Processo Ágil de Desenvolvimento de Software (Scrum)

Brasília-DF.

Elaboração

xxxxxxxxx

Produção

Equipe Técnica de Avaliação, Revisão Linguística e Editoração

Sumário

APRESENTAÇÃO **5**

ORGANIZAÇÃO DO CADERNO DE ESTUDOS E PESQUISA **6**

INTRODUÇÃO **8**

UNIDADE I
METODOLOGIAS ÁGEIS 9

CAPÍTULO 1
CONCEITO 9

CAPÍTULO 2
AGILE VERSUS TRADICIONAL 14

CAPÍTULO 3
PRINCÍPIOS ÁGEIS 19

CAPÍTULO 4
OUTROS MÉTODOS ÁGEIS 25

UNIDADE II
SCRUM 34

CAPÍTULO 1
HISTÓRICO E CONCEITO BÁSICO 34

CAPÍTULO 2
INTRODUÇÃO AOS PRINCÍPIOS DO SCRUM 41

CAPÍTULO 3
REQUISITOS E *USER STORIES* ***47***

CAPÍTULO 4
PRODUCT BACKLOG 54

UNIDADE III
PAPÉIS 59

CAPÍTULO 1
PRODUCT OWNER 61

CAPÍTULO 2
SCRUM *MASTER* ***64***

CAPÍTULO 3
EQUIPES DE DESENVOLVIMENTO 68

CAPÍTULO 4
GERENTES DE PROJETOS E SCRUM *MASTER* ***70***

UNIDADE IV
PLANEJAMENTO E SPRINTS – EVENTOS DO SCRUM 75

CAPÍTULO 1
SPRINT E GERENCIAMENTO DE RISCOS NO SCRUM..... 76

CAPÍTULO 2
SPRINT PLANNING OU PLANEJAMENTO DE SPRINT..... 84

CAPÍTULO 3
SPRINT REVIEW – REVISÃO DE SPRINT 88

CAPÍTULO 4
SPRINT RETROSPECTIVE 92

REFERÊNCIAS 96

Apresentação

Caro aluno

A proposta editorial deste Caderno de Estudos e Pesquisa reúne elementos que se entendem necessários para o desenvolvimento do estudo com segurança e qualidade. Caracteriza-se pela atualidade, dinâmica e pertinência de seu conteúdo, bem como pela interatividade e modernidade de sua estrutura formal, adequadas à metodologia da Educação a Distância – EaD.

Pretende-se, com este material, levá-lo à reflexão e à compreensão da pluralidade dos conhecimentos a serem oferecidos, possibilitando-lhe ampliar conceitos específicos da área e atuar de forma competente e conscienciosa, como convém ao profissional que busca a formação continuada para vencer os desafios que a evolução científico-tecnológica impõe ao mundo contemporâneo.

Elaborou-se a presente publicação com a intenção de torná-la subsídio valioso, de modo a facilitar sua caminhada na trajetória a ser percorrida tanto na vida pessoal quanto na profissional. Utilize-a como instrumento para seu sucesso na carreira.

Conselho Editorial

Organização do Caderno de Estudos e Pesquisa

Para facilitar seu estudo, os conteúdos são organizados em unidades, subdivididas em capítulos, de forma didática, objetiva e coerente. Eles serão abordados por meio de textos básicos, com questões para reflexão, entre outros recursos editoriais que visam tornar sua leitura mais agradável. Ao final, serão indicadas, também, fontes de consulta para aprofundar seus estudos com leituras e pesquisas complementares.

A seguir, apresentamos uma breve descrição dos ícones utilizados na organização dos Cadernos de Estudos e Pesquisa.

Provocação

Textos que buscam instigar o aluno a refletir sobre determinado assunto antes mesmo de iniciar sua leitura ou após algum trecho pertinente para o autor conteudista.

Para refletir

Questões inseridas no decorrer do estudo a fim de que o aluno faça uma pausa e reflita sobre o conteúdo estudado ou temas que o ajudem em seu raciocínio. É importante que ele verifique seus conhecimentos, suas experiências e seus sentimentos. As reflexões são o ponto de partida para a construção de suas conclusões.

Sugestão de estudo complementar

Sugestões de leituras adicionais, filmes e sites para aprofundamento do estudo, discussões em fóruns ou encontros presenciais quando for o caso.

Atenção

Chamadas para alertar detalhes/tópicos importantes que contribuam para a síntese/conclusão do assunto abordado.

Saiba mais

Informações complementares para elucidar a construção das sínteses/conclusões sobre o assunto abordado.

Sintetizando

Trecho que busca resumir informações relevantes do conteúdo, facilitando o entendimento pelo aluno sobre trechos mais complexos.

Para (não) finalizar

Texto integrador, ao final do módulo, que motiva o aluno a continuar a aprendizagem ou estimula ponderações complementares sobre o módulo estudado.

Introdução

O desenvolvimento de *software* se tornou uma das tarefas mais pesquisadas nos últimos anos. Muitos pesquisadores começaram a se incomodar em relação aos numerosos padrões de projetos que foram criados para esse propósito. Alguns tiveram suas vidas curtas devido às enormes burocracias que, fatalmente, resultariam em custos elevados. Aliás, o custo sempre foi uma variável utilizada pelo mercado para sempre tentar reduzir seu valor.

Muitos desenhos, gráficos e teorias foram criados com foco em produtividade. As métricas forma lançadas seguindo sempre algum cálculo ou teoria que cobrasse do time de desenvolvimento uma melhor atuação. E foi nesse momento da história que houve, basicamente, a revolução das equipes de desenvolvimento. Sempre dispostas a seguir o percurso ideal em busca da métrica perfeita, a qualquer custo, em determinado momento a equipe decidiu que seria ela quem determinaria essa métrica, o tamanho de seu time, os papéis etc. Foi dado o pontapé inicial da metodologia ágil ou *Agile*.

A metodologia ágil foi definida pelos grandes pensadores e pelas grandes empresas como a descoberta de ouro do desenvolvimento de *software* moderno. Ela estaria propensa não só a facilitar a vida das equipes, num primeiro momento; teria todas as condições de ser um organismo vivo, capaz de se reinventar e se melhorar conforme fosse sedo utilizada e aplicada em empresas ao redor do mundo. E foi exatamente isso o que aconteceu.

Portanto, é fundamental um arquiteto de *softwares* saber como esse conceito beneficiará a equipe, como se comportar dentro de uma equipe ágil, quais serão suas responsabilidades e quem faz parte do Time de Scrum.

Objetivos

- Compreender melhor as metodologias ágeis.
- Compreender como se comportar em um time ágil.
- Estudar as principais técnicas e metodologias ágeis, como o Scrum.

METODOLOGIAS ÁGEIS

UNIDADE I

CAPÍTULO 1
Conceito

"Métodos ágeis são métodos de desenvolvimento incremental de software, concentrado na velocidade do desenvolvimento, entregas frequentes do software, redução de *overheads* dos processos e produção de códigos de alta qualidade, envolvendo o cliente diretamente no processo de desenvolvimento" (SOMMERVILLE, 2010, p. 53).

O conceito de desenvolvimento ágil de *software* refere-se a um grupo de metodologias baseadas em colaborações de equipes multifuncionais e bem organizadas de desenvolvimento iterativo de *software*. Elas promovem um processo disciplinado no gerenciamento do projeto, preparando as equipes para ter um poder de aceitação para as adaptações e alterações. Também incentiva o trabalho em equipe e a responsabilidade, preparando-se para a entrega de *software* de alta qualidade, alinhando-se aos negócios e satisfazendo às necessidades dos clientes, entregando-lhes algo de valor a cada iteração.

Segundo Sommerville (2010, p. 39), "nas décadas de 80 e 90, os engenheiros de software tinham uma visão de que a melhor maneira de se conseguir um bom planejamento era por meio de processos cuidadosos e minuciosos, métodos apoiados em ferramentas CASE (*computer-aided software engineering)*. Softwares grandes e com prazos gigantescos, utilizavam grandes equipes que muitas vezes estavam geograficamente afastadas. Softwares governamentais e de grande criticidade demorava muito tempo e possuía um overhead significativo no planejamento produção e documentação".

Quando aplicado em processos de pequeno porte, esses planejamentos eram tão dispendiosos que dominavam o processo de desenvolvimento de *software*, ou seja, o tempo gasto em análises e planejamento era muito superior ao desenvolvimento do sistema. Essa insatisfação já era notória entre os desenvolvedores que, na década de

90, começaram a propor métodos ágeis, em que o foco era uma abordagem incremental para as especificações, o desenvolvimento e a entrega do produto.

Segundo Sommerville (2010, p. 40), "eles são mais adaptáveis ao desenvolvimento de software nos quais os requisitos de funcionais e de sistema mudam constantemente durante o processo de desenvolvimento. Se foco é entregar software funcionais rapidamente aos clientes, e poderão, na sequência, propor alterações para as próximas iterações. Com isso, reduz a burocracia do processo, evitando qualquer trabalho de valor duvidoso e qualquer documentação que nunca será usada".

O Manifesto Ágil foi o marco para que o movimento se tornasse uma filosofia de melhoria. O original desse manifesto pode ser visualizado pelo link https://www.manifestoagil.com.br/, acessado em 01/11/2019:

> "Estamos descobrindo maneiras melhores de desenvolver softwares, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:
>
> **Indivíduos e interações** mais que processos e ferramentas
>
> **Software em funcionamento** mais que documentação abrangente
>
> **Colaboração com o cliente** mais que negociação de contratos
>
> **Responder a mudanças** mais que seguir um plano
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda".

No Capítulo 4, veremos alguns métodos ágeis. Entretanto, todos compartilham algumas características, como foco no desenvolvimento, entrega incremental e, como propõe Sommerville (2010, p. 40):

Tabela 1. Os princípios dos métodos ágeis

Princípios	Descrição
Envolvimento do cliente	Essa etapa é essencial para priorizar novos requisitos e considerar iterações.
Entrega incremental	O programa é desenvolvido em incrementos juntamente com o cliente, especificando os requisitos para serem incluídos em cada um.
Pessoas, não processos	As habilidades do time de desenvolvimento devem ser reconhecidas, exploradas, e seus membros devem desenvolver suas próprias maneiras de trabalhar, sem processos prescritivos.
Aceitar as mudanças	Consciência de que os requisitos do sistema vão mudar.
Manter a simplicidade	Focalize a simplicidade, tanto do *software* a ser desenvolvido como do processo de desenvolvimento, e trabalhe ativamente para eliminar a dificuldade do sistema.

Fonte: Sommerville, 2010, p. 40.

Apesar de muitas empresas adotarem a metodologia Ágil para gerenciamento de *software*, em 2010, Sommerville (2010, p. 41) citou algumas dificuldades:

- alguns *softwares* não foram bem-sucedidos para alguns tipos de desenvolvimento de sistemas, como o desenvolvimento de produtos para empresa de pequeno e médio porte;
- o sucesso da implementação do processo depende da disposição do cliente de passar o tempo com a equipe de desenvolvimento;
- personalidade de algum membro adequada para o intenso envolvimento com a equipe;
- priorização de mudanças;
- mudanças culturais nas empresas para se adequar aos processos ágeis;
- utilização de organização externa para o desenvolvimento de sistemas com as alterações de requisitos;
- relação entre a alteração de requisitos e contrato com cliente. Quando algo dá errado fica difícil encontrar um responsável;
- manter o envolvimento do cliente no processo após a entrega do *software*;
- continuidade da equipe de desenvolvimento.

Quando a declaração de princípios Ágeis foi criada, o intuito era a otimização de processos de desenvolvimento de *software*, desencorajar a utilização de práticas ineficientes, como reuniões excessivas, rigidez nos processos, e documentação pesada.

Depois de quase vinte anos de lançamento de seu manifesto, a metodologia Ágil está sendo utilizada na maioria das empresas envolvidas no desenvolvimento de *software*. E, com certeza, algo tão sólido poderia oferecer vantagens por tanto tempo. Entre elas, podemos citar:

- **Vantagens para o cliente**

  A iteração mais frequente torna o fornecedor e o cliente parceiros que atenderão, mais harmonicamente, às solicitações. *Softwares* de valor são desenvolvidos e entregues mais rapidamente, em ciclos curtos, diferentemente dos ciclos longos de processos clássicos de desenvolvimento de *software*.

» **Vantagens para os fornecedores**

Algumas vantagens beneficiarão tanto os fornecedores quanto os clientes. Certamente, a entrega em ciclos menores reduzirá os custos de desenvolvimento, o tempo de lançamento será mais rápido, aumentando a eficiência, obtendo receitas rapidamente. Com isso, a fidelização será mais eficaz, e sobrará tempo para investir na aquisição de clientes.

» **Vantagens para as equipes de desenvolvimento**

As metodologias ágeis foram criadas com o foco nas pessoas. Além de clientes e fornecedores, a equipe de desenvolvimento também se beneficia quando adotado esse método. Nos métodos ágeis, a improdutividade dá espaço à eficiência, torna os prazos mais reais, e, com isso, as pessoas poderão trabalhar como se sentem mais à vontade. Elas se sentirão mais valorizadas, e haverá mais harmonia no trabalho.

» **Vantagens para gerentes**

Quando os prazos são determinados pela equipe de desenvolvimento, o trabalho do gerente se torna mais fácil em fazer cumprir o cronograma. Eles descobrem que o planejamento e a rastreabilidade são mais fáceis e possíveis no nível das tarefas. Com isso, as prioridades são definidas em um alinhamento melhor com as necessidades dos clientes, podendo ser oferecidas as entregas de valores. Essa conscientização faz com que o monitoramento, a captura e a resolução de problemas sejam definidos rapidamente.

Escalabilidade de Scrum

Uma das qualidades do Scrum é como sua estrutura de equipes foi pensada. Para seus criadores, o tamanho da equipe ou o *Scrum Team* não terá um tamanho fixo, pois, como sabido, os projetos possuem seus tamanhos variados. Entretanto, foi pensado um mecanismo de escalabilidade que se iniciaria com um tamanho mínimo e fosse livre para que a própria equipe decidisse pela quantidade de membros.

Segundo o SBOK (2017, p. 5), "para serem eficazes, o tamanho ideal dos Times Scrum deve ser de seis a dez membros, porém, esta prática pode induzir à concepção errônea de que o Scrum pode ser utilizado apenas para projetos pequenos, mas, ao contrário, o Scrum pode ser facilmente escalado para o uso eficaz em grandes projetos. Em situações

em que o tamanho do Time Scrum ultrapassa dez pessoas, vários Times Scrum podem ser formados para trabalhar no projeto".

Outra prática interessante é o processo de "Convocar o Scrum de Scrums", que facilita a coordenação entre os Times Scrum e permite o gerenciamento eficaz em projetos maiores.

Extreme Programming Explained. Esse foi o primeiro livro sobre XP e talvez ainda seja o mais lido. Ele explica a abordagem a partir da perspectiva de um de seus inventores, e seu entusiasmo fica explícito no livro (BECK, K. *Extreme Programming Explained*. Addison-Wesley, 2000).

Get Ready for Agile Methods, With Care. Uma crítica cuidadosa aos métodos ágeis, que discute seus pontos fortes e fracos, escrita por um engenheiro de *software* muito experiente (BOEHM, B. *IEEE Computer*, jan. 2002).

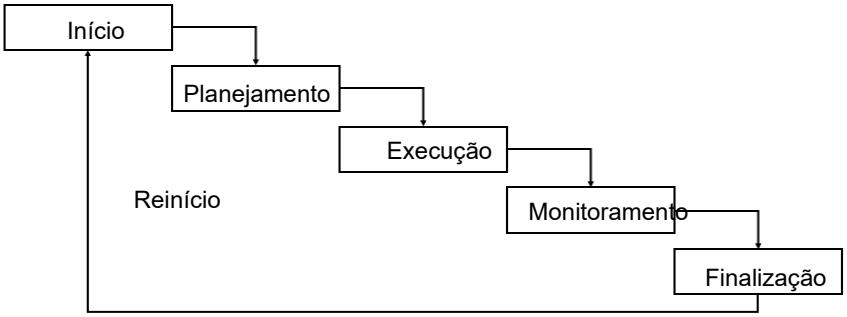
Scaling Software Agility: Best Practices for Large Enterprises. Embora com foco em questões de escalamento de desenvolvimento ágil, esse livro inclui também um resumo dos principais métodos ágeis, como XP, Scrum e Crystal (LEFFINGWELL, D. *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley, 2007).

Running an Agile Software Development Project. A maioria dos livros sobre métodos ágeis se concentra em um método específico, mas esse livro tem uma abordagem diferente e discute como colocar XP em prática em um projeto. A obra apresenta conselhos bons e práticos (HOLCOMBE, M. *Running an Agile Software Development Project*. John Wiley and Sons, 2008).

CAPÍTULO 2
Agile versus Tradicional

O gerenciamento de projetos tradicionais segue uma metodologia fixa na qual o projeto é executado em um ciclo sequencial: iniciação, planejamento, execução, monitoramento e finalização. O ciclo significa que um processo só será iniciado quando um anterior estiver terminado. O método tradicional está fortemente fixado em processos lineares, documentação, planejamento antecipado e priorizações. Por herdar semelhanças do PMBOK, o tempo e o orçamento são variáveis, e requisitos fixos.

Figura 1. Ciclo em metodologias tradicionais.



Fonte: elaboração própria do autor.

Em contrapartida, as metodologias ágeis usam abordagens iterativas, e seu principal objetivo é entregar algo de valor a cada ciclo. "Algo de Valor" significa entregar o *software* funcionando do módulo que o cliente definiu como prioridade alta, de maior importância ao negócio. As entregas têm um prazo curto, de uma a quatro semanas, ao contrário do método clássico, que é feito em sequências. As tarefas não concluídas têm um tratamento de priorização e serão incluídas em entregas futuras, e os estágios do ciclo de vida serão revisados conforme necessidades, ao contrário dos requisitos fixos das metodologias clássicas.

A principal diferença dos métodos ágeis e os tradicionais é a forma de como o projeto é tratado. Vejamos uma lista de principais diferenças entre essas duas metodologias.

Flexibilidade no escopo

Nas metodologias clássicas de gerenciamento de projeto, a flexibilidade de alteração de escopo é muito estressante, pois não oferece – ou oferece poucas – oportunidades de alterações ou atualizações de escopo.

Nas metodologias ágeis, as equipes já estão preparadas para receber alterações constantes de escopo e dispostas a encontrar melhores alternativas para um ciclo de manutenção e alterações mais rápidas. Vale lembrar que toda alteração de escopo passa antes por análise envolvendo todos os membros da equipe, e a decisão de alterações é tomada da melhor forma para que não influencie negativamente o produto final. O foco é entregar o melhor produto, e não ficar atrelado às estruturas rígidas.

Transparência

No gerenciamento tradicional, a responsabilidade pertence totalmente ao gerente de projeto, cabendo a ele planejar e documentar todo o ciclo de vida deste. Ele trata diretamente com o cliente, e os prazos em necessidades de desenvolvimento são negociados entre os dois. Os membros da equipe só terão ciência após tudo planejado. Como o prazo não envolveu a equipe de desenvolvimento, isso poderá gerar reações inesperadas, como atraso nos prazos pelo fato de o gerente não ter consultado sobre a complexidade dos funcionalidades a desenvolver.

Nas metodologias ágeis, os membros da equipe decidem e compartilham as propriedades do projeto. A equipe tem a voz mais ativa na hora de decidir sobre a complexidade e o tempo de determinada funcionalidade a desenvolver. Com isso, além de criarem um ambiente produtivo e altamente engajado, são capazes de visualizar o progresso do produto, ajustando desvios de atrasos rapidamente ou informando que o prazo será cumprido.

Solução de problemas

Nas metodologias tradicionais, o gerente de projeto é o ponto principal para que a equipe reporte um problema. Isso significa, no entanto, que todas as supostas soluções deverão depender única e exclusivamente da disponibilidade deste. Ele poderá facilmente se tornar um gargalo nesse ponto do projeto e aumentar os custos e exceder prazos.

Nas metodologias ágeis, o time possui mais autonomias e autoridades para tomar decisões por conta própria. Por estarem mais envolvidos no processo e compartilhando seus pensamentos a cada momento, eles tentarão decidir, imediatamente, todos os problemas, deixando para demandar os gerentes unicamente em questões extremas.

Feedback de alterações

Nas metodologias clássicas, o processo é "engessado" ao escopo inicial, ao prazo e orçamento estimado. Portanto, qualquer alteração ou *feedback* necessário, se não ignorado, é tratado com burocracias, como alterações de escopo etc.

Nas metodologias ágeis, o *feedback* é constante e altamente aceitável, fornecendo algo sempre útil ao projeto. As aceitações e alterações são respondidas a cada iteração validada com os clientes.

Complexidade do Projeto

A metodologia tradicional é destinada a projetos menores devido à sua abordagem linear, como o Cascata, por exemplo, e essa metodologia não se adéqua muito a alterações repentinas, e os membros da equipe de desenvolvimento as evitam.

As metodologias ágeis estão sendo muito aceitas no desenvolvimento de projetos grandes e complexos, pois utiliza-se de um método que divide trabalhos grandes e complexos em processos menores e mais fáceis de gerenciar.

Tabela 2. Principais diferenças entre as metodologias de projeto tradicional e ágil.

Características	Abordagem ágil	Abordagem tradicional
Estrutura organizacional	Iterativo	Linear
Escala de projetos	Pequena e média escala	Em larga escala
Requisitos do usuário	Entrada interativa	Claramente definido antes da implementação
Envolvimento de clientes	Alto	Baixo
Modelo de desenvolvimento	Entrega evolutiva	Ciclo da vida
Envolvimento do cliente	Os clientes estão envolvidos desde o momento em que o trabalho está sendo realizado	Os clientes se envolvem no início do projeto, mas não após o início da execução
Gerenciamento de escalação	Quando ocorrem problemas, toda a equipe trabalha em conjunto para resolvê-los	Encaminhamento para gerentes quando surgirem problemas
Preferência de modelo	Modelo ágil favorece adaptação	Modelo tradicional favorece antecipação
Produto ou processo	Menos foco nos processos formais e diretivos	Mais sério sobre os processos do que o produto
Documentação de teste	Planejamento abrangente de testes	Os testes são planejados um Sprint de cada vez
Estimativa de esforço	O Scrum Master facilita, e a equipe faz a estimativa	O gerente de projeto fornece estimativas e obtém aprovação do pedido de compra para todo o projeto
Revisões e aprovações	Revisões são feitas após cada iteração	Revisões e aprovações excessivas por líderes

Fonte: adaptada e traduzida de https://www.proofhub.com/articles/traditional-vs-agile-project-management.

