Scrum

Gestão ágil para produtos de sucesso

Edição atualizada

Casa do Código | alura

RAFAEL SABBAGH

DADOS DE ODINRIGHT

Sobre a obra:

A presente obra é disponibilizada pela equipe ***eLivros*** e seus diversos parceiros, com o objetivo de oferecer conteúdo para uso parcial em pesquisas e estudos acadêmicos, bem como o simples teste da qualidade da obra, com o fim exclusivo de compra futura.

É expressamente proibida e totalmente repudíavel a venda, aluguel, ou quaisquer uso comercial do presente conteúdo.

Sobre nós:

O ***eLivros*** e seus parceiros disponibilizam conteúdo de dominio publico e propriedade intelectual de forma totalmente gratuita, por acreditar que o conhecimento e a educação devem ser acessíveis e livres a toda e qualquer pessoa. Você pode encontrar mais obras em nosso site: ***eLivros***.

Como posso contribuir?

Você pode ajudar contribuindo de várias maneiras, enviando livros para gente postar ***Envie um livro*** ;)

Ou ainda podendo ajudar financeiramente a pagar custo de servidores e obras que compramos para postar, ***faça uma doação aqui*** :)

"Quando o mundo estiver unido na busca do conhecimento, e não mais lutando por dinheiro e

poder, então nossa sociedade poderá enfim evoluir a um novo nível."

Sumário

ISBN

Impresso: 978-65-86110-92-0

Digital: 978-65-86110-93-7

Caso você deseje submeter alguma errata ou sugestão, acesse: http://erratas.casadocodigo.com.br.

Sobre o autor

Rafael Sabbagh é instrutor oficial de Scrum (Certified Scrum Trainer, pela Scrum Alliance) e de Kanban (Accredited Kanban Trainer, pela Kanban University). Entre 2015 e 2017, ele foi membro do Board de Diretores da Scrum Alliance, organização internacional responsável pelo Scrum. Rafael já fez treinamentos e *coaching* em mais de vinte países da Europa, Américas e Ásia e é palestrante em eventos pelo mundo desde 2009, incluindo diversos Scrum Gatherings globais e regionais.

Engenheiro de Computação e Mestre em Administração de Empresas pela PUC-Rio, Rafael é cofundador da K21, onde atua na transformação de empresas que vão desde *startups* a multinacionais. Ele atualmente vive em Madri, Espanha, e conduz os negócios da empresa na Europa.

Agradecimentos

Agradeço aos revisores técnicos de diferentes partes desta edição, cujas sugestões levaram a importantes decisões sobre o conteúdo e sobre a estrutura do livro: Marcelo Paiva, Tadeu Marinho e Maria José Levy Ibarra. Agradeço também aos revisores técnicos das duas edições anteriores: Manoel Pimentel, Marcos Garrido e Rodrigo de Toledo.

Agradeço à minha mulher, Maria José Levy Ibarra, pelo apoio nessa longa jornada que é escrever (e atualizar) um livro.

Agradeço a tantos outros que, com seu apoio e feedback, me encorajaram e ajudaram a construir mais uma edição.

Dedicatória

Para as pessoas mais importantes da minha vida:

- meu pai, Miguel Armony, que não está mais aqui, mas se orgulharia mais do que ninguém ao ver este trabalho pronto;
- meus filhos, Clara e Henrique, e minha mulher, Maria José, que são a minha razão de estar aqui;
- minha mãe, Réjane, e meus irmãos, Nathália e Flávio, que, juntos com meu pai, me fizeram quem eu sou.

Prefácio

Este livro do Rafael Sabbagh é um marco para a Agilidade brasileira. Foi o primeiro livro de Scrum escrito em português, com dezenas de milhares de exemplares vendidos. Ao mesmo tempo que é excelente ver as atualizações desta terceira edição, é interessante perceber como o que foi dito na primeira edição há dez anos ainda faz sentido até hoje.

Neste livro você vai aprender muito mais do que um processo, mas uma cultura. Claro que o livro também está recheado de técnicas e em especial analogias, algo que o Rafael sabe fazer muito bem. Um destaque para a analogia do horizonte e o planejamento contínuo, em que afirmamos que o nível de detalhamento de um plano tem que ser inversamente proporcional à distância no tempo.

Fiquei muito empolgado em receber o convite do Rafael para escrever este prefácio. Tenho orgulho de ter sido o primeiro a falar sobre Scrum ao Rafael e desde então temos uma história lado a lado na Agilidade de mais de uma década. Começamos com palestras, participação em congressos, a criação de um treinamento que perdura até hoje - o CSPO (Certified Scrum Product Owner) - até fundarmos a K21 com Marcos Garrido e Carlos Felippe Cardoso (o CFC). Ao longo desses anos, todas essas iniciativas, juntamente com o livro, ajudaram a influenciar e a definir o mercado brasileiro de Agilidade.

Há quinze anos trabalhando com Agilidade, posso afirmar que a evolução alcançada na área vai além das técnicas, práticas, métodos e alcance (como novas áreas nas organizações). Uma grande evolução está na forma como quebramos resistências ao assunto. No início, alguns times queriam trabalhar com Agilidade, mas seus

gestores diretos eram contra pela não conformidade com o *status quo*. Com o tempo, e em especial com os resultados, esses gestores começaram a valorizar e incentivar outros times a fazerem o mesmo. A resistência então passou a ser de outras áreas da empresa. Essa barreira foi vencida quando começamos a ter exemplos em setores como financeiro, RH ou negócios, além de indústrias de todos os tipos: cosméticos, telecom, vestuário, farmacêuticas etc. Então a nova resistência estava no C-level, mas até mesmo o convencimento desses está se tornando mais natural. Claro que para cada uma dessas etapas, a forma de convencer mudou e se adaptou, como tudo na Agilidade.

Ao mesmo tempo em que ficamos muito felizes com o pensamento ágil se espalhando pelas empresas, transformando negócios e pessoas, ficamos também apreensivos com os mal-entendidos. Às vezes há uma interpretação errônea sobre o que é Agilidade: “é apenas um processo”, “é só uma mudança de nomes”, “só funciona em empresa pequena”, “aquele negócio em que não se planeja nada”. Além de um pensamento de que é algo que apenas se “instala”, sem considerar que na verdade Agilidade é uma cultura. É uma cultura de experimentação, uma cultura de ser *user-centric*, foco em valor e melhoria contínua. Existem até métodos que se dizem ágeis no nome, mas que criam processos gigantescos de várias etapas e entrega de valor em apenas a cada tantos meses.

Podemos dizer, no entanto, que o Brasil é um grande exemplo de Agilidade. Nos últimos anos, com a experiência da K21 internacional (EUA, México e Europa), várias vezes testemunhamos a admiração das pessoas com os cases que trazíamos do Brasil. Talvez em parte pela cultura de adaptação ser natural ao brasileiro, ou

pela nossa vontade de fazer diferença no que trabalhamos. Mas também pela didática de pessoas como o Rafael que foram capazes de formar milhares de pessoas, que se tornaram “agilistas”.

— Rodrigo de Toledo, PhD. (Certified Scrum Trainer e Accredited Kanban Trainer)

Prefácio da segunda edição

Imagine. Você contando aos seus netos que participou da grande transformação do início do século. Transformação na forma como as pessoas trabalham. Uma mudança de paradigma. Que começou com empresas de TI e, aos poucos, expandiu-se para todas as áreas da sociedade.

É como se o seu bisavô tivesse trabalhado com Taylor e lhe contasse como se sentiu com as primeiras mudanças daquela transformação do início do século passado.

> *Foi o Taylor aparecer e a empresa logo implementou as mudanças. O trabalho ficou mais específico, com alocação e especialização de tarefas para cada cargo. Criaram metas para a produtividade e um sistema de recompensas para o cumprimento delas. Ah, a empresa também instalou a tal esteirinha (assembly line, em inglês), meu bisneto.* — Conversa imaginária com seu bisavô que trabalhou com Taylor em 1916.

Pois bem. É isso que está acontecendo. Uma profunda transformação na essência de como trabalhamos. Uma transformação dos canais digitais, permitindo novos tipos de inovação e criatividade em vários domínios.

Uma transformação que afeta tanto os pequenos empreendimentos quanto os grandes segmentos da sociedade, tais como governo, transporte, comunicação de massa, arte e a ciência. Esta transformação não só mudará sociedades e economias inteiras, mas mexerá com a própria essência da natureza humana, do nosso comportamento e convívio em sociedade.

E você ali, na crista da onda, no lugar certo, na hora certa. Um protagonista deste admirável novo mundo buscando um modelo de transformação baseada em princípios e valores fortes que impulsione uma cultura de melhoria contínua. Tentando saber qual a parte do futuro que pode ser introduzida no presente.

Você já escolheu tomar a pílula vermelha (do filme *Matrix*). Não tem mais volta. Agora é fazer o upload do conhecimento, praticar e comemorar as conquistas.

Neste excelente livro, Rafael Sabbagh, um dos precursores do Scrum no Brasil, compartilha a teoria e a prática desse framework simples, efetivo e poderoso; um conhecimento essencial para a sua jornada.

Boa leitura.

— Paulo Caroli (cofundador do Agile Brazil e autor do livro *Lean Inception*)

Prefácio da primeira edição

Vivemos em um período de extrema riqueza na quantidade e qualidade de adoções de Agilidade em diferentes organizações. Hoje, é possível ver os assuntos ligados à Agilidade sendo discutidos nos mais diferentes níveis dentro dela. E um dos grandes catalisadores desse movimento de adoção é o Scrum. O Scrum fez com que Agilistas de todo o mundo ganhassem ferramentas para empatizar com os problemas gerenciais e econômicos das organizações.

