *performance* diferente, com 58% dos respondentes acreditando que não tem qualidade na entrega. Isto se deve à falha de comunicação entre as etapas e à baixa participação do cliente no período de desenvolvimento.

Com o resultado desse gráfico pode-se reafirmar os dados que Leffingwell e Smits (2005) apresentam: o *Scrum* entrega em poucas semanas um produto com qualidade e sem desperdícios.

5.3.4 Satisfação do Cliente

Os projetos, esforços, dedicação sempre deveriam estar focados em nossos clientes. Coutinho e Oliveira (2017) fazem diversos alertas sobre a qualidade que hoje é entregue aos clientes nos projetos de desenvolvimento de *software*. Tendo isto em vista, são analisados os próximos resultados.

Gráfico 17. Existe Satisfação do Cliente



Fonte: Elaboração própria (2018).

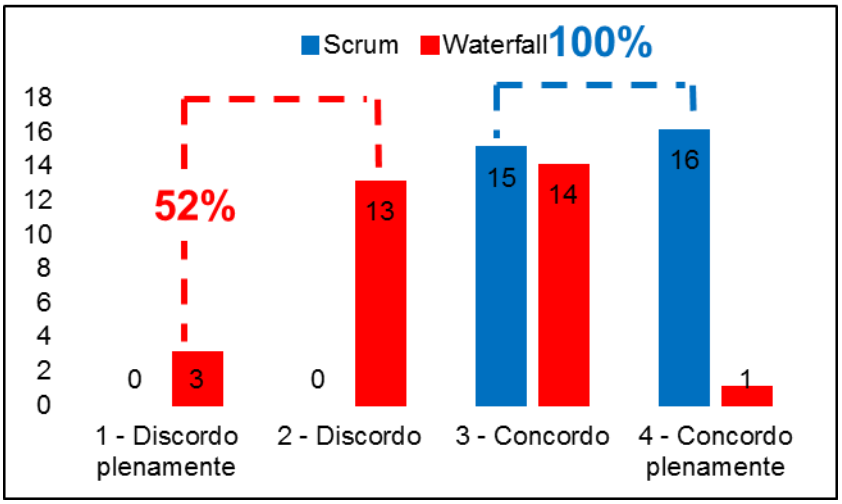
De acordo com o gráfico 17, tem-se 99% das respostas afirmando que existe satisfação dos clientes utilizando o *Scrum*. Isto significa que todas as questões

relacionadas a esta subcategoria estão em 100% e uma em 99%. Reafirmando o que Tabaka (2006), Elssamadisy (2006) e outros autores dizem sobre a importância do que é esperado pelo cliente, a colaboração durante a construção, entender suas necessidades e ser flexível frente às mudanças.

O que acontece com o *Waterfall* é uma característica contrária e, por isto, apenas 47% concordam com essa afirmação.

Para se obter mais detalhes sobre essa subcategoria, são avaliadas as respostas à questão “possível visualizar o *software* em funcionamento”.

Gráfico 18. Possível visualizar o *software* em funcionamento



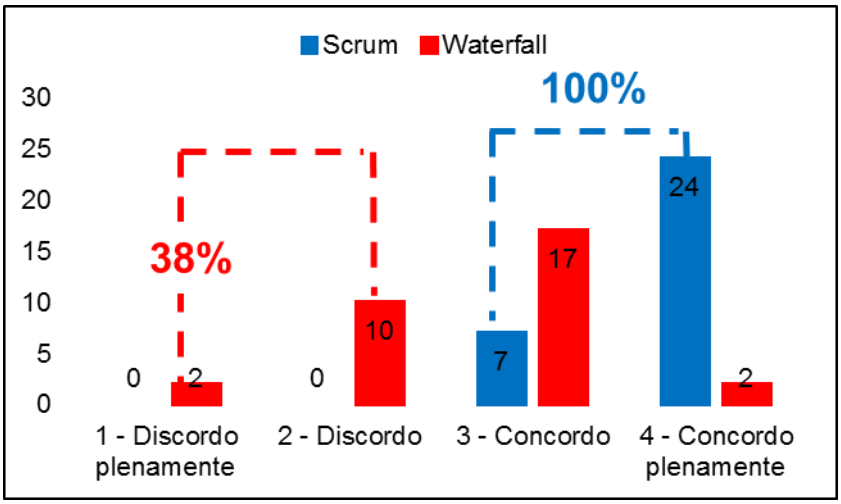
Fonte: Elaboração própria (2018).

No mundo de projetos existe uma grande expectativa dos clientes em visualizar o produto. De acordo com Kniberg (2007), o método *Scrum* reconhece essa ansiedade e busca atendê-la. Para Sutherland e Schwaber (2007), isto deve acontecer quando são seguidas as técnicas de priorização do *Product Backlog* em pequenas porções colocadas em produção e que geram valor agregado para o negócio. No gráfico 18 é possível visualizar que 100 % dos respondentes concordam com essa afirmação.

Segundo Capodieci (2014) e Boehm (1988) *Waterfall* é um método tradicional e sua origem é industrial, e por isto segue a definições e suas fases entregando o produto final somente no fim do ciclo. O que leva a 52% de respondentes que discordam que é possível visualizar o *software* em funcionamento nessa metodologia.

Seguindo esta análise tem-se a validação “Existe colaboração com o cliente”.

Gráfico 19. Existe colaboração com o cliente



Fonte: Elaboração própria (2018).

