AGILE CULTURE: MAKING PROCESSES FASTER

FRAMEWORK SCRUM

ANDRE LUIZ DIAS RIBEIRO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

2 FRAMEWORK SCRUM

2.1 Origem do Scrum

Já se vão mais de vinte anos desde a primeira divulgação do SCRUM em um evento nos Estados Unidos e mais de quinze desde a criação da Scrum Alliance...

Como vocês já devem ter observado, muita coisa evoluiu e diversos conceitos se consolidaram no mercado como práticas essenciais para se obter sucesso nos projetos, tais como: as entregas parciais, times auto-organizados, aceitação de mudanças, *timebox*, entre outras.

O SCRUM foi apresentado pela primeira vez em um evento anual da ACM (*Association for Computing Machinery*), o OOPSLA (*Object-Oriented Programming Systems, Languages & Applications*) em 1995, por Ken Schwaber, e desde então passou a divulgar e publicar seus conceitos no mercado, culminando com a criação da Scrum Alliance em 2002, inicialmente com a Agile Alliance de Mike Cohn e Jeff Sutherland.

Em 2009, por algumas divergências internas, Ken Schawber criou o Scrum.org, e o SCRUM passou a ter duas referências oficiais no mercado. Nessa mesma época foi criado o Scum Guide® que é o documento oficial sobre o framework e que é mantido uniforme para ambas organizações, sendo a base conceitual e fundamental do SCRUM.

No Brasil, as práticas ágeis com SCRUM começaram a entrar no mercado a partir de 2009, mas foi a partir de 2015 que realmente tomou força e tornou-se cada vez mais popular, incorporando inclusive outros conceitos ágeis que não pertencem ao framework SCRUM, como o design thinking, gestão visual, kanban e canvas, que estão cada vez mais populares no ambiente de projetos.

2.1.2 Certificações

Existem diversas linhas de certificação no mercado, mas as mais conhecidas são as fornecidas pelas duas organizações oficiais: o Scrum.org e a ScrumAllian. Vamos saber um pouco sobre essas certificações.

Para começar, ambas têm como fundamento básico para as avaliações o *Scrum Guide* ® e alguns conceitos adicionais que são complementados por algumas referências bibliográficas sugeridas e pelo *Nexus Guide*® que trata da escalabilidade do SCRUM para projetos maiores.

O programa de certificação do Scrum.org tem as certificações para cada um dos principais papéis do SCRUM, consistindo em provas avaliativas e sem a necessidade de realização de treinamentos específicos.

- **PSF** - Professional Scrum Foundations: Atinja seus objetivos com um treinamento prático de Scrum que estabelece os fundamentos para cada profissional de Scrum.

- **PSD** – Professional Scrum Developer: Aprenda a aplicar práticas de engenharia ágeis e modernas no Scrum para aumentar a sua capacidade de entrega de softwares

- **PSM** – Professional Scrum Master: Você é responsável pela implementação e uso efetivos do Scrum? Junte-se à nossa aula avançada de Professional Scrum Master.

- **PSPO** – Professional Scrum Product Owner: Um curso desenvolvido para todos aqueles responsáveis por maximizar o valor entregue por seus produtos e serviços de software.

O programa de certificação do Scrumalliance também possui as certificações para cada um dos principais papéis do SCRUM, porém amplia essa gama de certificações para outros níveis de experiência e conhecimento, conforme ilustrado na figura Certificações.

Figura 2.1 – Certificações
Fonte: Scrum Alliance (2017), adaptado por FIAP (2018)

Nas certificações Scrumalliance é obrigatório que o treinamento para cada certificação básica de função (*ProductOwner, ScrumMaster e Developer*), seja realizada por um CST (*Certified Scrum Trainning*) que é o profissional habilitado a realizar esses treinamentos, e fazer uma avaliação escrita ao final deles.

Para se tornar um CST, o profissional deve ser certificado CSP (*Certified Scrum Professional*) que exige cinco anos de experiência com as práticas SCRUM.

Importante: Embora existam outras certificações disponíveis no mercado, essas duas são as mais reconhecidas e que fazem diferença em um processo de seleção.

Se você quiser mais detalhes sobre as certificações, basta acessar os sites:

www.scrum.org

www.scrumalliance.org

2.2 O que é SCRUM?

Você deve estar se perguntando: o que é SCRUM? Uma sigla? Não. Scrum é o nome de uma das mais conhecidas jogadas do Rugby, que tem como principal característica uma formação ordenada dos times que pode ser visualizada na figura Jogada de Rugby.

Figura 2.2 – Jogada de Rugby
Fonte: Pandlr (2017)

Nessa jogada, oito jogadores de cada time disputam a reposição de bola atuando em conjunto com o mesmo objetivo, sendo que se um deles falhar, todos falham. Esse trabalho em equipe e esse conceito de comprometimento e união é que torna o framework um diferencial. O sucesso ou o fracasso é de todos, e não de uma pessoa, um herói.

O SCRUM é um framework ágil que permite manter o foco na entrega do maior valor de negócio, no menor tempo possível.

E por que é um framework? Porque permite que se façam adaptações ao seu uso, desde que respeitada a sua estrutura de organização e sequência.

Em SCRUM, as necessidades do negócio determinam as prioridades do desenvolvimento de um sistema e devem ser feitas pelo cliente para obter a cumplicidade no trabalho. Além disso, as equipes se auto-organizam para definir a melhor maneira de entregar as funcionalidades de maior prioridade num tempo que seja adequado ao time e ao cliente.

As entregas ocorrem a cada duas ou quatro semanas, quando todos podem ver o produto realmente funcionando, permitindo que seja avaliado, tendo um rápido

feedback e decidindo se ele deve ser liberado ou precisa continuar a ser aprimorado por mais um ciclo, chamado de *sprint*.

Como você pode observar, vários paradigmas devem ser quebrados para se atingir o que é preconizado.

Portanto, não se começa a utilizar SCRUM de um dia para o outro. É preciso apresentar aos envolvidos como ele funciona, suas cerimônias e suas restrições ao time, cliente e ao gerente de projeto, pois todos devem mudar de atitude frente à nova forma de executar um projeto. E isso não é uma tarefa trivial. Além de um treinamento inicial, é preciso executar projetos pequenos como experimentações para que todos conheçam e se acostumem com a nova abordagem e vejam o resultado real de sua utilização.

> "Quando um projeto está atrasado, adicionar pessoas ao projeto servirá apenas para atrasá-lo ainda mais. Devemos considerar o tempo que perdemos em gestão e comunicação quando temos pessoas demais trabalhando em um projeto.
>
> Ao calcular o tempo de desenvolvimento de qualquer coisa, temos que dobrá-lo. O programador precisa de "tempo para pensar" além do "tempo para programar".
>
> "The Mythical Man-Month: Essays on Software Engineering". Frederick Brooks, 1st Edition, Addison-Wesley 1975.

A afirmação de Brooks (1975) reitera outro fundamento do SCRUM que **é manter inalterada a equipe e o tempo de um ciclo**, pois só assim é possível aprender com os erros e melhorar no ciclo seguinte, sem postergar prazos e gerar esforços adicionais que acabam por desmotivar e prejudicar o desempenho do time.

2.3 Características do framework

O principal objetivo do SCRUM é permitir o controle das atividades de construção que utilizam técnicas ágeis para gerar informações gerenciais sobre o andamento e sobre a evolução do projeto.

Em uma visão genérica inicial, podemos considerar as seguintes características principais do framework SCRUM:

- Os requisitos formam o chamado ***Product Backlog,*** que é a lista de tudo que precisa ser feito no projeto.
- O produto evolui em ciclos curtos e imutáveis em tempo chamados de ***sprints***, cujos conceitos de sustentação são o desenvolvimento iterativo-incremental e o *timebox*.
- Durante a execução do *sprint*, cada equipe utiliza a técnica de desenvolvimento que melhor se adapta ao cenário do projeto.
- As Equipes que se auto-organizam, ou seja, todos decidem o que e como fazer.
- Há um processo de melhoria contínua: ao final de cada *sprint* executado, há uma série de coisas boas que devem ser repetidas e coisas ruins que não devem tornar a se repetir - lições aprendidas.