Se fizermos uma busca rápida pela internet, encontraremos vários artigos, apresentações e vídeos falando sobre Scrum. Na prática, entender a mecânica básica do seu fluxo de trabalho é algo fácil e sem grandes desafios pela sua extrema simplicidade. Mas, segundo o próprio Rafael Sabbagh:

> *A adoção mundial de Scrum não significa que todos os problemas estão resolvidos. Longe disso, Scrum é apenas uma ferramenta que pode trazer diversos benefícios em comparação a outras formas de se conduzir projetos, mas somente se bem utilizada.*

Essa frase resume muito bem o grande desafio em questão. "Ser bem utilizado" é um estado paradoxal em se tratando do Scrum. Por ser um framework iterativo e incremental para se desenvolver produtos, ele também estimula que o próprio aprendizado acerca dos seus detalhes seja construído de forma iterativa e incremental. Isso significa que não é necessário ter uma ampla compreensão do Scrum para começar a usá-lo. Logo, por definição, "usar bem" o Scrum é um estado

que pode demorar um considerável tempo para ser alcançado.

A dinâmica de framework, as peculiaridades dos papéis e suas respectivas interações são uma parte difícil do Scrum. Existe uma grande variedade de possibilidades no uso de seus papéis, cerimônias, artefatos e regras. Na prática, compreender esses detalhes do framework é fundamental para permitir uma extensibilidade saudável, de forma que possa ser adotado em qualquer tipo de ambiente organizacional.

Tipicamente, o maior desafio se dá exatamente em função de sua extensibilidade. Na verdade, muito mais difícil do que usar bem o Scrum é estender seus papéis, cerimônias, artefatos e regras de forma a gerar uma congruência do contexto organizacional com os pilares do Scrum e com os valores do Manifesto Ágil.

É comum, por exemplo, que, movidos por um desejo de gerar menos conflitos no processo de adoção, sejamos compelidos a retirar algum dos elementos básicos do framework. Outro comportamento comum é usarmos de maneira desenfreada a extensibilidade para fazer o Scrum conviver de forma harmônica com todos os outros papéis, cerimônias, artefatos e regras já existentes na organização. Em ambos os casos, os resultados gerados são muito efêmeros e não promovem um uso sustentável do Scrum.

Adotar Scrum traz uma série de ganhos relacionados, mas também traz uma gama de perdas inerentes. "Perdas", nesse caso, possui o sentido de "coisas de que precisamos abrir mão" para usá-lo. Como um framework ágil, ele promove invariavelmente uma reflexão organizacional acerca de "o que" e "por que" melhorar. Logo, é comum que, com seu uso, seja necessário abrir

mão de algum papel, de alguma cerimônia, de algum artefato ou de alguma regra existente na organização.

Na realidade, ao contrário do que muitas organizações buscam, o Scrum é um meio, e não o fim. Isso significa que ele sempre estará em uma contínua mutação, pois deve ser usado para ajudar a organização a chegar a uma forma melhor no que tange ao *mindset* e aos processos de gestão de projetos e de produtos.

Logo, o propósito do Scrum não é se perpetuar em uma organização. Muito pelo contrário, almeja-se que um dia a organização não mais precise dele.

Uma forma típica de compreender plenamente essa dinâmica é ter muitas horas de experimentação real do Scrum em algum ambiente organizacional complexo. Na verdade, somente com essa carga de experiência na adoção será possível que muitos de seus detalhes sejam revelados e compreendidos.

Dadas todas essas peculiaridades da adoção de Scrum, é possível afirmar que Rafael Sabbagh conseguiu um feito singular: com este livro, Rafael conseguiu sintetizar de maneira didática todos os principais detalhes do uso de Scrum. Acredito piamente que este livro poderá encurtar o caminho para uma compreensão plena da sua essência e de como adotá-lo de maneira efetiva dentro uma organização.

Tive a honra de revisar este livro. Na verdade, há alguns anos, tive a honra maior de trabalhar junto do Rafael em um complexo processo de adoção de Scrum para uma grande empresa de telecomunicações aqui do Brasil. Lá tive a felicidade de conhecer e ser inspirado pela forma analítica e questionadora do pensamento do Rafael.

Na época, um dos objetivos desse trabalho foi produzir materiais sobre como usar Scrum para aquela organização. Apesar da aspiração que Rafael já tinha para escrever um livro, gosto de pensar que foi ali que ele começou a se concretizar.

Uma marca inconfundível do Rafael é destilar o conhecimento como se fosse uma cebola. Nessa abordagem, cada camada representa um aprofundamento do conhecimento que fora iniciado na camada anterior. Este livro tem essa abordagem da cebola. Dessa forma, tenho plena certeza de que, muito mais do que de um simples livro, você está munido de uma poderosa ferramenta de trabalho. Portanto, aproveite cada linha deste livro para que ele lhe ajude a transformar o seu dia a dia de trabalho em direção a um estado Ágil de gerenciar e desenvolver produtos.

— Manoel Pimentel Medeiros (autor do livro *The Agile Coaching DNA*)

Como este livro está organizado

Este trabalho está dividido em sete partes: *Introdução ao Scrum*, *Time de Scrum*, *Artefatos e compromissos do Scrum*, *Eventos do Scrum*, *Técnicas Complementares* e *Final*.

Na **Parte I — Introdução ao Scrum**, começo com as principais razões para escolher e utilizar Scrum. Em seguida, ofereço uma definição formal e sigo explicando o que Scrum é em detalhes, indicando também onde melhor o framework pode ser aplicado. No capítulo *Como é o Scrum?*, o leitor pode fazer um passeio pelo trabalho com Scrum e com diversos elementos associados, muitos deles adicionais ao framework. A Parte I se encerra com uma explicação das origens do Scrum e quais são as suas principais influências.

A **Parte II — Time de Scrum** trata em detalhes do que é, o que faz e como é cada uma das responsabilidades do Time de Scrum, ou seja, Desenvolvedor, Product Owner e Scrum Master. Mostro também nessa parte ideias e práticas associadas a essas responsabilidades, como resolução de impedimentos, motivação e facilitação, entre outras.

Na **Parte III — Artefatos e compromissos do Scrum**, explico o que são e como utilizar os artefatos Product Backlog, Sprint Backlog, e Incremento, além de seus respectivos compromissos, o Objetivo do Produto, o Objetivo do Sprint e a Definição de Pronto. Acrescento também a Definição de Preparado, que não é parte oficial do Scrum.

A **Parte IV — Eventos do Scrum** descreve as reuniões de Sprint Planning, de Daily Scrum, de Sprint Review e de

Sprint Retrospective, indicando em detalhes boas práticas para sua execução e problemas comuns enfrentados por seus participantes. Adiciono também o planejamento de entregas, o Refinamento do Product Backlog e atividades de inicialização. O Sprint também é abordado na Parte IV, além dos fatores determinantes para a escolha de sua duração, motivos para seu cancelamento e a sua relação com a Definição de Pronto.

Na **Parte V — Técnicas complementares**, ofereço alguns conceitos adicionais que podem ser muito úteis na execução do trabalho, como: Contratos Ágeis, User Stories, Estimativas Ágeis com Story Points (que também incluem Planning Poker e Velocidade) e os Gráficos de Acompanhamento do Trabalho (Release Burndown, Release Burnup e Sprint Burndown).

Na **Parte VI — Final**, o leitor tem acesso a um glossário dos termos relacionados ao Scrum e à bibliografia utilizada neste livro.

Para mais discussões, material e complementos ao conteúdo deste livro, acesse o *site* do livro em http://livrodescrum.com.br.

Introdução ao Scrum

Capítulo 1
Por que Scrum?

Conteúdo

1. Por que Scrum?
2. Entregas de valor.
3. Entregas frequentes.
4. Maior qualidade no produto desenvolvido.
5. Transparência sobre o progresso.
6. Redução do desperdício.
    - Produzir incrementalmente o mais simples que funcione.
    - Desenvolver o produto com base em seu próprio uso.
    - Planejar utilizando apenas o nível possível de detalhes.
    - Utilizar apenas os artefatos necessários e suficientes.
7. Foco nas pessoas.
    - Trabalho em equipe e a autonomia.
    - Facilitação e remoção de impedimentos.
    - Melhoria contínua.
    - Ritmo sustentável de trabalho.
    - Trabalho de ponta a ponta.
8. Redução dos riscos do desenvolvimento do produto.

1.1 Por que Scrum?

Scrum existe desde o início dos anos 1990, mas foi só na década seguinte que se tornou popular. Ele ganhou o mundo, desbancou métodos tradicionais e se tornou a forma mais bem-sucedida de se trabalhar em desenvolvimento de software. Uma pesquisa de 2015 mostra que projetos de desenvolvimento de software que utilizam Scrum ou métodos similares têm 3,5 vezes mais chances de sucesso (THE STANDISH GROUP, 2015).

O crescimento espetacular na adoção de Scrum nos últimos anos não significa que todos os problemas estão resolvidos. Longe disso. O uso do Scrum implica em uma profunda mudança em como as organizações entendem e lidam com o trabalho, com as pessoas que realizam esse trabalho, com quem solicita e com quem utiliza de seus resultados. Assim, embora o Scrum seja de fácil compreensão, sua adoção pode ser difícil e dolorosa. Não é incomum essas organizações buscarem atalhos e simplificações sem, no entanto, transformarem o que, de fato, é necessário.