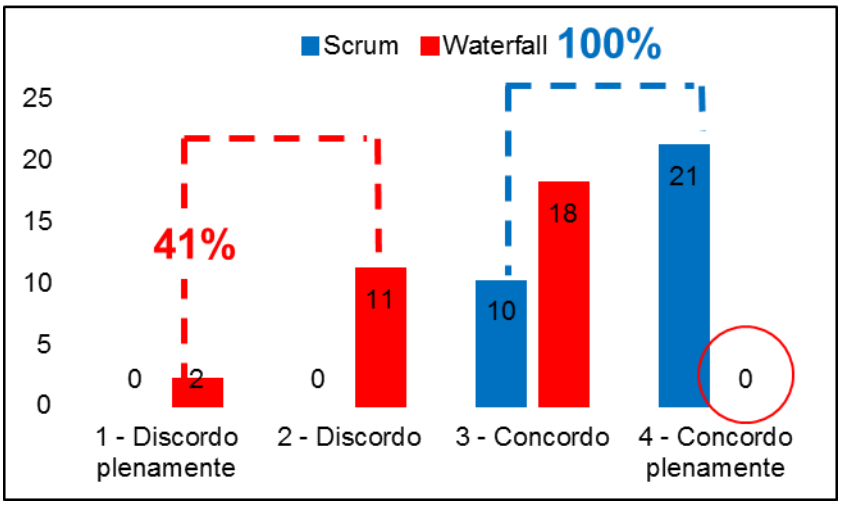
O gráfico19 mostra que 100 % dos respondentes concordam que existe colaboração com o cliente utilizando *Scrum*. Percentual indiscutível e com uma tendência enorme em concordo plenamente.

Isto acontece devido ao envolvimento ativo do cliente na figura do PO em todo o momento, desde o planejamento inicial, na proximidade do dia-a-dia, nas reuniões de revisão de trabalho entregue, fazendo com que o PO faça realmente parte da equipe.

Waterfall teve 12 pessoas que não acreditam que existe essa colaboração com o cliente. Acontece essa situação porque o envolvimento do cliente acontece apenas no início, com a definições de requisitos, e no fim, para visualizar a entrega total.

Para finalizar esta subcategoria é a analisada a questão “os resultados atendem os pedidos dos clientes”.

Gráfico 20 - Os resultados atendem os pedidos do cliente



Fonte: Elaboração própria (2018).

O último gráfico (20) volta a reafirmar a satisfação do cliente com o *Scrum* em 100% de afirmação.

Contudo, tem-se uma informação não positiva para o método *Waterfall*, diferente dos gráficos anteriores dessa categoria. Não houve nenhum respondentee que concordasse plenamente com a afirmação de que os resultados atendem os pedidos do cliente.

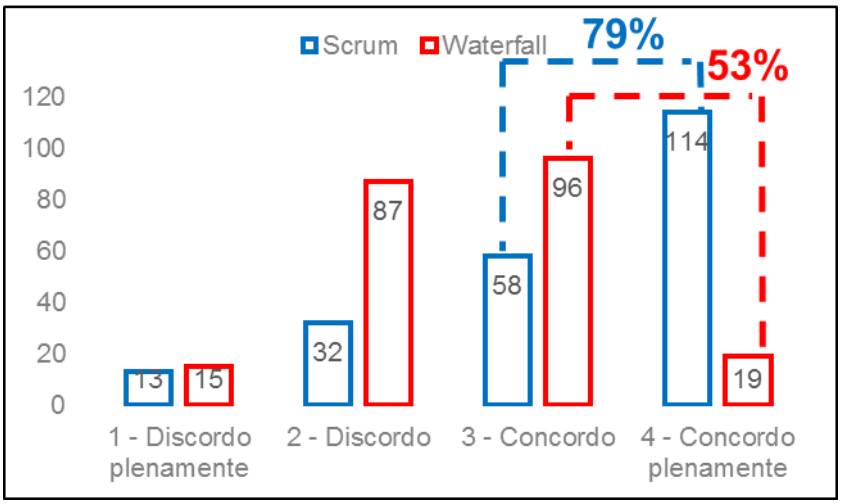
Segundo Adzic (2009), os métodos tradicionais têm dificuldades na comunicação de projetos tecnológicos que é um fator essencial para condução de projetos e atender as necessidades dos clientes.

5.3.5 Colaboradores

Em projetos tecnológicos os colaboradores deveriam estar sempre em primeiro lugar em qualquer execução. O motivo é simples, são eles que fazem acontecer e sem uma equipe não se tem nenhuma entrega em desenvolvimento de *software*.

Manter a equipe motivada, dar-lhe autonomia, permitir que sejam auto organizáveis, auto gerenciados, multidisciplinares, e com respeito, são alguns dos pontos cruciais para se ter uma equipe de alta *performance*, segundo Tabaka (2006) Mahadevan (2015) Sutherland e Schwaber (2007). Porém, é importante saber respeitar as quatro fases na construção de uma equipe, conforme explica Kaner (2007).

Gráfico 21 - Apoia os colaboradores



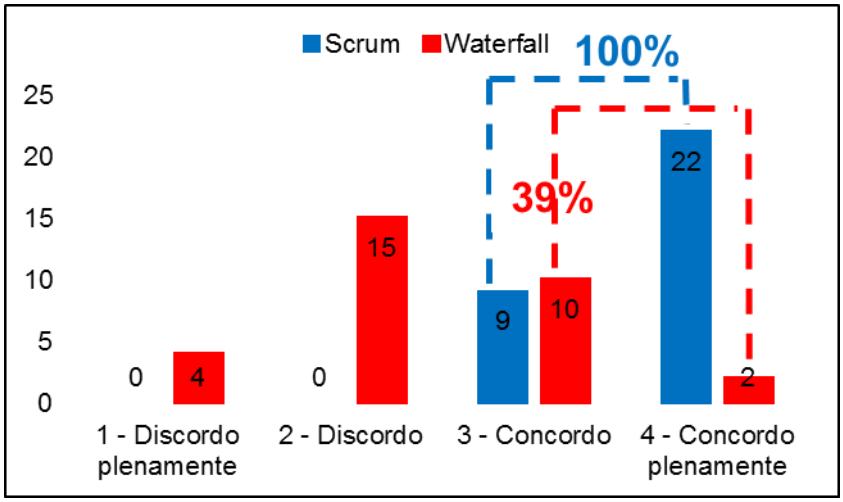
Fonte: Elaboração própria (2018).