Vamos detalhar duas dessas características que fundamentam o conceito que deve ser seguido para a elaboração e execução dos *sprints* e de todas as cerimônias do SCRUM.

Importante: Cerimônias são as reuniões que acontecem durante o ciclo SCRUM de desenvolvimento, como as reuniões de planejamento, reuniões diárias, de revisões e de lições aprendidas que veremos à frente

2.4 Desenvolvimento iterativo e incremental

É o modelo desenvolvido e caracterizado pela construção em vários ciclos, entregas constantes de produtos, permitindo a identificação de problemas nas fases iniciais do projeto e a tomada de ações visando corrigir o curso do desenvolvimento no tempo adequado e de maneira eficiente (Krutchen, 2000).

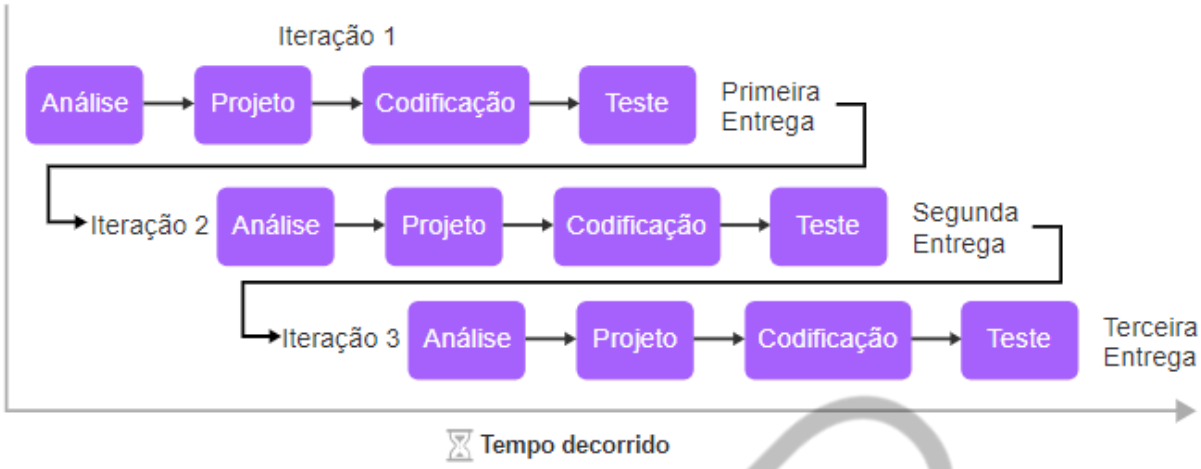

Figura 2.3 – Desenvolvimento iterativo
Fonte: Pressman (2005), adaptado por FIAP (2018)

Como você pode observar na figura Desenvolvimento iterativo, no modelo iterativo, a primeira iteração reúne os requisitos principais do produto que são validados pelo cliente. Um plano é desenvolvido para a próxima interação, visando satisfazer as necessidades do cliente. Esse processo é repetido após cada iteração até que o produto completo seja produzido. O modelo iterativo objetiva a elaboração de um produto a cada iteração (Pressman, 2005).

O modelo iterativo permite flexibilidade em ambientes de projeto onde existem indefinições de requisitos iniciais e a determinação do foco nos pontos mais críticos do projeto.

Outras vantagens desse modelo são a detecção, o mais cedo possível, de inconsistências entre os requisitos e a implementação, a produção de resultados tangíveis a cada iteração e a identificação de melhorias contínuas no processo (Krutchen, 2000).

2.5 Desenvolvimento *Timebox*

O desenvolvimento *timebox* é uma prática que auxilia a manter o foco nas principais características do produto, evidencia o senso de restrição de tempo à equipe do projeto e reduz o tempo de construção.

O ponto central dessa prática é enquadrar os principais requisitos do projeto ao tempo disponível, enquanto os demais requisitos são incorporados em outros *timeboxes* com menor prioridade. A prioridade desses requisitos é definida

conjuntamente pelo cliente e pela equipe do projeto, permitindo a redução do tempo de desenvolvimento.

A aplicação dessa prática requer a utilização conjunta com a prática de prototipação, além de necessitar do envolvimento significativo do usuário final e de revisões constantes da equipe do projeto. Após a construção, o sistema é avaliado pelo cliente, podendo ser aceito completamente ou retornar para ajustes de funcionalidades ou de qualidade, conforme ilustrado na Figura Prototipando o produto.

Figura 2.4 – Prototipando o produto
Fonte: Pressman (2005)

A definição dos *timeboxes* deve ser parte da fase de planejamento, em que o cliente participa diretamente da definição e limitação dos principais requisitos que são atendidos dentro de cada *timebox*, bem como da validação dos protótipos e das tomadas de decisões.

As principais condições de sucesso dessa prática é que a data final estabelecida para cada *timebox* não seja alterada, o cliente esteja de acordo com os requisitos definidos e a limitação rígida do escopo seja atendida. Qualquer mudança nessas condições deve ser tratada como um novo *timebox* para não comprometer os prazos estabelecidos.

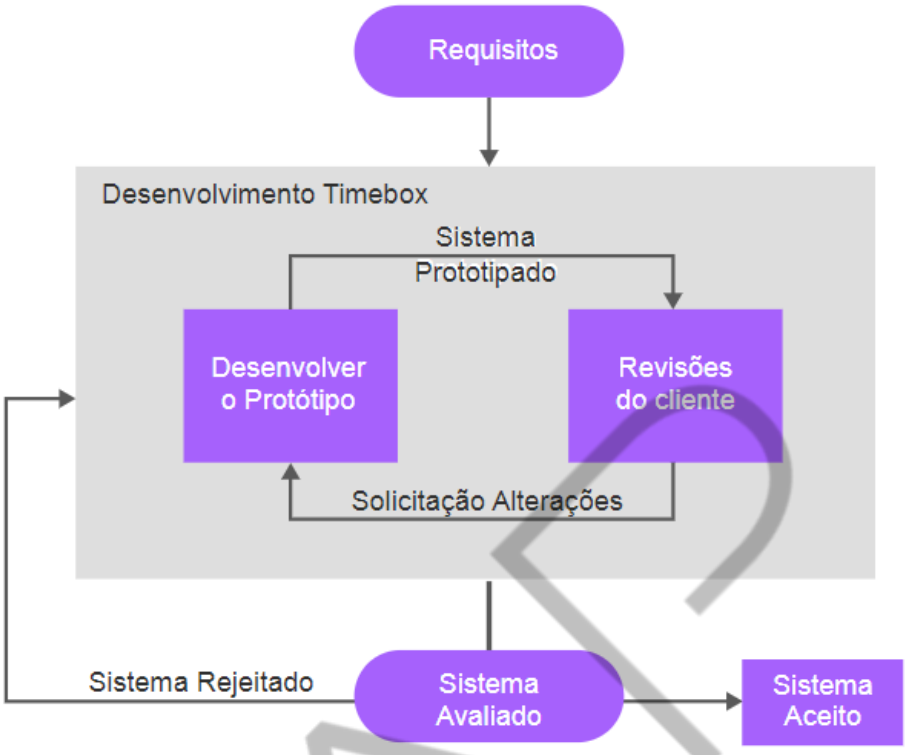

Figura 2.5 – Desenvolvimento *Timebox*
Fonte: McConnell, 1996, adaptado por FIAP (2018)

O primeiro método criado para a gestão de projetos ágeis foi o Extreme Project Management (XPM) com o objetivo de gerenciar os projetos.