Scrum permite aumentar as chances de sucesso, entregar valor mais rápido e, desde cedo, lidar com as inevitáveis mudanças de escopo, transformando-as em vantagem competitiva. Seu uso pode também aumentar a qualidade do produto entregue e melhorar a produtividade das equipes.

Em suma, o uso de Scrum permite aumentar a eficiência na realização do trabalho de desenvolvimento de produtos e, mais importante ainda, a eficácia dos seus resultados.

Eficiência e eficácia

"*Eficiência significa fazer certo as coisas; eficácia significa fazer as coisas certas.*"

- Peter Drucker

Scrum é aplicado no desenvolvimento de produtos com características igualmente variadas. Em produtos críticos de centenas de milhares de dólares e em produtos internos simples. No desenvolvimento de softwares comerciais, de negócios digitais, de softwares embarcados, de aplicativos para dispositivos móveis, de softwares financeiros e de jogos. É adotado com sucesso por organizações de diversos tamanhos e tipos, de multinacionais a *startups*, de famosas a desconhecidas.

Seu uso não se limita a desenvolvimento de software, embora tenha sido concebido originalmente com essa finalidade. Scrum é hoje também usado para diferentes fins, como criação de campanhas de *marketing*, desenvolvimento de produtos de *hardware* e a concepção e execução de serviços diversos.

Neste livro, considero **DESENVOLVIMENTO DE UM PRODUTO** o trabalho de realização das atividades necessárias para a criação e evolução do produto, ao adicionarmos e modificarmos funcionalidades, características e comportamentos, além da corrigirmos problemas e erros.

No caso de desenvolvimento de software, por exemplo, essas atividades podem incluir programação, diferentes tipos de testes e de documentação etc. Com Scrum, o trabalho de desenvolvimento de um produto é realizado continuamente.

Exemplo: loja virtual

Temos como objetivo vender nossos produtos on-line. Ao final do primeiro de vários ciclos curtos de desenvolvimento, já implementamos uma loja virtual que, quando colocada em produção, permitirá a venda, mesmo que básica, de nossos artigos mais rentáveis para os usuários compradores com maior potencial de compra. Em seguida, recebemos o feedback de pessoas relevantes e implementamos, no ciclo seguinte, algumas melhorias decorrentes que consideramos necessárias.

Somente agora disponibilizaremos a loja para os nossos usuários compradores, visando apenas àquele perfil com maior potencial de compra. Esses usuários, ao começarem a acessar, realizar buscas e comprar, gerarão métricas que nos ajudarão a definir o que melhorar na loja e o que implementar em seguida, nos próximos ciclos. Continuaremos com o trabalho, ciclo após ciclo e, aos poucos, alcançaremos mais usuários compradores e

ofereceremos mais produtos, e em cada vez melhorando a experiência.

Exemplo: campanhas de marketing

Somos uma agência de marketing digital e desenvolvemos nossas campanhas trabalhando em ciclos curtos de uma semana. Nossas equipes são multifuncionais e cada uma possui fortes conhecimentos em SEO, ciência de dados, criação de roteiro e desenho gráfico, entre outros. Os membros das equipes trabalham em conjunto, sempre em busca de um objetivo bem definido, planejado no início do ciclo. Ao seu final, fazemos uma videoconferência com os clientes que foram o foco do nosso trabalho e coletamos feedback, que vai alimentar o trabalho do ciclo seguinte.

Exemplo: automação de processos

Desenvolveremos a automação de processos internos repetitivos em nossa organização, com o objetivo primário de "liberar colaboradores para realizarem trabalhos mais nobres". De acordo com esse objetivo, o maior valor está em desenvolver a automação daquelas rotinas que ocupam mais tempo do nosso pessoal. E é por aí que começaremos, desde o primeiro de vários ciclos curtos de desenvolvimento do produto, gerando rotinas de automação e substituindo de parte em parte o trabalho manual. Seguiremos priorizando o trabalho perseguindo esse objetivo e entregando o mais frequentemente possível, liberando cada vez mais tempo do nosso pessoal.

Exemplo: telecomunicações

Em nossa empresa de telecomunicações, possuímos produtos de software que nos ajudam na operação do dia a dia e na interação com os consumidores dos nossos serviços. Mantemos equipes razoavelmente estáveis, que cuidam tanto do desenvolvimento quanto da operação e evolução desses produtos. São esses produtos que, de fato, apoiam todo o funcionamento da nossa empresa e nos permitem ter um negócio lucrativo. As equipes trabalham com Scrum em ciclos curtos, continuamente adicionando funcionalidades e modificando os produtos de forma a nos ajudar a atender cada vez melhor às necessidades dos nossos usuários, tanto internos quanto externos. Essa constante evolução nos permite estar à frente da concorrência e responder rapidamente a qualquer movimento relevante do mercado.

Ao aprender Scrum, você passará por termos como entregas de valor, trabalho em equipe, autonomia, facilitação, motivação e relacionamento com os clientes e usuários, entre tantos outros. Scrum utiliza-se de poucos conceitos realmente novos e, na minha opinião, essa é uma de suas grandes qualidades: juntar diversas ideias e práticas de mercado, muitas já conhecidas e consagradas, de uma forma organizada e direcionadas a contextos em que se mostram adequadas.

Clientes e usuários

Neste livro, chamo de clientes aqueles que solicitam ou contratam o desenvolvimento do produto, e de usuários aqueles que efetivamente utilizam o produto. Clientes podem ser usuários, mas não necessariamente o são.

É importante lembrar que não existe uma solução única para todos os problemas. Scrum é um framework simples e pequeno e, assim, funciona bem em cada contexto se for usado em conjunto com outras técnicas e práticas a serem escolhidas, experimentadas e adaptadas. É uma excelente escolha para o desenvolvimento de produtos, para as organizações que realizam esse trabalho e para os seus clientes. Os benefícios obtidos com o uso do Scrum serão explicados ao longo deste livro. A seguir, sumarizo alguns deles.

1.2 Entregas de valor

Um estudo de 2001 informa os resultados de uma pesquisa em que foram analisados 400 projetos de desenvolvimento de software durante 15 anos, em uma época em que pouquíssimos projetos utilizavam Scrum e similares. Em nenhum dos projetos analisados mais de 20% das funcionalidades entregues foram utilizadas por seus usuários (COHEN et al., 2001).

Esses números evidenciam o que nossa experiência já nos mostra: o quanto somos ineficazes em nosso trabalho, já que a maior parte do que produzimos não atende às necessidades dos usuários.

No cenário mais tradicional, é definida uma longa lista de funcionalidades a serem desenvolvidas antes de qualquer entrega, sob a crença de que uma extensa e detalhada análise de requisitos inicial vai se traduzir em um produto de sucesso. Assim, clientes ou pessoas de negócio decidem antecipadamente os detalhes do produto que será criado durante meses adiante e mudanças posteriores nessa definição não são bem-vindas. São, desse modo, induzidos a listar tudo o que

puderem imaginar naquele momento, gerando como consequência um grande desperdício. Essa prática nociva de definir e detalhar o escopo antes de iniciar o desenvolvimento do produto é conhecida como BDUF (veja em *Scrum é ágil*, no capítulo *O que é Scrum?*).

Em outro cenário comum, o trabalho é realizado em ciclos curtos e com um escopo flexível e evolutivo, assim como fazemos com o Scrum. No entanto, embora o produto seja demonstrado e receba feedback no decorrer de seu desenvolvimento, ele apenas sofrerá algum tipo de uso real após um longo período.

Em ambos os cenários, cometemos o erro de definir o produto nos mantendo distantes daqueles que vão utilizá-lo, bem como do uso em si. A entrega para os usuários será realizada apenas ao final do projeto ou após a conclusão de uma grande etapa. Dessa forma, investimos no trabalho por um longo tempo sem obter qualquer retorno, que é esperado apenas como resultado dessa entrega. Não há, no entanto, qualquer garantia de que a entrega realmente vai atender às necessidades dos usuários e, portanto, de que vai gerar sequer algo próximo do valor esperado.

O USO DEFINE O PRODUTO.

Essa afirmação circular que criei e que tenho utilizado em meus treinamentos indica que é somente ao interagir com partes funcionando do produto, e que potencialmente lhe gerem valor, que o usuário começa a ser capaz de entender melhor como esse produto pode ajudá-lo a resolver suas necessidades. Como consequência, uma das práticas de mercado mais arriscadas é a de detalharmos um grande escopo e

demorarmos a oferecer aos usuários algo pronto que eles possam ver, utilizar e, somente então, oferecer feedback a partir desse uso tardio. O que criamos e entregamos para nossos usuários constitui apenas uma hipótese de algo útil para eles, que assim deveria ser validada ou invalidada o mais rapidamente possível. Como conclusão, o uso do que já foi entregue é necessário para guiar a definição e construção do produto. Quanto mais postergarmos essa possibilidade de feedback, maior é o potencial desperdício gerado com a construção de um produto realizada sob premissas erradas.

Com o uso do Scrum, o desenvolvimento do produto ocorre em ciclos curtos. Em cada um deles, implementamos ao menos um incremento pronto desse produto, definido a partir do que acreditamos serem as maiores necessidades para os clientes e usuários naquele momento. Cada um desses incrementos, somado aos anteriores, é um produto funcional. Esse produto é, portanto, um candidato a entrega, que recebe algum feedback de clientes e demais partes interessadas ao verem e interagirem com ele em uma reunião ao final de cada ciclo.