No Gráfico 21 tem-se uma boa proporção indicando que a metodologia *Scrum* apoia as pessoas. Cerca de 79% dos respondentes acreditam que apoia as pessoas, e 21% discorda.

Quando se fala de *Waterfall*, 53% concordam e 47% discordam. Um dos motivadores desse resultado deve estar relacionado ao gerenciamento e tomada de decisão centralizada, além da baixa visibilidade da equipe.

Continuando a análise, olham-se as questões separadas e identificam-se os principais erros e acertos dos métodos. A primeira é se “existe visibilidade do grupo”.

Gráfico 22. Existe visibilidade do grupo



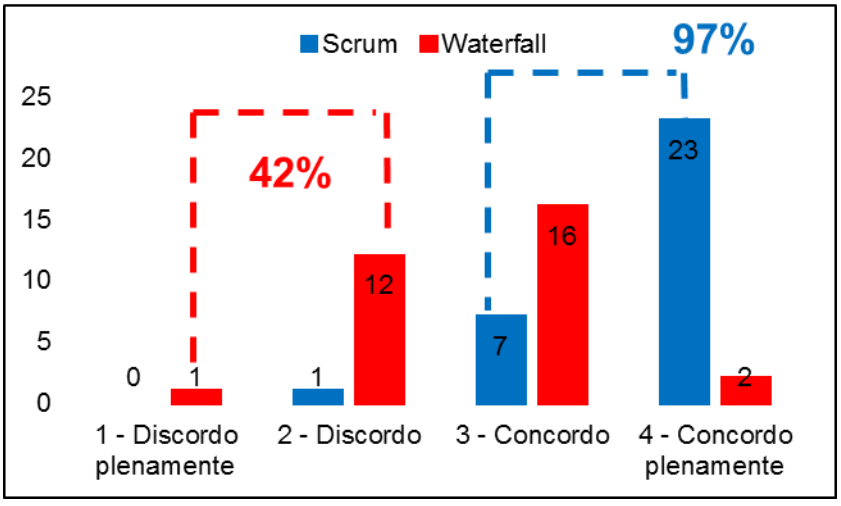
Fonte: Elaboração própria (2018).

No gráfico 22 tem-se uma visão individual que está diferente da subcategoria. O motivo que indica esse resultado está relacionado às referências de Tabaka (2006) que não se pode considerar a equipe como apenas um grupo de trabalho, mas sim, como pessoas que se completam, promovem comunicação, *feedback* e respeito. Dado isto, tem-se um resultado em que todos os respondentes concordam que o *Scrum* busca a visibilidade do grupo. Além claro, da Review em que, segundo Mahadevan (2015), o grupo tem uma oportunidade de apresentar o trabalho para os clientes e envolvidos do projeto.

Diferente do que acontece no *Waterfall* que 19 pessoas discordam. Talvez devido à centralização dos gerentes de projetos, tanto em contato com os clientes e divulgação de resultados.

Para continuar o raciocínio, segue-se com questão sobre se os métodos “têm comunicação”.

Gráfico 23. Tem comunicação



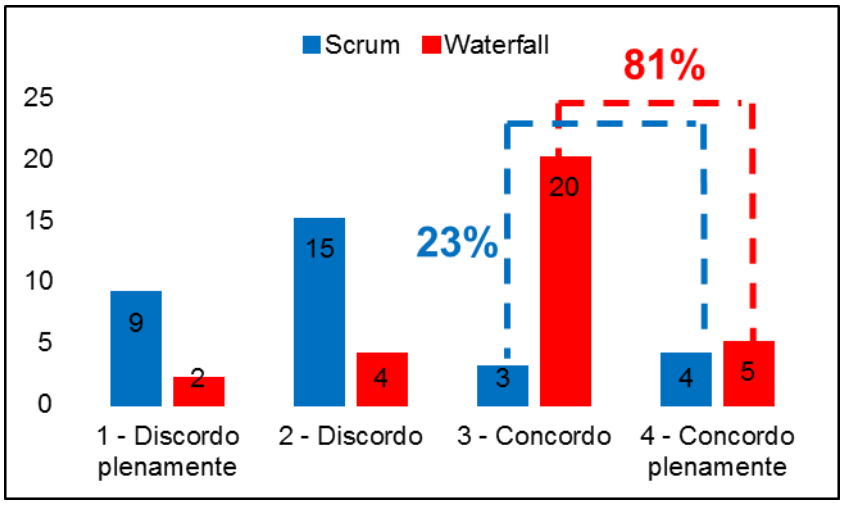
Fonte: Elaboração própria (2018).

O gráfico 23 é importante para falar sobre o apoio às pessoas, mas também, para reforçar uma promessa, os métodos Ágil sempre falam sobre melhorias na comunicação.

Sabe-se que sem comunicação não existe um desenvolvimento ou apoio às equipes e, olhando o gráfico 23 observa-se que 97% concordam que o *Scrum* tem comunicação.

Os resultados individuais até o momento somente explicam a boa *performance* do *Scrum*, por isto, vamos analisar o quadro “Mudança de cultura na empresa” e “Baixo turn over”, que deveriam indicar algumas preocupações.