2.6 Pilares do SCRUM

No Scrum Guide® estão definidos os três pilares básicos do SCRUM que sustentam o framework e trazem os verdadeiros ganhos aos projetos e são ilustrados na figura, são eles:

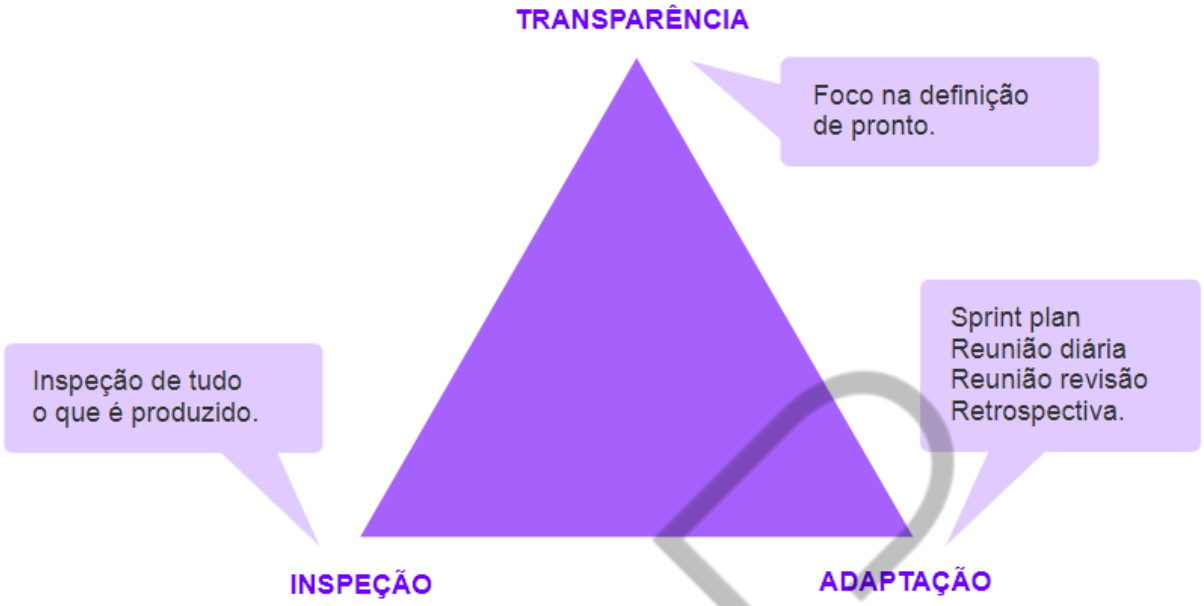

Figura 2.6 - Pilares Scrum
Fonte: Scrum Guide (2016) adaptado por FIAP (2018)

A transparência refere-se a deixar claro para todos qual é a situação real das atividades, alinhando expectativas e realizando feedback constante com o cliente, permitindo a verificação da evolução do projeto.

A inspeção é um preceito de qualidade que deve ser observado sempre. Tudo que é produzido precisa ser inspecionado para garantir que está aderente ao que foi solicitado. Ágil não significa ser somente rápido, mas também fazer com qualidade e entregar com qualidade.

Aqui aplica-se o conceito de um faz e outro revisa para melhorar o resultado final.

O processo de adaptação é obtido por meio das diversas reuniões que ocorrem durante o ciclo Scrum, em que todos avaliam o desempenho atual, criticam, fazem adequações imediatas e/ou melhorias para o próximo ciclo.

2.6.1Valores do SCRUM

Na sua versão 2016, o Scrum Guide® incluiu o conceito de valores importantes para serem compreendidos por todo o time, com o objetivo de tornar claras algumas definições que conflitam com a visão tradicional dos projetos e traz à tona algumas quebras de paradigmas.

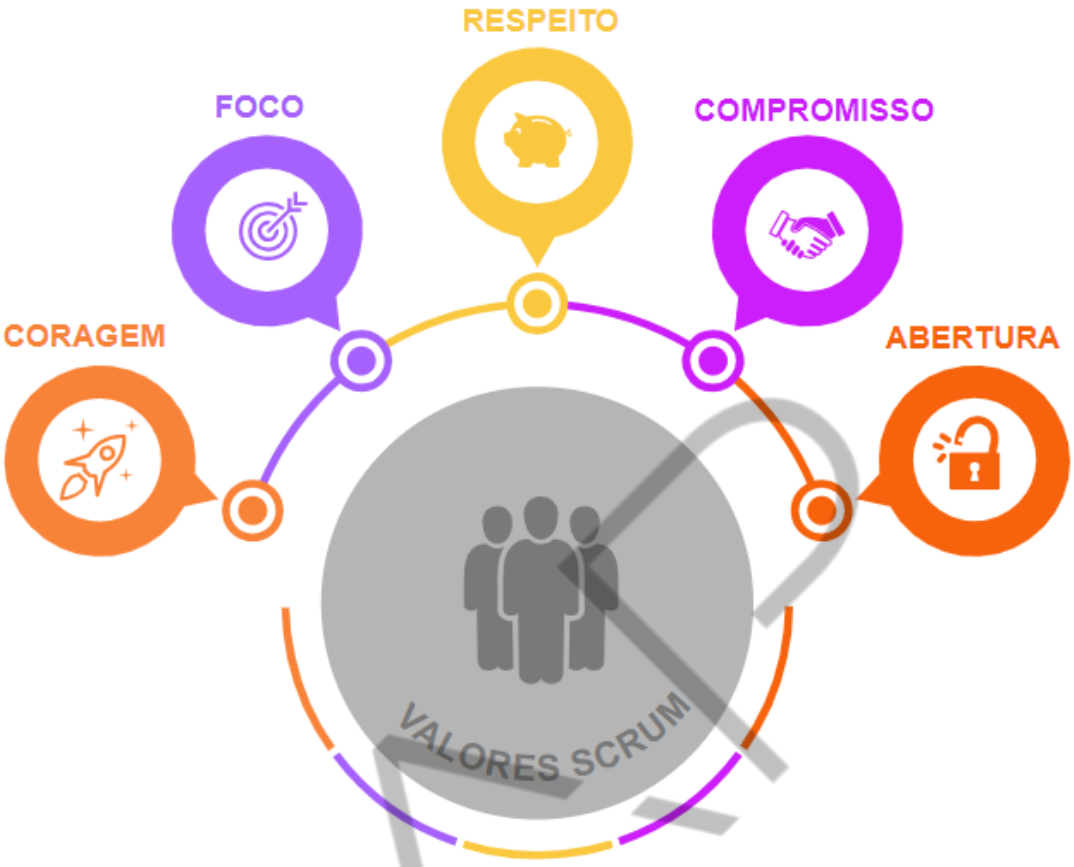

Figura 2.7 - Valores do Scrum
Fonte: Scrum Guide (2016), adaptado por FIAP (2018)

Vamos agora analisar, no quadro, cada um desses valores e sua interpretação correta em relação ao que o Scrum preconiza:

Valores	Visão tradicional	Visão scrum
Compromisso	Se comprometer só porque o seu chefe mandou.	Comprometimento com o time e com a meta do *sprint*.
Foco	Manter o cliente satisfeito.	Entregar o combinado no prazo combinado.
Abertura	Falar bem do que você faz	Destacar o sucesso de todos e alertar para os impedimentos
Respeito	Você como herói para resolver os problemas	Auxiliar o time a fazer melhor e não julgar os outros

Coragem	Culpar a tudo e a todos pelos problemas.	Aceitar opiniões dos outros, críticas e assumir seus erros.

Quadro 2.1 - Interpretação de valores do Scrum
Fonte: Adaptado de West (2016)

Segundo Dave West (2016), algumas ações podem ser tomadas para auxiliar os envolvidos a cultivarem esses valores e romperem com os preconceitos estabelecidos e melhorar o ambiente e o desempenho do time:

- Coloque os valores em uma parede e peça para cada membro da equipe escrever como eles estão sendo aplicados em seu dia a dia no trabalho.
- Adicione um “momento dos valores” em sua retrospectiva. Isso dará a todos uma oportunidade para inspecionar e adaptar os valores em sua equipe.
- Considere introduzir um prêmio ‘valores’ para estimular a prática e reconhecimento de todos envolvidos.

2.6.2 Papéis no SCRUM

No framework SCRUM os papéis são muito bem definidos e reduzidos, minimizando conflitos e aumentando o grau de comunicação entre todos os envolvidos. São três os papéis descritos pelo SCRUM:

2.6.2.1 Product Owner

O *Product Owner* é o representante do cliente no projeto e deve ter autonomia e autoridade para a tomada de decisões e validações necessárias no projeto.