Desenvolvimento ágil e desenvolvimento dirigido a planos

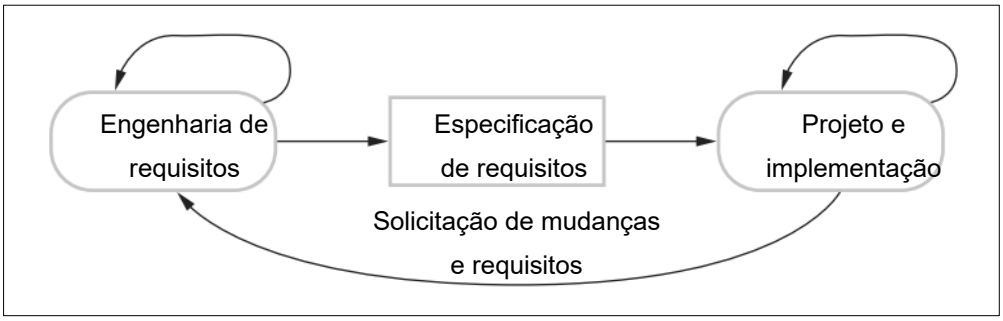
Existe uma comparação feita por Sommerville (2010, p. 42) entre desenvolvimento ágil e desenvolvimento de *software* dirigido a planos. "Na verdade, a maioria dos projetos de software inclui práticas das abordagens dirigidas a planos e ágil".

Desenvolvimento dirigido a planos *versus* desenvolvimento ágil

É baseado em estágios de desenvolvimento separados, com produtos a serem produzidos em cada um dos estágios, antecipadamente planejados, que é muito bem representado pelo modelo Cascata.

Segundo Sommerville (2010, p. 42), "em uma abordagem dirigida a planos, ocorrem iterações no campo das atividades com documentos formais, usados para determinar a comunicação entre os estágios do processo. As saídas de um estágio são usadas como apoio para o planejamento da atividade do processo seguinte. Um processo de software dirigido a planos pode amparar o desenvolvimento e a entrega incremental. É completamente factível alocar requisitos e preconceber as fases de projeto e desenvolvimento como uma sucessão de incrementos".

Figura 2. Desenvolvimento baseado em planos.



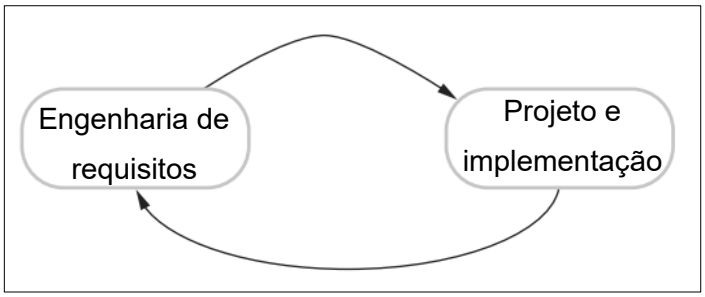
Fonte: Sommerville, 2010, p. 43.

Nas metodologias ágeis, tanto o projeto, a elicitação, os testes, os requisitos, quanto a implementação são atividade centrais.

Seguindo um exemplo de Sommerville (2010, p 43), "os requisitos vão evoluir e, finalmente, será produzida uma especificação de requisitos. Essa é, então, uma entrada para o processo de projeto e implementação. Em uma abordagem ágil, iterações ocorrem em todas as atividades. Portanto, os requisitos e o projeto são desenvolvidos em conjunto, e não separadamente".

Veremos mais adiante que um processo ágil não é, inevitavelmente, focado no código e pode produzir alguma documentação de projeto.

Figura 3. Desenvolvimento ágil.



Fonte: Sommerville, 2010, p. 43.

Segundo SBOK (2017, p. 53), "o destaque do modelo clássico de gerenciamento de projetos está na execução do planejamento inicial do projeto, com ênfase na fixação do escopo, custo e cronograma, e gerenciamento desses parâmetros. O modelo tradicional de gerenciamento de projetos pode muitas vezes levar a uma circunstância em que, apesar o plano tenha sido bem sucedido, o cliente não está contente".

O *framework* Scrum acredita muito que os colaboradores agregam valores que vão além dos seus conhecimentos técnicos e que um planejamento mapeado é muito ineficiente em ambientes de mudanças constantes. Seu foco está em satisfazer as necessidade dos clientes em entregas incrementais utilizáveis e, para isso, promove a priorização de tarefas e escopos, *Time-boxing* e a auto-organização das equipes.

CAPÍTULO 3
Princípios ágeis

Como dito no Capítulo 1, o Manifesto Ágil iniciou todo o processo rumo ao sucesso do movimento e das metodologias ágeis. O manifesto é composto por quatro valores fundamentais e doze princípios que orientaram a construção das metodologias e *frameworks* ao longo do tempo.

Os quatro valores do manifesto ágil:

1. **Indivíduos e interações sobre processos e ferramentas**

Depois de criar muitas ferramentas que ditassem as regras, nas mais variadas metodologias de gestão de projetos, foi necessário promover uma inversão de atenção, voltada às pessoas. São as pessoas que respondem às regras ligadas ao negócio. Se o processo é baseado totalmente nas ferramentas, a equipe se tornará refém de seus fluxos e, muitas vezes, poderá não ser tão flexível como deveria.

Com isso, por exemplo, a equipe interagiria menos e responderia menos às mudanças necessárias para atender às necessidades dos clientes.

2. ***Software* de Trabalho sobre Documentação Abrangente**

Desenvolver documentação para especificações, requisitos, prospecto, documentos de *design* de interface, planos de teste, planos de documentação e, ainda, aprovações necessárias para cada um, historicamente, consumiu muito tempo e recursos para que, ao final, fosse entregue juntamente com o produto de *software*. Isso quando não se utilizava a equipe de desenvolvimento para participar desse processo – ao final, a entrega do produto e a da documentação nunca eram ao mesmo tempo.

A metodologia ágil não tem a proposta de eliminar as documentações, pois estas são fundamentais para o produto final. Ela simplifica a forma com que a equipe de desenvolvimento realiza o trabalho, tornando as estórias dos clientes requisitos suficientes para a documentação, deixando-os focar nas novas funcionalidades.

3. **Colaboração do cliente sobre negociação de contrato**

Na negociação de contrato, o cliente e o gerente do produto elaboram detalhes, inclusive com os pontos negociáveis, ao longo do caminho. Nos modelos clássicos, o cliente participa no início de cada etapa, negocia os requisitos do produto e só volta a aparecer no momento da entrega ou quando algum ponto precisa ser negociado.

No método Ágil, um cliente estará envolvido e colaborando em toda a etapa do desenvolvimento. Com essa colaboração efetiva, fica muito mais fácil negociar mudanças de escopo, prazo, funcionalidades etc. Isso também inclui as demonstrações e testes.

4. **Respondendo às mudanças após um plano**

Ao longo do desenvolvimento e à medida que o trabalho avança, é comum surgirem novas ideias que melhoram o entendimento do domínio do negócio, mudanças de tecnologias, mudanças de pessoas etc. As mudanças são realidades no processo de desenvolvimento do *software*. As metodologias clássicas consideram as mudanças como uma despesa que deveria ser evitada.

As metodologias ágeis não desconsideram a elaboração de planos de projeto. Entretanto, um plano de projeto deve ser flexível e deve haver conscientização da equipe para alterá-lo conforme a situação mude.

Princípios do Manifesto Ágil

A metodologia ágil está em constante expansão devido a alguns fatores, como a flexibilização, a auto-organização e autoindependência das equipes, a possibilidade de alterar os Backlog a qualquer momento etc. Porém, existem alguns princípios que devem ser mantidos. Foi escolhido o temo "princípio" justamente para não ser algo rígido, obrigatório, "engessado".

Segundo o *site* https://www.significados.com.br/principios/, acessado em 01/11/2019, princípio está ligado a vários ambientes, desde o direito, a filosofia etc.:

> "São um conjunto de normas ou padrões de conduta a serem seguidos por uma pessoa ou instituição, sua conceituação dos princípios está relacionada ao começo ou início de algo. Os princípios também podem estar associados às proposições ou normas fundamentais que norteiam os estudos, sobretudo os que regem o pensamento e a conduta. Os princípios, enquanto regidos pelas leis morais, são valores que o indivíduo considera adotar de acordo com o que diz sua consciência. Eles estão associados à liberdade individual, já que são normas propostas sem pressão externa, ligadas a fatores externos e instituições sociais que possuem determinada influência no comportamento social. No entanto, cada indivíduo terá seus princípios que estarão de acordo com a educação e a experiência de vida de cada um. E eles serão acionados cada vez que a consciência humana exigir."

O Scrum é uma metodologia que, ao ser adotada, consegue ser tão admirada pelas pessoas envolvidas que acaba se tornado uma filosofia de vida. E isso foi o suficiente para que seus princípios acessassem o âmbito filosófico. Chega a pesar na consciência quando algum princípio está sendo quebrado por outras práticas senão o Scrum. Essa foi uma das polêmicas envolvendo Ken Schwaber e o pessoal da Scrum Alliance que pode ser vista, na íntegra, no Apêndice 1. Nela, Ken critica o guia SBOK justamente por ele achar que o guia está quebrando alguns princípios. Mas isso ainda será tema de muita discussão, inclusive filosófica.

A seguir, veremos os doze princípios.

1. **Nossa maior prioridade é satisfazer o cliente por meio da entrega adiantada e contínua de *software* de valor.**

Por incrível que pareça, eliminar de vez resquícios da forma tradicional de desenvolvimento de *software* ainda está no DNA da equipe de desenvolvimento, e isso faz com que esqueçam uma das prioridades de qualquer empresa: a satisfação do cliente. E esse foi o motivo para que se tornasse a primeira opção dos 12 princípios.

» **[Nossa]:** significa que toda a equipe estará envolvida nas responsabilidades de satisfazer o cliente.

» **[Entrega adiantada]:** nesse momento definimos que as entregas devem ser quebradas em partes menores, melhorando sempre a cada entrega.

» **[*Software* de Valor]:** significa que estamos entregando um *software* funcionando, exatamente como a expectativa do cliente.

2. **Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adéquam a mudanças, para que o cliente possa tirar vantagens competitivas.**

Um dos processos mais dispendiosos para a aceitação da metodologia ágil é a aceitação às mudanças. É preciso que haja um trabalho muito forte de conscientização da equipe para aceitar e ver as mudanças como uma oportunidade, e não com algo estressante. Não promover essa oscilação no escopo certamente não atenderá às expectativas do cliente para entregar algo de valor.

As palavras-chaves para esse tópico são:

» **[mudar]:** enfatiza o foco principal desse princípio. Faz com que a equipe, praticamente, dê as boas-vindas ao cliente.

» **[tarde]:** é importante enfatizar ao cliente que nunca é tarde para fazer alteração nos requisitos. Entretanto, para evitar que ela ocorra muito tarde, é importante entendermos a necessidade do cliente e deixar claro que quanto mais tempo gastarmos nesse processo, melhor atenderemos ao primeiro princípio.

» **[vantagens competitivas]**: liberando as ideias do cliente mais rapidamente, por meio do *software*, certamente teremos vantagens competitivas em relação aos clientes e no mercado de atuação.

3. **Entregar *software* funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.**

As metodologias tradicionais são dependentes da documentação e da conclusão total de cada etapa. A metodologia ágil de gerenciamento dá mais ênfase ao próprio desenvolvimento do *software*. Por isso, a maioria dos projetos que utilizam essa metodologia conseguem cumprir o prazo e o escopo do produto.

» **[*software* funcionando]:** esse princípio enfatiza apenas a entrega, a funcionalidade e a satisfação do cliente, baseando-se nas condições acertadas de "feito" e "valor".

» **[aos períodos mais curtos]:** quanto mais cedo entregar, melhor, pois, além de reduzir desperdícios, melhora cada ciclo entregue.

4. **Pessoas relacionadas a negócios e desenvolvedores devem trabalhar em conjunto e diariamente durante todo o curso do projeto.**

Esse é um dos princípios mais fundamental do desenvolvimento ágil: é quando as pessoas que cuidam do *core* da empresa e desenvolvedores trabalham em conjunto, estreitando as ideias, rumo ao objetivo do projeto. É imprescindível que as equipes estejam sob o mesmo ambiente físico. Caso isso não seja possível, devem se reunir por conferências *on-line*.

» **[Pessoas relacionadas a negócios]:** normalmente é o *Product Owner* que faz a ponte entre o cliente e a equipe de desenvolvimento.

» **[devem trabalhar juntos]:** ênfase na colaboração de áreas distintas.

» **[o curso do projeto]:** compromisso total com o objetivo do produto final.

5. **Construir projetos ao redor de indivíduos motivados, dando a eles ambiente e suporte necessário, e confiar que farão seu trabalho.**

Deixar a equipe motivada é o foco principal. E como manter os indivíduos motivados? Isso é válido para qualquer membro de equipe de qualquer projeto, funcionário etc.: deixá-los em um ambiente propício e dar-lhes a confiança para executar suas tarefas.

» **[indivíduos motivados]:** indivíduo motivado, equipe inteira motivada. É o foco principal desse princípio.

6. **O Método mais eficiente e eficaz de transmitir informações para e por dentro de um time de desenvolvimento é por meio de uma conversa cara a cara.**

Na metodologia de gerenciamento ágil de projetos, precisamos obter respostas rápidas para as perguntas, e não há melhor maneira de alcançar esse feito do que a conversa cara a cara.

» **[eficiente e eficaz]:** enfatiza a comunicação eficiente e eficaz em reunião presencial **[cara a cara].**

7. ***Software* funcional é a medida primária de progresso.**

Software funcional é o foco para atingir a satisfação do cliente, entregando algo de valor. O cliente satisfeito significa que todas as funcionalidades prometidas para aquele momento foram entregues.

8. **Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter indefinidamente passos constantes.**

Esse princípio tem uma forte ligação com o quinto em relação aos indivíduos motivados. Motivados também significa que estarão em pleno descanso para trabalhar. Quando dizemos “trabalhar”, referimo-nos ao trabalho intelectual. Pode haver intercalações entre trabalhos intelectuais e trabalhos que não requerem a utilização de pensamentos lógicos.

» **[Desenvolvimento sustentável] [ritmo constante]:** a produtividade diminui, assim como o ritmo cai, quando se trabalha muito tempo continuamente.

9. **Contínua atenção à excelência técnica e ao bom *design* aumenta a agilidade.**

Design e excelência no código é iniciar corretamente a trabalhar com refatorações. A baixa qualidade do código poderá comprometer muitos outros requisitos anteriores.

10. **Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.**

É essencial se concentrar em atividades de alto valor, maximizando o retorno de investimento, eliminando ou automatizando certas atividades. Entretanto, colocar isso em prática, como o princípio diz, é uma arte.

11. **As melhores arquiteturas, requisitos e *designs* emergem de times auto-organizáveis.**

Esse é um princípio mais crítico e delicado das metodologias ágeis. Fazer emergir requisitos, arquiteturas e designs é abordado constantemente como Métodos Orientados a Modelos Ágeis e Design Orientado a Testes.

12. **Em intervalos regulares, o time reflete em como ficar mais efetivo. Então, ajustam-se e otimizam seu comportamento de acordo.**

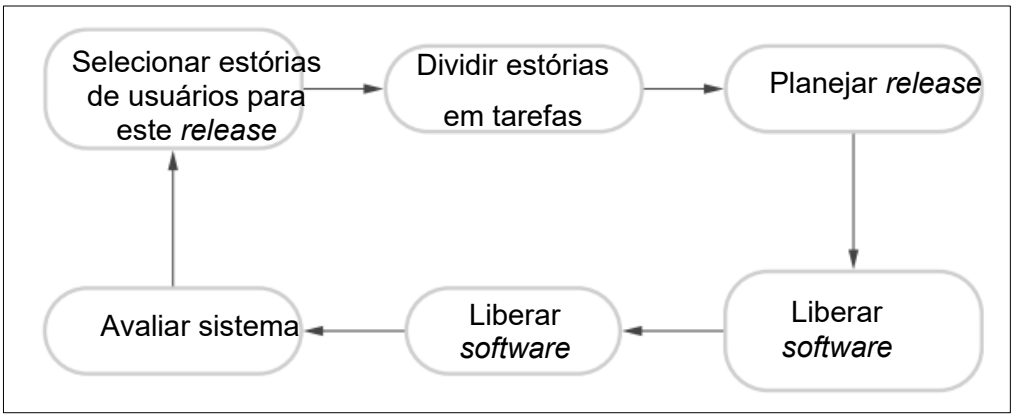
As equipes que estiverem bem sincronizadas com o princípio onze certamente aplicarão melhor as atividades dos processos de melhorias, aprimorando o processo de desenvolvimento de *software*.

CAPÍTULO 4
Outros métodos ágeis

Extreme Programming

Segundo Sommerville (2010, p. 44), "em XP (*Extreme Programming*) os requisitos são cenários, chamados de estórias de usuários e são implementados diretamente como uma série de tarefas, os programadores trabalham em pares, desenvolvem testes para cada tarefa, mesmo antes de escrever o código. Os releases de sistema são entregues em curtos intervalos".

Figura 4. O ciclo de um release em *Extreme Programming*.



Fonte: Sommerville (2010, p. 44).

O XP está envolvido em algumas práticas dos métodos ágeis.

O **desenvolvimento incremental** é sustentado por meio de pequenos e frequentes releases do sistema. Os requisitos são baseados em cenários ou em simples histórias de clientes, usadas como base para decidir a funcionalidade que deve ser incluída em um incremento do sistema. O engajamento entre a equipe de desenvolvimento com cliente é contínuo, e a equipe do cliente é a responsável por efetuar os testes de aceitação.

É sustentado em pessoas, e não em processos, mudanças aceitas por *releases* de ciclo contínuo, não envolve horas longas de trabalho por ser mais eficiente. Em relação à codificação, são feitas refatorações e testes constantes.

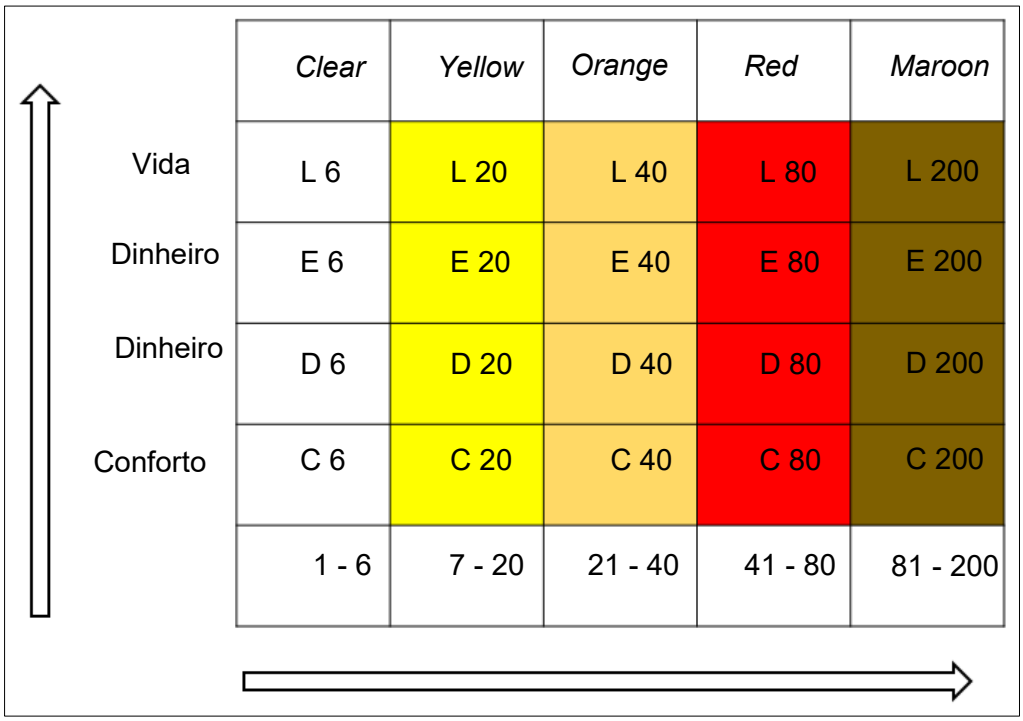
Tabela 3. Práticas de *Extreme Programming*.

Princípio ou prática	Descrição
Planejamento incremental	Requisitos são priorizados por estórias de usuários, organizados e combinados com a equipe de desenvolvimento.
Pequenos *releases*	*Releases* são entregues por meio de pequenas funcionalidades úteis, fornecendo valor ao negócio.
Projeto simples	Todo projeto é enxuto, apenas para atender as necessidades e nada a mais.
Desenvolvimento *test-first*	Os testes para as funcionalidades são escritos antes que seja implementada.
Refatoração	Refatoração contínua para melhoria do código. Mantém o código simples e manutenível.
Programação em pares	Programação em pares, os desenvolvedores interagem com opiniões sobre o código do outro.
Propriedade coletiva	Todos os conhecimentos e todos os desenvolvedores assumem responsabilidade por todo o código.
Integração contínua	Assim que o trabalho em uma tarefa é concluído, ele é integrado ao sistema como um todo.
Ritmo sustentável	Grandes escalas de horas de trabalho não são aceitáveis por não serem produtivas.
Cliente no local	Em um processo de *Extreme Programming*, o cliente é um membro da equipe de desenvolvimento e é responsável por levar a ela os requisitos de sistema para implementação.

Fonte: Sommerville (2010, p. 45).

Crystal

Figura 5. Parametrização de nível crítico.



Fonte: adaptada de http://assets.devx.com/articlefigs/17424.jpg.

Crystal é uma família de metodologias preparada para variar de acordo com o tamanho e a complexidade do projeto. Seu nome foi dado pelo Dr. Alistair Cockburn, e cada família recebe uma cor correspondente à dureza de um cristal: *Clear, Yellow, Orange, Orange web* e *Maroon*. Seu fundador sugere que os projetos não migrem de

fase, de uma cor para outra, ou seja, se um projeto começa *Clear*, permanecerá assim até o final e, independentemente da escolha, seus princípios em comum são:

» entrega frequente;

» *feedback* contínuo;

» comunicação constante;

» segurança;

» foco;

» acesso aos usuários.

As letras nas células representam a criticidade do projeto da seguinte maneira:

» C: conforto;

» D: dinheiro discricionário;

» E: dinheiro essencial;

» L: vida.

Os números nas células representam o tamanho da equipe do projeto.

As funções dessa metodologia variarão conforme o tamanho e a criticidade do projeto aumentam, ou seja, seguindo essa lógica, a função *Clear* possui menos papéis que a *Maroon*.