Gráfico 24. Baixa mudança de cultura na empresa



Fonte: Elaboração própria (2018).

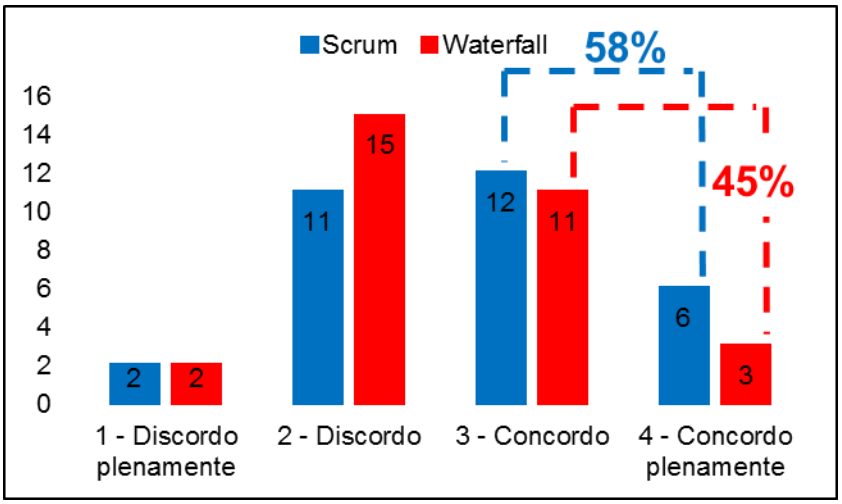
Mudança na cultura da empresa é um tema que impacta diretamente todos os colaboradores. Segundo *Poppendieck* e *Poppendieck* (2003), é necessária uma mudança cultural na organização para implantação de métodos Ágil.

Este é um verdadeiro risco de insucesso na condução de utilização do *Scrum*. De acordo com Leffingwell e Smith (2005) é de grande importância o envolvimento dos executivos, até porque, será trabalhoso, complexo e o envolvimento deverá ser de toda organização.

O gráfico 24 deixa clara a preocupação dos respondentes, devido ao baixo resultado do *Scrum* com apenas 23%.

Totalmente contrário do *Waterfall* que tem 81% de aceitação e não olham a mudança de cultura como um risco.

Gráfico 25. Baixo Turnover



Fonte: Elaboração própria (2018).

No gráfico 25 é possível visualizar ainda uma insatisfação em relação entre as mudanças de pessoas. Tanto *Scrum* com 58% e *Waterfall* 45% têm um baixo desempenho neste tópico.

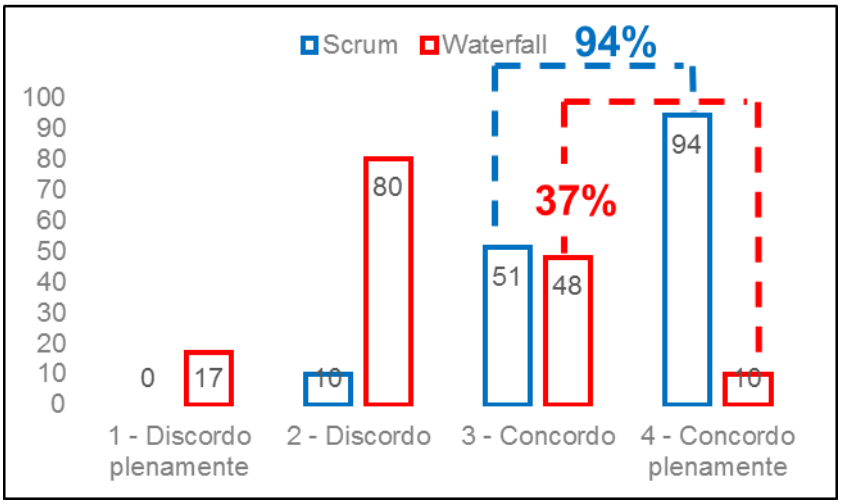
De acordo com Beck (2000), em métodos Ágil acredita-se que com estímulo, visibilidade, *feedback* e respeito, o *turnover* não deveria ser alto. Porém, mesmo que o resultado deste gráfico seja um pouco superior ao *Waterfall*, vê-se um item que deve ser trabalhado: existe espaço para melhorias no método *Scrum*.

5.3.6 Resultados

A subcategoria de resultados um dos temas que “brilham os olhos” das organizações. Que devem reconhecer as necessidades anteriores para chegar nos resultados esperados. E claro, nenhuma empresa vai se submeter a mudanças se não tiver resultados significativos. Afinal, o mercado espera isso com suas exigências e competitividade.

Dado isto, vamos avaliar as respostas dessa subcategoria.

Gráfico 26. Traz resultados para empresa



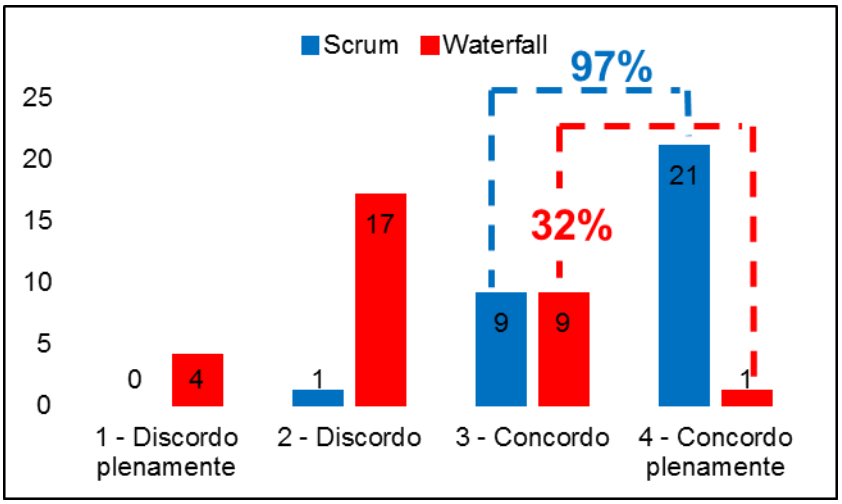
Fonte: Elaboração própria (2018).