- Define as funcionalidades do produto, mantendo o *Product Backlog* atualizado;
- Decide datas de lançamento e conteúdo dos *releases*;
- Responsável pela rentabilidade (ROI);
- Prioriza funcionalidades de acordo com o valor;
- Ajusta funcionalidades e prioridades;

- Apresenta ao time os requisitos necessários para a entrega do produto;
- Aceita ou rejeita o resultado dos trabalhos;
- Atua como facilitador quando há mais clientes envolvidos;
- Garante que especialistas estejam disponíveis para o time.

2.6.2.2 Scrum Master

O *Scrum Master* é o representante da gerência e atua como facilitador para o time, removendo impedimentos; e como elo do time com o *Product Owner*, protegendo o time para que esse não perca o foco na construção do produto.

- Representa a gerência para o projeto;
- Responsável pela aplicação dos valores e práticas do Scrum;
- Remove os impedimentos;
- Garante a plena função e produtividade da equipe;
- Garante a colaboração entre os envolvidos;
- Escudo para interferências externas;
- Responsável por garantir que o produto atenda às necessidades do cliente;
- Auxilia o *Product Owner* com o *Product Backlog*;
- Facilita as reuniões.

2.6.2.3 Time

O time é auto-organizado e autogerenciado e sua responsabilidade é a execução dos trabalhos acordados nos planejamentos com o *Product Owner* e o *Scrum Master*. Não existe diferença entre testador e arquiteto. Todos compartilham o mesmo título de *Scrum Team Member*.

As equipes SCRUM são pequenas e devem ter no máximo 9 (nove) *Team Members*, um *Scrum Master* e um *Product Owner*, sendo a palavra-chave para o sucesso o COMPROMETIMENTO de todos.

- O trabalho entregue ao cliente é gerado por um “Time Scrum” dedicado;
- Times Scrum são auto-organizados, multidisciplinares e comprometidos;
- Responsáveis por entregar o que foi acordado como meta do *Sprint*;
- Definir como as tarefas devem ser divididas para gerar o resultado esperado;
- Definir quem irá executar as tarefas e em que ordem elas são realizadas.

Agora que você entendeu o papel de cada um, vamos detalhar as principais responsabilidades de cada um no projeto SCRUM, segundo descrito no Scrum Guide®:

2.7 O Framework Scrum

Como já vimos, o *framework* define papéis claros com times reduzidos, descreve um conjunto de processos básicos que definem o que deve ser feito (o “como” fica a critério de cada time), um conjunto mínimo de artefatos que devem ser produzidos durante o ciclo de execução (cabendo ao time definir outros que julgue necessários), faz uso de ferramentas ágeis (outras que julgue necessárias para facilitar a execução e o acompanhamento dos trabalhos) e parte da premissa que os envolvidos já se desvencilharam de paradigmas tradicionais e já incorporaram conceitos como:

O cliente é uma parte presente e colaborativa no projeto, tornando “o projeto de todos” e não só do time, usando o conceito de líder servidor e produzindo com qualidade.

Figura 2.8 - Analogia a produzir com qualidade
Fonte: Banco de imagens Shutterstock (2017)

Esse conjunto de definições são detalhadas na figura Resumo do framework.

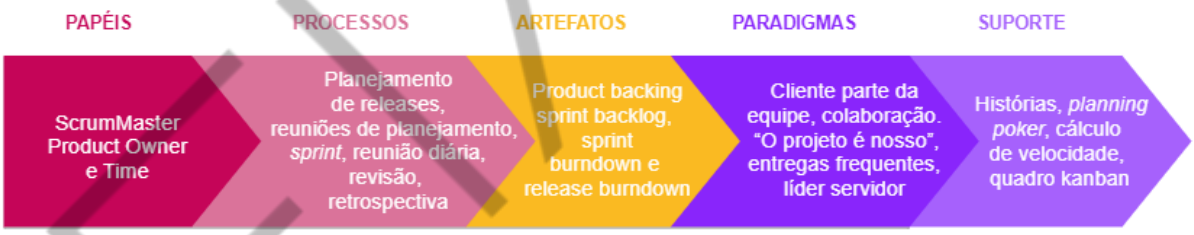

Figura 2.9 – Resumo do framework
Fonte: Elaborado pelo autor (2017), adaptado por FIAP (2018)

Vamos detalhar cada uma das etapas do *framework* adiante, mas primeiramente vamos alinhar as terminologias para facilitar o entendimento.

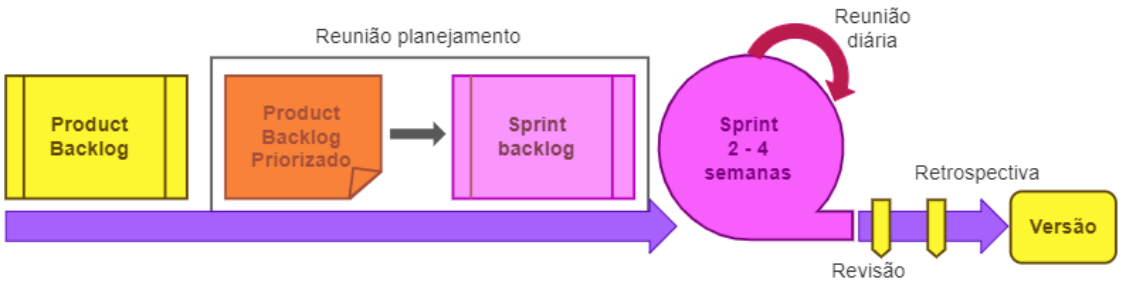

Figura 2.10 – O framework SCRUM
Fonte: Scrum Guide®, adaptado por FIAP (2018)

***Product backlog*:** É um documento elaborado pelo *Product Owner* a partir dos requisitos iniciais fornecidos pelo cliente e descreve tudo que deverá ser feito no projeto, sendo a referência para a execução do trabalho. As tarefas que o compõem devem ser organizadas em ordem de importância.

Reunião de planejamento: é uma reunião realizada antes de cada *Sprint*, com o objetivo de definir o que será feito no *Sprint*. Todos os membros do projeto participam.

***Sprint Backlog*:** é a lista detalhada de atividades que precisam ser executadas para gerar o produto esperado. Deve ser elaborada e estimada pelo time do projeto.

***Sprint*:** tem duração pré-determinada de 2 a 4 semanas, não pode ser estendido e nem reduzido. É no *sprint* que o produto é elaborado e entregue com qualidade.

Reunião diária: é uma reunião breve, em pé e no local de trabalho, com duração máxima de 15 minutos. O objetivo dessa reunião é acompanhar as atividades que estão sendo realizadas e melhorar a comunicação do time.

Reunião de revisão: é uma reunião em que o time apresenta o resultado do trabalho ao *Product Owner* e seus convidados. O objetivo é fazer uma avaliação e definir se o *sprint* será aceito. Eventuais apontamentos de erros e melhorias devem ser inseridos no *product backlog*.

Reunião de retrospectiva: é uma reunião na qual são discutidos os pontos positivos e negativos da execução de cada *sprint*, com o objetivo de melhorar o desempenho dos próximos *sprints*.

Versão: nem todo *sprint* gera um produto final a ser utilizado pelo cliente. Às vezes são necessários mais *sprints* para produzir uma versão fechada para utilização.

2.8 Iniciando o Projeto

Um projeto SCRUM começa a partir de um documento recebido pelo *Product Owner* denominado Visão do produto. Esse documento contém todas as necessidades que devem ser atendidas pelo projeto, juntando as informações fornecidas pelos principais clientes, usuários, patrocinadores, entre outros, de modo

a refletir como o produto deve ser ao final do projeto, conforme figura Documento Visão.

É importante ressaltar que não é o *Product Owner* que elabora a Visão. Esse documento é recebido por ele para ser transformado no *Product Backlog*.