O DSDM (*Dynamic Systems Development Method*)

É uma estrutura de entrega de projeto ágil usada no desenvolvimento de *software*, que incorpora grande parte do conhecimento sobre gerenciamento de projetos, tanto projetos ágeis quanto tradicional. O DSDM é muito difundido na comunidade de desenvolvimento de *software*, porém, é utilizado em vários negócios, mudando sua estrutura, tornando geral para tarefas complexas de solução de problemas.

Suas principais vantagens são as mesmas da maioria das metodologias ágeis:

» pelo fato de a Equipe de desenvolvimento estar envolvida diretamente com o cliente, o resultado é prontamente visível;

- as funcionalidades básicas e de valor são entregues em intervalos menores e regulares;
- a burocracia é eliminada entre as partes envolvidas;
- o *feedback* é constante; com isso, a probabilidade de sucesso do sistema é maior;
- certamente, o produto será entregue no prazo e no orçamento combinado;
- os usuários ditam o ritmo e a direção do projeto.

Existem 9 princípios que são essenciais para qualquer implementação do DSDM. Ignorar um deles quebrará a filosofia das estruturas ágeis e aumentará os riscos do projeto:

1. envolvimento ativo do usuário;
2. as equipes devem ter poderes para tomar decisões;
3. concentre-se na entrega frequente;
4. critério para entrega aceita;
5. desenvolvimento iterativo e incremental;
6. todas as alterações durante o desenvolvimento devem ser reversíveis;
7. os requisitos são alinhados com a base em alto nível;
8. o teste é integrado ao longo do ciclo de vida;
9. abordagem colaborativa e cooperativa.

O DSDM possui 7 fases para a organização e criação de Projetos:

1. pré-projeto;
2. estudo de viabilidade;
3. estudo de negócios;
4. iteração de modelo funcional;
5. iteração de design e construção;
6. implementação;
7. pós-projeto.

Além disso, possui várias técnicas e regras para diversos pontos e papéis.

Timeboxing

É um intervalo, geralmente não superior a duas, quatro ou seis semanas, em que determinado conjunto de tarefas deve ser alcançado.

Prototipagem

» Protótipo de negócios: permite a avaliação do sistema em evolução.

» Protótipo de usabilidade: verifique a interface do usuário.

» Protótipo de desempenho: garanta que a solução ofereça desempenho ou controle o volume.

» Protótipo de capacidade: avaliar opções possíveis.

Loop de Feedback

É implementada primeiramente uma funcionalidade crítica, para que sejam descobertas falhas ou dificuldades logo no início do processo de desenvolvimento e permitir que as entregas antecipadas obtenham *feedback* do usuário. O *loop* de *feedback* necessário é fornecido por uma oficina, que é a última técnica importante em um projeto do DSDM.

O consórcio DSDM compilou 10 fatores mais importantes de seus membros:

1. aceitação da filosofia do DSDM antes de iniciar o trabalho;
2. os poderes de decisão de usuários e desenvolvedores dentro da equipe de desenvolvimento;
3. o compromisso do gerenciamento de usuários sênior em fornecer um envolvimento significativo do usuário final;
4. entrega incremental;
5. fácil acesso dos desenvolvedores aos usuários finais;
6. a estabilidade da equipe;
7. as habilidades da equipe de desenvolvimento;
8. o tamanho da equipe de desenvolvimento;

9. um relacionamento comercial favorável;
10. a tecnologia de desenvolvimento.

Fraqueza do DSDM

» Custo de licenciamento.

» Barreira relativamente alta à entrada.

» Mudança cultural na organização.

Processo Unificado Ágil

O Processo Unificado (UP) é uma estrutura de processo de desenvolvimento de *software* iterativa e incremental. Existem várias adaptações, sendo a mais popular o *Rational Unified Process* (RUP), da IBM.

O *Agile Unified Process* (AUP) é uma adaptação ágil da UP formalizada por Scott Ambler e escrita por outros, incluindo Craig Larman. Ao contrário do RUP, o AUP possui apenas sete disciplinas:

1. *model*;
2. implementação;
3. teste;
4. implantação;
5. gerenciamento de configuração;
6. gerenciamento de projetos;
7. meio ambiente.

O Agile UP é baseado nas seguintes filosofias:

» **Sua equipe sabe o que está fazendo**

A Equipe não será orientada a ficar lendo a documentação dos processos nos mínimos detalhes. Porém, terá orientações de alto nível de tempos em tempos.

» **Simplicidade**

As descrições são simplificadas e concisas, usando poucas páginas resumidas. Podem conter *links* para detalhes, porém não será obrigatório consultá-los para entender o conceito.

» **Agilidade**

O Agile UP está em conformidade com os valores e princípios do desenvolvimento de *software* ágil e da Agile Alliance.

» **Concentre-se em atividades de alto valor**

Também está concentrado em priorizações de alto valor que sejam realmente interessantes no momento da entrega.

» **Independência da ferramenta**

As ferramentas específicas sempre surgem depois. Primeiramente se desenvolve o conceito, apoiado em alguma ferramenta, como editor de texto ou planilhas, e, mais tarde, começam surgir ferramentas próprias para o conceito ágil. Entretanto, o Time poderá escolher a que mais for útil.

XP

» **Vantagens**

› Cliente está a par do andamento das tarefas.

› *Feedback* frequente.

» **Desvantagens**

› Requer cliente no local.

› Documentação informal.

Scrum

» **Vantagens**

› Equipes de auto-organização e *feedback*.

› Participação do cliente.

› Prioridades baseadas no valor do negócio.

» **Desvantagens**

› Fornece apenas suporte ao gerenciamento de projetos; outras disciplinas estão fora do escopo.

› Não especifica práticas técnicas.

› Pode levar algum tempo para que a empresa forneça prioridades exclusivas para cada requisito.

Cristal

» **Vantagens**

› Família de metodologias projetadas para escalar por tamanho e criticidade do projeto.

› À medida que o tamanho do projeto aumenta, equipes multifuncionais são utilizadas para garantir consistência.

› O componente "humano" foi considerado para todos os aspectos da estrutura de suporte ao projeto.

› A ênfase no teste é tão forte que se espera que pelo menos um testador esteja em cada equipe do projeto.

» **Desvantagens**

› Espera que todos os membros da equipe sejam colocados juntos. Pode não funcionar bem para equipes distribuídas.

› Mudar de uma cor do Crystal para outra no meio do projeto não funciona.

AUP

» **Vantagens**

› Metodologia robusta, com muitos artefatos e disciplinas para escolher.

› Escala muito bem.

› A documentação ajuda a se comunicar em ambientes distribuídos.

› Prioridades definidas com base no maior risco. O risco pode ser comercial ou técnico.

» **Desvantagens**

› Níveis mais altos de cerimônia podem ser um obstáculo em projetos menores.

› Atenção mínima à dinâmica da equipe.

› A documentação é muito mais formal do que a maioria das abordagens mencionadas aqui.

DSDM

» **Vantagens**

› A ênfase no teste é tão forte que se espera que pelo menos um testador esteja em cada equipe do projeto.

› Projetado desde o início por pessoas de negócios, o valor da empresa é identificado, e espera-se que seja a entrega a mais alta prioridade.

› Possui abordagem específica para determinar a importância de cada requisito para uma iteração.

› Define as expectativas das partes interessadas, desde o início do projeto, de que nem todos os requisitos chegarão à entrega final.

» **Desvantagens**

› Provavelmente é o projeto mais pesado comparado nesta pesquisa.

› Espera envolvimento contínuo do usuário.

› Define vários artefatos e produtos de trabalho para cada fase do projeto; documentação mais pesada.

› O acesso ao material é controlado por um consórcio, e as taxas podem ser cobradas apenas para acessar o material de referência.

CAPÍTULO 1

Histórico e conceito básico

Segundo Sutherland (2017, p. 3), "Scrum é um framework dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. Scrum é leve, simples de entender e difícil de dominar, é um framework estrutural que está sendo usado para gerenciar o trabalho em produtos complexos desde o início de 1990, não é um processo, técnica ou um método definitivo".

Desde mesmo antes de sua criação até ao longo de seu amadurecimento, o Scrum vem gerando algumas controvérsias. A primeira está relacionada à autoria de sua criação. Muitos profissionais afirmam que os autores do SCRUM foram Jeff Sutherland, John Scumniotales e Jeff McKenna, 1993. E há outros que afirmam que Hirotaka Takeuchi e Ikujiro Nonaka inventaram o Scrum em 1986.

Segundo seu artigo denominado "*Inventing and Reinventing SCRUM in Five Companies*" (2001), que pode ser visualizado pelo link http://faculty.salisbury.edu/~xswang/research/papers/serelated/scrum/inventingscrum.pdf, Jeff Sutherland afirma que implementou o primeiro processo de SCRUM quando era vice-presidente de Tecnologia na *Easel Corporation*, em 1993. O SCRUM foi implementado primeiramente em um projeto de *software* orientado a objetos. Em seguida, foi criada uma ferramenta para automatizar o mapeamento de objetos relacional, e foi auxiliado por *Jeff Mckenna e John Scumniotales*.

Entretanto, suas inspirações surgiram no artigo da HBR intitulado "*The New Product Development Game*" (1986), podendo ser visualizado pelo link https://hbr.org/1986/01/the-new-new-product-development-game, em que *Takeuchi* e Nonaka compararam uma nova abordagem holística da inovação ao esporte do *rugby,* "onde uma equipe tenta percorrer a distância como uma unidade, passando a bola para frente e para trás – pode servir melhor aos requisitos competitivos de hoje".

Portanto, podemos concluir que a formalização da estrutura, as nomenclaturas e os testes reais do SCRUM foram criados por Sutherland, Scumniotales e McKenna, baseando-se na ideia de se comparar com o *rugby* de Takeuchi e Nonaka.

Sua primeira aparição pública foi em 1995, em Austin, Texas, onde Jeff Sutherland e Ken Schwaber apresentaram em conjunto o artigo "*The SCRUM Development Process*", na Conferência de Programação Orientada a Objetos, Sistemas, Idiomas e Aplicações (OOPSLA). O artigo pode ser acessado pelo link https://www.scrum.org/resources/scrum-development-process.

Em 2011, Sutherland e Schwaber reuniram quinze colegas em SnowBird, Utah, e redigiram o manifesto Ágil, que se tornou um documento orientador para projetos de desenvolvimento de *software* e pode ser acessado pelo link https://www.manifestoagil.com.br/.

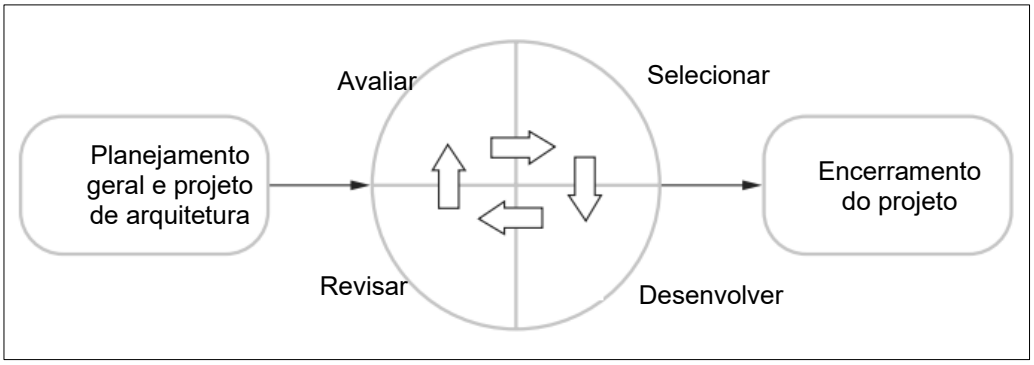
Pelo fato de se tornar uma estrutura popular e considerada por muitos uma das maiores invenções relacionadas ao gerenciamento de desenvolvimento de *software*, suas responsabilidades aumentam, e suas controvérsias surgem no mesmo ritmo. Logo no início de sua ascensão, uma plataforma foi criada para que pudessem ser geradas certificações. Surge a Scrum Alliance e, com ela, demais controvérsias que se deram pela forma como cada entidade aplicaria o Scrum na prática e para a emissão de certificações.

Um fato curioso está relacionado aos guias Scrum e ao SBOK. Um dos criadores do Scrum, *Ken Schwaber*, fez uma crítica ao guia publicado pela Amazon e criado pela ScrumStudy. A crítica pode ser encontrada no Apêndice 1 e acessada pelo link https://abrachan.wordpress.com/2013/11/04/agile-cheating-stories-see-how-the-founders-comment-about-scrum-got-blocked-on-amazon.

Este material está focado em demonstrar aos alunos o embasamento teórico e didático acerca da metodologia Scrum. Esse foco posicionará o futuro arquiteto em *software* a entender e se posicionar dentro de uma equipe que utiliza as boas práticas de Scrum.

Sommerville (2010, p. 50) demonstra como estão divididas as fases no Scrum, seguindo um referencial teórico mais próximo da engenharia de *software*. "A primeira fase define o planejamento geral onde se estabelece os objetivos do projeto e a arquitetura do *software*. Na sequência iniciam-se os ciclos de Sprint, onde cada ciclo realiza um incremento do sistema e, na última fase, ocorre o encerramento do projeto, definindo a documentação e avalia as lições aprendidas".

Figura 6. Processo de Scrum – ciclo Sprint.



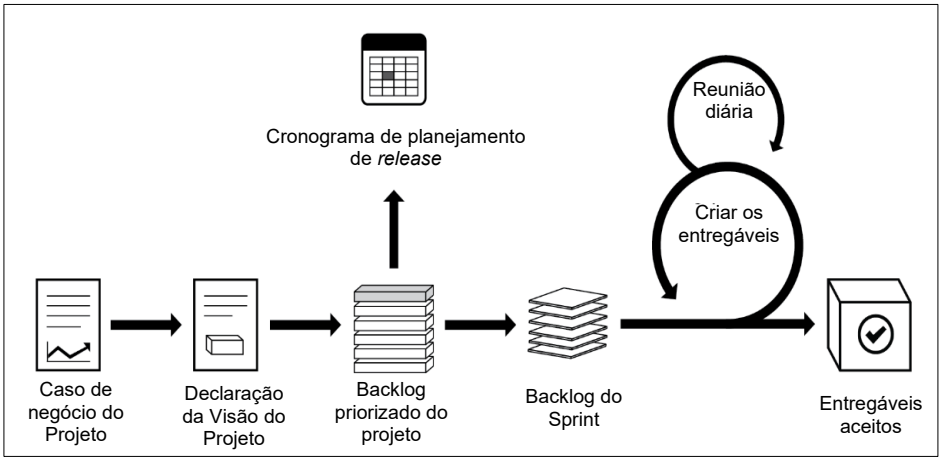
Fonte: Sommerville (2010 p. 50).

O guia SBOK (2017, p. 3) detalha mais a definição dos conceitos básicos de Scrum, com um referencial mais atualizado na prática:

1. o ciclo do Scrum se inicia com uma reunião que resultará na **Visão do Projeto**;
2. o Dono do Produto desenvolve o ***Product Backlog***;
3. na sequência, inicia-se o Sprint que, normalmente, dura entre uma e seis semanas;
4. Durante o **Sprint**, são realizadas Reuniões Diárias, curtas e altamente focadas, em que os membros do time discutem o progresso diário;
5. Perto do final do Sprint, uma **Reunião de Planejamento do Sprint** é realizada, e o *Product Owner* e os *Stakeholders* recebem uma demonstração dos entregáveis;
6. o ciclo Sprint termina com uma **Reunião de Retrospectiva do Sprint**, em que o time apresenta maneiras de melhorar os seus processos e o seu desempenho à medida que avançam para o próximo Sprint.

Já em Sutherland (2017, p. 3), "o *framework* Scrum consiste em times Scrum associados a papéis, eventos, artefatos e regras. Cada componente dentro do framework serve a um propósito específico e é essencial para o uso do Scrum".

Figura 7. Fluxo de Scrum.



Fonte: guia SBOK, p. 2.

SBOK (2017, p. 3) ainda cita algumas das principais vantagens da utilização do Scrum em qualquer projeto.

1. **Adaptabilidade**

O Controle de Processos Empíricos e a Entrega Iterativa são as peças fundamentais para que os projetos sejam adaptáveis.

2. **Transparência**

Toda fonte de informação, *Scrumboard* e o Gráfico *Burndown* do Sprint são compartilhadas com toda equipe, proporcionando um ambiente de trabalho aberto.

3. ***Feedback* Contínuo**

É possível, por meio de processos denominados como Conduzir a Reunião Diária e Demonstrar e Validar o Sprint.

4. **Melhoria Contínua**

Por meio do Refinamento do *Product Backlog*, as entregas melhoram progressivamente, Sprint por Sprint;

5. **Entrega Contínua de Valor**

Os processos iterativos atendem à demanda de entregas exigida pelo cliente.

6. **Ritmo Sustentável**

Os processos são projetados para que as pessoas envolvidas trabalhem em um ritmo sustentável.

7. **Entrega Antecipada de Alto Valor e Eficaz**

Criar o *Product Backlog* garante que as exigências de maior valor ao cliente sejam antecipadas.

8. **Processo de Desenvolvimento Eficiente**

O *Time-boxing* e a minimização de trabalho não essencial conduzem bons níveis de eficiência.

9. **Motivação**

Processos como o de Conduzir a Reunião Diária e de Retrospectiva do Sprint produzem motivação entre os colaboradores.

10. **Solução de Problemas de Forma mais Rápida**

A colaboração de times multifuncionais conduz à resolução de problemas de maneira mais rápida.

11. **Com Foco no Cliente**

Garantido pela abordagem colaborativa com *stakeholders* e ênfase no valor de negócio.

12. **Ambiente de Alta Confiança**

O baixo atrito entre os colaboradores e a confiança são garantidos pelas reuniões diárias e pela Sprint *Retrospective*.

13. **Responsabilidade Coletiva**

O processo de Aprovar, Estimar e Comprometer as Estórias de Usuário permite que os membros do time se sintam responsáveis pelo projeto e por seu trabalho.

14. **Alta Velocidade**

Graças à estrutura de colaboração, é permitido que os times atinjam o seu pleno potencial e alta velocidade.

15. **Ambiente Inovador**

Os processos de Retrospectiva do Sprint e de Retrospectiva do Projeto criam um ambiente de introspecção, aprendizagem e adaptabilidade.

Tabela 4. Scrum x O Modelo Tradicional de Gerenciamento de Projetos.

	Scrum	Modelo Tradicional de Gerenciamento de Projetos
A ênfase está em ...	Pessoal	Processos
Documentação	Mínima	Exaustiva
Estilo de processos	Iterativo	Linear
Planejamento antecipado	Baixo	Alto
Priorização de requisitos	Com base no valor de negócio e atualizado regularmente	Fixo no Plano de Projeto
Garantia de qualidade	Centrada no cliente	Centrada no processo
Organização	Auto-organizada	Gerenciada
Estilo de gerenciamento	Descentralizado	Centralizado
Mudança	Atualizações no *Product Backlog*	Sistema formal de Gerenciamento da Mudança
Liderança	Colaborativa, liderança servidora	Comando e controle
A medição do desempenho	Valor do negócio	Conformidade em relação ao plano
Retorno sobre o investimento	No Início e durante projeto	Final do projeto
Participação do cliente	Alta durante todo o projeto	Varia de acordo com o ciclo de vida do projeto

Fonte: SBOK 2010, p. 19.

Scrum Guidance Body

Na tentativa de se criar uma documentação formal no momento de se implementar o Scrum, cria-se o *Scrum Guidance Body* (SGB). Segundo o SBOK (2017, p. 43), "consiste de um conjunto de documentos e/ou um grupo de especialistas que estão geralmente envolvidos na definição de objetivos relacionados com a qualidade, regulamentações governamentais, de segurança e outros parâmetros-chave da organização. Estes objetivos orientam o trabalho realizado pelo Dono do Produto, Scrum Master e Time Scrum, ajuda a capturar as melhores práticas que devem ser usadas na organização, ele não toma decisões relacionadas ao projeto, porém, atua como uma consultoria ou estrutura de orientação para todos os níveis de hierarquia da organização do projeto; no portfólio, programa e projeto".

Definição de Projeto, Programa e Portfólio

No decorrer do material, citaremos exemplos de alguns conceitos e atuações dos papéis em Projeto, Programa e Portfólio. É muito importante que, neste momento, você entenda o significado de cada um. Utilizaremos o exemplo citado várias vezes no guia SBOK (2017, p. 365).

Projeto

Um projeto é um empreendimento colaborativo com o objetivo de criar novos produtos ou serviços, ou para entregar resultados conforme definido na Declaração da Visão do Projeto. Eles são geralmente afetados por restrições de tempo, custo, escopo, qualidade, pessoas e capacidades organizacionais, e o objetivo do time do projeto é o de criar entregas conforme definido no *Product Backlog*.

Programa

Um programa é um grupo de projetos com o objetivo de entregar resultados de negócio, conforme definido na Declaração da Visão do Programa. O *Backlog Program* incorpora os *Product Backlog* para todos os projetos do programa.

Portfólio

O portfólio é um grupo de programas relacionados, com o objetivo de entregar resultados de negócio, conforme definido na Declaração da Visão do Portfólio. O *Backlog* Priorizado do Portfólio integra os *Backlogs Priorizados* dos Programas para todos os programas do portfólio.

Estudo de Caso

Neste estudo de caso, foi explorado como a *Avanade* usa o Scrum e formou um Nexus+ em sua organização para impulsionar sua transformação digital e comercial, enquanto impulsiona o progresso para se tornar uma empresa ágil. Adotar uma abordagem ágil foi a escolha lógica. Não apenas atendeu aos critérios necessários, mas a *Avanade* já tinha fortes capacidades ágeis que estava usando com os clientes. Então, fazia sentido aplicar essa mesma abordagem internamente. Isso também alinhado com o objetivo estratégico da empresa de entregar maior valor a seus clientes, tornando-se uma empresa mais ágil em geral. Link: https://www.scrum.org/resources/avanade-uses-agile-accelerate-its-digital-and-business-transformation. Acesso em: 01 out. 2019.

CAPÍTULO 2
Introdução aos princípios do Scrum

Segundo o SBOK (2017, p. 21), os princípios do Scrum são a base do *framework* Scrum. Podem ser aplicados a qualquer tipo de projeto ou organização e devem ser respeitados, a fim de assegurar a aplicação adequada do Scrum.

Para entendermos melhor os princípios básicos do Scrum, utilizaremos a organização do SBOK e mencionaremos trechos do Guia Scrum, quando os tópicos coincidirem.

Guia de Papéis

A responsabilidade do bom andamento do Scrum está baseada nos papéis do Time Central do Scrum: Dono do Produto, Scrum Master e a Equipe de Desenvolvimento, que serão detalhados na Unidade III.