De acordo com Leffingwell e Smits (2005), com a implantação do *Scrum* tem-se bons resultados espalhados pela empresa naturalmente. Também Sutherland e Schwaber (2007) afirmam que com mudanças simples na forma de conduzir projetos tecnológicos será possível se obter resultados positivos. O gráfico 26 demonstra que 94% dos respondentes concordam com a visão destes autores. Isso traz uma segurança para as empresas buscarem essa mudança.

Observando o resultado do *Waterfall* tem-se que apenas 37% dos respondentes concordam. Este valor é preocupante para qualquer empresa no ramo tecnológico, e o motivo é simples, estão fazendo um investimento em métodos que perdem tempo, aumentam custos e não trazem resultados.

Para indicar alguns pontos de atuação vamos avaliar os resultados de forma individual. Começando pela questão “a empresa fica mais competitiva”.

Gráfico 27. A empresa fica mais competitiva



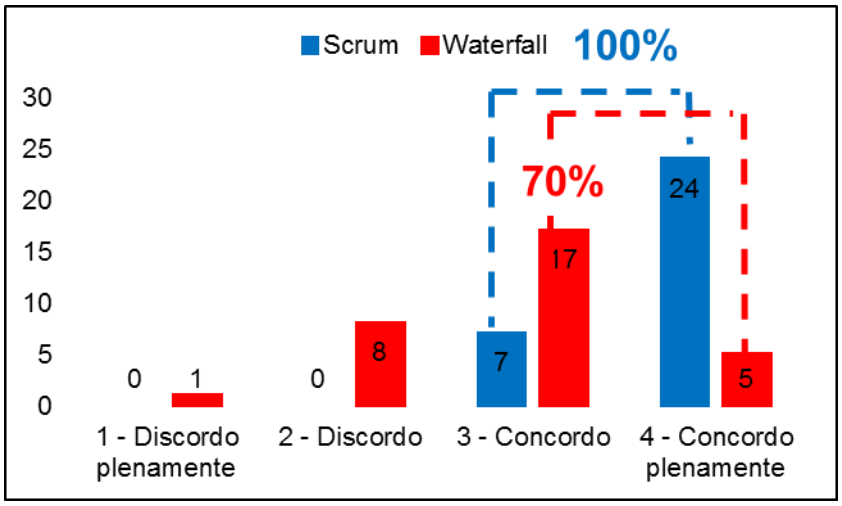
Fonte: Elaboração própria (2018).

O gráfico 27 mostra que esses resultados para tornar a empresa mais competitiva são possíveis. Tem-se 97% dos entrevistados concordando que *Scrum* ajuda a empresa a ficar mais competitiva.

Diferente do *Waterfall* que tem 32% que estão de acordo com a utilização desse método para ser mais competitivo.

Seguindo com a análise vamos abrir a questão se a metodologia “busca gerar valor para empresa”.

Gráfico 28. Busca gerar valor para empresa



Fonte: Elaboração própria (2018).

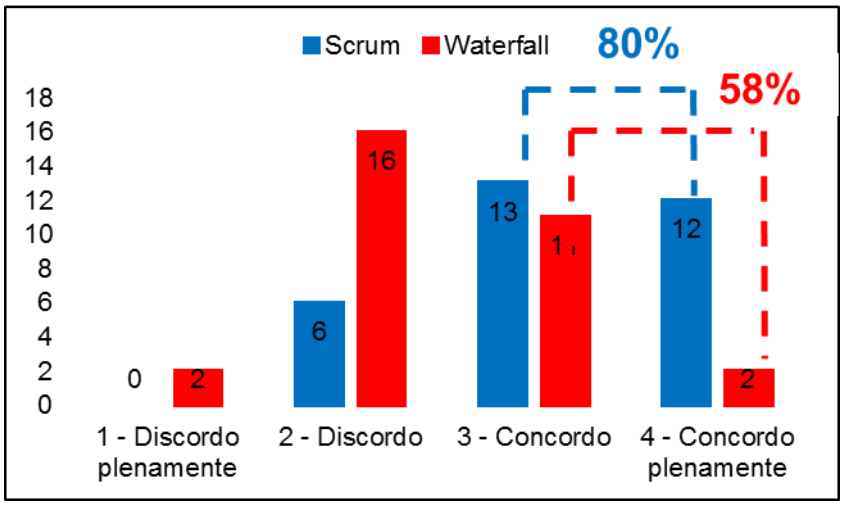
Schwaber (2004) reforça que um dos objetos do *Scrum* é gerar valor para o cliente e a organização.

No gráfico 28, existe um dado em grande destaque, e diz que 100% dos respondentes concordam que o *Scrum* gera valor para a empresa. Além de uma boa parte indicar que concordam plenamente.

Outra informação, é que *Waterfall* teve uma *performance* boa com 70% dos respondentes. É visível que nos dois métodos existe o interesse de gerar valor, porém a eficiência e eficácia na execução são diferentes.

Para finalizar, vamos seguir com os dados da questão de “baixo risco na implantação”.

Gráfico 29. Baixo risco de implantação



Fonte: Elaboração própria (2018).