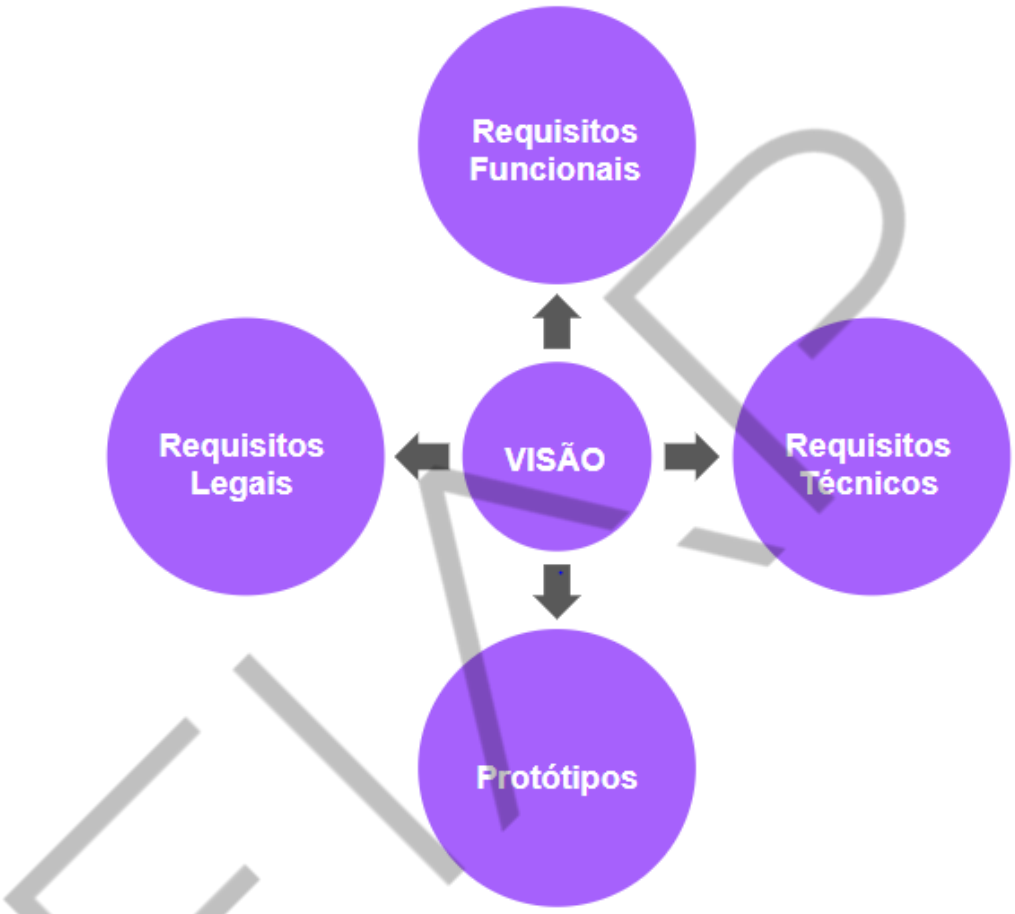

Figura 2.11 – Documento Visão
Fonte: Elaborado pelo autor (2017), adaptado por FIAP (2018)

2.8.1Criando o *Product Backlog*

Na definição mais simples, o *Product Backlog* é uma lista de todas as coisas que precisam ser feitas dentro do projeto, elaborada a partir da Visão do Produto. Ele não substitui os artefatos de especificação de requisitos tradicionais, pois apenas reflete o conteúdo macro do que será executado.

O proprietário do *Product Backlog* é o *Product Owner*, mas o o *Scrum Master* e o Time podem contribuir para se ter uma ampla e completa lista de tarefas para garantir o atendimento às necessidades do usuário.

Como o *Product Backlog* não é um documento detalhado, ele pode ser completado por outros documentos que trazem maior riqueza de detalhes e vão permitir melhor entendimento das funcionalidades. Alguns exemplos de artefatos adicionais que podemos ter são:

- Memorial descritivo;
- Um resumo das várias funções de usuário;
- Descrições de fluxo de trabalho;
- Diretrizes de interface de usuário;
- *Storyboards*;
- Requisitos técnicos;
- *Bugs* identificados previamente;
- Melhorias;
- Casos de uso ou protótipos de interface de usuário.

O *Product Backlog* não é um documento estático e vai existir durante todo o ciclo de vida do projeto, devendo ser continuamente atualizado pelo *Product Owner* para refletir tudo o que acontece com os requisitos durante o projeto.

À medida que novos itens são descobertos, eles são descritos e adicionados à lista. Os já existentes podem ser alterados ou eliminados, conforme o projeto evolui, portanto, o *Product Backlog* nunca está completo.

IMPORTANTE: O dono e responsável pelo *Product Backlog* é o *Product Owner*.

Segue um exemplo de *Product Backlog*:

#	Descrição
1	Como atendente, posso consultar os veículos disponíveis para locação
2	Como atendente, posso fazer locação de um veículo por km livre ou controlada
3	Como atendente, preciso aprovar o crédito do cliente antes de fazer a locação do veículo
4	Como atendente, posso cadastrar os clientes que vão fazer locação de veículos
5	Como atendente, posso fazer o pagamento da locação com cartão de crédito validando no sistema da operadora
6	Como atendente, posso fazer o pagamento da locação com cartão de crédito validando no sistema bancário
7	Como atendente, posso imprimir um comprovante da locação de veículos
8	Como atendente, posso receber a devolução do veículo locado na data prevista

Quadro 2.2 – Exemplo de um Product Backlog
Fonte: Elaborado pelo autor (2017)

Cartões de histórias não fazem parte do *framework* SCRUM, mas é a técnica mais utilizada para descrever o *Product Backlog*. Uma história representa uma pequena parte da funcionalidade a ser construída, ou seja, não é uma completa especificação de requisitos, mas deve ser suficiente para que o time possa estimar sua complexidade.

Tipicamente uma história é descrita seguindo o formato:

"COMO UM <<papel>>, EU GOSTARIA DE <<funcionalidade>> PARA <<benefício>>

Exemplo:

EU, COMO vendedor, GOSTARIA DE ver a lista de pedidos comissionados

PARA saber quanto irei receber de comissão.

Então, a montagem do *Product Backlog* comumente é uma lista de histórias representando cada item que precisa ser realizado no projeto.

Vamos agora dar um exemplo da construção de um *Product Backlog* para um projeto de um Sistema de Compra de Automóveis pela Internet a partir da Visão de Produto descrita abaixo.

Visão do Produto:

- Uma empresa tem apenas uma loja de vendas de carros e todo o processo é manual, sem nenhuma automação.
- A empresa deseja desenvolver um sistema para realizar a venda de automóveis pela Internet.
- O usuário deverá se cadastrar no site, procurar e detalhar os automóveis disponíveis, escolher o automóvel que deseja comprar e efetuar a compra por meio de Cartão de Crédito.
- O usuário também pode cadastrar o seu veículo para venda que alimenta o site para novos negócios.

Para criar as histórias, vamos identificar os papéis e as funções que cada um espera que o sistema possua:

Nesse cenário temos um site em que você pode fazer a compra de um automóvel e também pode publicar seu veículo para a venda. Tudo feito pelo próprio usuário. Então temos:

#	História
1	Eu, como usuário, posso me cadastrar no site para fazer compra e venda de veículos
2	Eu, como usuário, posso realizar buscas para encontrar o tipo de veículo que desejo
3	Eu, como usuário, posso ver os detalhes do veículo à venda

4	Eu, como usuário, posso fazer a compra de um veículo com cartão de crédito
5	Eu, como usuário, posso cadastrar meu veículo para a venda no site

Quadro 2.3 – Priorizando as Histórias
Fonte: Elaborado pelo autor (2017)

Observe que não precisamos entrar no detalhe para definir a história, mas ela é informação suficiente para ser entendida, mas não sabemos ainda os detalhes que serão adicionados depois no *Sprint Planning*.

2.9 Priorizando as histórias

Após listar todas as histórias, o *Product Owner* deve priorizá-las em ordem de importância com o objetivo de criar uma sequência que gere valor para o negócio e faça sentido como um *sprint* ou uma versão.