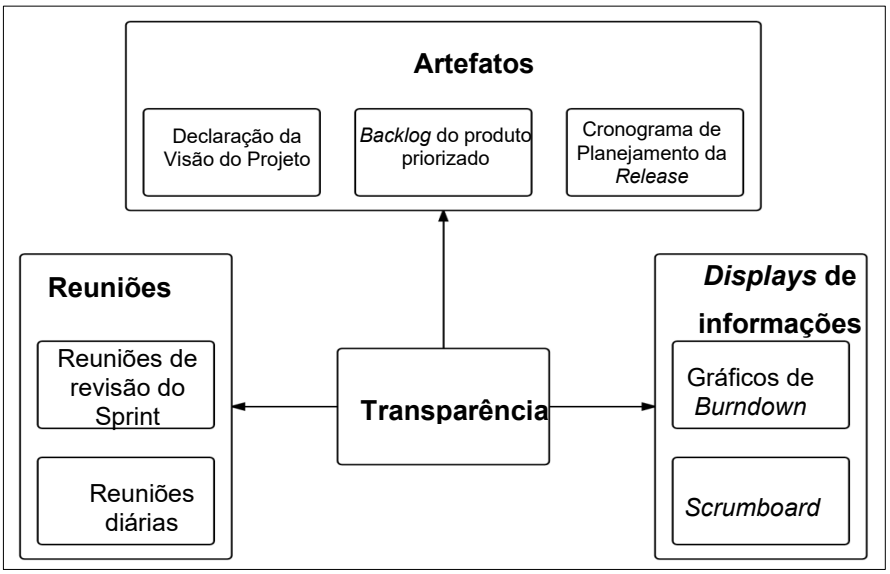
Controle de Processo Empíricos

Segundo Sutherland (2017, p. 4), "Scrum possui suas bases nas teorias empíricas de controle de processo, ou empirismo, onde afirma que o conhecimento vem da experiência e de tomada de decisões baseadas no que é conhecido. O Scrum emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos".

O controle do processo Empírico está implementado em três pilares.

Transparência

Figura 8. Transparência em Scrum.



Fonte: SBOK (2017, p. 22).

Toda informação classificada como significativa deverá estar disponível àqueles que tomarão as decisões.

"A transparência requer que estes aspectos tenham uma definição padrão comum para que os observadores compartilharem um mesmo entendimento comum do que está sendo visto" (SBOK, 2017, p. 22).

Inspeção

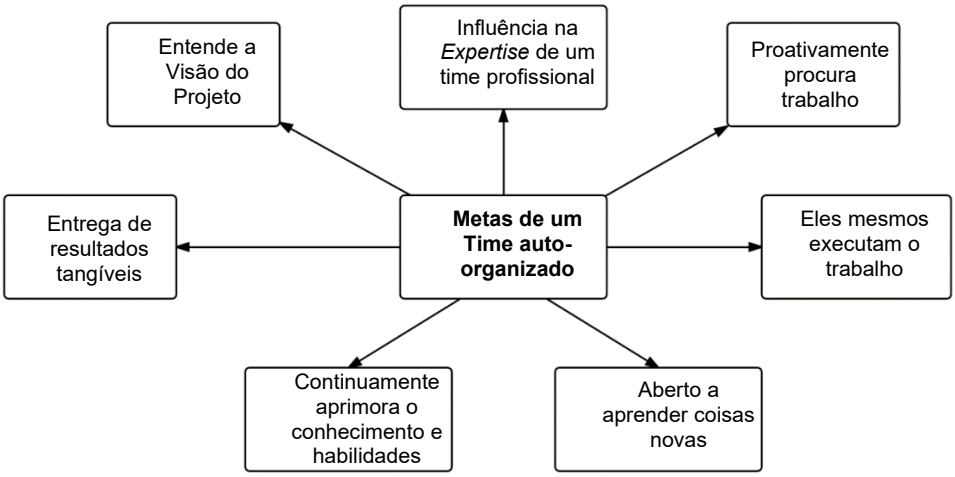
Os artefatos de Scrum devem ser inspecionados para ajustar possíveis variações indesejadas, o que ocasionaria algum problema ao objetivo do projeto.

"Esta inspeção não deve ser tão frequente que atrapalhe o objetivo dos trabalhos. As inspeções são mais benéficas quando realizadas de forma diligente por inspetores especializados no trabalho a se verificar" (SUTHERLAND, 2017, p. 5).

Segundo o SBOK (2017, p. 24), "a Inspeção em Scrum é representada pelo do uso de um *Scrumboard* habitual e de outras fontes de informação que mostrem a evolução do *Team Scrum* em concluir as tarefas do Sprint atual, coleta de *feedback* dos clientes e de outros stakeholders durante os processos de Desenvolver os Épicos, Criar o *Product Backlog* e Conduzir o Planejamento da Release, controle e aceitação das entregas, feitas pelo *Product Ownere* pelo cliente no processo de Demonstrar e Validar o Sprint".

Adaptação

Figura 9. Adaptação em Scrum.



Fonte: SBOK (2017, p. 24).

A adaptação é a consequência da inspeção e transparência. Com elas bem alinhadas, fica fácil efetuar as adaptações aos ajustes de desvios inaceitáveis que prejudicarão o produto final.

Também prescreve eventos formais para inspeção e adaptação:

- planejamento da Sprint;
- reunião diária;
- revisão da Sprint;
- retrospectiva da Sprint;
- identificação dos riscos.

Auto-organização

Quando colaboradores estão confiantes, motivados e possuem liberdades, avançam rumo a novas responsabilidades, efetuando entregas de valor. Isso se refere à auto-organização.

Segundo SBOK (2017, p. 27):

> A auto-organização não significa que os membros do time estejam autorizados a agir da maneira que eles quiserem. Significa apenas que, uma vez que a Visão do Produto é definida no processo de Criar a Visão do Projeto, o Dono do Produto, o Scrum Master e o Time Scrum são

identificados. A experiência do time é utilizada para avaliar as entradas necessárias para executar o trabalho planejado para o projeto. Esse julgamento e experiência são aplicados a todos os aspectos técnicos e de gerenciamento do projeto durante o processo de Criar os Entregáveis.

Os principais objetivos de times auto-organizados, segundo SBOK (2017, p. 27), são:

» compreender a Visão do Projeto;

» estimar Estórias;

» criar tarefas de forma independente durante o processo de Criar as Tarefas;

» aplicar e aprimorar os seus conhecimentos;

» entregar resultados tangíveis e de valor

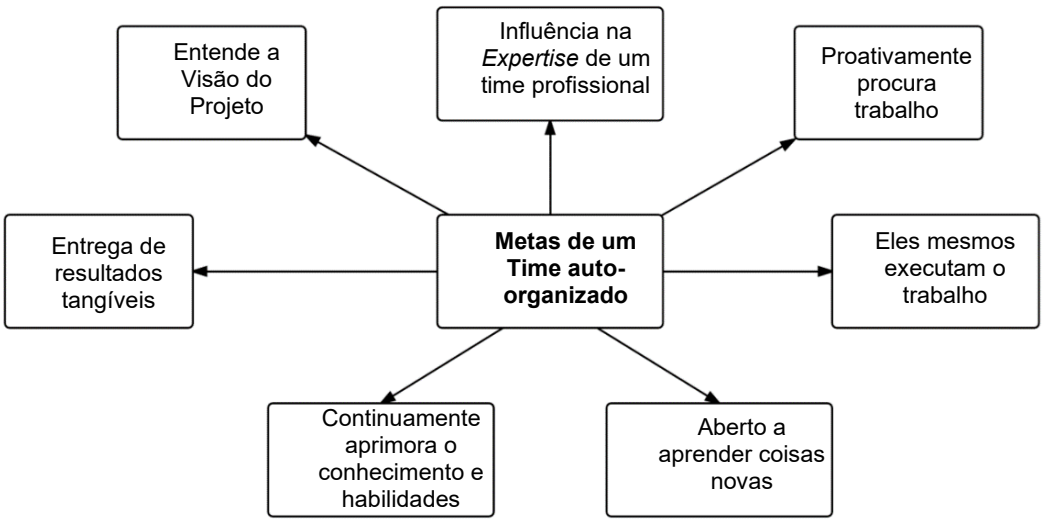
» resolver em conjunto problemas individuais abordados durante as Reuniões Diárias;

» esclarecer quaisquer discrepâncias ou dúvidas e estar aberto para aprender coisas novas;

» atualizar o conhecimento e a habilidade de forma contínua;

» manter a estabilidade dos membros do time durante toda a duração do projeto.

Figura 10. Objetivos de um time auto-organizado.



Fonte: SBOK (2017, p. 27).

Colaboração

Segundo SBOK (2017, p. 28), "a Colaboração em Scrum refere-se ao Time Central do Scrum trabalhando e interagindo simultaneamente com os Stakeholders para criar e validar as entregas do projeto, para assim alcançar os objetivos delineados na Visão do Projeto. É imprescindível notar a diferença entre cooperação e colaboração. A Cooperação ocorre quando o produto do trabalho consiste na soma dos esforços de trabalho de várias pessoas em um time. A Colaboração ocorre quando um time trabalha em conjunto, contribuindo uns com os outros para produzir algo maior".

O Manifesto Ágil (*Fowler e Highsmith*, 2001) enfatiza a "colaboração do cliente acima da negociação de contrato. Assim, o *framework* Scrum adota uma abordagem em que os membros do Time Central do Scrum (Dono do Produto, Scrum Master e Time Scrum), colaboraram uns com os outros e com os Stakeholders para criar os entregáveis que proporcionem o maior valor possível para o cliente. Essa colaboração ocorre durante todo o projeto".

Segundo SBOK (2017, p. 29 e 30), "colaboração garante que os seguintes benefícios do projeto sejam realizados: a necessidade de mudanças devido a requisitos mal esclarecidos é minimizada; os riscos são identificados e tratados de forma eficiente; o verdadeiro potencial do time é realizado; a melhoria contínua é assegurada através de lições aprendidas."

Priorização baseada em valor

Herdando um dos pilares da metodologia ágil, um dos objetivos do Scrum é oferecer o máximo valor ao negócio, em um período cada vez menor. Contando com várias ferramentas e artefatos, foi provado que é extremamente possível de se implementar.

Uma de suas ferramentas mais eficazes para atingir tais objetivos é a priorização baseada em valor, e quanto mais cedo ela for feita, mais eficiente será o projeto.

Priorizado do Produto, três fatores são considerados:

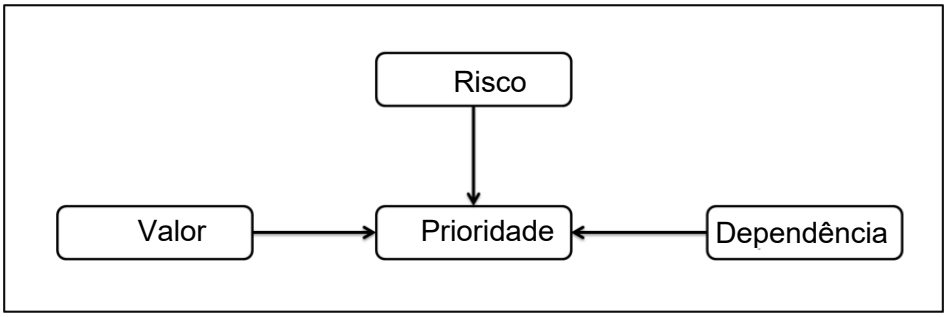
1. valor;

1. risco ou incerteza;

2. dependências.

A priorização resulta em entregas que satisfazem os requisitos do cliente com o objetivo de fornecer o maior valor de negócio no menor tempo possível.

Figura 11. Priorização Baseada em Valor.



Fonte: SBOK (2017, p. 33).

Time-boxing

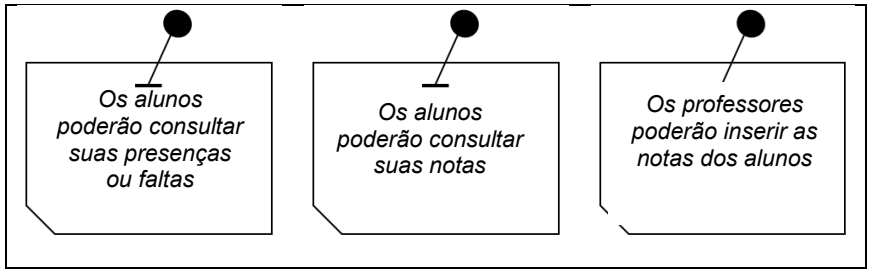
Para tratar do cronograma, o Scrum utiliza o conceito de tempo por meio de um artefato chamado *Time-boxing*. Segundo o SBOX (2017, p. 33), "o *Time-boxing* propõe a fixação de um certo intervalo de tempo para cada processo e atividade de um projeto Scrum, que garante com que os membros do Scrum não usem muito tempo (ou pouco) em um trabalho exclusivo, e não gastem o seu tempo e energia em um trabalho no qual eles tenham menos conhecimento".

Algumas das vantagens de *Time-boxing* são a eficiência no processo de desenvolvimento e a redução de despesas gerais. O *Time-boxing* está presente em todas as etapas, todas as tarefas, como nas reuniões diárias e para evitar *gold-plating*.

CAPÍTULO 3
Requisitos e *User Stories*

No *Produtct Backlog,* são efetuadas entradas que costumam ser escritas em forma de *User Stories* ou Estórias de Usuários. Elas são compostas de uma breve narrativa curta, um nome, que conta a estória de um usuário utilizando o sistema e a condução para que ela seja concluída. Dessa forma, sua principal vantagem está na objetividade, pois se concentra exatamente nas necessidades do usuário, sem entrar em detalhes de como alcançá-lo. As Estórias nada mais são do que algumas frases em linguagem simples que descrevem o resultado desejado. Elas não entram em detalhes, e os requisitos podem ser adicionados mais tarde, uma vez acordado pela equipe.

Figura 12. Estórias.



Fonte: elaboração própria do autor.

Por serem descrições curtas, as estórias são muito utilizadas em kanban. As estórias são anexadas ao kanban como uma lista de pendências sendo executadas no fluxo de trabalho que ajudam a melhorar o planejamento de estimativas de *Sprints*. Graças a esse processo, as equipes de kanban gerenciam melhor o trabalho em andamento, chamado de WIP, como já estudamos, e ainda refinam seus fluxos de trabalho.

Segundo *Rehkopf* (2017), "uma história de usuário é a menor unidade de trabalho em uma estrutura ágil. É um objetivo final, não um recurso, expresso da perspectiva do usuário do software, seu objetivo é definir como será entregue algo de valor ao cliente, que podem ser clientes internos ou externos, ou membros que depende de sua equipe".

Existem várias recomendações no modo de criar estórias de usuários. A recomendação do Scrum Institute pode ser acessada pelo link https://www.scrum-institute.org/Scrum_User_Stories.php. Na frase:

Como [**ator**], eu [**quero** | **preciso**] [**ação**] para que [**realize**].

- O **Ator** é um usuário que normalmente é o proprietário da estória. É mais prático para o contexto utilizar o tipo de usuário que será no sistema: administrador, cliente conectado, visitante autenticado.
- A **Ação** é o que o ator quer fazer no sistema. Ela pode ser definida como obrigatória ou não.
- A **Realização** é o desejo do ator para alcançar a ação.

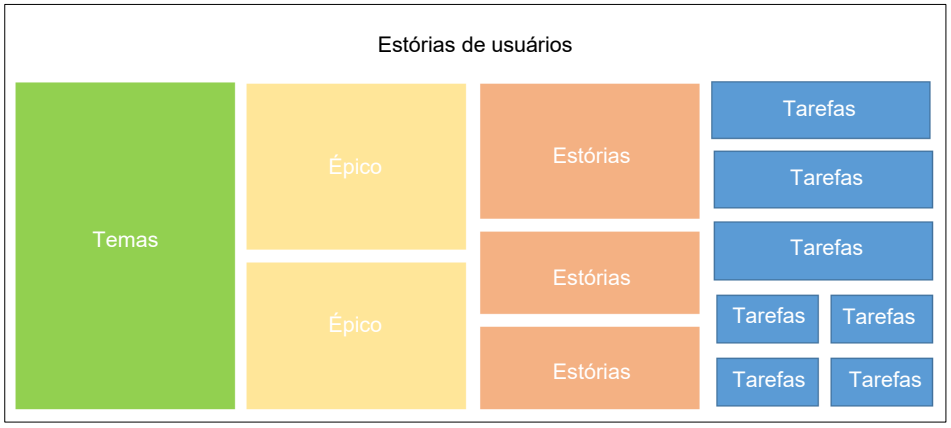
Muitos são os benefícios das estórias:

- as equipes mantêm o foco nos usuários e nas tarefas que precisam ser marcadas;
- uma coleção de estórias mantém a equipe focada nas solução de problemas de usuários reais;
- definindo como atingir o objetivo, a equipe trabalha junta para servir ao usuário;
- incentivam a criatividade da equipe;
- a equipe desfruta de pequenos desafios.

Em componentes maiores de estruturas ágeis, as estórias podem ser encontradas em forma de épicos. Os épicos geralmente são histórias de usuários com prioridade mais baixa, porque, uma vez que o épico se aproxima da parte superior da pilha de itens de trabalho, ela é reorganizada em menores. Não faz sentido desagregar um épico de baixa prioridade, porque você estaria investindo tempo em algo que talvez nunca consiga abordar.

Um conjunto de estórias de usuários relacionadas é chamado de Temas, e a razão para que sejam agrupadas pode definir uma regra de negócio, provenientes de um mesmo épico.

Figura 13. Estrutura de *User Stories*.



Fonte: elaboração própria do autor.

Todo processo de desenvolvimento de *software* está baseado em fluxo de trabalho que inclui a própria criação das estórias. Depois que a estória é devidamente escrita, temos que integrá-la ao fluxo de trabalho. As equipes discutirão quais estórias serão abordadas nos Sprints, seus requisitos e funcionalidades, suas prioridades, com base em complexidades.

Scrum *Planning*

No Scrum, para se definir a complexidade das tarefas e assim definir suas prioridades, o Scrum *Game Poker* ou Scrum *Planning* é muito utilizado. Primeiramente é preciso definir uma unidade de medida para determinada tarefa. Normalmente se utiliza uma unidade abstrata chamada *Story Point*. Essa unidade surge de forma rápida e sem a rigidez da Engrenharia de *software* tradicional. Segundo Cohn (2006, p. 36), "os *Storys Points* são uma unidade de medida para expressar o tamanho geral de uma história do usuário, recurso ou outro trabalho. Quando estimamos com pontos da história, atribuímos um valor em pontos para cada item".

Existe muita controvérsia em se associar os *Story Points* com Tempo. Isso é muito relativo, possui um comportamento diferente entre equipes e linguagens de programação a se utilizar. Portanto, deixaremos para que os grandes mestres filosóficos de Scrum aperfeiçoem suas técnicas. Enquanto isso, podemos obter nossas métricas de diversas maneiras.

Para que se inicie a definição de complexidade, algum membro da equipe lê em voz alta uma estória, e o restante da equipe, programadores, testadores, menos o *Product Owner* e Scrum Master, tentará entrar em um acordo para definir sua complexidade, tentar compreendê-la, ter a certeza de que todos também entenderam e fornecer uma estimativa. Em seguida, cada membro escolhe uma carta do baralho que representa a sua estimativa para a Estória de Usuário. Se a maioria, ou todos os membros do time, selecionar a mesma

carta, a estimativa indicada nessa carta será a estimativa para a Estória de Usuário. Se não houver um consenso, os membros do time discutem as razões pelas quais selecionaram diferentes cartas ou estimativas.

O Scrum Master será o mediador em caso de os membros da equipe não chegarem a uma definição. O *Product Owner* será o responsável por explicar a estória caso seja necessário.

Após essa discussão, eles escolhem novamente uma carta. Essa sequência continua até que todas as suposições sejam compreendidas, mal-entendidos sejam resolvidos e o consenso ou acordo seja alcançado.

Não podemos deixar o Scrum Master ou o *Product Owner* participar da definição de complexidade do produto. Somente a equipe que desenvolverá, testará etc. sabe a real complexidade do trabalho. O Scrum Master e o *Product Owner* estarão interessados que o produto seja entregue o quanto antes e, muitas vezes, passando sob requisitos como a qualidade.

A equipe de programação recebe um baralho com cartas enumeradas com a sequência de Fibonacci: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100. Também possui as cartas interrogação, infinito e a carta café.

» **Carta zero (0)**

A carta zero é usada para quando a tarefa já está pronta.

» **Carta meio (1/2)**

A carta meio é utilizada quando a tarefa está muito fácil, porém não está pronta.

» **Cartas de um a três (1, 2, 3)**

Essas carta são utilizadas para tarefas relativamente fáceis.

» **Cartas de cinco a treze (5, 8, 13)**

Quando as tarefas começam a apresentar certo grau de complexidade.

» **Cartas de vinte e quarenta (20, 40, 100)**

Tarefas relativamente complexas e que certamente terão atenção especial na priorização.

» **Carta infinita**

A carta infinita é para quando a tarefa é muito grande e a equipe não se sente à vontade em definir um valor.

» **Carta interrogação (?)**

A carta interrogação é utilizada quando a equipe está com dúvidas sobre a estória.

» **Carta café**

A carta café funciona para quando o time precisa de uma pausa.

Fist of Five

O *Fist of Five* é mais um mecanismo para chegar a um consenso. Lembrando que todas essas técnicas são fáceis, rápidas e práticas. Depois de tudo lido, os membros da equipe utilizam seus dedos como escala, de um a cinco. Em todas as técnicas, os membros são convidados a dar explicação sobre o motivo de ter mostrado aquele valor e de expressar quaisquer problemas ou preocupações. Assim que o time discutir o assunto, uma decisão coletiva será tomada.

O número de dedos usados na votação indica o nível de acordo e desejo de discussão:

1. um dedo: discordo do grupo ou tenho dúvidas;
2. dois dedos: discordo do grupo e quero discutir os problemas;
3. três dedos: incerteza e apoio o consenso do grupo;
4. quatro dedos: concordo com o grupo, mas gostaria de discutir os problemas;
5. cinco dedos: concordo, sem restrições, com o grupo.

Wideband Delphi

A *Wideband Delphi* é uma técnica de estimativa que se assemelha muito, em seu propósito, com as demais: determinar a quantidade de trabalho que está envolvida e quanto tempo vai demorar para esse trabalho ser concluído. A diferença, entretanto, é que os indivíduos, **anonimamente**, fornecem estimativas para cada recurso, e essas estimativas iniciais são então adicionadas em um gráfico.