No entanto, é apenas quando colocarmos essas partes funcionando do produto nas mãos de usuários que receberemos o feedback mais valioso, provindo do uso real do produto. Esse feedback permite que os detalhes do produto emerjam ao longo de seu desenvolvimento. Por essa razão, buscamos realizar entregas para usuários reais desde cedo e com frequência. E quando, por exemplo, tomamos decisões equivocadas sobre o produto, esse feedback obtido rapidamente a partir do uso gera transparência sobre o problema e permite correções de rumo. Dessa forma, o produto é continuamente definido e seu trabalho de

desenvolvimento é ordenado a partir do entendimento em constante evolução das necessidades a serem atendidas, em consequência ao uso do próprio produto.

Esse processo de definição do produto a partir do seu uso busca garantir o desenvolvimento de um produto de valor.

1.3 Entregas frequentes

Em projetos tradicionais, o trabalho consiste em realizarmos tarefas definidas em um longo plano, que geralmente devem ser cumpridas quase em sua totalidade para que possamos entregar o produto para seus usuários o utilizarem. Nesses cenários, não existe a preocupação em convergir desde cedo para algo entregável, pois há a crença equivocada de que "tudo" deve ser criado para que possamos entregar algo. Assim, apenas uma ou poucas entregas são programadas.

Para o desenvolvimento de software, por exemplo, é comum ordenarmos as tarefas de forma a desenvolver o produto módulo após módulo, camada após camada, passo após passo no fluxo do uso do produto ou, mais frequentemente, utilizando alguma combinação dessas possibilidades. Algo similar acontece com outros tipos de produtos e serviços: o trabalho é orientado ao cumprimento integral das tarefas planejadas.

Essa forma de trabalhar nos leva ao risco máximo de **tudo ou nada** para cada entrega planejada. Ou seja, ou chegamos à data estabelecida com tudo pronto, ou não teremos nada a entregar. Infelizmente o pior cenário é o mais provável de acontecer. Não é possível, por exemplo, entregar um software sem testes, ou sem a camada de

interface com o usuário, ou sem um passo essencial no seu fluxo de uso, como o pagamento de uma compra realizada. Dessa forma, com frequência geramos atrasos nas entregas.

O uso de equipes funcionais, cenário em que cada equipe realiza uma função distinta e bem definida, reforça o problema nas entregas. Essas equipes, em conjunto, não são geralmente capazes de convergir para algo de valor com frequência para seus usuários. O trabalho a ser realizado, nesses casos, deve passar por uma sequência de equipes funcionais, cada uma realizando a sua parte no desenvolvimento do produto, até que possa ser entregue. Sempre que há essa transferência de responsabilidade sobre um item de trabalho (ou seja, uma *passagem de bastão*), esse item entra na fila de trabalho a ser realizado da equipe funcional que o recebe. Há, consequentemente, a geração de um tempo de espera, pois a equipe estará trabalhando de acordo com suas prioridades quando o item chegar. O mesmo acontecerá quando essa última equipe funcional passar o item de trabalho à equipe seguinte, e assim por diante.

As esperas aumentam consideravelmente o tempo necessário para implementarmos algo que possa ser entregue aos usuários e, portanto, diminuem as oportunidades de entregas. Elas também podem tornar a possibilidade de entrega ainda mais imprevisível do que já é, potencializando os riscos de não as realizar nas datas previstas. O problema piora ainda mais quando ocorre o refluxo, ou seja, quando itens de trabalho são devolvidos a alguma das equipes funcionais anteriores, em geral por conta de defeitos encontrados (por exemplo, por uma equipe de controle de qualidade ou de auditoria).

No trabalho com Scrum, o produto é construído em ciclos curtos e incrementalmente, partindo-se do que cremos serem suas partes mais importantes. Em cada ciclo, o foco não está no simples cumprimento de tarefas planejadas, mas sim na realização de um trabalho convergente para algo entregável. Ciclo a ciclo, o time produz algo pronto, que funciona e que potencialmente representa valor para seus clientes e usuários. Cada incremento produzido em cada ciclo representa um passo seguro de trabalho pronto, já que oferece uma oportunidade de entrega. Como consequência, o produto pode, inclusive, já ser introduzido no mercado em um curtíssimo prazo.

O time que desenvolve o produto com Scrum é multifuncional ou multidisciplinar, o que significa que possui entre seus membros todos os conhecimentos e habilidades necessários para o desenvolvimento do produto, de ponta a ponta. Esse time produz partes prontas e funcionando do produto em cada ciclo curto do seu desenvolvimento. Dessa forma, Scrum evita o cenário em que um time deveria esperar pelo trabalho de outros times para poder começar a trabalhar em algo.

No desenvolvimento de software não é incomum o item ter que passar necessariamente pelo trabalho de validação de requisitos, de *design* e de segurança, por exemplo, antes de chegar ao time de desenvolvimento do produto. Na outra ponta, Scrum também evita ainda que o trabalho realizado por esse time tenha que passar por outros times antes de chegar nas mãos dos usuários. Ou seja, o trabalho ainda poderia ter que passar, por exemplo, pelos testes de um time de qualidade, que por sua vez devolveria ao time uma lista de problemas a serem corrigidos e, somente após aprovado em novos testes, o produto chegaria a um time de operações que,

dentro de suas prioridades, finalmente poderia o colocar nas mãos dos usuários.

O uso de times multifuncionais, portanto, elimina ou, ao menos, minimiza dependências externas. Ou seja, buscamos eliminar esperas no desenvolvimento do produto, sejam as esperas por entradas ou, na outra ponta, por validação, maximizando a eficiência no trabalho. Aumentamos, portanto, a chance de termos partes do produto prontas em cada ciclo de seu desenvolvimento. Seguindo a mesma ideia, ainda que haja mais de um time trabalhando no desenvolvimento de um mesmo produto, cada time será multifuncional.

Caso identifique uma habilidade necessária em que é deficiente, o time busca formas de incorporar aprendizado suficiente para ser capaz de desenvolver o produto. Além disso, ao trabalharem juntos e dividirem responsabilidade sobre o produto sendo desenvolvido, os membros do time aprendem uns com os outros e o conhecimento vai gradualmente se disseminando entre eles. Esse conhecimento compartilhado ajuda a minimizar as dependências internas, reduzindo-se ainda mais possíveis filas e esperas e aumentando a eficiência do time.

Além de convergir com frequência para algo pronto, o trabalho com Scrum é ordenado e o escopo, flexível. Uma vez que trabalhamos sempre nas próximas partes mais importantes do produto, em ciclos curtos, raramente é necessário atrasarmos uma entrega. Ao chegarmos à data preestabelecida, as chances são de que já teremos produzido um valor suficiente para realizarmos os objetivos de negócios da entrega. Em outras palavras, o problema já estará suficientemente bem resolvido. Se necessário, deixaremos de fazer, apenas, as partes menos importantes do que foi imaginado, e que assim

não são essenciais para os objetivos da entrega. Da mesma forma, caso haja um prazo final estabelecido para todo o trabalho de desenvolvimento do produto, é natural esperar que, ao chegarmos nele, a visão inicialmente definida tenha sido suficientemente bem realizada, por termos realizado o trabalho a partir das partes mais importantes dos problemas mais importantes.

1.4 Maior qualidade no produto desenvolvido

As partes do produto só serão consideradas prontas, em cada ciclo de seu desenvolvimento, se possuírem a qualidade necessária para serem entregues aos seus clientes e usuários. Dessa forma, evitamos que problemas sejam tardiamente detectados.

O time que trabalha com Scrum é integralmente responsável pelo trabalho de desenvolvimento do produto e, assim, possui todas as habilidades e conhecimentos necessários para fazê-lo. É parte integral disso, portanto, realizar dentro do ciclo de desenvolvimento do produto todas as atividades necessárias para garantir a qualidade desse produto, ou seja, que as partes do produto implementadas estejam funcionando adequadamente.

Além dessas atividades, faz parte do trabalho realizado em cada ciclo a integração com o resto do produto já implementado. Também faz parte do trabalho garantir a integração com outros produtos, caso haja esse tipo de dependência. Resolver essas questões no próprio ciclo nos permite identificar e corrigir problemas desde cedo.

Para o desenvolvimento de software, ferramentas de automação para testes, sejam técnicos ou de negócios, e integração contínua auxiliam nessas atividades. Não existem no Scrum times externos responsáveis por avaliar ou garantir a qualidade do produto em desenvolvimento, já que esse trabalho é de inteira responsabilidade do próprio time que desenvolve o produto.

O produto desenvolvido até então é visto, experimentado e utilizado. Ao final de cada ciclo, clientes e demais partes interessadas participam de uma reunião para fornecerem seu feedback sobre o produto desenvolvido até então. Um feedback ainda mais profundo e valioso é obtido dos usuários do produto ao utilizarem as partes do produto já entregues, o que é possibilitado pelas entregas frequentes. Em ambos os casos, o produto será modificado a partir do feedback obtido para melhor atender aos clientes e usuários do produto. Tudo isso busca assegurar a qualidade de negócios.

Assim, ao invés de postergarmos a validação do que é produzido e acumularmos problemas, o produto já é desenvolvido com a qualidade necessária para ser utilizado.

1.5 Transparência sobre o progresso

Diversas práticas e artefatos do Scrum ou associados a ele visam a garantir a transparência sobre o progresso do desenvolvimento do produto para seus participantes e envolvidos e uma consequente alta visibilidade.