Existem diversas técnicas de priorização, mas vamos analisar três delas

2.9.1 Técnica MoSCOW

Esta é uma técnica muito utilizada para priorização de escopo, priorização de requisitos e para classificação de mudanças. O objetivo dessa técnica é classificar as funcionalidades com base no seu valor de negócio. O funcionamento da técnica é explicado no quadro:

#	Significado	-	O requisito é:
M	**Must have**	**Deve ter**	**Essencial para o negócio**
O	**-**	**-**	
S	**Should have**	**Deveria ter**	**Importante, mas não essencial**
C	**Could have**	**Poderia ter**	**Seria bom ter, mas não é importante**

O	-	-	
W	**Won´t have**	**Não precisa agora**	**Não gera valor para o negócio agora**

Quadro 2.4 - Técnica MoSCoW
Fonte: Elaborado pelo autor (2017)

Ao classificar os requisitos com MUST HAVE, por exemplo, você tem certeza que precisam ser feitos primeiro e devem estar no primeiro *sprint* e compor a primeira versão, pois esses terão prioridade em relação ao SHOULD HAVE e assim sucessivamente.

2.10 Matriz GUT

A matriz GUT é muito utilizada para priorização de tomada de decisões e para a tomada de ações corretivas e leva em consideração a gravidade da falta de um requisito, a urgência para se resolver o problema e a tendência, caso o problema não se resolva, conforme quadro.

Importância = G x U x T		
G	Gravidade	impacto do problema sobre coisas, pessoas, resultados, etc
U	Urgência	relação com tempo disponível ou necessário para resolver o problema
T	Tendência	avaliação da tendência de crescimento, redução ou desaparecimento do problema

Quadro 2.5 – Matriz GUT
Fonte: Elaborado pelo autor (2017)

Para cada uma dessas variáveis são atribuídos pesos de acordo com critérios pré-estabelecidos que vão direcionar a categorização, conforme quadro.

Pontos	Gravidade	Urgência	Tendência
5	Os prejuízos ou dificuldades são extremamente graves	É necessária uma ação imediata	Se nada for feito o agravamento será imediato
4	Muito grave	Com alguma urgência	Vai piorar a curto prazo
3	Grave	O mais cedo possível	Vai piorar a médio prazo

2	Pouco grave	Pode esperar um pouco	Vai piorar a longo prazo
1	Sem gravidade	Não tem pressa	Não vai piorar ou pode até melhorar

Quadro 2.6 – Pesos da matriz GUT
Fonte: Elaborado pelo autor (2017)

Após isso, multiplica-se os pesos de cada variável para se obter um número que, quanto maior, maior a prioridade, conforme mostrado na tabela.

Tabela 2.1 – Priorização com matriz GUT

#	Descrição	Prior.	G	U	T	total
18	Como supervisor da agência de locação, posso cadastrar, alterar consultar e excluir os veículos para locação	1	5	5	5	125
7	Como atendente, posso consultar os veículos disponíveis para locação	3	3	3	5	45
10	Como atendente, posso cadastrar os clientes que vão fazer locação de veículos	1	5	5	5	125
8	Como atendente, posso fazer locação de um veículo por km livre ou controlada.	2	5	4	3	60
9	Como atendente, preciso aprovar o crédito do cliente antes de fazer a locação de veículo.	4	1	3	3	9

Fonte: Elaborado pelo autor (2017)

2.11 Importância subjetiva

Nessa técnica, o *Product Owner* avalia onde a empresa se encontra e define o que é mais importante a ser feito a fim de atender às necessidades do negócio.

Por exemplo, imagine que uma locadora de automóveis não possui nenhum sistema informatizado. Quais seriam as primeiras funcionalidades que deveriam ser entregues? Provavelmente as funções de locação, cadastro de veículos e de clientes, pois, a locadora já poderia começar a utilizar essa parte enquanto é desenvolvido o processo de devolução de uma locação, como no quadro:

#	Descrição	Prioridade
18	Como supervisor da agência de locação, posso cadastrar, alterar, consultar e excluir os veículos para locação	1

7	Como atendente, posso consultar os veículos disponíveis para locação	2
10	Como atendente, posso cadastrar os clientes que vão fazer locação de veículos	3
8	Como atendente, posso fazer locação de um veículo por km livre ou controlada	4
9	Como atendente, preciso aprovar o crédito do cliente antes de fazer a locação do veículo	5
20	Como atendente, posso fazer reservas de locações de veículos para clientes cadastrados	5
14	Como atendente, posso receber a devolução do veículo locado na data prevista	6
15	Como atendente, posso receber a devolução do veículo locado com diferenças do previsto	7
19	Como supervisor da agência de locação, posso consultar as locações feitas no dia	8

Quadro 2.7 – Exemplo de um *Product Backlog* Priorizado
Fonte: Elaborado pelo autor (2017)

2.12 Estimativas

Após a priorização do *Product Backlog*, as histórias precisam ter seu tamanho estimado para que possa ser avaliado quantas histórias cabem num *Sprint* e se as histórias estão com um tamanho que seja possível estimar. Caso contrário, ela precisa ser quebrada em histórias menores para melhorar a compreensão do time.

O time é o responsável pelas estimativas, mas a presença do *Product Owner* é essencial para detalhar ou esclarecer cada história.

A técnica de estimativa mais utilizada em projetos *Scrum* é o *Planning Poker*. Essa técnica consiste em cartas com numeração seguindo a tabela de Fibonacci, vide figura Sequência de Fibonacci, e os membros do time avaliam um tamanho para cada um dos itens do *Product Backlog*.

Importante: tamanho de uma história não é em dias ou em horas, é normalmente relacionado à complexidade de cada história e também serve de comparação

entre elas. Essa numeração atribuída a cada item é chamada de pontos de complexidade.

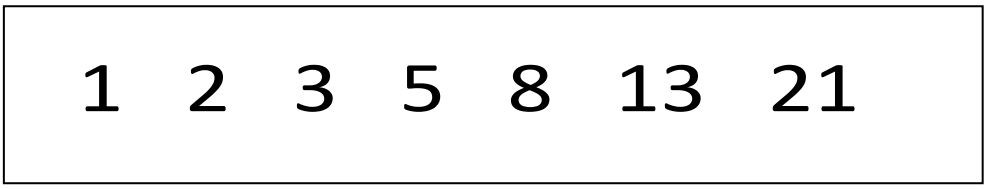

Figura 2.12 – Sequência de Fibonacci
Fonte: Elaborado pelo autor (2017)

O passo a passo para se jogar o *planning poker* é o seguinte:

- Cada membro do time tem um baralho com as cartas numeradas com a sequência de Fibonacci;
- Escolhe-se uma história do *Product Backlog*;
- O *Product Owner* detalha o que espera dessa história, mas não participa da estimativa;
- O time compreende e tira suas dúvidas;
- As cartas são apresentadas ao mesmo tempo pelo time;
- O time compara e discute suas estimativas;
- Em caso de divergências, a história é refeita e nova rodada acontece;
- O ciclo se repete até que todos tenham jogado a mesma carta, ou seja, o tamanho seja o mesmo.

Figura 2.13 - *Planning Poker*
Fonte: Banco de imagens Shutterstock (2017)

Para exemplificar, vamos tomar uma história do *Product backlog*:

História: Como atendente, eu gostaria de consultar os veículos disponíveis para a locação.

Esclarecimento do *Product Owner*: ao entrar no site, deve ser apresentada uma lista de todos os veículos disponíveis para a locação e o detalhe de cada veículo deve ser apresentado com a sua tarifa diária.

O time faz a 1ª jogada de cartas:

Tabela 2.2 – Exemplo *Planning poker*

Nomes	Rodada 1	Rodada 2	Rodada 3
Huguinho	3		
Zezinho	2		
Luizinho	5		

Fonte: Elaborado pelo autor (2017)

Como houve divergência, novos detalhes da história são fornecidos e o time debate o tamanho que cada um jogou. Nova rodada é feita.

Tabela 2.3 – Exemplo *Planning poker* (2)

Nomes	Rodada 1	Rodada 2	Rodada 3
Huguinho	3	3	
Zezinho	2	3	
Luizinho	5	5	

Fonte: Elaborado pelo autor (2017)

Como ainda há uma divergência, novas explanações são realizadas e, não pode haver “vencido pela maioria”, deve haver um consenso natural.