A Equipe discute os fatores que influenciaram em suas estimativas e, se necessário, esse processo é repetido até que as estimativas dos indivíduos sejam similares, ou um consenso possa ser alcançado para uma estimativa final.

Tamanho Relativo/Pontos da Estória

Essa abordagem atribui um valor de ponto da estória baseado em uma avaliação geral do tamanho de uma Estória de Usuário, considerando o risco, a quantidade de esforço exigido e o nível de complexidade. Essa avaliação será realizada pelo Time Scrum, e um valor de ponto da estória será atribuído. Além de serem usados para estimar o custo, os pontos da estória também podem ser usados para estimar o tamanho geral de uma Estória de Usuário ou característica. Uma vez que a avaliação é feita para uma Estória de Usuário do *Product Backlog*, o Time Scrum pode então avaliar outras Estórias de Usuários relacionadas a essa primeira estória.

Por exemplo, uma característica com um valor de ponto da estória igual a dois deve ser duas vezes mais difícil de ser concluída do que uma característica com um valor de ponto da estória igual a um; uma característica com um valor de ponto da estória igual a três deve ser três vezes mais difícil de ser concluída do que uma característica com um valor de ponto da estória igual a um.

Estimativa de Afinidade

É uma técnica utilizada para estimar rapidamente um grande número de Estórias de Usuário. Utilizando os mecanismos mais simples, como *post it*, o time coloca as Estórias de Usuário em uma superfície, como parede, lousa ou quadro de vidro, na ordem do menor para o maior. Para isso, cada membro do time começa com um subconjunto de Estórias de Usuário do *Product Backlog*, colocando de acordo com o tamanho relativo. Nessa primeira parte, não haverá discussões. Depois que todo mundo coloca as suas Estórias de Usuário na parede, o time as analisa e pode reposicioná-las de acordo com o que achar mais apropriado. Essa segunda parte envolve discussão. Finalmente, o *Product Owner* vai indicar algumas categorias de dimensionamento na parede, que podem ser pequena, média ou grande, ou podem ser numeradas usando valores de ponto da estória para indicar o seu tamanho relativo. O time, então, moverá as Estórias de Usuário para essas categorias como parte da etapa final desse processo. O grande benefício desse processo é a transparência.

Estimativa de Intervalo

Nesse processo, as estimativas para os projetos devem ser apresentadas em intervalos e devem ser baseadas no nível de confiança que o time tem em cada estimativa. O intervalo pode ser estreito quando o time está confiante e amplo quando o time não está tão confidente.

CAPÍTULO 4
Product Backlog

Segundo Kniberg (2007, p. 9), "o *Product Backlog* é o coração do Scrum e é o ponto de partida para as demais fases. Ele é basicamente uma lista de requisitos, estórias, ou coisas que o cliente deseja e são descritas utilizando a terminologia do cliente, muitas vezes chamados de itens do *backlog*". Esses itens podem ser de natureza técnica ou centrados no usuário na forma de estórias de usuários e são priorizados e ordenados de acordo com o nível de detalhe. Todas as entradas são estimadas.

Existem características do *Product Backlog* que o diferenciam de uma simples lista de tarefas. Por exemplo, cada entrada sempre agregará valor ao cliente. Entradas sem nenhum valor ao cliente podem ser consideradas desperdícios. Essas entradas são incluídas para explorar as necessidades do cliente, requisitos funcionais e não funcionais, correção de defeitos, preparação do ambiente, novo produto etc. Incidentes, entretanto, poderão ser evitados ou reduzidos quando as tarefas não agregarem valor diretamente à funcionalidade, porém aumentam sua qualidade em longo prazo.

Product Backlog é um documento ativo, pois é alterado ao longo de todo o projeto. Com certeza, novos requisitos serão adicionados, ou os requisitos existentes modificados constantemente, excluídos, ou ainda pode ser incluídos mais detalhes. A equipe estará preparada para toda essa modificação, pois já sabem que os requisitos, no Scrum, diferentemente das técnicas tradicionais, não são mais congelados desde o início.

Segundo o Scrum Institute, por meio do link https://www.scrum-institute.org/The_Scrum_Product_Backlog.php, "em vez disso, o conjunto final de requisitos no *Product Backlog* Scrum também é desenvolvido iterativamente, junto com o software resultante. Isso é diferente da engenharia de requisitos tradicional, mas permite maximizar o valor do cliente e minimizar o esforço de desenvolvimento".

No *Product Backlog*, não se deve inserir detalhes dos requisitos, pois esse nível de detalhamento deverá ser definido, juntamente com o cliente, no momento da especificação de Sprint, e quem define sua distribuição é a equipe de Scrum. As entradas do *Product Backlog* devem ser estimadas utilizando as técnicas vistas no Capítulo 3. Elas são usadas para priorizar as entradas e planejar liberações. Inicialmente, a Equipe Scrum e o *Product Owner* escrevem tudo o que podem pensar facilmente, e isso, quase sempre, é mais do que suficiente para um primeiro Sprint. Nada é feito sem que

esteja no *Product Backlog*, mesmo que seja para agradar o cliente, perdendo tempo desnecessário (*gold-plating*[1]).

Segundo o SBOK (2017, p. 170), "durante a criação do *Product Backlog*, podem incluir informações sobre as regras, regulamentos, padrões e melhores práticas, para seu desenvolvimento. Também são utilizadas metodologias de priorização, como as mesmas utilizadas nas Estórias de usuários".

Para a criação dos Épicos, podem ser incluídas regras, regulamentações ou padrão de melhores práticas, pode existir um time de especialistas de negócios, arquitetos, desenvolvedores sênior, que geralmente não fazem parte do time que atua diretamente no projeto e que auxiliem o *Product Owner* na criação do *Product Backlog* Priorizado.

O *Product Backlog* deve ser mantido em um processo contínuo que compreende as seguintes etapas:

- conforme novos itens vão sendo descobertos, eles são documentados, adicionados à lista. Quando existirem, poderão ser alterados ou excluídos, conforme necessidades;
- a priorização do *Product Backlog,* em que os itens mais importantes são movidos para o topo;
- preparando as entradas de alta prioridade para a próxima Reunião de Planejamento da Sprint;
- estimando, sempre que preciso, as entradas no *Product Backlog* Scrum.

Podemos basear o *Product Backlog* três fatores:

- **Valor**

  O *Product Owner* garante a entrega por ordem de prioridade. Produtos valiosos podem não fazer parte da primeira *release* se houverem outros produtos de maior valor que sejam suficientes para a primeira *release*.

- **Risco e Incerteza**

  Os produtos com maior grau de riscos e incertezas também farão parte da priorização do *Product Backlog*. Quando essas ações de riscos são priorizadas no *Backlog*, o resultado é o *Product Backlog* de Risco Ajustado. No início do projeto, os ajustes de riscos são muito efetuados.

1 Expressão usada quando o que será feito for supérfluo.

» **Dependências**

Não tem como criar um *Product Backlog* Priorizado em que as estórias de usuários não sejam dependentes. Isso acontece mesmo entre requisitos funcionais e não funcionais. Essas dependências podem dificultar a priorização das Estórias. Quando isso acontecer, as duas formas mais comuns para solucionar são: a divisão de uma única estória em várias partes ou a combinação de estórias interdependentes.

As equipes de Scrum são preparadas para aceitar as mudanças em todas as etapas do Scrum, e no desenvolvimento do *Product Backlog* não seria diferente. O *Product Backlog* evolui à medida que o produto e o ambiente em que será usado evoluem; é dinâmico; muda constantemente para identificar o que o produto precisa para ser apropriado, competitivo e útil.

Durante o refinamento do *Product Backlog*, os itens são inspecionados e revisados. O Time de Scrum decide como e quando o refinamento está "Pronto". Esse refinamento usualmente não consome mais de 10% da capacidade do Time de Desenvolvimento. Contudo, os itens do *Product Backlog* podem ser atualizados a qualquer momento pelo *Product Owner* ou a critério deste.

"O refinamento do *Product Backlog* é o ato de adicionar detalhes, estimativas e pedidos aos itens do *Product Backlog*. Esse é um processo contínuo no qual o *Product Ownere* a Equipe de Desenvolvimento colaboram nos detalhes dos itens do *Product Backlog*. Durante o refinamento do *Product Backlog*, os itens são revisados e revisados" SBOK (2017, p. 86).

Os itens do *Product Backlog* de ordem mais alta (topo da lista) devem ser mais claros e mais detalhados que os itens de ordem mais baixa. Estimativas mais precisas são feitas baseadas em maior clareza e maior detalhamento. Quanto menor a ordem na lista, menos detalhes. Os itens do *Product Backlog* que ocuparão o Time de Desenvolvimento na próxima Sprint são mais refinados, de modo que todos os itens possam ser "Prontos" dentro do *time-boxed* da Sprint. Os itens do *Product Backlog* que podem ser "Prontos" pelo Time de Desenvolvimento dentro de uma Sprint são considerados "Preparados" para seleção no Planejamento da Sprint. Itens do *Product Backlog* geralmente adquirem esse grau de transparência por meio das atividades de refinamento descritas acima.

O Time de Desenvolvimento é responsável por todas as estimativas. O *Product Owner* deve influenciar o Time de Desenvolvimento, ajudando no entendimento

e nas decisões conflituosas de troca, mas as pessoas que realizarão o trabalho fazem a estimativa final.

Critérios de Aceitação e *Product Backlog*

Veremos mais adiante que, em determinado momento, serão apresentados os resultados dos trabalhos executados pela Equipe. Veremos também que esse trabalho poderá ser aceito ou não, normalmente pelo *Product Owner*. Porém, ele deve obedecer a alguns princípios acordados para ter a certeza das decisões que tomará.

Segundo o SBOK, (2017, p. 86), "o *Product Backlog* é um documento de requisitos individuais que definem o escopo do projeto, fornecendo uma lista de prioridades das características do produto ou serviço a serem entregues pelo projeto, os recursos necessários são descritos na forma de Estórias de Usuário, que são requisitos específicos descritos por vários *stakeholders*, no que se refere ao produto ou serviço proposto". Sendo assim, para cada Estória de Usuário, teremos Critérios de Aceitação.

Quem desenvolve os Critérios de Aceitação é o *Product Owner*, de acordo com os critérios dos requisitos do cliente. Ele comunica ao Scrum *Team* e, se houver alguma discordância, busca um acordo. Os Critérios de Aceitação devem conter as condições que as Estórias de Usuário devem satisfazer, e os Critérios de Aceitação, claramente definidos, são muito importantes para a entrega da funcionalidade de forma eficaz e no prazo.

Ao final dos Sprints, cabe ao *Product Owner* utilizar esses critérios para verificar as entregas concluídas, podendo ***aceitar*** ou ***rejeitar*** as entregas individuais e suas respectivas Estórias de Usuário. Se aceitas, a Estória de Usuário será considerada "**Pronta"**. O Tema Scrum precisa ter muito bem definido os critérios que o *Product Owner* utilizou para definir o "Pronto", pois isso ajudará a esclarecer os requisitos, aderir a normas de qualidade e a pensar a partir da perspectiva do usuário, enquanto trabalham com as suas Estórias.

Definição de "Pronto"

Segundo o SBOK (2017, p. 8), "existe uma diferença fundamental entre os 'Critérios de Pronto' e os 'Critérios de Aceitação'. Enquanto que os Critérios de Aceitação são exclusivos para Estórias de Usuário individuais, os Critérios de Pronto são um conjunto de regras que são aplicáveis a todas as Estórias de Usuário em um determinado Sprint".

Os Critérios de Pronto geralmente podem incluir:

- avaliação por outros membros do time;
- conclusão do teste unitário e da qualidade da Estória de Usuário;

» conclusão de toda a documentação relacionada com a Estória de Usuário;

» demonstração para os *stakeholders* e/ou representantes do negócio.

Da mesma maneira que acontece com os Critérios de Aceitação, todas as condições dos Critérios de Pronto devem ser satisfeitas para que a Estória de Usuário seja considerada **Pronta**.

O Time Scrum deve utilizar uma lista de verificação dos Critérios Gerais de Pronto para garantir que uma tarefa foi concluída e que o resultado atende à Definição de Pronto. Uma definição clara de Pronto é fundamental, pois ajuda a eliminar a ambiguidade e permite ao time aderir aos padrões de qualidade exigidos. A definição de Pronto é tipicamente determinada e documentada pelo *Scrum Guidance Body*.

Entregáveis Aceitos e Entregáveis Rejeitados

O SBOK (2017, p. 250) enfatiza na fase de *Release* que são considerados Entregáveis Aceitos os que podem ser liberados para o cliente, se assim o desejarem. Após cada Reunião de Revisão do Sprint, uma lista de Entregáveis Aceitos é mantida e atualizada. Se um "***Entregável***" não cumprir os Critérios de Aceitação definidos, não será considerado aceito e, geralmente, será transferido para um Sprint subsequente para a correção de quaisquer problemas. Isso é altamente indesejável, porque o objetivo de cada Sprint é que os entregáveis satisfaçam os Critérios de Aceitação.

Existe a possibilidade de os Entregáveis serem rejeitados ou não atenderem ao critério de aceitação. Se isso acontecer, as Estórias de Usuário associadas a esses entregáveis serão adicionadas ao *Product Backlog*, de modo que tais entregáveis possam ser considerados como parte de um Sprint posterior.

PAPÉIS

UNIDADE III

Os papéis no Scrum são separados para deixar claro o que cada colaborador da equipe fará e, segundo o SBOK (2017, p. 39), podemos dividi-los em duas categorias.

Papéis Centrais

São papéis que estarão diretamente envolvidos na produção do projeto, estão comprometidos com o resultado e o valor, e são os responsáveis pelo sucesso do Projeto como um todo. Existem três papéis que são considerados o **Scrum *Team***, ou seja, o Time Central do Scrum. São eles:

- ***Product Owner*:** responsável por atingir o maior valor de negócio para o projeto, pela organização das necessidades dos clientes e pela manutenção da justificativa de negócio para o projeto;
- **Scrum *Master*:** é um facilitador que garante ao Time Scrum o fornecimento de um ambiente ideal para concluir o projeto com sucesso;
- ***Team* Scrum:** é o time responsável pelo desenvolvimento das entregas do projeto e por entender os requisitos especificados pelo *Product Owner*.

O SBOK (2017, p. 56) faz um resumo das responsabilidades dos papéis do Scrum *Team*.

Tabela 5. Resumo das Responsabilidades Relevantes à Organização.

Papéis	Responsabilidades
Stakeholder(s)	É um termo coletivo que inclui clientes, usuários e patrocinadores: interagir frequentemente com o Dono do Produto, Scrum *Master* e com o Time Scrum, para fornecer *inputs* e facilitar a criação das entregas do projeto.
Dono do Produto	Criar os requisitos iniciais gerais do projeto e manter o projeto em andamento, nomear as pessoas adequadas para os papéis de Scrum *Master* e Time Scrum, fornecer os recursos financeiros para o início do projeto e durante o seu andamento, determinar a Visão do Projeto, avaliar a viabilidade e garantir a entrega do produto ou serviço, garantir a transparência e clareza dos itens do *Backlog* Priorizado do Produto, decidir o conteúdo mínimo para *release* comercial, fornecer os Critérios de Aceitação para as Estórias de Usuário a serem desenvolvidas em um Sprint, inspecionar as entregas, decidir a duração do Sprint.
Scrum *Master*	Garantir que os processos do Scrum sejam corretamente seguidos por todos os membros do time, incluindo o Dono do Produto, assegurar que o desenvolvimento do produto ou serviço está ocorrendo sem problemas e que os membros do Time Scrum têm todas as ferramentas necessárias para a realização do trabalho, supervisionar a Reunião de Planejamento da *Release* e agendar as outras reuniões.
Time Scrum	Assumir a responsabilidade coletiva e garantir que as entregas do projeto sejam criadas de acordo com os requisitos, garantir ao *Product Owner* e ao Scrum *Master* que o trabalho alocado está sendo realizado de acordo com o projeto.

Fonte: SBOK (2017, p. 56).

Papéis não essenciais

Segundo o SBOK (2017, p. 40), “são aqueles papéis que não são obrigatoriamente necessários para o projeto Scrum, e podem incluir membros do time que estão interessados no projeto, que não têm nenhum papel formal no time do projeto, que podem interagir com o time, mas que não podem ser responsáveis pelo sucesso do projeto. Os papéis não essenciais também devem ser levados em consideração em qualquer projeto Scrum”.

CAPÍTULO 1
Product Owner

Segundo Sutherland (2017, p. 6), "o *Product Owner*, ou Dono do Produto, é o responsável por maximizar o valor do produto resultado do trabalho do Time de Desenvolvimento. Como isso é feito pode variar amplamente através das organizações, Times Scrum e indivíduos. Ele faz parte de uma equipe de desenvolvimento de produtos responsável pelo gerenciamento da lista de pendências de produtos, a fim de alcançar o resultado desejado que uma equipe de desenvolvimento de produtos procura alcançar".

O *Product Owner* representa uma pessoa, e não um comitê. Porém pode representar o desejo de um comitê no *Product Backlog*. Mas aqueles que quiserem uma alteração nas prioridades dos itens de *Backlog* devem endereçar ao *Product Owner*. Para o sucesso do *Product Owner*, a organização deverá respeitar suas decisões. Como consequência, é claro que isso significa que o *Product Owner* deve trabalhar em estreita colaboração com o *Team Scrum* e coordenar suas atividades durante toda a vida útil do projeto. Ninguém mais pode dizer à equipe de desenvolvimento que trabalhe com um conjunto diferente de prioridades.

O *Product Owner* é a voz do cliente no projeto, uma vez que ele garante que as necessidades do cliente sejam convertidas em Estórias de Usuário no *Product Backlog* e, posteriormente, utilizadas na criação dos "Entregáveis" do projeto para o cliente.

O gerenciamento do *Product Backlog* pelo *Product Owner* inclui:

- expressar claramente os itens do *Product Backlog*;
- ordenar os itens do *Product Backlog* para alcançar melhor as metas e missões;
- otimizar o valor do trabalho que o Time de Desenvolvimento realiza;
- garantir que o *Product Backlog* seja visível, transparente, claro para todos, e mostrar o que o *Team Scrum* vai trabalhar a seguir;
- garantir que o Time de Desenvolvimento entenda os itens do *Product Backlog* no nível necessário.

O *Product Owner Scrum* tem várias responsabilidades, segundo SBOK (2017, p. 43).

Tabela 6. Responsabilidades do *Product Owner*.

Processos	As Responsabilidades do Dono do Produto
Criar a Visão do Projeto	Definir a Visão, a Patente e Orçamento do Projeto
Identificar *Scrum Master* e o(s) *Stakeholder*(s)	Ajudar a identificar o *Scrum Master* e *Stakeholder*(s) para o Projeto
Formar o Time Scrum	Ajudar a determinar os membros do Time Scrum, um Plano de Colaboração, o Plano de *Team Building* com o(s) *Scrum Master*(s)
Desenvolver o(s) Épico(s)	Criar os Épico(s) e *Personas*
Criar o *Product Backlog*	Priorizar os Itens do *Product Backlog*
Conduzir o Planejamento da *Release*	Criar o Cronograma de Planejamento da *Release* e a duração do Sprint
Criar as Estórias de Usuário	Ajudar a criar as Estórias de Usuário e os critérios de aceitação
Aprovar, Estimar e Comprometer as Estórias de Usuário	Aprovar as Estórias de Usuário
Criar as Tarefas	Explicar as Estórias de Usuário para o Time Scrum, enquanto cria a lista de tarefas
Estimar as Tarefas	Fornecer orientações e esclarecimentos para o Time Scrum na estimativa de esforço para as tarefas
Criar o *Backlog* do Sprint	Esclarecer os requisitos para o Time Scrum, enquanto cria o *Backlog* do Sprint
Criar os Entregáveis	Esclarecer os requisitos de negócios para o Time Scrum
Refinamento do *Product Backlog*	Refinar o *Product Backlog*
Demonstrar e Validar os Sprints	Aceitar/Rejeitar os Entregáveis, fornecer o *feedback* necessário para o Scrum Master e para os Times Scrum, atualizar o Plano da Release no *Product Backlog*
Envio de Entregáveis	Ajudar a implantar a *Release* de Produtos, coordenação feita com o cliente
Retrospectiva do Projeto	Participar de Reuniões de Retrospectiva do Sprint

Fonte: SBOK (2017, p. 43).

Outra função do *Product Owner* é o gerenciamento das partes interessadas, pois é justamente para isso que existe a divisão dos papéis. Para tanto, os papéis não essenciais não deverão se comunicar diretamente com o *Team* Scrum, mas sim com o *Product Owner*, que deverá coletar as funcionalidades necessárias, com os diferentes *stakeholders*, que, em forma de itens de *Backlog*, deverão ser filtrados e entregues para a equipe priorizá-los. Nesse momento, é importante que o *Product Owner* Scrum e a Equipe Scrum trabalhem juntos muito de perto.

Outras responsabilidades do *Product Owner* relacionados diretamente ao Portfólio e ao Serviço ou Produto e Programa:

- determinar os requisitos gerais iniciais do projeto e dar início às suas atividades; isso pode envolver a interação com o *Product Owner* do Programa e com o *Product Owner* do Portfólio, para garantir que o projeto esteja alinhado de acordo com a orientação dada pela alta administração;
- representar o(s) usuário(s) do produto ou serviço com um profundo conhecimento sobre a comunidade dos usuários;
- garantir os recursos financeiros iniciais e em andamento para o projeto;
- focar na criação de valor e, de forma geral, no Retorno sobre Investimento;
- avaliar a viabilidade e garantir a entrega do produto ou serviço.

Product Owner Chefe

O guia SBOK (2017, p. 45) prevê situações em que o projeto seja muito grande. Para isso, quando o Time Scrum é muito grande, haverá a necessidade de um *Product Owner* Chefe, e seu papel é ser responsável pela coordenação do trabalho de vários Donos do Produto.

"O *Product Owner* Chefe prepara e mantém todo o *Product Backlog* para o projeto, coordenando o trabalho entre os Donos do Produto dos Times Scrum. Os Donos do Produto, por sua vez, gerenciam suas respectivas partes no *Product Backlog*. O Product Owner Chefe também interage com o *Product Owner* do Programa para garantir o alinhamento de grandes projetos, com as metas e objetivos do programa".

Dono do Produto do Programa

O SBOK (2017, p. 139) prevê um papel de Dono do Produto do Programa, que é a pessoa responsável por maximizar o valor do negócio para um programa, por articular as necessidades do cliente e por manter a justificativa de negócio para o programa. Gerencia o *Backlog* do Produto do Programa, interage com o Dono do Produto do Portfólio para garantir o alinhamento do programa com as metas e objetivos do portfólio.