Pode-se averiguar no gráfico 29 que o método *Scrum* tem um resultado positivo com 25 (80%) respondentes concordando com o baixo risco de implantação, e isso está relacionando às implantações em pequenas proporções, a equipe pronta para se adaptar a mudanças, e o cliente podendo visualizar o *software* em funcionamento.

Gostaríamos de reforçar a atenção para o fato de que 58% de respondentes acreditam que *Waterfall* tem um alto risco na implantação de seus projetos. Sendo assim, temos uma informação importante relacionada a uma falsa ilusão de gestão de seus projetos nas empresas. Segundo Mahadevan (2015), os diversos controles disponíveis nos métodos tradicionais levam aos executivos uma tranquilidade que não deveria existir, daí, o baixo resultado na pesquisa esboçado no gráfico 29.

5.3.7 Pesquisa descritiva

A última informação colocada na pesquisa foi uma pergunta direta e relacionada com o tema desta dissertação. A pergunta foi “É possível reduzir custos e tempo em projetos tecnológicos utilizando a metodologia *Scrum* ao invés da *Waterfall*? A resposta deveria ser descritiva para conseguir trazer um pouco das experiências das pessoas.

Observando o resultado tem-se três pessoas que não colocam a resposta, todos os outros indicando que “SIM”, ou seja, todos acreditam que a utilização do *Scrum* reduz custos e o tempo de projetos com desenvolvimento de *software*.

Além disto, deixaram algumas palavras ou pequenas frases que simbolizam a resposta deles com seguintes temas:

- Redução do retrabalho;
- Responsiva as necessidades;
- O que era uma necessidade ontem, talvez não seja hoje;
- Flexibilidade;
- Busca produtividade da equipe;
- Fazer entregas aos poucos;
- Comprometimento da equipe.

Os pontos acima reforçam os itens discutidos nas subcategorias descritas pelos respondentes.

Um resultado importante para esta dissertação e com respostas claras sobre tema. Agora vamos utilizar esses resultados para indicar uma conclusão e refinar os dados finais.

6 CONSIDERAÇÕES FINAIS

A condução de projetos tecnológicos e as organizações realmente devem ter uma preocupação inerente ao conduzir seus recursos e ao como trabalhar, considerando que se tem à frente um mercado competitivo e que não perdoa a ineficiência e o desperdício.

A metodologia Scrum, Ágil proposta assegura a redução de tempo e custos em projetos tecnológicos. *Scrum* é um método que impulsiona a cooperação entre as pessoas, autogerenciamento, entrega de resultados, e qualidade na entrega. Seguindo as práticas indicadas do *Scrum* as empresas terão velocidade nas respostas ao mercado e mudanças.

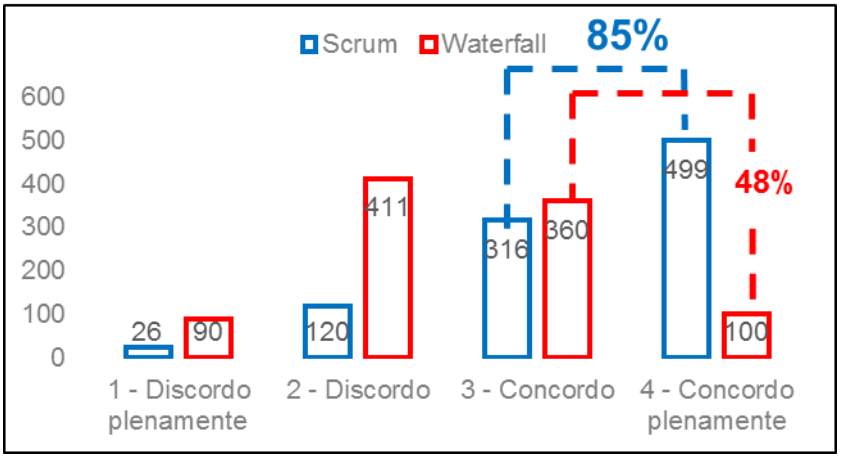
Waterfall ainda é um método muito presente nas empresas e qualquer alteração nesse sentido terá um impacto diretamente na cultura da organização, o que representa uma das principais barreiras na implantação do *Scrum* nas organizações. Os executivos têm que estar presentes nesta mudança e direcionando sua equipe, principalmente mostrando que é possível bons resultados para organização e para a equipe. Caso não tenham cuidado com este item podem tornar a experiência frustrante. Além disso, a rotatividade de pessoas nos dois métodos continua com um percentual com margem de melhoria, e por isto, é necessária atenção desse item na utilização em qualquer dos métodos.

Sabemos que o *Scrum* é um método leve e simples de entender, ainda que complicado para dominar. Os benefícios são excelentes, motivados na utilização do *Scrum*, pois não é um método caro de ser implantado, é simples de ser entendido, reduz os custos dos projetos, melhora a performance do tempo de entrega, aumenta a satisfação dos clientes, motiva os colaboradores da empresa e potencializa os resultados.

Sua mudança pode ser realizada aos poucos, *squad* por *squad* e com apenas oito pessoas para iniciar, no entanto, é importante colocar nesse piloto, pessoas engajadas como o tema e com a metodologia, além claro, de especialistas do produto esperado pela equipe. Com esta abordagem inicial pequena e simples se tem uma disseminação natural do tema, logo, diversas frentes terão interesse em saber que método é este que traz tanto resultado.

O gráfico 30 consolida todas respostas realizadas pelos respondentes, ajudando na conclusão, buscando a referência na pesquisa.

Gráfico 30. Resultado final consolidado



Fonte: Elaboração própria (2018).