Tabela 2.4 – Exemplo *Planning poker* (3)

Nomes	Rodada 1	Rodada 2	Rodada 3
Huguinho	3	3	3
Zezinho	2	3	3
Luizinho	5	5	3

Fonte: Elaborado pelo autor (2017)

O final só é alcançado quanto todos chegam ao mesmo tamanho. No início pode haver uma demora natural, mas à medida que são feitas jogadas, o consenso ocorre de forma natural.

Com a evolução das estimativas, os tamanhos das histórias passam a ser comparados entre si e podem sofrer ajustes normais durante todo o ciclo do projeto.

DICA: O objetivo da estimativa é obter o comprometimento de todos com o prazo que será utilizado nos *sprints*.

Todo o *Product Backlog* deve ser estimado para permitir uma visão geral do tamanho do projeto, mas se for muito longo, pode ser feito por partes, começando pelas histórias mais prioritárias.

2.13 Planejamento de *Releases*

O planejamento de release é um plano de muito alto nível para determinar o tempo total do projeto, o ciclo de versões que serão geradas no projeto e definir quais funcionalidades serão construídas, sua prioridade e quando serão feitas. Um *release* pode ter um ou mais *sprints* para que possa atingir sua meta, conforme figura Hierarquia do release.

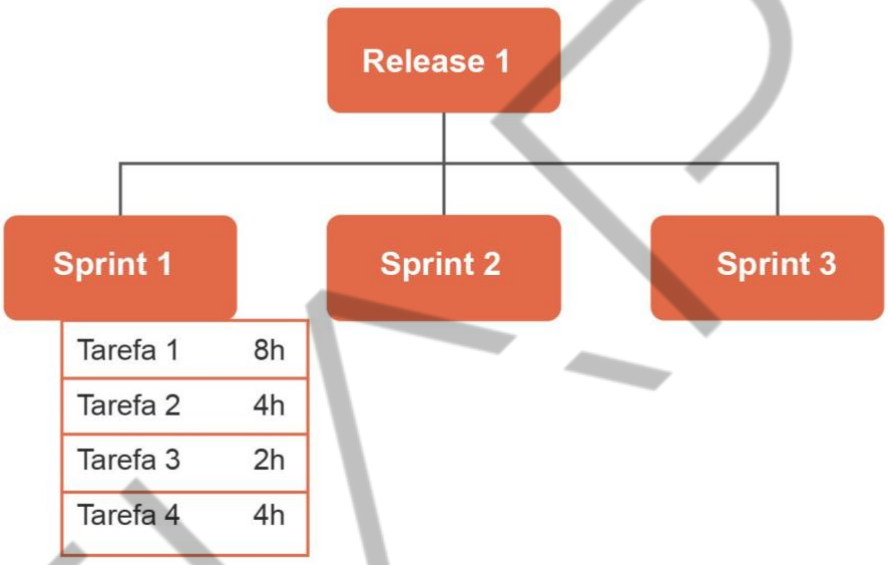

Figura 2.14 – Hierarquia do release
Fonte: Elaborado pelo autor (2017)

O seu objetivo principal é proporcionar o entendimento claro do projeto, sua estratégia de entrega e, com isso, a melhoria da comunicação entre todos os envolvidos.

Segundo o ScrumGuide®:

“O plano estabelece a meta do *release*, as maiores prioridades do *Product Backlog*, os principais riscos e as características gerais e funcionalidades que estarão contidas no *release*. Ele estabelece também uma data de entrega e custo prováveis que devem se manter. A organização pode então inspecionar o progresso e fazer mudanças nesse plano do *release* a cada *Sprint*. O planejamento do *release* é inteiramente opcional. Se um Time de *Scrum* iniciar o trabalho sem essa reunião, a ausência de seus artefatos aparecerá como um impedimento que deverá ser resolvido. ”

A primeira atividade do planejamento de um *release* é determinar sua META. Todo *release* deve ter uma meta e seu objetivo deve estar alinhado com as necessidades do negócio e ser definida pelo *Product Owner* com a ajuda do *Scrum Master*. A meta serve para guiar o time na realização das atividades e no cumprimento dos *sprints* planejados.

Para se criar um plano de *releases*, devemos seguir os seguintes passos, conforme figura “Sequência Planejamento de releases”:

- **Determinar a meta (objetivo) do *release*;**
- **Ter o *Product Backlog* priorizado e estimado.**
- **Definir o tamanho do *Sprint* (de 2 a 4 semanas).**
- **Estimar a velocidade do time.**
- **Determinar quantos, quais as histórias e quantos *sprints* são necessários para cumprir a meta do *release*.**

IMPORTANTE: velocidade é o número de pontos que um time é capaz de realizar durante um *Sprint* de tamanho determinado. Veremos como calcular adiante.

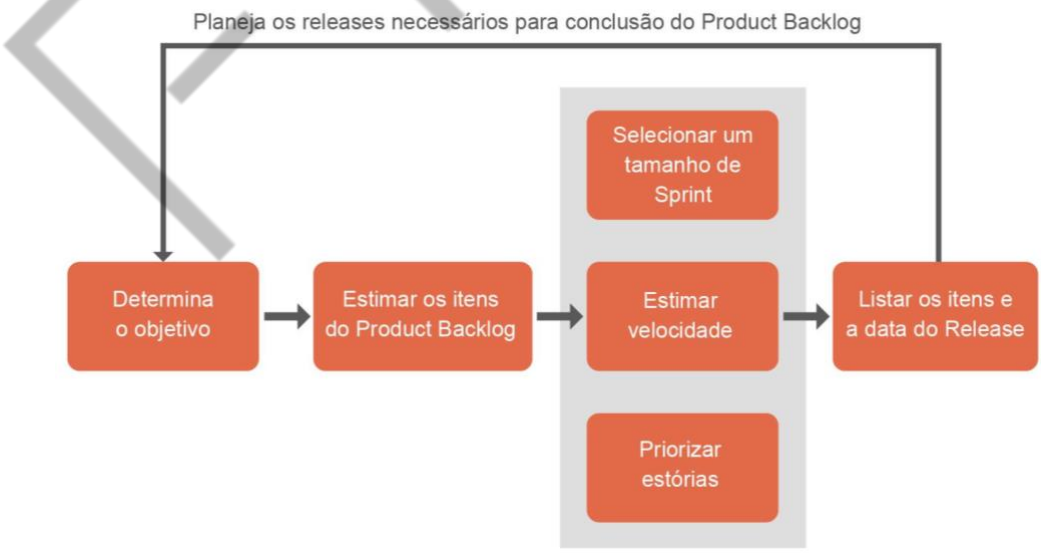

Figura 2.15 – Sequência Planejamento de releases
Fonte: Scrum Guide®, adaptado por FIAP (2018)

Como o *Product Backlog*, o plano de *releases* não é um plano estático. Ele pode ser alterado durante todo o ciclo do projeto, à medida que a equipe for adquirindo conhecimento maior. Por isso, o plano de *releases* deve ser revisto e atualizado em intervalos regulares, por exemplo, após cada *sprint*.

Podemos orientar o planejamento dos *releases* de duas formas:

1. Orientado ao escopo

A forma mais utilizada é a orientada ao escopo, na qual o *Product Backlog* priorizado e estimado é dividido de acordo com a meta do *release*, listando as histórias necessárias para atingir a sua meta e definindo quantos *sprints* são necessários para cumprir o *release*.

Essa definição do número de *Sprints* é feita considerando o tamanho das histórias (soma) e a velocidade do time.

PLANEJAMENTO DE RELEASES					
R	**#**	**Descrição**	**Prior.**	**Tam**	**Pts.**
	18	Como supervisor da agência de locação, posso cadastrar, alterar, consultar e excluir os veículos para locação	2	5	
	7	Como atendente, posso consultar os veículos disponíveis para locação	3	8	
	10	Como atendente, posso cadastrar os clientes que vão fazer locação de veículos	4	5	
	8	Como atendente, posso fazer locação de um veículo por km livre ou controlada	5	13	
	9	Como atendente, preciso aprovar o crédito do cliente antes de fazer a locação do veículo	5	8	
	20	Como atendente, posso fazer reservas de locações de veículos para clientes cadastrados	5	0	39

Quadro 2.8 – Exemplo plano de releases orientado ao escopo
Fonte: Elaborado pelo autor (2017)

No exemplo do quadro, temos um release com tamanho de 39 pontos. Considerando hipoteticamente que a velocidade do time é de 13 pontos por *sprint* de 2 semanas, temos que são necessários 3 *sprints* para entregar esse *release*.