CAPÍTULO 2
Scrum *Master*

Segundo Sutherland (2017, p. 6), "o **Scrum *Master* do Produto** é responsável por promover e apoiar o Scrum, ele faz isso ajudando todos a entender a teoria, práticas, regras e valores do Scrum, é um líder-servidor da equipe Scrum, ajuda as pessoas de fora da equipe Scrum a entender quais de suas interações com a equipe Scrum são úteis e quais não são e ajuda todos a mudar essas interações para maximizar o valor criado pela equipe Scrum".

Segundo o guia SBOK (2017, p. 44), "o Scrum *Master* é o 'líder servidor' do Time Scrum, aquele que modera e facilita a interação do time, agindo como motivador e mentor do time. O Scrum *Master* é responsável por garantir que o time tenha um ambiente de trabalho produtivo, protegendo o time de influências externas, removendo qualquer impedimento, e aplicando os princípios, aspectos e processos do Scrum".

Por se tratar de uma peça fundamental no projeto, que pode ter acesso direto a outros *stakeholders*, outras equipes etc., o guia SBOK separou as funções correlativas entres as peças essenciais do projeto.

Scrum *Master* atende ao ***Product Owner*:**

- garantir que as metas, escopo e domínio do produto sejam compreendidos por todos da equipe Scrum;
- encontrar técnicas para o gerenciamento eficaz do *Product Backlog*;
- ajudar a equipe Scrum a entender a necessidade de itens claros e concisos do *Product Backlog*;
- compreender o planejamento de produtos em um ambiente empírico;
- garantir que o *Product Owner* saiba como organizar o *Product Backlog* para maximizar o valor;
- compreender e praticar agilidade; e
- facilitar eventos Scrum, conforme solicitado ou necessário.

Scrum Master atende à **Equipe de Desenvolvimento**:

» ajudando a equipe de desenvolvimento em auto-organização e a criar produtos de alto valor;

» remoção de impedimentos ao progresso da equipe de desenvolvimento;

» facilitar eventos Scrum, conforme solicitado ou necessário;

» *coaching* da equipe de desenvolvimento em ambientes organizacionais nos quais o Scrum ainda não foi totalmente adotado e compreendido.

Scrum Master atende **à Organização:**

» liderar e treinar a organização na adoção do Scrum;

» planejando implementações de Scrum dentro da organização;

» ajudar os funcionários e as partes interessadas a entender e implementar o Scrum e o desenvolvimento empírico de produtos;

» causando mudanças que aumentam a produtividade da equipe Scrum;

» trabalhando com outros Scrum *Masters* para aumentar a eficácia da aplicação do Scrum na organização.

A tabela a seguir mostra as várias funções do Scrum *Master* em vários processos do Scrum, baseando-se em SBOK (2017, p. 45).

Tabela 7. Responsabilidades do Scrum *Master* nos vários processos Scrum.

Processos	Responsabilidades do Scrum Master
Identificar o(s) *Stakeholder*(s)	Ajudar a identificar o(s) *Stakeholder*(s) para o projeto
Formar o Time Scrum	Facilitar a seleção do Time Scrum, criação do Plano de Colaboração e do Plano de *Team Building*
Desenvolver o(s) Épico(s)	Facilitar a criação de Épico(s) e de *Personas*
Criar o *Product Backlog*	Ajudar o *Product Owner* na criação do *Product Backlog* e na definição dos Critérios de Pronto
Conduzir o Planejamento da *Release*	Coordenar a criação do Cronograma de Planejamento da *Release*, Determinar a Duração do Sprint
Criar as Estórias de Usuário	Auxiliar o Time Scrum na criação das Estórias de Usuário e em seus Critérios de Aceitação
Aprovar, Estimar e Comprometer as Estórias de Usuário	Facilitar as reuniões do Time Scrum para estimar e Criar as Estórias de Usuário
Criar as Tarefas	Facilitar ao Time Scrum a criação da Lista de Tarefas para o próximo Sprint
Estimar as Tarefas	Auxiliar o Time Scrum em estimar os esforços necessários para completar as tarefas de acordo para o Sprint
Criar o *Backlog* do Sprint	Auxiliar o Time Scrum no desenvolvimento do *Backlog* do Sprint e do Gráfico *Burndown* do Sprint
Criar os Entregáveis	Suportar o Time Scrum na criação das entregas acordadas para o Sprint, Ajudar a atualizar o *Scrumboard* e o Registro de Impedimentos
Conduzir a Reunião Diária	Garantir que o *Scrumboard* o Registro de Impedimentos continuem sendo atualizados

Refinamento do *Product Backlog*	Facilitar as Reuniões de Revisão do *Product Backlog*
Convocar o Scrum de Scrum	Garantir que os problemas que afetam o Time Scrum sejam discutidos e resolvidos
Demonstrar e Validar o Sprint	Facilitar a apresentação de entregas concluídas pelo Time Scrum, para a aprovação do Dono do Produto
Retrospectiva do Sprint	Garantir a existência de um ambiente ideal para o projeto, para o Time Scrum durante os Sprints seguintes
Retrospectiva do Projeto	Representar o Time Central do Scrum, fornecendo lições do projeto atual, se necessário

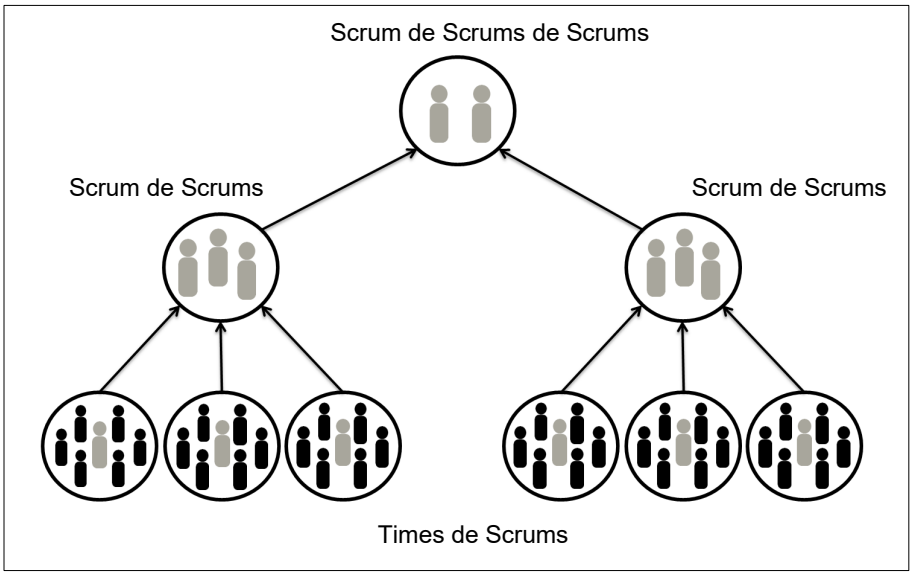
Fonte: SBOK (2017, p. 45).

Em relação à equipe, o Scrum *Master* assegura que sigam os valores, as práticas do Scrum, protege a equipe assegurando que ela não se comprometa excessivamente, atua como facilitador no *Daily Scrum* e torna-se responsável por remover quaisquer obstáculos que sejam levantados pela equipe durante essas reuniões.

Scrum Master Chefe

Com certeza os grandes projetos vão requerer vários Times de Scrum trabalhando em paralelo, e um dos papéis que aumentará conforme o projeto aumenta é o de Scrum *Master*. Todas as informações que serão comunicadas entre os Times serão de responsabilidade do Scrum *Master* Chefe, e a coordenação entre os vários Times Scrum envolvidos será feita por meio da Reunião do Scrum de Scrums (*SOS – Scrum Of Scrum*).

Figura 14. Reuniões do Scrum de Scrums.



Fonte: SBOK (2017, p. 52).

Segundo o SBOK (2017, p. 51), "em uma reunião SOS, estão presentes um representante de cada Time Scrum, que é facilitada por um Scrum *Master* Chefe e seu foco é destinado em áreas de coordenação e integração entre os diferentes Times Scrum".

A escala de reuniões de Scrum pode ter vários níveis dependendo do tamanho da organização. É muito complexo quando as reuniões se tornam Scrums de Scrums de Scrums. Podem não obter benefício nenhum.

Scrum *Master* do Programa

Assim como o Scrum *Master* do Produto, o papel do Scrum *Master* do Programa é facilitar e garantir que todos os times do projeto, no programa, sejam favorecidos com um ambiente propício à conclusão do projeto com sucesso, fornecer diretrizes aos Scrum *Masters* de projetos individuais, remover os impedimentos encontrados pelos diferentes times do projeto, coordenar com o *Scrum Guidance Body*, definir objetivos relacionados com a qualidade e assegurar que os processos do Scrum sejam seguidos durante o programa.

Segundo o SBOK (2017, p. 139), o Scrum *Master* do Programa interage com o Scrum *Master* do Portfólio para garantir o alinhamento do programa com as metas e objetivos do portfólio. Também está envolvido na nomeação de Scrum *Masters* para projetos individuais e em garantir que a visão, os objetivos, os resultados e os lançamentos dos projetos individuais se alinham com os do programa.

CAPÍTULO 3
Equipes de Desenvolvimento

Todo trabalho no Scrum entregue ao cliente é realizado por uma coleção de pessoas trabalhando juntas e organizadas para fornecer as funcionalidades de valor. Para que haja toda essa motivação, todos aderem às mesmas normas, sincronizadas e organizadas, mostrando respeito mútuo, e isso deve ser uma estratégia oferecida desde o início. Essa é a Equipe de Desenvolvimento.

A Equipe de Desenvolvimento não possui uma divisão de organizacional ou funcional, como em uma equipe clássica de desenvolvimento de *software*: programador, analistas, *testers*, arquitetos etc. Todos os membros têm uma única missão, que é o comprometimento, em conjunto, com a entrega do Sprint.

Alguns autores se referem ao Scrum *Team* como Time de Desenvolvimento, uma vez que é responsável pelo desenvolvimento do produto, serviço ou de outro resultado, ou seja, é um grupo de indivíduos que trabalham nas Estórias de Usuário do *Backlog* do Sprint para criar as entregas para o projeto. Entretanto, Sutherland (2017, p. 6) deixa claro: "O Time Scrum consiste em um *Product Owner*, o Time de Desenvolvimento e um *Scrum Master*".

A equipe de desenvolvimento é uma das equipes mais importantes do Time Scrum e é composta por profissionais que realizam os trabalhos e efetuam as entregas para serem testadas no final de cada Sprint. São somente esses membros que criam incremento.

As equipes de desenvolvimento são criadas, organizadas e capacitadas pela organização para estruturar e gerenciar seu próprio trabalho. A sinergia entre os membros otimiza a eficiência e a eficácia gerais da equipe como um todo.

As equipes de desenvolvimento têm as seguintes características:

- **auto-organizadas**: ninguém, nem mesmo o Scrum *Master*, diz à equipe de desenvolvimento como transformar o *Product Backlog* em incrementos de funcionalidade potencialmente liberável;
- **multifuncionais:** possui todas as habilidades necessárias para criar um incremento de produto;
- **sem títulos ou subequipes:** o Scrum não reconhece títulos ou subequipes para os membros da equipe de desenvolvimento, independentemente do trabalho que está sendo realizado pela pessoa;

» **habilidades individuais:** cada membro da equipe de desenvolvimento pode ter habilidades e áreas de foco especializadas, mas a responsabilidade pertence à equipe como um todo.

Tamanho da equipe de desenvolvimento

Não existe um tamanho predefinido para o tamanho de uma Equipe de Desenvolvimento. O lema é: pequeno o suficiente para permanecer ágil e grande o suficiente para concluir um trabalho significativo dentro de um Sprint.

Sutherland (2017, p. 7) dá uma sugestão sobre o tamanho das equipes de desenvolvimento. Menos de três diminuem a interação e resultam em menores ganhos de produtividade, podem encontrar restrições de habilidades durante o Sprint, fazendo com que a equipe de desenvolvimento não consiga fornecer um incremento potencialmente liberável. Ter mais de nove membros requer muita coordenação, pois as grandes equipes de desenvolvimento geram muita complexidade para que um processo empírico seja útil. As funções *Product Owner* e Scrum *Master* só estarão incluídas no tamanho da equipe caso estejam executando o trabalho no *Sprint Backlog*.

Tabela 8. Responsabilidades do Time Scrum em Processos do Scrum.

Processos	Responsabilidades do Time Scrum
Formar o Time Scrum	Fornecer *inputs* para a criação do Plano de Colaboração e Plano de *Team Building*
Desenvolver os Épicos	Garantir uma compreensão clara sobre os Épicos
Product Backlog	Compreender as Estórias de Usuário no *Backlog* Produto
Conduzir o Planejamento da *Release*	Concordar com outros membros do Time Central do Scrum sobre a Duração do Sprint, buscar esclarecer novos produtos, ou mudanças nos produtos já existentes, se houver, no *Backlog* do Produto refinado
Criar as Estórias de Usuário	Fornecer *inputs* para o *Product Owner* na criação das Estórias de Usuário
Aprovar, Estimar e Comprometer as Estórias de Usuário	Estimar as Estórias de Usuário aprovadas pelo Dono do Produto, Comprometer as Estórias de Usuário a serem concluídas no Sprint
Criar as Tarefas	Desenvolver a Lista de Tarefas com base em Estórias de Usuário e dependências acordadas
Estimar as Tarefas	Estimar os esforços para as tarefas identificadas e, se necessário, atualizar a Lista de Tarefas
Criar o *Backlog* do Sprint	Desenvolver o *Backlog* do Sprint e o Gráfico *Burndown* do Sprint
Criar os Entregáveis	Criar os Entregáveis, Identificar riscos e implementar ações de mitigação de risco, se houver, atualizar o Registro de Impedimento e dependências
Conduzir a Reunião Diária	Atualizar o Gráfico *Burndown*, *Scrumboard*, e Registro de Impedimentos, discutir problemas enfrentados por membros individuais, e buscar soluções para motivar o time, identificar riscos, se houver, submeter Solicitações de Mudança, se necessário
Refinamento do *Product Backlog*	Participar em Reuniões de Revisão do *Product Backlog*
Convocar o Scrum de Scrums	Fornecer *inputs* ao Scrum Master para Reuniões do Scrum de Scrums (SoS)
Demonstrar e Validar o Sprint	Demonstrar ao *Product Owner* as entregas concluídas, que requerem aprovação
Retrospectiva do Sprint	Identificar oportunidades de melhorias, se houver, no Sprint atual e concordar com todas as melhorias viáveis para o próximo Sprint
Retrospectiva do Projeto	Participar da Reunião de Retrospectiva do Projeto

Fonte: Sutherland (2017, p. 47).

CAPÍTULO 4
Gerentes de Projetos e Scrum *Master*

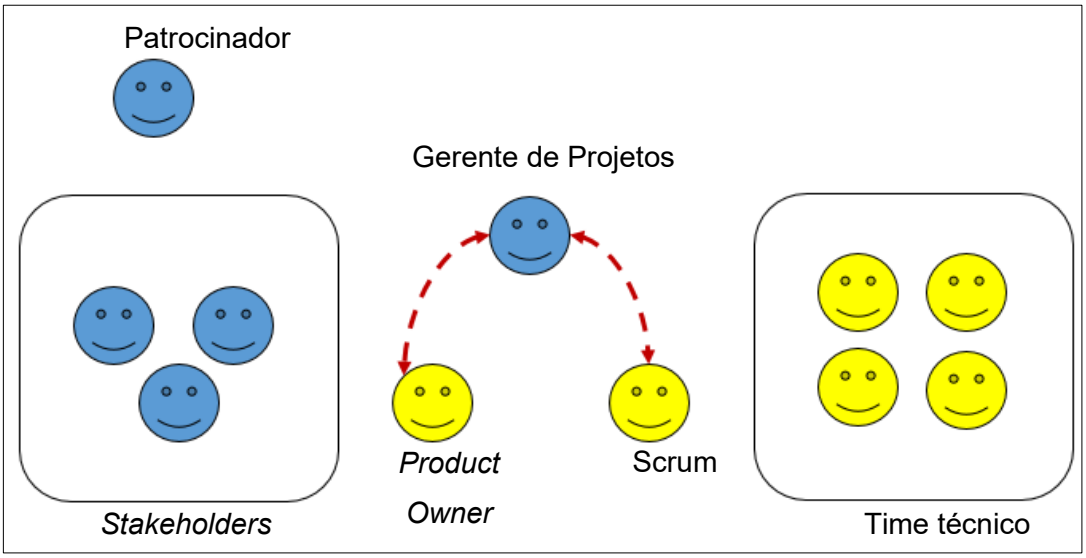
Quando se é um profissional de TI e ainda não teve contato com o Scrum, é muito difícil perceber a diferença entre um Scrum *Master* e um Gerente de Projetos. Entretanto, seus papéis são muito separados um do outro e ambos podem ter abordagens diferentes em projetos.

Segundo o PMBOK (2010, p. 13), "o gerente de projetos é a pessoa designada pela organização executora para atingir os objetivos do projeto. O papel de um gerente de projetos é diferente de um gerente funcional ou gerente de operações. Normalmente, o gerente funcional está concentrado em proporcionar a supervisão de gerenciamento de uma área administrativa e os gerentes de operações são responsáveis por um aspecto do negócio principal".

Segundo Sutherland (2017, p. 6), "o **Scrum *Master*** é responsável por promover e apoiar o Scrum, ele faz isso ajudando todos a entender a teoria, práticas, regras e valores do Scrum, é um líder-servidor da equipe Scrum, ajuda as pessoas de fora da equipe Scrum a entender quais de suas interações com a equipe Scrum são úteis e quais não são e ajuda todos a mudar essas interações para maximizar o valor criado pela equipe Scrum".

Segundo o guia SBOK (2017, p. 44), "o Scrum *Master* é o "líder servidor" do Time Scrum, aquele que modera e facilita a interação do time, agindo como motivador e mentor do time. O Scrum *Master* é responsável por garantir que o time tenha um ambiente de trabalho produtivo, protegendo o time de influências externas, removendo qualquer impedimento, e aplicando os princípios, aspectos e processos do Scrum".

Figura 15. Gerentes de projetos e Scrum *Master*.



Fonte: elaboração própria do autor.

Quando listamos as diferenças entre o Gerente de Projetos e o Scrum *Master*, percebemos que o Gerente de Projetos possui papel fundamental na liderança do Projeto, ou seja, lidera o planejamento para a execução do Projeto, e o Scrum Master desempenha um papel de suporte para os membros da equipe, trabalhando em estreita colaboração com a equipe e garantindo que eles estejam seguindo os princípios do Agile corretamente.

Quando houve a transição do modelo cascata para o ágil, muitas pessoas, querendo resolver o quanto antes essas transição, apenas acreditaram que mudando o nome de Gerente de Projeto para Scrum *Master* bastaria. Entretanto, veremos que não é tão simples assim.

Nas metodologias ágeis, as responsabilidades de um "Gerente" são distribuídas entre as variedades dos membros da equipe, sendo algumas Gerente do Programa, outras do *Product Owner*, outras do Scrum *Master* e outras da Equipe de Desenvolvimento. Percebemos, quando não estamos analisando somente o projeto de Scrum, que o Scrum *Master* não terá a função de Gerente de Projetos. Ao contrário disso, o *Product Owner* está, sim, intimamente alinhado à função de Gerente de Projetos.

Tanto o Gerente de Projetos quanto o *Product Owner* precisam manter uma lista constantemente atualizada de pendências do produto e garantir que o produto atenda aos requisitos de negócios, e, no caso de qualquer alteração no produto, o *Product Owner* deve ajustar e priorizar novamente o *Backlog* atual do produto para ajustar essas alterações e manobrar o projeto.

O Scrum *Master* está lá para desempenhar um papel de facilitador, orientar o *Product Owner* sobre como gerenciar o trabalho em equipe com o uso de *Backlog* do produto, planejamento de Sprint e reuniões. Ele oferece suporte ao *Product Owner* no gerenciamento do trabalho em equipe, treina a equipe e garante que a esta esteja alinhada adequadamente ao processo Scrum, gerencia o processo Scrum, garante sua correta implementação e aumenta o escopo de seus benefícios ao longo do projeto.

O Gerente de Projeto é líder, tomador de decisão e é responsável pelo gerenciamento do projeto, recursos e escopo dos requisitos de negócios. Às vezes, o gerente de projeto precisa garantir que este esteja alinhado com os requisitos de negócios necessários ou não, enquanto o Scrum *Master* precisa cuidar de uma equipe de projeto. Além disso, o Scrum *Master* deve ser um mediador entre o projeto e o cliente.

Vejamos como as funções de Gerente de Projeto e Scrum *Master* diferem entre si em termos discretos.

- **Objetivos**
  - **Gestor de projetos:** define metas como concluir o projeto no prazo, orçamento planejado e escopo.
  - **Scrum *Master*:** garante que os membros da equipe sejam bem treinados para seguir as práticas ágeis de maneira apropriada.
- **Garantia da Qualidade**
  - **Gestor de projetos:** sabe a importância da qualidade, mas não sabe como conseguir isso. Geralmente, um consultor é contratado para corrigir os erros.
  - **Scrum *Master*:** garante a qualidade e sabe muito bem a importância disso.
- **Tamanho da equipe**
  - **Gestor de projetos:** normalmente o gerente de projetos trabalha com mais pessoas e um orçamento enorme.
  - **Scrum *Master*:** o Scrum *Master* sempre tenta manter as coisas menores, trabalhando com equipes pequenas, independentemente do orçamento.

» **Descrição do trabalho**

› **Gestor de projetos:** planejamento, criação de orçamento e documentos relacionados, trabalha com a alta gerência para garantir o escopo e a direção de um projeto, trabalha com outro departamento também, em caso de emergência. Às vezes, eles precisam trabalhar sozinhos ou instruir a equipe a terminar um objetivo.

› **Scrum *Master*:** resolve barreiras e controla os processos Scrum, torna uma equipe ciente do Agile e do Scrum para entregar com sucesso, facilita as cerimônias do Scrum, garante que um projeto esteja funcionando sem problemas com a ajuda das ferramentas, executa o *Backlog* do Produto de acordo com a priorização do Product Owner, resolve conflitos de equipe com boas habilidades de comunicação, motiva a equipe, monitora os processos Scrum para aumentar a eficiência.