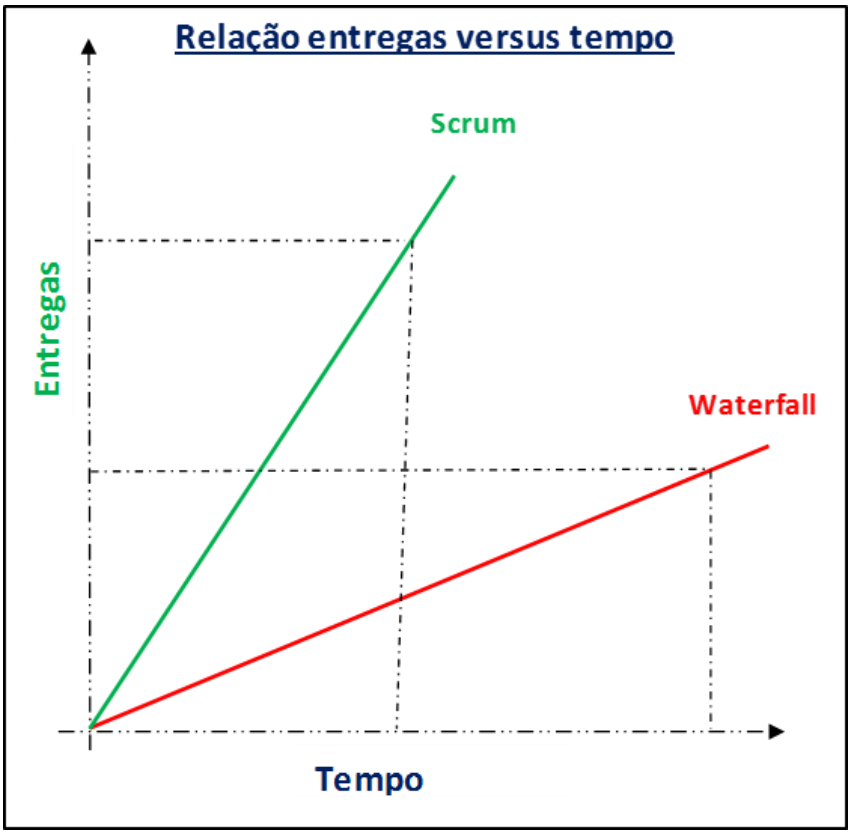
O resultado do gráfico 30 traz segurança para ajudar na resposta do tema desta dissertação. Principalmente quando se analisa o quadro na visão do *Scrum* que tem uma aceitação consolidada de 85% dos respondentes indicando que essa metodologia é a mais adequada para conduzir projetos tecnológicos.

Quando se realizam a mesma análise para a metodologia *Waterfall* tem-se uma aceitação de 48% das respostas, indicando que não é o melhor método. Esse resultado está relacionado diretamente à baixa comunicação, controles ineficientes, resistência a mudanças, foco no cliente, decisões centralizadas, desperdícios e riscos na implantação.

O *Scrum* é uma metodologia que busca constantemente o valor agregado com qualidade e investimento na moral das equipes, minimizando riscos e focando na satisfação do cliente.

Por fim, o Quadro 10 ajuda a visualizar uma das conclusões desta dissertação.

Quadro 10 - Relação entregas vs tempo



Fonte: Elaboração própria (2018).

Scrum tem uma maior quantidade de entregas em um menor intervalo de tempo quando comparado às metodologias tradicionais em projetos tecnológicos, saindo do mesmo ponto de partida e com uma quantidade menor de recursos.

Seguem os melhores resultados do Scrum:

- Eliminar desperdício 94%;
- Negociação do contrato 97%;
- Redução de tempo 98%;
- Velocidade na entrega 97%;
- Entrega no prazo 94%;
- Satisfação do cliente 99%;
- Software em funcionamento 100%;
- Colocaboração com o cliente 100%;
- Resultados atendem os pedidos 100%;

- Visibilidade do grupo 100%;
- Comunicação 97%;
- Resultados para empresa 94%;
- Valor agregado 100%.

Podemos concluir que Scrum é a arte de fazer o dobro do trabalho na metade do tempo, conforme explica Sutherland (2014) em seu livro e está demonstrado no Quadro 10.

REFERÊNCIAS

ADAMS, J. **The Agile with Scrum user's guide**: a compilation of thoughts and practices. 2006. Disponível em: https://danielettinger.files.wordpress.com/2010/12/the-agile-with-scrum-users-guide.pdf. Acesso em 23 dez. 2018.

ADZIC, G. **Bridging the communication**. London: Neuri Limited, 2009.

APPELO, J. Management 3.0: Leading Agile developers. Grupo de Tecnologia Pearson, 2010.

BARDIN L. **Análise de conteúdo**. 6 ed. São Paulo: Almedina, 2011

BASAK, B. G. Cost management in an imperfect world: bridging the gap between theory and practice. **ICEC Cost Management Journal**, 2006. Disponível em: http://www.icoste.org/SloveniaPlenaryLectures/icecFinal00185.pdf. Acesso em 8 fev. 2019.

BECK, K. **Extreme programming explained**: embrace change. 2 ed. Massachusetts: Addison Wesley Professional, 2000.

BOEHM, B. W. **A spiral model of software development and enhancement**, 1988.

CAPODIECI, A. **A case study to enable and monitor real IT companies migrating from Waterfall to Agile**. Lecce: Switzerland Springer International Publishing, 2014.

CARVALHO, L. **Metodologia qualitativa em pesquisa sobre formação de professores**: narração de uma experiência. Campos dos Goytacazes: Instituto Superior de Educação do ISECENSA, 2007.

CLELAND, D. I. **Project management: strategic design and implementation**. 2. ed. Chicago: McGraw-Hill, 1994.

COCKBURN, A. **Agile software development**: the cooperative game. Salt Lake: Cockburn & Highsmith, 2001.