Com o uso do Scrum, os clientes e demais partes interessadas estão muito mais próximos, e colaboram

com o time (ou times) na evolução do produto. Ao final de cada ciclo curto de seu desenvolvimento, é apresentado a eles o que foi criado e está funcionando. Mais determinante ainda é o fato de que os clientes e usuários recebem com frequência as próximas partes mais importantes do produto, prontas para uso. Com Scrum, portanto, o senso de progresso é real, já que se dá a partir de algo concreto. Já em projetos tradicionais, com escopo detalhado e entregas realizadas após longos períodos, o progresso é informado a partir de percentuais ou de etapas cumpridas e, por muitas vezes, se mostra distante da realidade.

O time que trabalha com Scrum também tem grande visibilidade de seu próprio progresso. O feedback que o time obtém ao final de cada ciclo, diretamente dos clientes e demais partes interessadas, permite descobrir desde cedo se está na direção certa e o quanto efetivo seu trabalho está sendo.

Cada time realiza uma reunião todo dia exatamente para criar transparência sobre o trabalho para os seus próprios membros e, assim, tornarem-se capazes de se adaptar ao que aconteceu e planejar o que será realizado até o próximo dia.

Além dessas práticas, existem ferramentas que podem ser utilizadas para aumentar a transparência sobre o progresso do trabalho de desenvolvimento do produto, como um quadro de tarefas e os Gráficos de Acompanhamento do Trabalho, explicados nos capítulos *Sprint Backlog* e *Burndown e Burnup: acompanhando o trabalho*, respectivamente.

As reuniões de retrospectiva, também realizadas ao final de cada ciclo, permitem criar transparência sobre como o time está realizando seu trabalho. Dessa forma, o próprio

time pode inspecionar e adaptar suas práticas para se tornar mais efetivo.

1.6 Redução do desperdício

Em seguida, identifico como as regras e práticas do Scrum ajudam a evitar alguns tipos de desperdícios:

- produzir incrementalmente o mais simples que funcione;
- desenvolver o produto com base em seu próprio uso;
- planejar utilizando apenas o nível possível de detalhes;
- utilizar apenas os artefatos necessários e suficientes.

Produzir incrementalmente o mais simples que funcione

O desperdício é reduzido e evitado com Scrum principalmente por meio da busca pela simplicidade. A ideia central é que, no processo de desenvolvimento do produto, produzamos, de pouco em pouco, o mais simples que funcione para resolver os problemas que o produto se propõe a resolver.

A próxima parte a ser implementada é sempre uma parte pequena e simples do produto, mas que estará pronta para uso, e que resolve apenas a parte mais importante do próximo problema mais importante. Para fazê-lo, implementamos apenas o suficiente e necessário, tanto da estrutura e partes técnicas que sustentam o produto, quanto das próprias funcionalidades, características e comportamentos desse produto.

De simples em simples, podemos construir o complexo. Mas apenas o suficiente e necessário.

Com essa frase, eu busco traduzir como a natureza incremental do trabalho com Scrum ajuda a reduzir o desperdício. Uma ideia de solução complexa e custosa, por mais bela que pareça, sempre carrega muitas funcionalidades que não serão utilizadas e leva à utilização de soluções técnicas mais complexas que o necessário. Ao construirmos o produto de incremento simples em incremento simples, criamos a complexidade apenas suficiente e necessária para resolver o problema proposto.

Desenvolver o produto com base em seu próprio uso

Em 2002, o presidente do Standish Group, um reconhecido grupo de pesquisa norte-americano, apresentou em uma conferência na Itália os resultados de uma pesquisa limitada, que mostrou que 64% das funcionalidades presentes em produtos de software nunca ou raramente eram utilizadas (veja a figura a seguir). Embora as origens desses números sejam nebulosas, a pesquisa passou a ser largamente referenciada e, por essa razão, apresento-a aqui.

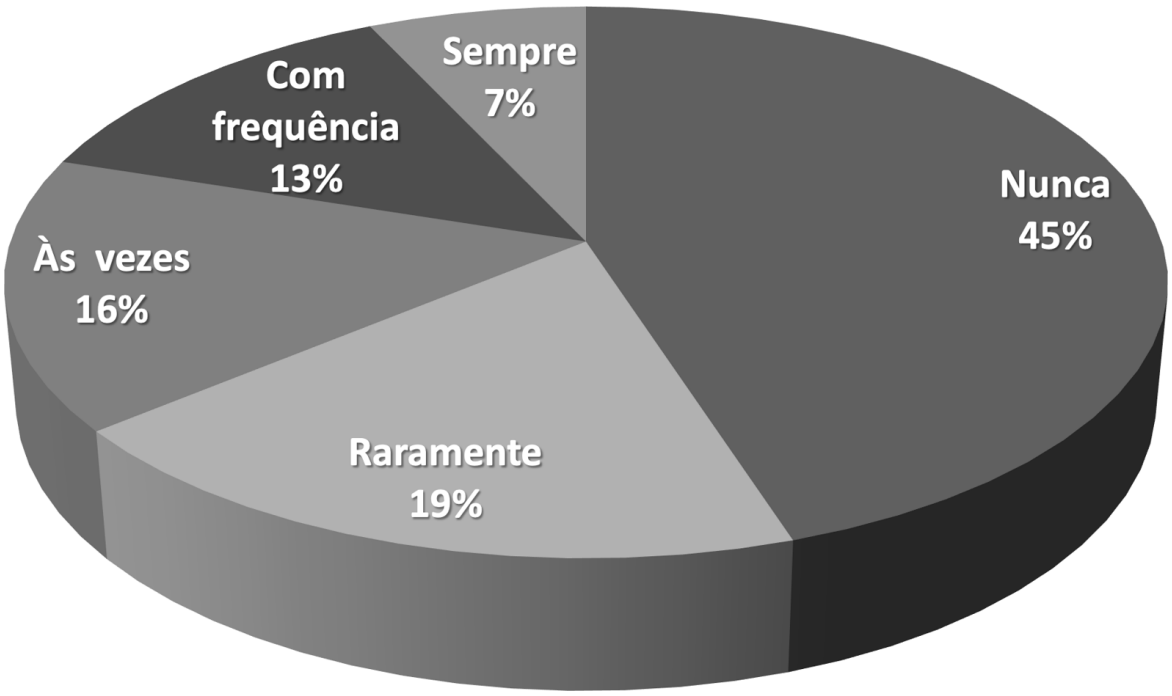


Figura 1.1: Percentual de uso das funcionalidades em software, de acordo com o presidente do Standish Group em 2002

Também mencionei anteriormente um estudo de 2001 que mostra que não menos que 80% das funcionalidades implementadas em projetos de software eram puro desperdício. Para piorar, esse mesmo estudo mostra que apenas algo entre uma em cada dez e uma em cada cem linhas de código produzidas foram de fato entregues, variando com a complexidade e tamanho do projeto (COHEN et al., 2001). Esses números mostram um gigantesco desperdício gerado no desenvolvimento de software.

No trabalho com Scrum, o produto emerge e é entregue ao longo de seu desenvolvimento a quem o utilizará, em vez de ter seus detalhes definidos no começo. A definição de como o produto vai evoluir, ou seja, do que será feito em seguida, parte principalmente do feedback e métricas provindos do uso do que já foi entregue. Assim, o trabalho de desenvolvimento do produto é

ordenado e realizado a partir das necessidades dos clientes e usuários e, então, as partes entregues do produto deverão ser utilizadas.

Ainda que exista um desperdício, uma vez que partes já prontas do produto são modificadas a partir do feedback e métricas de uso, este não é comparável àquele gerado quando tentamos prever os detalhes antecipadamente.

Planejar utilizando apenas o nível possível de detalhes

Métodos tradicionais buscam planejar, no início do projeto, todo o trabalho a ser realizado no desenvolvimento do produto (ou, ao menos, da próxima entrega) com um alto nível de detalhes. Os planos gerados muitas vezes descrevem cada uma das funcionalidades do produto, quem as produzirão e quando serão produzidas. O resultado é que esses planos, em geral, não se traduzirão em uma realidade futura.

Scrum não possui essa fase inicial de planejamento detalhado, mas isso não significa que o produto não possa ser descrito desde o seu início até o fim. No entanto, o princípio básico aqui é que planejemos apenas com o nível de detalhes que é possível enxergarmos, e o façamos com uma alta frequência.

Com o uso do Scrum, utilizamos um objetivo ou uma visão de produto, que representa, em alto nível, o problema a ser resolvido a partir do uso do produto. Para realizar esse objetivo, podemos talvez definir um plano em alto nível que o desmembre. Esse plano pode prever os principais objetivos de negócios a serem realizados em cada entrega, por exemplo, e imaginar quando isso vai acontecer aproximadamente. Para o trabalho ser feito

até a próxima entrega, podemos criar um plano com o trabalho provável a ser feito e o próximo objetivo de negócios a ser realizado a partir desse trabalho, que é um incremento à realização do objetivo do produto. Para cada ciclo de desenvolvimento do produto de um time, seus membros planejam, com detalhes, que trabalho farão nesse ciclo e como vão fazê-lo, visando a realizar um objetivo de negócios, que por sua vez é um incremento à realização do objetivo estabelecido para a entrega.

Exceto pelo plano do ciclo atual, que já possui uma granularidade mais fina, esses planos são modificados com frequência, de forma a refletir a realidade que se apresenta em cada momento. Esse comportamento é melhor demonstrado pela analogia do horizonte.

Definição: analogia do horizonte

Digamos que você está olhando para uma cidade e pode enxergar desde as construções, carros e pessoas que estão mais próximos de você até o que está no horizonte distante.

Ao olhar para o que está mais próximo, você consegue descrever o que vê com um excelente nível de detalhes e com uma granularidade fina, ou seja, em pequenas porções de informação. O que está um pouco mais distante, você ainda pode descrever com um bom nível de detalhes, ainda que menor, e com uma granularidade um pouco mais grossa.