2. Orientado ao prazo

Na forma orientada ao prazo, o *Product Owner* determina o tempo em que deseja que um *release* seja entregue e dividem-se as histórias em *sprints* para se enquadrarem no prazo definido.

No exemplo da tabela, vamos supor que o *Product Owner* deseja um *release* a cada 4 semanas. Considerando que a velocidade do time é de 10 pontos por *sprint*, temos que entregar 20 pontos por *release*, ou seja, com 2 *sprints*.

IMPORTANTE: Como determinar a velocidade é abordado no item 6.15.

2.14 Cálculo da Velocidade

Como já mencionado, para determinar quantos pontos de histórias o time é capaz de realizar durante um Sprint, precisamos fazer o cálculo da velocidade do time, ou seja, saber se as histórias descritas e priorizadas no *Product backlog* pelo *Product Owner* cabem dentro do *timebox* do *Sprint* que está sendo planejado.

Algumas considerações...

Antes de entrar no cálculo da velocidade, vamos analisar alguns fatores importantes relacionados ao time e ao ambiente que devemos considerar nos cenários de projeto:

- Quando começa a usar *Scrum*, há um ciclo natural de aprendizado do time e de todos os envolvidos;
- Não espere que um *Scrum team* vá começar a trabalhar com o mais alto desempenho possível desde o início;
- A estrutura e conhecimento do time melhoram à medida que executam os *sprints*.

Segundo Tuckman, todo time passa pelas seguintes etapas: formação, conflito, acordo, desempenho e dispersão, ilustrado na figura Modelo de Tuckman em que cada time tem tempos diferentes para atingir a fase de desempenho.

Ao iniciar o projeto, um time é montado e entra na fase de formação. Em seguida, no dia a dia do trabalho, o time se ajusta, descobrindo as forças e fraquezas de cada membro e estabelecendo padrões e limites até atingir a fase de acordo. Somente após essas 3 fases o time começa a entrar na fase de desempenho. Ao final do projeto, entra a fase de dispersão do time e o ciclo recomeça.

Figura 2.16 – Modelo de Tuckman
Fonte: FIAP (2018)

Em estudos realizados, um time *Scrum* demora cerca de 3 a 4 *sprints* para entrar em ritmo de desempenho e produzir o seu melhor.

Agora sim! Calculando a velocidade.

Existem diversas formas de calcular a velocidade de um time Scrum. Vamos apresentar uma que é bastante utilizada no mercado e fácil de utilizar.

Nesse método, o primeiro passo é definir qual é o percentual de dedicação de cada membro do time Scrum durante o Sprint, por exemplo: 100% dedicado, 50% dedicado. Esse nível é chamado de fator de foco. Podemos dizer que seria o percentual de produtividade do time, dado pela fórmula:

Fator de Foco = Número de pontos produzidos / Capacidade de produção do time

Em que:

Número de pontos produzidos = quantos pontos foram produzidos no *Sprint*.

Capacidade de produção do time = Somatória do número de dias de cada membro do time durante o *sprint*.

IMPORTANTE:

Quando NÃO se tem histórico ou algum *sprint* executado pelo time, o fator de foco inicial pode variar entre 50% e 70% para um novo time SCRUM. O *Scrum Master* deve avaliar o melhor fator para aplicar de acordo com as características do time. Sugerido é usar o fator de foco inicial em 60%.

Exemplo:

Vamos considerar um time com 3 recursos, sendo dois 100% e um 50% dedicado ao projeto. Em um *Sprint* de 2 semanas foram produzidos 18 pontos.

Qual é o fator de foco?

Tabela 2.5 – Exemplo de quadro de alocação

Equipe	Recursos	Alocação	Produção
3 pessoas	**Huguinho**	**100%**	**100% / 10 dias = 10 dias/homem (d/h)**
	Zezinho	**100%**	**100% / 10 dias = 10 dias/homem (d/h)**
	Luizinho	**50%**	**50% / 10 dias = 5 dias/homem (d/h)**

Fonte: Elaborado pelo autor (2017)

Somando-se a capacidade de produção dos recursos do projeto, temos:

10 + 10 + 5 = 25 d/h de capacidade de produção

Aplicando a fórmula:

Fator de Foco = número de pontos produzidos / capacidade produção time

Temos:

Fator de foco = 18 / 25 = 0,72 => **fator de foco = 72%**

Para calcular a velocidade usamos a fórmula:

Velocidade = Capacidade de produção do time x Fator de foco

Dessa forma, vamos calcular a velocidade no cenário abaixo, considerando o fator de foco calculado anteriormente (72%):

Tabela 2.6 – Exemplo de cálculo de velocidade de produção

Equipe	Recursos	Alocação	Produção
3 pessoas	**Huguinho**	**100%**	**100% / 10 dias = 10 dias/homem (d/h)**
	Zezinho	**100%**	**100% / 10 dias = 10 dias/homem (d/h)**
	Luizinho	**100%**	**100% / 10 dias = 10 dias/homem (d/h)**

Fonte: Elaborado pelo autor (2017)

Velocidade = capacidade de produção do time x fator de foco
Velocidade = 30 d/h x 72% = **22 pontos por *sprint*.**

Sendo assim, nenhum *sprint* planejado pode ter mais do que 22 pontos.

Outro exemplo:

Um time novo com 3 pessoas alocadas 100% do tempo, para a realização de um *sprint* de 2 semanas (10 dias), com fator de foco igual a 60%.

Tabela 2.7 – Exemplo de cálculo de velocidade de produção (1)

Equipe	Recursos	Alocação	Produção
3 pessoas	**Huguinho**	**100%**	**100% / 10 dias = 10 dias/homem (d/h)**
	Zezinho	**100%**	**100% / 10 dias = 10 dias/homem (d/h)**

	Luizinho	**100%**	**100% / 10 dias = 10 dias/homem (d/h)**

Fonte: Elaborado pelo autor (2017)

A capacidade produtiva do time é calculada da seguinte forma:

Capacidade produtiva = Número de dias do *sprint* x número recursos dedicados

Portanto, temos:

Capacidade produtiva = 10 dias x 3 recursos = 30 h/d
Velocidade = 30 h/d x 60% / 100
Velocidade = 18 pontos por *sprint*.

Resumo das fórmulas de cálculo da velocidade.

Velocidade = capacidade de produção do time x fator de foco
Capacidade produção = Número de dias *sprint* x número recursos
Fator de Foco = Número de pontos produzidos / Capacidade de produção do time

O cálculo da velocidade deve ser refeito ao final de cada *Sprint*. Esse recálculo é necessário para que o time refine seu planejamento e melhore a capacidade produtiva do time. Nesse recálculo, devem ser considerados os pontos que foram construídos.

Após tanta informação você deve estar ansioso para colocar o Scrum em ação! Após o detalhamento do backlog chegou a hora de colocarmos a mão na massa e definir nossos primeiros Sprints. Veja no próximo capítulo o uso deste framework na prática!

REFERÊNCIAS

COHN, M. **Desenvolvimento com Scrum:** Aplicando métodos ágeis com sucesso. Bookman: New York, 2000.

SCHWABER, K.; SUTHERLAND, J. **Scrum Guide**. Scrum Org, 2016.

_____. **Software em 30 dias**. John Wiley, 2012.

SCRUM. **The home of Scrum**. [s.d.]. Disponível em: <http://www.scrum.org>. Acesso em: 17 out. 2018.

SCRUM ALLIANCE. **About Scrum Alliance**. [s.d.]. Disponível em: <https://www.scrumalliance.org>. Acesso em: 17 out. 2018.

WEST, D. **Updates to the Scrum Guide: The 5 Scrum values take center stage**. 6 jul. 2016. Disponível em: <https://blog.scrum.org/updates-scrum-guide-5-scrum-values-take-center-stage/>. Acesso em: 19 jun. 2017.