COHN, M. **Agile estimating and planning**. New York: Addison Wesley, 2005.

COHN, M. **User stories applied**: for agile software development. Boston: Addison Wesley, 2004.

COUTINHO, E da S.; OLIVEIRA, V. R. de. Gestão de custos em projetos: desafios para uma indústria. São Bernardo do Campo: **Revista de Administração IMED**. v. 7, n. 2, 2017.

DAVIES, R.; SEDLEY, L.; **Prepared exclusively Agile Coaching**. North Carolina Dallas: The Pragmatic Bookshelf, 2009.

DeMARCO, T. de. The choir and the team. **The Dorset House Quarterly**. Nov.4, 1995, p. 4.

DERBY, E. **Agile retrospectives**: making good teams great. Texas: The Pragmatic Programmers, 2006.

ELSSAMADISY, A. **Patterns of agile practice adoption**. New York: InfoQ.com., 2007.

FERNANDES, J. H. C. Qual a prática do desenvolvimento do software? **Cienc. Cult**. vol.55 no.2 São Paulo Apr./June 2003.

FREITAS, H. M. R.; CUNHA JUNIOR, M. V. M., MOSCAROLA, J. Aplicação de sistemas de software para auxílio na análise de conteúdo. São Paulo, **RAUSP** v. 32, n. 3, jul./set. 1997. p. 97-109.

HARPER, Jacque. **Why the name "scrum"?** 2012. Disponível em: https://agilecoachjacque.wordpress.com/2012/12/04/why-the-name-scrum/. Acesso em 20 fev. 2019.

HARRIS, M. L., Collins, R. W., Hevner, A. R. Control of flexible software development under uncertainty. **Information Systems Research**. Linthicum: 2009.

HRON, M.; OBWEGESER, N. Scrum in practice: an overview of Scrum adaptations. Waianae: Hawaii International Conference on System Sciences, 2018.

KANER, S. **Facilitator's guide to participatory decision-making**. San Francisco: Jossey-Bass, 2007.

KNIBERG, H. **Kanban vs** Scrum. Stockholm: Crisp, 2009.

KNIBERG, H. Scrum and XP from the Trenches. Stockholm: Crisp, 2007.

KORHONEN, K. **Migrating defect management from waterfall to agile software development in a large-scale multi-site organization: a case study**. Tampere: Springer-Verlag, 2009.

LADAS, C. **Scrumban**. Seattle: Modus Cooperandi Press, 2008.

LEFFINGWELL, D.; SMITS, H. A CIO's playbook for adopting the scrum method. Boulder: **Rally Software Development Corp**, 2005: 8-26.

LEVISON, M. **Choosing a sprint length** – shorter trumps longer**.** Agile pain relief consulting. 2013. Disponível em: https://agilepainrelief.com/notesfromatooluser/2013/07/choosing-sprint-length-shorter-trumps-longer.html#.XHVCT8BKjcs. Acesso em: 25 fev. 2019.

MAHADEVAN, L. **Running on hybrid**: control changes when introducing an agile methodology in a traditional Waterfall system development environment. California: CAIS, 2015.

MANTELLINI, G. A Revisão e a análise como metodologias científicas conteudísticas. Campinas: **Revista científica internacional**, v.2, n. 1, p. 1-13, 2009

MARQUES JÚNIOR, L. J.; PLONSKI, G. A. Gestão de projetos em empresas no Brasil: abordagem "tamanho único"? **Gest. Prod.,** São Carlos, v. 18, n. 1, p. 1-12, 2011.

MARQUES, J. R. **Tipos de mudança organizacional mais praticados nas empresas.** 2017. Disponível em https://www.jrmcoaching.com.br/blog/tipos-de-mudanca-organizacional-mais-praticados-nas-empresas/. Acesso em 10 fev. 2019.

MOUNTAIN GOAT SOFTWARE. **Planning Poker**. Disponível em: https://www.mountaingoatsoftware.com/agile/planning-poker. Acesso em: 15 fev. 2019.

NORTON, D.; KAPLAN, R.; **The strategy-focused organization**. Nova York: Harvard Business School Publishing, 2001.

POMPERMAYER, C. B.; LIMA, J. E. P. **Gestão de custos**.) Sao Paulo: Atlas, 2003 (Coleção Gestão Empresarial

POPPENDIECK, M.; POPPENDIECK, T. **Lean software development:** an Agile toolkit. Boston: Addison Wesley, 2003.

QUICKSCRUM. **Managing quickscrum**. Disponível em: https://www.quickscrum.com/ScrumGuide/174/sg-Sprint-Planning. Acesso em 10 fev. 2019.

RUPING, A. **Agile documentation:** a pattern guide to producing lightweight documents for software projects. Chichester: John Wiley & Sons Ltd, 2003.

SCHWABER, K. **Agile project management with scrum**. Washington: Microsoft Press, 2004.

SCHWABER, K. **Guia do scrum.** . Boston: Scrum Alliance, 2009.

SCHWABER, K. **Scrum: it isn't scrum if…**. Boston: Agile Alliance, 2007.

SUTHERLAND, J. **Scrum**: a arte de fazer o dobro do trabalho na metade do tempo. São Paulo: LeYa, 2014.

SUTHERLAND, J; SCHWABER, K. **The scrum papers**: nuts, bolts, and origins of an Agile process. Washington: PatientKeeper, 2007.

TABAKA, J. Collaboration explained: facilitation skills for software project leaders. Crawfordsville: Addison Wesley Professional, 2006.

TAKEUCHI, H. **The new new product development game.** Boston: Harvard Business School Publishing, 1986.

WILLIAMS, L. **Pair programming illuminated.** Boston: Addison Wesley, 2002.