Mas, à medida que olha para cada vez mais longe, os detalhes que você pode usar em sua descrição vão gradualmente diminuindo e a granularidade vai ficando cada vez mais grossa. Se você então tentar descrever

tudo o que vê, até o horizonte, com um alto nível de detalhes, sua descrição certamente não corresponderá à realidade e você vai errar muito. Para quanto mais distante você estiver olhando, mais incorreta estará essa sua descrição detalhada e maior será o desperdício gerado.

Mas se, ao contrário, você descrever apenas o que consegue enxergar, somente com o nível de detalhes possível, à medida que caminha na direção do horizonte você poderá refinar a sua descrição, adicionando detalhes conforme os enxerga melhor, bem como refletir as mudanças que ocorrerem durante a caminhada.

Um planejamento tradicional geralmente descreve em detalhes o que será feito durante todo o desenvolvimento do produto ou, ao menos, todo o trabalho até a próxima entrega, meses à frente. É muito comum ver esses planos descritos em gráficos de Gantt, que mostram tarefas detalhadas e alocação de "recursos" no tempo.

Essa prática pode ser comparada a descrevermos detalhadamente todo o caminho, desde o que está mais próximo de você até o que está mais próximo da linha do horizonte. O resultado é um plano de altíssima precisão, ou seja, detalhado, porém com baixíssima chance de acerto, levando à geração de desperdício.

Com o uso do Scrum, somente planejamos com o nível de detalhes que podemos enxergar. Para planejarmos um trabalho a ser realizado, por exemplo, até o próximo dia, podemos utilizar um nível de detalhes bastante alto e uma granularidade bem fina. Para as próximas duas semanas de trabalho — um ciclo de desenvolvimento do produto, por exemplo — podemos usar um nível de

detalhes ainda razoavelmente alto, porém mais baixo do que para apenas um dia.

Já ao planejarmos uma entrega que acontecerá daqui a várias semanas ou ao planejarmos o semestre inteiro de trabalho, quanto mais longe olhamos no tempo, a quantidade de detalhes diminui e a granularidade fica mais grossa, de forma que pouquíssima informação pode ser utilizada para planejarmos o que está mais distante. À medida que o desenvolvimento do produto caminha no tempo, esse plano é refinado e mais detalhes são gradualmente adicionados, além de se refletirem as mudanças que acontecerem pelo caminho.

O plano com itens a serem desenvolvidos pelo time ou times no Scrum reflete essa forma de planejamento. Os itens do alto dessa lista, chamada de Product Backlog, possuem uma granularidade mais fina. Ou seja, são itens menores e que representam mais detalhes. Ao descermos na lista, os itens vão ficando cada vez maiores e com menos detalhes. Mais detalhes do que o necessário e suficiente significam geração de desperdício.

Em outras palavras, não trabalhamos com o escopo detalhado desde o início do trabalho de desenvolvimento do produto. E esse é um princípio básico do Scrum. O escopo de curtíssimo prazo é suficientemente detalhado para ser desenvolvido. Já o escopo de curto prazo recebe um nível um pouco menor de detalhes, enquanto que, se for necessário detalhar os escopos de médio e longo prazos, eles terão tão menos detalhes quanto mais distantes no tempo estiverem.

Utilizar apenas os artefatos necessários e suficientes

Ferramentas podem ser úteis, mas até o ponto em que começam a gerar desperdício. Aquelas usadas no suporte à gestão do trabalho e ao desenvolvimento do produto devem cumprir seu papel sem adicionar burocracia ao trabalho. De forma geral, as ferramentas mais simples e que funcionam permitem a geração dos melhores resultados.

Ferramentas utilizadas pelos times para esboçar, planejar e monitorar seu trabalho de desenvolvimento do produto são mais bem representadas em meios físicos do que em meios virtuais. Quadros brancos na parede são bons exemplos de meios para essas ferramentas. Eles funcionam como irradiadores de informação, impondo uma transparência que não é possível, por exemplo, ao usarmos ferramentas de software. Os softwares podem ser úteis e trazer diversas facilidades. No entanto, quando mal utilizados, terminam por impor uma forma de se trabalhar, que frequentemente é equivocada. Por colocar mais empecilhos do que facilitar o trabalho, geram assim desperdícios bastante custosos.

Documentos que refletem planos, esquemas e especificações também podem ser úteis no suporte à gestão e ao próprio trabalho. O time, no entanto, concentra-se em criar e manter apenas aquilo que será, de fato, utilizado. Produzir documentação além do suficiente e necessário é um desperdício.

A documentação em excesso desloca o foco e diminui a usabilidade daquela que é realmente necessária, o que podemos ver como um desperdício ainda maior.

Discussão: quanta documentação devemos produzir?

Produzir documentação além do suficiente e necessário é um desperdício. Não é incomum a exigência da geração de uma grande quantidade de documentos no desenvolvimento de produtos. Os pretextos mais utilizados para esse tipo de obrigação incluem: a necessidade de facilitar futuras manutenções, a garantia da entrega do que foi acordado e a conformidade com algum padrão ou com alguma exigência legal. Em muitos casos, esse trabalho de geração de documentos é tão grande quanto ou até maior do que o trabalho de se construir o produto. E, na realidade, quando muito apenas uma pequena parte dessa documentação é realmente necessária. Dessa forma, um grande desperdício é gerado.

No desenvolvimento de software, raramente conseguimos utilizar de forma efetiva a documentação escrita para a manutenção do produto em um momento futuro. Essa documentação, na prática, é extensa e dificilmente se mantém atualizada. Boas práticas de codificação e os próprios testes automatizados constituem uma documentação suficiente e necessária para a manutenção do produto.

Quanto à garantia das entregas do que foi acordado, o uso da documentação é substituído pela relação mais próxima com os clientes, construída pela própria dinâmica do Scrum. Os ciclos curtos do framework geram, para os clientes e demais partes interessadas, oportunidades frequentes de interagir com o time para, a partir do feedback sobre o que foi produzido, influenciar diretamente o produto em desenvolvimento, fomentando a criação de um ambiente de confiança.

Sobre padrões e exigências legais, posso dizer por experiência que estes sempre podem ser desafiados, tanto com relação à sua forma quanto com relação à sua

necessidade como um todo. Ou seja, recomendo buscar sempre reduzir e eliminar o que for possível. Devemos gerar apenas a documentação que realmente se prova necessária.

1.7 Foco nas pessoas

Em vez de tratar as pessoas como recursos, Scrum as entende como seres criativos, com aspirações e sentimentos e, assim, seu uso potencializa a motivação e a produtividade dos times de desenvolvimento de produto. Diversos fatores no Scrum trazem esse foco para o ser humano, entre os quais posso citar:

- o trabalho em equipe e a autonomia do time na realização desse trabalho;
- a existência de facilitação e de remoção de impedimentos;
- a melhoria contínua;
- um ritmo sustentável de trabalho;
- realização do trabalho de ponta a ponta.

Trabalho em equipe e autonomia

O time de desenvolvimento do produto com Scrum é propositalmente pequeno, de forma que seus membros possam interagir e se comunicar com eficiência durante todo o dia para realizar o trabalho. Ainda que haja mais de um time trabalhando no desenvolvimento de um mesmo produto, cada time será pequeno. Esse trabalho em equipe é considerado essencial para o sucesso no desenvolvimento do produto.

A responsabilidade pelo trabalho não recai apenas sobre o indivíduo, mas sobre seu time como um todo, que tem a autonomia e se organiza para decidir o quanto é possível de se fazer em cada ciclo de desenvolvimento do produto, definir como vai fazê-lo e monitorar seu próprio progresso.

Em cada ciclo, o próprio time planeja uma quantidade de trabalho que acredita ser capaz de implementar, gerando cada parte do produto para realizar um objetivo de negócios bem definido, que guia e dá propósito ao seu trabalho. Os membros do time tornam-se igualmente responsáveis por realizar o trabalho, o que estimula a cooperação entre eles em busca desse objetivo. Dessa forma, a responsabilidade sobre o trabalho não é individual, mas sim coletiva.

O time conta com um ambiente que apoie seu trabalho, o que inclui a disponibilidade de todos os meios necessários para realizá-lo e a confiança da organização em sua capacidade em tomar as decisões que lhe cabem.

Essa autonomia ajuda os times que trabalham com Scrum a criarem um senso de propriedade sobre seu trabalho, de forma que os próprios membros de um mesmo time incentivam, cobram e ajudam seus colegas a fazerem o seu melhor para, juntos, realizarem os objetivos. Esse comportamento naturalmente leva os times a serem mais motivados e produtivos, além de estimular uma maior adaptabilidade.

Facilitação e remoção de impedimentos

Cada time conta com a ajuda de um facilitador, que trabalha para que o time torne-se cada vez mais efetivo e busca aumentar a qualidade das interações entre seus

integrantes, bem como com outras pessoas da organização.

Esse facilitador, chamado no framework de *Scrum Master*, é responsável por ensinar Scrum ao time e a quem mais na organização julgar necessário para apoiar o seu trabalho. Ele também pode facilitar as reuniões e outras interações no dia a dia do time, e trabalha para que os impedimentos que ameaçam os objetivos a serem realizados sejam removidos. O facilitador também ajuda o time a entender como trabalhar com maior autonomia e como se autogerenciar.

Melhoria contínua

Um time que utiliza Scrum está sempre em busca de melhorar para se tornar cada vez mais eficiente e eficaz. Melhorar seus conhecimentos e habilidades, melhorar seu trabalho de equipe, melhorar os seus processos, melhorar a qualidade do seu trabalho e melhorar o produto em desenvolvimento.