Quando as empresas estão iniciando a jornada Scrum, cometem alguns erros por não definir, com clareza, os papéis do Scrum *Master* e o Gerente de Projetos, e isso acontece em empresas com times menores, em que esses dois papéis se sobrepõem. Apesar de existirem semelhanças entre as funções de ambos, não podemos ignorá-las no momento de designá-las. Veja algumas semelhanças:

» ambos aprendem com os erros e têm recursos para aprender com eles. Ambos comunicam entre si, recebem *feedback*, mitigam os riscos e possibilitam um ótimo vínculo dentro de uma equipe, ou ainda podem errar ignorando opiniões de membros da equipe;

» em relação à autoridade, nem o gerente de projeto nem o Scrum *Master* são a autoridade suprema. Ambos se reportam às partes interessadas, o gerente de projeto deve se reportar ao cliente e o Scrum Master deve se reportar ao *Product Owner*.

Outro fator relacionado tanto ao Gerente de Projetos quanto ao Scrum *Master* é seu comportamento diante da mediação de conflitos. O SBOK (2017, p. 61) mostra variações de estilo de lideranças:

» **Liderança Servidora**

Os líderes servidores alcançam resultados focando as necessidades do time. Este estilo é a personificação do papel do Scrum Master.

» **Delegação**

Este estilo de liderança é apropriado em situações em que o líder está focado em detalhes específicos do projeto e quando o seu tempo é limitado.

» **Autocrático**

Os Líderes autocráticos tomam decisões por conta própria, permitindo aos membros do time pouco ou nenhum envolvimento na tomada de decisões. Este estilo de liderança deve ser usado somente em raras ocasiões.

» **Direção**

O Líder de Direção instrui os membros do time sobre as tarefas que são necessárias, quando e como elas devem ser realizadas.

» ***Laissez Faire***

Com este estilo de liderança, o time é deixado sem supervisão, e o líder não interfere nas atividades diárias de trabalho. Isso muitas vezes leva a um estado de anarquia.

» **Apoio/Treinamento**

Os Líderes de apoio e treinamento emitem instruções e, em seguida, apoiam e monitoram os membros do time pela escuta, ajudando, incentivando e apresentando uma perspectiva positiva em momentos de incerteza.

» **Orientador de Tarefa**

Os Líderes Orientadores de Tarefas impõem a conclusão de tarefas e o cumprimento de prazos.

» **Assertivo**

Os Líderes assertivos enfrentam problemas e demonstram confiança para estabelecerem autoridade com respeito.

PLANEJAMENTO E SPRINTS – EVENTOS DO SCRUM

UNIDADE IV

Eventos são artefatos do Scrum para criar uma constância e diminuir a necessidade de reuniões não definidas no Scrum. Todos os eventos são *time-boxed* e têm uma duração definida, ou seja, toda vez que o Sprint começa, a duração é fixada, não sendo diminuída nem aumentada. Os demais eventos podem terminar sempre que seu propósito for alcançado, garantindo que uma quantidade adequada de tempo seja gasta, não permitindo desperdícios no processo.

Esses eventos possuem oportunidades utilizadas para monitorar, melhorar alguma coisa, permitir a transparência e uma inspeção mais criteriosa. As falhas na inclusão de qualquer um destes eventos resultarão na redução da transparência e na perda de oportunidades para inspecionar e adaptar.

Na sequência, veremos dois eventos muito importantes no Scrum. Com eles, poderemos monitorar, no caso do gerenciamento de risco, e melhorar continuamente nossos ciclos por meio do Sprint.

CAPÍTULO 1

Sprint e Gerenciamento de Riscos no Scrum

Segundo o SBOK (2017, p. 116), na metodologia Scrum, os riscos são comumente minimizados, em grande parte devido ao trabalho que está sendo sucedido nos Sprints, em que uma sucessão contínua de Entregáveis é produzida em ciclos muito curtos, e o *Product Owner* está ativamente implicado no projeto. No entanto, mesmo durante o projeto mais discreto, as coisas podem dar errado. Por isso, é essencial ter uma estratégia para distinguir e apontar os riscos. Conforme o próprio guia SBOK (2017, p. 116), o risco é aplicável aos portfólios, produtos ou serviços e projetos.

O risco é um acontecimento incerto que pode afetar positivamente, o que chamamos de oportunidades, ou negativamente o projeto, o que chamamos de ameaças. A Gestão do Projeto é uma tarefa de monitoramento contínuo que deve ser feita do início ao fim de cada tarefa e continuar ao longo do ciclo de vida do projeto.

A Gestão e o Monitoramento dos riscos servem também para identificar e avaliar os riscos com base em dois fatores:

- a probabilidade de ocorrência de cada risco;
- o impacto que o risco terá, caso ele ocorra.

Os riscos deverão ser classificados da mais alta probabilidade e valor impactante para a menor, para compreender seus possíveis potenciais efeitos caso venham a acontecer.

Umas das classificações que são feitas é a distinção entre riscos e problemas. Riscos, segundo o SBOK (2017, p. 117), "são fatores incertos que estão relacionados a um projeto que podem alterar significativamente o resultado do projeto de forma negativa ou positiva. Por exemplo: • Mesmo depois de um treinamento intenso, os usuários do sistema podem não estar preparados para emitir pedidos na data da implantação".

Já os problemas são fatos certos bem definidos que estão acontecendo no projeto, e, por isso, não há necessidade de se realizar uma avaliação de probabilidade como seria feito para um risco. Por exemplo, o financiamento não foi aprovado, os requisitos não estão claros etc.

Também é importante entender como será determinada a atitude ao risco pelos *stakeholders*. O SBOK (2017, p. 119) sugere três opções, entretanto, como cada projeto é único e a base de conhecimento do Scrum cresce a cada ciclo, fica livre para cada empresa adotar sua estratégia em relação às pessoas:

1. **apetite de riscos:** refere-se à quantidade de incerteza que um *stakeholder* ou uma organização está disposta a assumir;
2. **tolerância aos riscos:** indica o grau, quantidade ou volume de risco ao qual os *stakeholders* resistirão;
3. **limite de riscos:** refere-se ao nível aceitável de risco para uma organização.

A Função de Utilidade existe para medir como o *stakeholder* reagirá diante do risco:

1. **avesso ao risco**: o *stakeholder* não está disposto a aceitar um risco;
2. **risco neutro:** o *stakeholder* não se opõe ao risco mas também não demonstra interesse por ele;
3. **buscando o risco**: o *stakeholder* está disposto a aceitar o risco.

Basicamente, as etapas de Gerenciamento de Riscos consistem nas mesmas cinco etapas listadas no Gerenciamento tradicional de risco, orientado pelo PMBOK, porém com algumas técnicas específicas do Scrum. Um bom exemplo da diferença é que, na forma tradicional, a maioria das decisões eram tomadas pelo Gestor do Projeto. No Scrum, as decisões são tomadas pelo próprio Time Scrum.

Identificação de riscos

Algumas técnicas de identificação de riscos são específicas do Scrum, devido à sua filosofia e organização:

- rever as lições aprendidas nos processos de retrospectiva do sprint ou de retrospectiva do projeto;
- *checklists* de risco;
- listas de risco prompt;
- *brainstorming*;
- Estrutura Analítica de Risco (EAR);
- *Risk-Based* Spike.

Avaliação de riscos

As principais técnicas de avaliação de riscos são:

» reunião de risco;

» árvores de probabilidade;

» análise de Pareto;

» tabela de probabilidade e de impacto;

» Valor Monetário Esperado (VME).

Priorização de riscos

Segundo o SBOK (2017, p. 125), "o Scrum permite a rápida identificação e avaliação dos riscos". Os Riscos Identificados são considerados na criação do *Product Backlog*,em que os passos para essa priorização são:

1. crie uma lista de riscos priorizados;
2. selecione os Riscos Identificados que podem ser mitigados;
3. crie uma lista de Estórias de Usuário no *Product Backlog* que sejam priorizadas pelo valor;
4. combine as listas do passo 2 e passo 3 e as priorize pelo valor para chegar ao *Product Backlog* Priorizado e Atualizado.

Mitigação de riscos

A resposta a cada um dos riscos vai depender da probabilidade e impacto dos riscos. No entanto, a natureza iterativa do Scrum com seus ciclos rápido de tempo e *feedback* permite a detecção precoce de falhas. Portanto, na prática, já possui uma característica natural de mitigação. O risco pode ser mitigado por meio da implementação de uma série de respostas. Na maioria dos casos, as respostas são proativas ou reativas. Em Scrum, o *Product Owner* é claramente responsável pelo gerenciamento de riscos relacionados a aspectos do negócio, e o Time Scrum é responsável pela implementação de respostas aos riscos, durante o desenvolvimento de um Sprint. O Scrum *Master* mantém-se atento aos potenciais riscos que possam afetar o projeto e mantém informados o *Product Owner* e o Time Scrum.

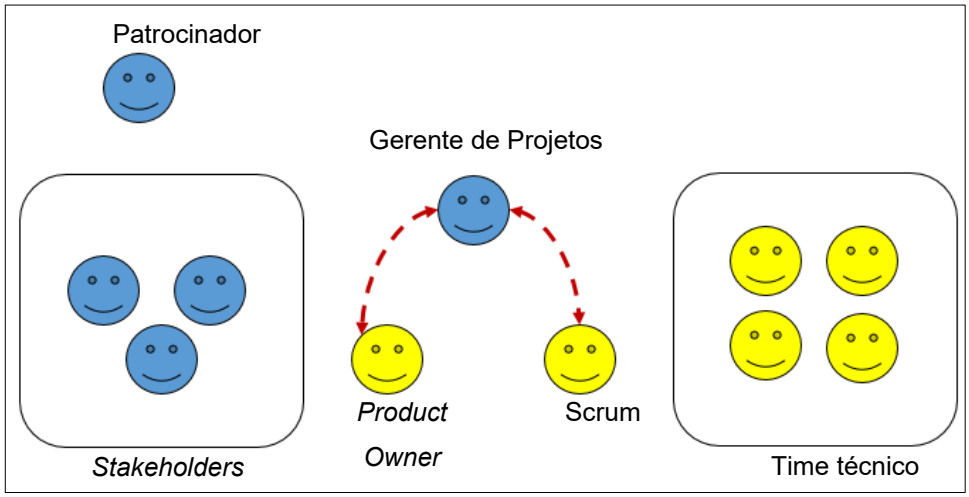
Comunicação de riscos

A transparência das informações é uma das grandes vantagens do Scrum. Um dos motivos é a flexibilização em utilizar a ferramenta adequada para cada situação. Entretanto, um gráfico muito utilizado no Scrum, seja em qualquer demonstração que envolva um relacionamento entre período e valores, é o gráfico de *Burndown*.

Sprint

Sommerville (2010, p. 50) "define o Sprint do Scrum sendo uma unidade de planejamento na qual o trabalho a ser feito é avaliado, os recursos para o desenvolvimento são selecionados e o *software* é implementado. No fim de um Sprint, a funcionalidade completa é entregue aos *stakeholders*".

Figura 16. Sprint.



Fonte: adaptado de http://metodoagil.com/wp-content/uploads/2017/02/metodologia_scrum.gif.

Sutherland (2017, p. 9) acrescenta: "um *time-boxed* de um mês ou menos, durante o qual um incremento de produto potencialmente liberável é criado, tem durações consistentes ao longo de todo o esforço de desenvolvimento e, tão logo uma Sprint termina, uma nova é iniciada".

As Sprints contêm e consistem em:

» *Sprint Backlog* – um planejamento da Sprint;

» *Daily* Scrum – reuniões diárias;

» Sprint – o trabalho de desenvolvimento.

Após o incremento, ou seja, quando toda a funcionalidade definida naquela Sprint for entregue, haverá:

» uma revisão da Sprint;

» uma retrospectiva da Sprint.

Durante o Sprint, algumas regras precisam ser seguidas:

» não é feita nenhuma alteração que coloque em risco o objetivo da Sprint;

» a qualidade não é negociável, ou seja, metas de qualidade não diminuem;

» o escopo pode ser esclarecido e renegociado entre o *Product Owner* e a Equipe de Desenvolvimento à medida que se ganha conhecimentos que agreguem valor ao ciclo.

Duração de Sprint

A duração de Sprint deve ser determinada pelo Scrum *Master*, que é o facilitador da equipe, conduzindo para que todos entrem em um consenso sobre esse prazo. Quando a equipe de Scrum – normalmente quem dá esse prazo é a Equipe de Desenvolvimento – definir que o período de Sprint será de trinta dias, todos os próximos Sprints deverão seguir essa medida como base.

Deve haver uma limitação para Sprint, pois, quando sua duração é muito longa, a definição do seu objetivo poderá mudar, aumentar a complexidade, assim como o seu risco.

Segundo Sutherland (2010, p. 9), "Sprints permitem previsibilidade que garante a inspeção e adaptação do progresso em direção à meta pelo menos a cada mês corrido. Sprints também limitam o risco ao custo de um mês corrido".

Assim que o Sprint for iniciado, o *Product Owner* sai de cena e deixa a equipe fazer seu trabalho de discutir o progresso e desafios. O *Product Owner* apenas participa das reuniões como um observador, limitando-se a responder dúvidas. Ele não pode fazer solicitações, deixando ao Scrum *Master* as interrupções de Sprint.

Logo que o Sprint for finalizado, o trabalho é demonstrado pela equipe, ao *Product Owner*, que utiliza de critérios preestabelecidos, para aceitar ou rejeitar o projeto.

Daily Scrum

É a reunião diária que ocorre idealmente no mesmo horário todas as manhãs, por se acreditar que as mentes estão descansadas, e sempre no mesmo local para diminuir a complexidade. O Scrum *Master* deverá entusiasmar a equipe ao máximo e garantir que a equipe de desenvolvimento tenha a reunião, mas a equipe de desenvolvimento é responsável pela condução do *Daily* Scrum, ensina a Equipe de Desenvolvimento a manter o Daily Scrum dentro do prazo de 15 minutos, e o Scrum *Master* garante que outras pessoas sejam ouvintes da reunião, porém não atrapalhe a equipe.

A ideia é fazer com que os membros respondam a três perguntas:

- o que fiz ontem que ajudou a Equipe de Desenvolvimento a atingir a meta da Sprint?
- o que farei hoje para ajudar a Equipe de Desenvolvimento a atingir a meta da Sprint?
- vejo algo que impeça a mim ou à equipe de desenvolvimento de atingir a meta da Sprint?

O objetivo é que todos os membros da equipe se comprometam e se mantenham todos focados nos objetivos do Sprint. Além disso, o objetivo é criar repetidamente novos planos de Sprint dentro de prazos fixos em um projeto, não criar novos planos de longo prazo.

Backlog da Sprint

Antes de entrar em uma reunião de Sprint, a equipe separa um conjunto de itens oriundos do *Backlog* do Produto e um plano de incremento do produto, denominado Sprint *Backlog*.

Segundo Sutherland (2017 p. 16), "o Sprint *Backlog* é uma previsão da equipe de desenvolvimento sobre qual funcionalidade será no próximo incremento e o trabalho necessário para entregar essa funcionalidade em um incremento 'Concluído'. O Sprint *Backlog* torna visível todo o trabalho que a Equipe de Desenvolvimento identifica como necessário para atingir a meta do Sprint. Para garantir a melhoria contínua, inclui pelo menos uma melhoria de processo de alta prioridade identificada na reunião Retrospectiva anterior". O Sprint *Backlog* surge durante o Sprint conforme a Equipe de Desenvolvimento o modifica, atualiza ou adquire mais conhecimento para atingir a meta da Sprint.

Durante a Sprint, algumas decisões estarão sendo tomadas para deixá-la mais flexível e menos complexa. Por exemplo:

» surgiu novo trabalho, a equipe adiciona no Sprint *Backlog*;

» quando um trabalho for concluído, o restante estimado é atualizado;

» quando alguma tarefa for desnecessária, imediatamente é removida.

Isso faz com que o Sprint *Backlog* seja um *snapshot* que a Equipe de Desenvolvimento planeja realizar durante o Sprint e pertença exclusivamente a ela. Esse processo ajuda no monitoramento da Sprint. Segundo Sutherland (2017, p. 16), "a equipe de desenvolvimento rastreia esse trabalho total restante, pelo menos para cada *Daily* Scrum, para projetar a probabilidade de atingir a meta da Sprint, podendo rastrear seu progresso".

Ao finalizar uma Sprint, somamos a todos os itens do *Product Backlog* concluídos e o valor de todas as Sprint anteriores, e a esse processo damos o nome de Incremento. Um novo incremento deverá atender à definição de concluído, ou seja, em condições de uso baseadas nos parâmetros da equipe.

Segundo Sutherland (2017, p. 16), um incremento é um corpo de trabalho que pode ser inspecionado e realizado que apoia o empirismo no final do Sprint. O incremento é um passo em direção a uma visão ou objetivo. O incremento deve estar em condições utilizáveis, independentemente de o *Product Owner* decidir liberá-lo.

Scrumboard

Em sua forma mais simples, um quadro do Scrum mostra uma lista pendências do Sprint de trabalho que precisa ser realizada para concluir um projeto. O trabalho de uma equipe Scrum e seu projeto são exibidos em colunas verticais: histórias, tarefas, trabalhos em andamento, concluídos. Cada linha é uma história. Cada unidade de trabalho – usando "cartões" em *software* ou notas adesivas – é exibida e movida pelas colunas durante um Sprint, que é o coração do Scrum. O *Scrumboard* é exatamente igual a um quadro Kanban.

Estas são as colunas normalmente incluídas em um quadro Scrum:

» estórias do usuário;

» o que fazer;

» trabalho em andamento;

» concluído.

Gráfico *Burndown* do Sprint

"O Gráfico *Burndown* do Sprint é um gráfico que mostra a quantidade de trabalho restante durante o desenvolvimento do Sprint. Este gráfico mostra o progresso que tem sido feito pelo Time Scrum e também permite a detecção de estimativas que podem ter sido incorretas. Se o Gráfico *Burndown* do Sprint mostra que o Time Scrum não será capaz de terminar as tarefas do Sprint em tempo, o Scrum *Master* deve identificar quaisquer obstáculos ou impedimentos para a conclusão bem sucedida e tentar removê-los" (SBOK, 2017, p. 210).

Cancelando Sprint

Seguindo a flexibilidade do Scrum, pode acontecer de uma Sprint ser cancelada. Isso pode acontecer se, por algum motivo, o curso para atingir seu objetivo estiver sendo desviado, por exemplo, se seu objetivo estiver ficando obsoleto, os negócios principais forem mudados, se a tecnologia ou o mercado mudarem, ou seja, não faz mais sentido seguir com a Sprint devido às circunstancias, e somente o *Product Owner* tem autoridade para cancelar, sob influência da Equipe de Desenvolvimento ou das partes interessadas.

Em relação aos itens, quando um Sprint é cancelado, haverá uma revisão de estimativa dos itens marcados como pronto no Product *Backlog*, porém, se algum desses itens estiver quase pronto para ser entregue, é previsto que o *Product Owner* aceite, e os itens incompletos terão sua estimativa recalculada.
Segundo Sutherland (2017, p. 9), "os cancelamentos do Sprint consomem recursos, já que todos se reagrupam em outro *Sprint Planning* para iniciar outro Sprint, eles são frequentemente traumáticos para a equipe Scrum e são muito incomuns".
Os componentes principais de um Sprint são:

- Sprint Planning – Planejamento do Sprint;
- Sprint Review – Revisão do Sprint;
- Sprint Retrospective – Retrospectiva do Sprint.

CAPÍTULO 2

Sprint Planning ou Planejamento de Sprint

O *Sprint Planning* é um evento que inicia o Sprint, e seu objetivo é definir o que pode ser entregue no Sprint e como esse trabalho será alcançado. O planejamento da Sprint é feito em colaboração com toda a equipe do Scrum.

Vimos no Capítulo anterior que o Sprint, é um período que define como todo trabalho será realizado. Entretanto, antes de entrarmos em ação, é necessário que haja um planejamento do Sprint, que definirá o foco e a motivação da equipe.

Normalmente, o Planejamento da Sprint tem um prazo de oito horas, no máximo, para uma Sprint de um mês. Obviamente, para Sprints mais curtos, o evento geralmente é mais curto, e quem garante que o evento ocorra, que os participantes entendam seu propósito e ensina a Equipe Scrum a mantê-lo dentro do prazo é o Scrum Master.

Segundo Sutherland (2017, p. 10), o planejamento da Sprint responde às seguintes questões:

- o que pode ser entregue como resultado do incremento da próxima Sprint?
- como o trabalho necessário para entregar o incremento será realizado?

Em seu artigo "*Sprint Planning*", visualizado pelo link https://www.atlassian.com/agile/scrum/sprint-planning, Dave West, *Product Owner* e CEO da scrum.org, define algumas perguntas a serem feitas para que planos ruins de Sprint não atrapalhem as expectativas da equipe, as entradas e os resultados:

- **O quê?**

  Após o **Product Owne**r descrever o objetivo do Sprint e quais itens da lista de pendências devem entrar nesses objetivos, a equipe do Scrum decide o que pode ser feito no próximo Sprint e o que eles farão durante o Sprint para que isso aconteça, o que pode ser entregue no Incremento resultante do próximo Sprint.

  Já o **Time de Desenvolvimento** trabalha para prever as funcionalidades que serão desenvolvidas durante a Sprint.

» **Como**?

A equipe de desenvolvimento planeja o trabalho necessário para atingir a meta do Sprint, e o plano de Sprint resultante é uma negociação entre a equipe de desenvolvimento e o proprietário do produto com base em valor e esforço, como o trabalho necessário para entregar o incremento será alcançado.

» **Quem**?

Você não pode fazer o planejamento de Sprint sem o Product Owner ou a Equipe de Desenvolvimento. O Product Owner define a meta com base no valor que eles buscam. A Equipe de Desenvolvimento precisa entender como eles podem ou não alcançar esse objetivo. Se um desses eventos estiver ausente, o planejamento do Sprint é quase impossível.

» **As entradas**

Um excelente ponto de partida para o plano de Sprint é o *Product Backlog*, pois fornece uma lista de "coisas" que poderiam fazer parte do Sprint atual. A equipe também deve analisar o trabalho existente realizado no incremento e ter uma visão da capacidade.

» **Os resultados**

O resultado mais importante para a reunião de planejamento do Sprint é que a equipe pode descrever o objetivo do Sprint e como eles começarão a trabalhar em direção a esse objetivo. Isso é visível no *Backlog* do Sprint.

Outro planejamento que deve ser feito é o *Time-boxing*, que nada mais é que uma configuração do limite superior de tempo para que a equipe realize as tarefas de Sprint, ou seja, planejar o Sprint. Podemos ainda estimar um período para esse artefato. Por exemplo, o Sprint não deve ser limitado a mais de duas horas para um Sprint de duas semanas. E assim como for definido com consenso da equipe.