Na contínua busca por melhorias, o time é estimulado a experimentar, aprender e adaptar-se de acordo com o aprendizado. Assim, tão importante quanto realizar experimentos que possam melhorar o seu trabalho é descobrir cedo o que não funciona adequadamente, para então modificá-lo ou descartá-lo. É melhor falhar cedo para aprender com a falha em vez de postergar a validação de algo que pode não estar funcionando.

Buscando garantir que o produto seja melhorado continuamente, ele é desenvolvido com Scrum em ciclos curtos de feedback. E buscando garantir que o time se torne cada vez mais efetivo, uma reunião de retrospectiva é realizada obrigatoriamente ao final de cada ciclo de desenvolvimento do produto. Nessa

reunião, pontos a melhorar sobre como realizam o seu trabalho são levantados e discutidos, e planos de ação para colocar as melhorias em prática são definidos.

Ritmo sustentável de trabalho

As práticas do uso intensivo de horas extras e de aceleração forçada do ritmo de trabalho, embora muito utilizadas quando estamos próximos de um prazo limite, geram estresse nos membros de times que desenvolvem o produto, destroem a motivação e não tardam em derrubar sua produtividade e a qualidade do produto em desenvolvimento. Times que utilizam Scrum, ao contrário, buscam um ritmo constante e sustentável no trabalho de desenvolvimento do produto.

Visando a essa sustentabilidade, Scrum busca trazer uma cadência para o trabalho dos times. Os ciclos de desenvolvimento do produto têm uma duração curta, fixa e regular, e o time tem a autoridade para definir quanto trabalho será planejado para cada ciclo. Assim, ao trabalhar em ciclos com a mesma duração, um após o outro, e não exercer pressão sobre o time para que planeje mais trabalho do que acredita que pode realizar, o time naturalmente tende a atingir um ritmo constante e sustentável.

Todas as reuniões prescritas pelo Scrum possuem uma duração máxima dentro da qual devem ocorrer. São os chamados *timeboxes* que, quando respeitados, também podem auxiliar o time a criar uma cadência de trabalho e alcançar esse ritmo constante e sustentável. A reunião de planejamento de cada ciclo é um exemplo de *timebox*. Ela não deve durar mais que oito horas para quando os ciclos têm duração de um mês. As outras reuniões do Scrum também têm duração máxima definida.

Mesmo quando o time, perto do final do período estabelecido, tiver certeza de que não cumprirá com o objetivo proposto, seu *timebox* não será estendido. Da mesma forma, disfunções como horas extras e a aceleração forçada do ritmo de trabalho devem ser evitadas. Ao contrário, o escopo é flexibilizado, já que o time pode naturalmente chegar ao final do *timebox* sem ter cumprido todo o plano. A contrapartida é que o time trabalhe na ordem definida, de forma que, ao final, sobre apenas o trabalho menos importante dentre o planejado.

Trabalho de ponta a ponta

Conforme detalhei anteriormente, o time que usa Scrum é multidisciplinar e realiza o trabalho de desenvolvimento do produto de ponta a ponta. Esse time, portanto, não produz partes fragmentadas ou etapas isoladas do trabalho, tal como máquinas realizando um trabalho repetitivo sobre o qual dificilmente reconhece o valor final produzido. Ao produzir um valor identificável, esse trabalho gera nas pessoas um senso de realização, aumentando sua motivação.

Ao final de cada ciclo, eles ficam frente a frente com clientes e demais partes interessadas para mostrar esse trabalho realizado e coletar seu feedback e, assim, o valor gerado fica evidente. Ao entregar com frequência para os usuários do produto, coletar feedback e medir o uso, esse valor gerado lhes fica ainda mais evidente.

Os membros do time multidisciplinar, ao trabalharem juntos e compartilharem as responsabilidades sobre as diversas tarefas a serem realizadas, trocam conhecimentos e são incentivados a desenvolverem novas competências, o que estimula ainda mais a sua motivação.

1.8 Redução dos riscos do desenvolvimento do produto

Scrum não oferece um processo formal de gestão de riscos. Ao invés disso, a redução dos principais riscos é inerente ao próprio processo de desenvolvimento do produto com o uso do framework.

O CHAOS Report de 2018, um estudo do Standish Group, indica que, de 2013 a 2017, apenas 26% dos projetos de desenvolvimento de software que não utilizaram Scrum e similares terminaram com o orçamento e no prazo previstos e com o cliente satisfeito (THE STANDISH GROUP, 2018). Esses números indicam que, entre os principais riscos nesse tipo de trabalho, estão o atraso ou a falha na realização das entregas e a implementação de funcionalidades que não atendam às necessidades dos usuários e do negócio, que levam à insatisfação dos clientes.

O mesmo estudo aponta entre os principais elementos de sucesso com Scrum o trabalho em equipe, a capacidade de tomar decisões rápidas, a responsabilidade compartilhada e as entregas curtas, alimentadas por feedback.

Como vimos nas seções anteriores, ao tratar diretamente dessas questões, e levando em conta o foco em qualidade e nas pessoas, Scrum pode ajudar a minimizar os principais riscos no desenvolvimento de produtos.

Capítulo 2
O que é Scrum?

Conteúdo

1. O que é Scrum?
2. Scrum é ágil.
    - Os Valores Ágeis.
        - Indivíduos e interações.
        - Produto em funcionamento.
        - Colaboração com o cliente.
        - Responder a mudanças.
    - Os Princípios Ágeis.
3. Scrum é um framework.
4. Scrum se aplica a problemas complexos.
    - Sistemas Adaptativos Complexos.
    - Onde utilizar Scrum?
        - Domínio Óbvio.
        - Domínio Complicado.
        - Domínio Complexo.
        - Domínio Caótico.
        - Domínio da Desordem.
        - Scrum e Cynefin.
5. Scrum é embasado no empirismo.
6. Scrum é iterativo e incremental.
7. Os valores do Scrum.

2.1 O que é Scrum?

No capítulo anterior, tratei dos benefícios de utilizarmos Scrum, ou seja, por que ele pode nos ajudar a gerar produtos de sucesso. Mas então o que é Scrum? O guia

oficial define sucintamente (SCHWABER; SUTHERLAND, 2020):

> ***Scrum*** *é um framework leve que ajuda pessoas, times e organizações a gerar valor a partir de soluções adaptativas para problemas complexos.*

Podemos complementar essa definição adicionando:

> ***Scrum*** *é embasado no empirismo e usa uma abordagem iterativa e incremental para entregar valor desde cedo e com frequência, para otimizar a capacidade de adaptação e para reduzir os riscos no desenvolvimento de um produto. Essa abordagem permite também criar transparência sobre a eficácia das práticas de gestão, de trabalho e do ambiente de uma forma sistemática. Dessa forma, Scrum possibilita a melhoria contínua tanto do produto quanto do processo para desenvolvê-lo.*

Vamos entender em seguida o que esses termos significam.

2.2 Scrum é ágil

O Michaelis Dicionário Brasileiro da Língua Portuguesa define "ágil" como: *que se movimenta com rapidez (...)* (ÁGIL, 2015).

Mas o nosso "ágil" não significa "fazer rápido" ou "entregar rápido" - ele tem muito mais a ver com ter e exercitar a capacidade de se adaptar. O nome "Ágil" (ou "Agilidade") foi escolhido para representar um

movimento que surgiu em meados dos anos 90 em resposta aos pesados métodos de gerenciamento de desenvolvimento de software que predominavam na época, que por simplicidade aqui chamo de "métodos tradicionais".

O mais conhecido representante dos métodos tradicionais para o desenvolvimento de software é o modelo em cascata, ou *waterfall*. Ele foi inicialmente descrito por Winston W. Royce em seu artigo de 1970 e se caracteriza por uma sequência de fases do trabalho, em que cada fase somente se inicia quando a anterior termina e a saída de uma fase é a entrada da fase seguinte, como mostrado na figura a seguir. Royce, no entanto, criticava o modelo, afirmando que, para o desenvolvimento de software, seu uso era arriscado (ROYCE, 1970).