Sutherland (2010, p. 10) cita dois tópicos relacionados ao Planejamento do Sprint. O **tópico um** refere-se à pergunta "o que pode ser **feito** neste Sprint?". Na sequência, a equipe de desenvolvimento trabalhará para prever a funcionalidade que será desenvolvida durante o Sprint, e o *Product Owner* discute se a meta do Sprint será atingida:

> "A entrada para esta reunião é o *Product Backlog*, o Incremento do produto mais recente, a capacidade projetada da Equipe de

> Desenvolvimento durante o Sprint e o desempenho passado da Equipe de Desenvolvimento. O número de itens selecionados no *Product Backlog* para o Sprint depende exclusivamente da equipe de desenvolvimento. Somente a equipe de desenvolvimento pode avaliar o que pode realizar no próximo Sprint".

Já no **tópico dois,** é questionado como será realizado o trabalho escolhido. Depois de definido o tópico "**o que fazer**", a equipe de desenvolvimento decide como criar essa funcionalidade em um incremento de produto "**Concluído**" durante a Sprint. "Os itens do *Product Backlog* selecionados para este Sprint mais o plano para entregá-los são chamados de *Sprint Backlog*" (SUTHERLAND, 2010, p. 11).

Os papéis são definidos, e todos já devem saber o que cada um fará na Sprint *Planning*. Por exemplo, a **Equipe de Desenvolvimento** geralmente começa projetando o sistema e o trabalho necessário para converter o *Product Backlog* em um incremento do produto em funcionamento. O tamanho do trabalho e o esforço estimado poderão variar, porém deve ser mensurado um trabalho, no Planejamento do Sprint, que a Equipe de Desenvolvimento acredita conseguir fazer na Sprint seguinte. Sutherland (2017, p. 11) completa que "o trabalho planejado para os primeiros dias do Sprint pela equipe de desenvolvimento é decomposto no final desta reunião, geralmente em unidades de um dia ou menos, ela se auto-organiza para realizar o trabalho no *Backlog* da Sprint, durante o Planejamento da Sprint e conforme necessário em toda a Sprint".

O ***Product Owner*** pode ajudar a esclarecer os itens selecionados do *Product Backlog* e fazer trocas. Se a Equipe de Desenvolvimento definir que tem muito ou pouco trabalho, poderá renegociar os itens selecionados do *Product Backlog* com o *Product Owner*. A Equipe de Desenvolvimento também pode convidar outras pessoas a comparecer para fornecer conselhos técnicos ou de domínio. No final do planejamento da Sprint, a Equipe de Desenvolvimento poderá explicar ao *Product Owner* e ao Scrum Master como pretende trabalhar como uma equipe auto-organizada para atingir a meta da Sprint e criar o incremento esperado.

A meta do Sprint

Metas, seja em qualquer tipo de processo, são o objetivo a ser alcançado, podendo ser Qualitativas (Regular, Boa, Ótima) ou Quantitativas (10%, 50%, 100%), e serão definidas em planejamentos do processo.

No Scrum, essa meta tem o objetivo definido no *Product Backlog*, e sua classificação é definida, juntamente com o cliente para entregar funcionalidades de valor e fornecer orientação para a equipe de desenvolvimento sobre por que está sendo criado o incremento. Ela é criada na reunião de Planejamento de Sprint, os itens são os elencados no *Product Backlog* e devem oferecer uma função coerente que definirá a meta do Sprint, fazendo com que a Equipe trabalhe sincronizada e não em tarefas separadas.

Segundo Sutherland (2017, p. 11), "à medida que a equipe de desenvolvimento trabalha, ela mantém o objetivo da Sprint em mente, para satisfazer o objetivo da Sprint, implementa funcionalidade e tecnologia e, se o trabalho for diferente do esperado pela equipe de desenvolvimento, eles colaboram com o *Product Owner* para negociar o escopo do *Backlog* da Sprint dentro da Sprint".

Após a Reunião Diária, a Equipe de Desenvolvimento ou membros da equipe frequentemente se encontram imediatamente para adaptar, ou redefinir, o restante do trabalho da Sprint.

CAPÍTULO 3

Sprint Review – Revisão de Sprint

Após a Sprint ser planejada, esse trabalho será demonstrado. Como opção, os membros da equipe se reúnem em torno de uma mesa para demonstrações informais e descrevem o trabalho que fizeram para essa iteração. É hora de fazer perguntas, experimentar novos recursos e dar *feedback*. A equipe demonstra o incremento com foco no objetivo da Sprint, de acordo com a definição de feito, e o *Product Owner* revisa e aceita o Incremento entregue.

O *Sprint Review* é realizado ao final de cada Sprint. O time, formado pelo *Product Owner*, Scrum *Master* e *Stakeholders* do cliente, demonstra tudo o que foi desenvolvido. Para otimizar o tempo, não é recomendado utilizar *slides* ou apresentações prontas.

Durante o *Sprint Review*, o projeto deve ser revisado de acordo com a Meta do Sprint, que foi definida no *Sprint Planning*. O ideal é que o time consiga alcançar a meta estipulada, mesmo que alguns itens ainda não tenham sido finalizados, tomar nota dos comentários realizados durante o *Sprint Review*, para que consigam considerar esses itens no planejamento do próximo ciclo de desenvolvimento.

A sequência do processo alinhada com o papel de cada membro é a seguinte:

- a **Equipe de Desenvolvimento** apresenta os resultados do Sprint;
- o ***Product Owner*** revisa e aceita o incremento de produto entregue;
- **o *Product Owner*, a Equipe de Desenvolvimento** e os ***Stakeholders***, durante o *Sprint Review*, revisam o que foi feito. Normalmente, isso assume a forma de uma demonstração dos novos recursos;
- a **Equipe do Scrum** analisa os itens previstos do *Product Backlog* e analisa como as estórias de usuários foram atendidos no incremento do produto.

Após a demonstração, o *Product Owner* e outras partes interessadas relevantes relatam suas impressões e esclarecem seus requisitos (histórias de usuários) se um requisito não foi implementado corretamente. O *Product Owner* identifica o que foi feito e o que não foi feito (de acordo com a Definição de Concluído). O *Product Owner* aceita as histórias de usuário que foram feitas. Os resultados

dessa reunião podem ser novos requisitos no *Product Backlog* e uma nova priorização dos itens existentes do *Product Backlog*.

Uma revisão de Sprint pode durar até 4 horas em Sprints de 4 semanas. A regra geral é que a revisão do Sprint não deve demorar mais de uma hora por semana de duração do Sprint. A tabela a seguir ilustra um pouco melhor o que seria essa regra.

Tabela 9. Regra básica de duração do *Sprint Review*.

Duração total da Sprint	Duração da Revisão da Sprint
1 semana	1 hora
2 semana	2 horas
3 semanas	3 horas
4 semana	4 horas

Fonte: https://www.visual-paradigm.com/scrum/what-is-sprint-review/.

Segundo Sutherland (2017, p. 13), a Revisão da Sprint inclui os seguintes elementos:

» os participantes incluem o Time Scrum e os *Stakeholders*-chaves convidados pelo *Product Owner*;

» o *Product Owner* esclarece quais itens do *Product Backlog* foram "Prontos" e quais não foram "Prontos";

» o Time de Desenvolvimento discute o que foi bem durante a Sprint, quais problemas ocorreram dentro da Sprint e como esses problemas foram resolvidos;

» o Time de Desenvolvimento demonstra o trabalho que está "Pronto" e responde as questões sobre o incremento;

» o Product Owner discute o *Product Backlog* tal como está. Ele (ou ela) projeta os prováveis alvos e datas de entrega baseado no progresso até a data (se necessário);

» o grupo todo colabora sobre o que fazer a seguir, e é assim que a Revisão da Sprint fornece valiosas entradas para o Planejamento da Sprint subsequente;

» revisão de como o mercado ou o uso potencial do produto pode ter mudado e o que é a coisa mais importante a se fazer a seguir; e

» revisão da linha do tempo, orçamento, potenciais capacidades e mercado para a próxima versão esperada de funcionalidade ou de capacidade do produto.

Como resultado, teremos um *Product Backlog* revisado, ajustado completamente para atender novas oportunidades e que define os prováveis Itens de para a próxima Sprint.

Reunião de Revisão da Sprint

O objetivo da reunião é que a equipe mostre aos clientes e às partes interessadas o trabalho que eles realizaram durante o Sprint. A reunião é facilitada pelo *Product Owner*, porém é comum que os membros da equipe executem a reunião. Nessa reunião, devem ser revisados os seguintes dados:

» o trabalho comprometido e o realizado pela equipe;

» principais decisões tomadas durante a Sprint;

» métricas do projeto;

» demonstração do próprio trabalho;

» revisão de prioridade para a próxima Sprint;

O *Team* Scrum deverá manter o foco da Reunião da Sprint em:

» o **Scrum *Master*** deve garantir que a Revisão do Sprint seja sobre processo, não sobre pessoas;

» os ***stakeholders*** devem operar em equipe, e não como especialistas;

» o ***Product Owner*** deve incentivar a equipe a manter um ritmo constante durante o diálogo do produto. Também deve evitar a tentação de formular soluções;

» o ***Product Owner*** e o **Scrum *Master*** devem focar na revisão do produto em questão.

Seguem algumas sugestões para realizar Reunião de *Sprint Review*.

Transparência

Garanta que todos os envolvidos, os *stakeholders*, saibam o motivo da Revisão da Sprint.

Comunicação

Lembre a equipe antes do *Sprint Planning* que haverá uma Revisão no final do Sprint e descreva o que esperar.

Prepare o local da apresentação

A equipe deve considerar maneiras e métodos de como melhor pode mostrar o trabalho concluído. Pode-se considerar demonstrar coletivamente subconjuntos relevantes de testes de aceitação ou, alternativamente, cada membro da equipe pode se voluntariar para demonstrar uma história de usuário cada.

Ajude na preparação da Equipe

- A equipe deve assumir a propriedade do trabalho pelo qual é responsável.
- Os líderes devem se afastar e ajudar a equipe a se auto-organizar, facilitando o processo.
- O Scrum *Master* deve fazer perguntas como:
  - Como é o sucesso da Revisão?
  - Qual é o próximo passo?
  - Como saberemos que conseguimos isso?

Cuide do tempo

É recomendado que um relógio fique na visão de todos para que tenham noção do tempo decorrido. Muitas empresas não têm esse artefato, mesmo sabendo que o tempo, no Sprint, é fixo. Isso ajudará a respeitar o *Time-boxed* também.

Reserve tempo para discutir a próxima Sprint

O *Product Owner* deve poder discutir o *Product Backlog* como está atualmente e convidar discussões abertas sobre se é o que a equipe deve fazer em seguida.

Registre os resultados

Além de registrar tudo o que for necessário, nesse momento é primordial registrar os principais pontos do que foram acordados.

CAPÍTULO 4
Sprint Retrospective

Logo após o *Sprint Review* e antes do próximo *Sprint Planning*, ocorre a *Sprint Retrospective*. Com ela, o Scrum *Team* tem a oportunidade de se inspecionar e preparar o planejamento de melhorias para a próxima Sprint. Ela deverá seguir a mesma tabela de tempo do *Sprint Review,* e o Scrum *Master* garante que o evento ocorra dentro do mais harmônico ambiente, seja claro ao explicar o propósito, e que os participantes realmente entendam, que o evento seja positivo e produtivo, ensina todos a manter o evento dentro do *time-box* e participa da reunião como um membro auxiliar do time devido à sua responsabilidade pelo processo Scrum.

O propósito da Retrospectiva da Sprint é:

- inspecionar como a última Sprint foi em relação aos artefatos de Sprint;
- identificar e ordenar os principais itens que foram bem e as potenciais melhorias;
- criar um plano de melhorias no modo como o Time Scrum faz seu trabalho.

O Registro de Retrospectiva do Sprint, segundo SBOK (2017, p. 255), é um registro das opiniões, discussões e de itens acionáveis apontados durante uma Reunião de Retrospectiva do Sprint. O Scrum *Master* pode facilitar a criação desse registro com a colaboração de membros do Time Central do Scrum. A coleção dos Registros de Retrospectiva do Sprint tornam-se o diário do projeto, onde são detalhados os sucessos, problemas e resoluções do projeto. Os registros são documentos públicos disponíveis para qualquer pessoa na organização.

É essencial realizar essa reunião em um ambiente aberto e descontraído para incentivar a plena participação de todos os membros do time. As discussões durante a Reunião de Retrospectiva do Sprint abrangem tanto o que deu errado como o que deu certo. O objetivo principal da reunião é identificar:

1. as coisas que o time precisa continuar fazendo: melhores práticas;
2. as coisas que o time precisa começar a fazer: melhorias de processo;
3. as coisas que o time precisa parar de fazer: problemas do processo e gargalos.

4. Estas áreas são discutidas, e é criada uma lista de Pontos de Melhorias Acordados.

Segundo Sutherland (2017, p. 14), é papel do Scrum *Master* encorajar o Scrum *Team* a melhorar seu processo de desenvolvimento e suas práticas para torná-lo mais efetivo e agradável para a próxima Sprint. Durante cada Retrospectiva da Sprint, o Scrum *Team* planeja formas de aumentar a qualidade do produto melhorando o processo de trabalho ou adaptando a definição de "Pronto".

Já o SBOK (2017, p. 235) "sugere que todos os membros do Time Scrum participam da reunião, que é recomendada, mas não necessária, a participação do *Product Owner*, um membro deverá documentar as discussões e os itens para ação futura. Os Pontos de Melhoria Acordados são as saídas primárias do processo de Retrospectiva do Sprint. Uma lista de itens de ações que é criada pelo time para direcionar os problemas e melhorar os processos, com a finalidade de aprimorar o seu desempenho em Sprints futuros".

Técnicas para as Reuniões de *Sprint Retrospective*

Existem várias ferramentas ligadas ao gerenciamento de pessoas que podem contribuir para a realização de reuniões. A seguir, existem algumas sugestões do SBOK que utilizam as melhores práticas para o Scrum.

ESVP

ESVP é o acrônimo de *Explorer, Shopper, Vacationer, Prisoner*. Basicamente, a atividade é sobre todos os participantes relatando sua própria atitude em relação à reunião no início da Reunião de Retrospectiva do Sprint, e isso pode fornecer informações sobre o que está acontecendo com a equipe e pode ajudar a equipe a raciocinar sobre seu curso de ação. Os participantes são convidados a indicar anonimamente o que melhor representa sua visão na reunião.

A ideia é que alguns dos participantes de uma retrospectiva sejam como ***Explorers***, que analisam todos os detalhes, procurando tirar o máximo proveito da retrospectiva. Outros são mais parecidos com ***Shoppers***. Querem ouvir tudo e escolher o que ele pode tirar da retrospectiva. Depois, há os ***Vacationer***s, que apenas apreciam a retrospectiva por lhes dar algum tempo de folga diária. A última categoria é de pessoas que se sentem ***Prisoner*** na retrospectiva e preferem fazer outra coisa. O Scrum *Master,* em seguida, coleta as respostas, prepara, e compartilha a informação com o grupo.

Lancha

Pouco utilizada, a Lancha é mais uma técnica que pode ser usada para realizar a Reunião de Retrospectiva do Sprint. Os membros do time simulam o papel da tripulação de uma Lancha que deverá chegar a uma ilha: o objetivo do Projeto.

Símbolos são usados pelos participantes para indicar motores, que são as coisas que os ajudam a chegar à ilha, e âncoras são as coisas que estão impedindo-os de chegar à ilha. Sua principal vantagem é ser um *time-boxed* de alguns minutos. Assim como na maioria dos términos das reuniões, os itens são documentados, a informação é discutida e priorizada por meio de processo de votação, os impasses são resolvidos pelo Scrum *Master*. Segundo o SBOK (2017, p. 254), "com base na prioridade, os motores são reconhecidos, e ações de mitigação são planejadas para as âncoras".

Técnicas de comparação

Várias medidas podem ser usadas para comparar o desempenho do time no Sprint atual com o seu desempenho em Sprints anteriores:

- a velocidade do time;
- a taxa de sucesso de Pronto;
- a eficácia da estimativa;
- a revisão das classificações de *feedback;*
- as classificações da moral do time;
- o *feedback* dos membros do time;
- o progresso para liberar ou lançar.

Espera-se que o Time Scrum auto-organizado aprenda com os erros cometidos durante o Sprint, pois as lições aprendidas ajudam os times a melhorar o seu desempenho nos próximos e poderão ser documentadas nas Recomendações do Scrum *Guidance Body*, para serem compartilhadas com outros Times Scrum.

Essas lições são a parte fundamental da retrospectiva e devem ser devidamente compartilhadas entre os membros do time e com o Scrum *Guidance Body*, já que os times buscam o autoaperfeiçoamento contínuo.

Segundo SBOK (2017, p. 256), “ao final da Retrospectiva da Sprint, o Time Scrum deverá ter identificado melhorias que serão implementadas na próxima Sprint. A implementação destas melhorias na próxima Sprint é a forma de adaptação à inspeção que o Time Scrum faz a si próprio. Apesar de que melhorias podem ser implementadas a qualquer momento, a Retrospectiva da Sprint fornece uma oportunidade formal focada em inspeção e adaptação”.

Referências

COHN, M. ***Agile estimating and planning***. Pearson Education: 2006.

KNIBERG, H. **Scrum e XP direto das Trincheiras**. C4Media Inc., 2007.

PMBOK. **Guia do Conhecimento em gerenciamento de projetos**. 6 ed., 2017, *Project Management Institute*. Disponível em: <https://www.atlassian.com/agile/project-management/user-stories>. Acesso em: 01 out. 2019.

REHKOPF, M. ***User Stories | Examples and Template***, 2017, Atlassian: https://www.atlassian.com/agile/project-management/user-stories. Acesso em: 01 out. 2019.

SBOK. ***A Guide to the SCRUM BODY OF KNOWLEDGE (SBOK™GUIDE) 3rd Edition***. SCRUMstudy, 2017. Avondale, Arizona, USA.

SUTHERLAND, J.; SCHWABER, K. **Guia Scrum 2017**. Disponível em: https://www.scrumguides.org/scrum-guide.html. Acesso em: 01 nov. 2019.

<https://www.thescrummaster.co.uk/scrum/short-history-scrum/>.

<https://www.thescrummaster.co.uk/wp-content/uploads/2016/09/The-New-New-Product-<Development-Game.pdf>.

<https://www.thescrummaster.co.uk/wp-content/uploads/2016/09/SCRUM-Development-Process-K-Schwaber.pdf>.

<http://agilemanifesto.org/>.

<http://www.controlchaos.com/>.

<https://www.techwell.com/techwell-insights/2012/10/brief-history-scrum>.

<http://wiki.c2.com/?ScrumProcess>.

<https://upraise.io/blog/traditional-vs-agile-project-management-3/>.

<http://www.devx.com/architect/Article/32761>.

<http://www.metodoagil.com/wp-content/uploads/2017/02/metodologia_scrum.gif>.

<https://www.proofhub.com/articles/traditional-vs-agile-project-management>.

<http://assets.devx.com/articlefigs/17424.jpg>.

Apêndice

Tridibesh Satpathy, com quem eu nunca conheci ou me comuniquei (nem com nenhum de seus coautores ou revisores), remove o coração, a alma e os valores de Scrum neste livro. Com uma única mão, ele tenta transformar Scrum em uma metodologia fórmula que pode ser usada sem pensamento ou empirismo. O Guia Scrum que define o Scrum tem 17 páginas. Tridibesh *et al.* adicionaram todas as práticas, técnicas e processos definidos conhecidos a ele para criar um monumento de mais de 300 páginas às falhas das metodologias preditivas. Você pode pegar este livro, aplicá-lo, obter as certificações e sentir-se confortável ao ver que tudo está no lugar. Não é. Tridibesh *et al.* nunca viram sua organização, seus projetos, seu contexto ou seus objetivos. Como eles podem acreditar que podem formular uma solução para você? Eu poderia ter pensado que a arrogância motivou esse esforço, principalmente porque nenhum dos autores ou editores é conhecido na comunidade ágil. No entanto, a experiência me ensinou que isso é puramente impulsionado pela necessidade de dinheiro. Estudado de todos os ângulos, é um esquema de ganhar dinheiro que deve ser evitado por quem entende a base da agilidade, empirismo, e pensamento enxuto. Como o primeiro passo no pensamento enxuto, para identificar e eliminar desperdícios, jogue fora este livro e evite seus autores.

Ken Schwaber desenvolvedor do Scrum, signatário do Agile Manifesto, fundador da *Agile Alliance, Scrum Alliance* e Scrum.org.

O suporte ao SCRUMstudy diz:

Esta revisão é completamente imprecisa e antiética e deve ser removida pela Amazon pelos seguintes motivos:

1. Esta revisão é inadequada porque ele trabalha na competição Scrum.org. Ele quer promover livros de pessoas em sua organização (por exemplo, Scrum.org) e desacreditar os livros escritos por SCRUMstudy. Portanto, isso não deve ser permitido pela Amazon conforme “Promoção de conduta ilegal ou imoral” - Material censurável.

2. Ele não leu o livro (não é uma compra verificada na Amazon); portanto, como ele pode comentar que o livro não é relevante e chamá-lo de “farsa ‘”?

3. Essa pessoa só está interessada em vender livros de sua organização e de autores que trabalham para sua organização (e não fornece nenhum motivo específico para o fato de o SBOK não ser bom, exceto que o livro discute conceitos com os quais ele não está familiarizado). Aqui, essa pessoa é concorrente direta do SCRUMstudy e está publicando críticas negativas sobre o SCRUMstudy por seus benefícios financeiros pessoais. Portanto, ele contém conteúdo inapropriado.

4. SCRUMstudy. Como é uma organização de renome para ensinar Scrum globalmente. O Conhecimento em Scrum (SBOK) foi escrito por 18 autores especialistas em Scrum Practitioners e está sendo amplamente apreciado no setor. O SBOK foi revisado por 25 especialistas e baseia-se no conhecimento e insights combinados adquiridos em milhares de projetos em várias organizações e indústrias. Toda essa revisão parece desacreditar o SCRUMstudy e o Conhecimento em Scrum (SBOK) para o benefício de um concorrente (Scrum Alliance) pelos benefícios financeiros.

5. Solicitaremos aos alunos interessados que façam um curso introdutório gratuito sobre Scrum no SCRUMstudy.com (que inclui o primeiro capítulo do SBOK, vídeos instrutivos e um simples estudo de caso do Scrum da vida real) e julgue por si mesmo sobre a qualidade dos cursos oferecidos pelo SCRUMstudy (em vez de ler críticas negativas de interesses e concorrentes). O primeiro capítulo do SBOK está disponível para você visualizar no SCRUMstudy.com ou na Amazon.