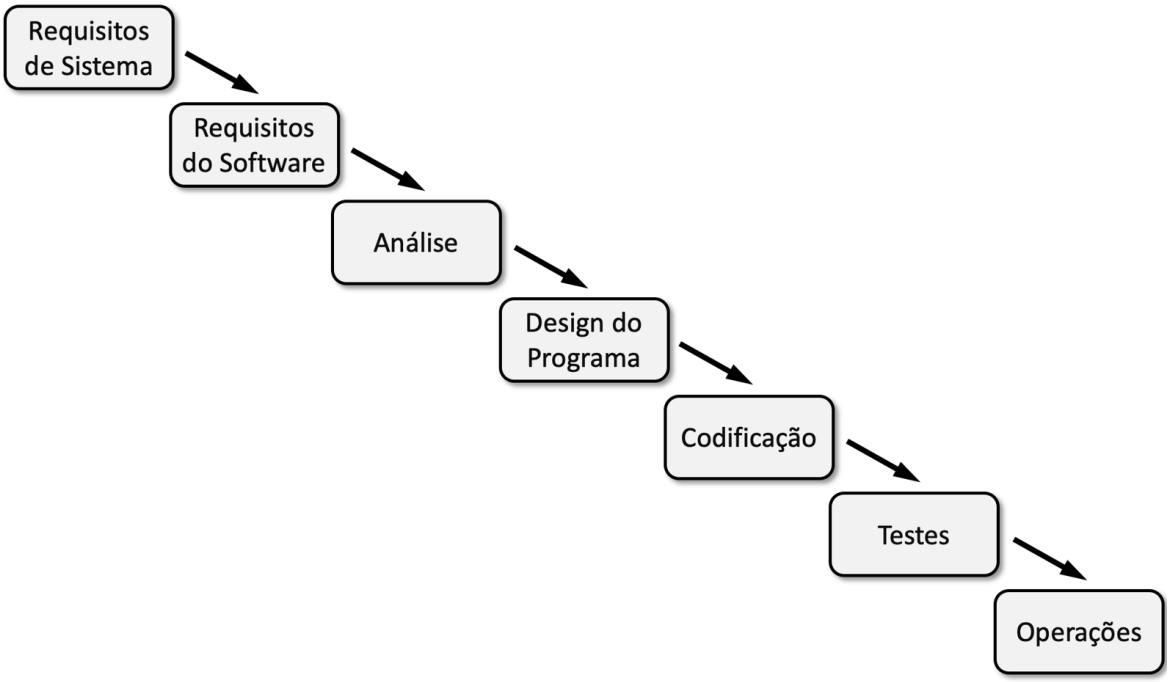


Figura 2.1: O modelo em cascata, ou waterfall, conforme definido por Royce (1970)

Os métodos tradicionais são fortemente prescritivos e, de forma geral, se caracterizam pelo BDUF, ou seja, o foco em planos detalhados e definidos no princípio do projeto (com custo, escopo e cronograma), pelo microgerenciamento do trabalho com o poder centralizado, pelos processos complicados e pela extensa documentação. Mudanças são fortemente indesejadas e o sucesso está em conseguirmos seguir o plano com a maior fidelidade possível.

Acreditava-se que, pelo uso desses métodos, seria possível tratar o desenvolvimento de software como um processo previsível. Na realidade, quanto mais falhavam, mais pesados e complexos esses métodos se tornavam.

Definição: BDUF, o grande inimigo

Big Design Up Front, ou BDUF, é um termo usado para descrever a prática de especificarmos detalhadamente um trabalho a ser realizado (originalmente, de desenvolvimento de software), antes de começar esse trabalho (BIG DESIGN UP FRONT, 2013). Associada a essa prática está a forte crença, equivocada em minha opinião, de que o BDUF é o modo mais correto e seguro de desenvolvermos um produto.

Considero o BDUF a prática mais nociva do modelo cascata (ou *waterfall*) quando tratamos do desenvolvimento de software. O Scrum foi criado justamente para combater o BDUF. No Scrum, não há espaço para "bedufar", nem para "bedufeiros".

Enquanto o paradigma do BDUF é planejar ao máximo de forma antecipada, o paradigma do Scrum é desenvolver um produto altamente adaptável, ou seja, capaz de responder às mudanças e à compreensão emergente do que deve ser construído.

Scrum é ágil porque, assim como outros métodos, metodologias e frameworks, sua utilização deve seguir os princípios e valores do **Manifesto para o Desenvolvimento Ágil de Software** (BECK et al., 2001). Esse manifesto foi criado em fevereiro de 2001 em uma reunião que aconteceu na estação de esqui de Snowbird no estado de Utah, Estados Unidos. O Manifesto Ágil, como ficou conhecido, foi assinado por dezessete líderes representantes de ideias, metodologias e processos que, em contraste com as práticas predominantes na época, estavam trazendo valor para seus clientes e usuários por meio de abordagens leves e empíricas para o desenvolvimento de software.

Nesse encontro histórico, não havia entre os participantes a intenção de unificar suas formas de trabalhar. A expectativa de chegarem a qualquer tipo de consenso era limitada e variada. No entanto, apesar das diferentes práticas defendidas, os participantes encontraram um conjunto de valores em comum e, ao final de três dias, estabeleceram o termo "Ágil" para representar o novo movimento. Neste livro, "Ágil" é usado como substantivo, e "ágil" é usado como adjetivo, ambos se referindo a esse movimento.

O Manifesto Ágil pode ser encontrado em *http://agilemanifesto.org* em diversas línguas, inclusive em português do Brasil (eu mesmo participei desse esforço de tradução, embora a versão apresentada aqui esteja um pouquinho diferente). Ao procurar na internet

pelos nomes dos signatários originais do Manifesto e por suas ideias, podemos ver o quanto representativa a grande maioria deles ainda é.

Para efeitos didáticos, criei uma definição extraoficial para "ágil", que tenho usado em aula:

> **Ágil**
>
> *Ser ágil significa constantemente adaptar-se, com o objetivo de entregar a seus clientes, em cada momento, o maior valor possível, a partir do trabalho de indivíduos e de suas interações - definição minha.*

Scrum é a escolha de 70% das organizações que utilizam métodos, metodologias ou frameworks que seguem o Manifesto Ágil, de acordo com uma pesquisa realizada entre 2017 e 2018 (VERSIONONE, 2018).

Curiosidade: Mike Beedle

Tive o prazer de conversar e interagir com um dos signatários do Manifesto, o saudoso Mike Beedle, durante o Scrum Gathering do Rio de Janeiro de 2014 (http://scrumrio.com). Eu, ele e meus sócios na K21 tomamos algumas cervejas quando Mike nos mostrou em seu celular uma lista dos nomes que foram cogitados para o movimento naquela reunião em 2001. Foi ele quem sugeriu o termo "Ágil". Naquela lista, entre outras palavras, eu pude ler: "adaptativo".

Mike Beedle foi coautor do primeiro livro de Scrum, publicado em 2002 (SCHWABER; BEEDLE, 2002) e era um Certified Scrum Trainer da Scrum Alliance (http://scrumalliance.org), assim como eu sou.

Mike Beedle faleceu em 23 de março de 2018, em Chicago, após um aparente assalto. Ele deixou seis filhos.

Os Valores Ágeis

O Manifesto Ágil reconhece que a utilização de processos, ferramentas, documentação, contratos e planos pode ser importante para o sucesso no desenvolvimento do produto, mas são ainda mais importantes os chamados Valores Ágeis: os indivíduos e interações entre eles, produto (software, originalmente) em funcionamento, colaboração com o cliente e responder a mudanças.

> **Manifesto para Desenvolvimento Ágil de Software**
>
> Estamos descobrindo maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Por meio deste trabalho, passamos a valorizar:
>
> - **Indivíduos e interações** mais do que processos e ferramentas.
> - **Software em funcionamento** mais do que documentação abrangente.
> - **Colaboração com o cliente** mais do que negociação de contratos.
> - **Responder a mudanças** mais do que seguir um plano.
>
> Ou seja, mesmo havendo valor aos itens à direita, valorizamos mais os itens à esquerda.

Não há como negar que o Scrum, assim como o Ágil, tem as suas origens no desenvolvimento de software. Hoje,

embora sua aplicação se estenda a diferentes tipos de produtos e serviços, não ouso modificar o Manifesto original. Discorro a seguir sobre o significado de cada um desses valores a partir da visão de dois signatários originais do Manifesto, Alistair Cockburn e Jim Highsmith, e de um autor bastante relevante, Craig Larman, criador do framework LeSS baseado em Scrum (HIGHSMITH, 2002, 2004; LARMAN, 2003; COCKBURN, 2007). Mas tomo a liberdade de adaptar essa explicação para um contexto mais genérico de desenvolvimento de produtos.

Indivíduos e interações

Valorizar indivíduos e interações mais do que processos e ferramentas leva em conta que, em última instância, o trabalho de desenvolvimento do produto é uma atividade humana e, assim, depende de questões humanas para seu sucesso. Ou seja, quem desenvolve os produtos são as pessoas, que possuem características únicas individualmente e em equipe, como talentos, habilidades e conhecimentos. Além dessas questões, são fundamentais para o sucesso as personalidades e peculiaridades dos indivíduos, os seus interesses, os seus momentos de vida, a sua vida social e familiar, e como esses elementos afetam e são afetados pelo trabalho.

Levadas em conta as questões individuais, podemos afirmar que o desenvolvimento do produto é baseado no trabalho em equipe. A qualidade da interação entre os membros do time que desenvolve o produto é crítica para a solução de problemas. Nessas interações, o uso contínuo da comunicação e do feedback é uma diretiva essencial para a prática ágil, especialmente a partir da conversação face a face.

As pessoas envolvidas no desenvolvimento do produto não podem ser trocadas como peças, pois elas são mais

determinantes para o sucesso do que seus papéis em diagramas de processos. Embora possam ser importantes para guiar e apoiar o desenvolvimento do produto, processos e ferramentas não são substitutos para as pessoas e seu trabalho em equipe, e utilizá-los por si só não desenvolverá bons produtos. É com os indivíduos e seu trabalho em conjunto que podemos contar para tomar as decisões críticas para o desenvolvimento do produto.

Buscamos processos e ferramentas que ajudem a equipe em vez de ditar como seu trabalho deve ser feito, de forma que são os processos e ferramentas que se adaptam ao time, e não o oposto. Por princípio, bons processos e ferramentas devem ser os mais simples possíveis que funcionem para apoiar o trabalho de geração de valor.

Produto em funcionamento

O produto em funcionamento é o único indicador do que a equipe de fato construiu. Os clientes se interessam por resultados, ou seja, por algo que de fato entregue valor de negócio.

O produto em funcionamento não exclui a necessidade de documentação. Ela pode ser muito útil para o desenvolvimento do produto, pois tem o potencial de facilitar a comunicação e colaboração, melhorar a transferência de conhecimento, preservar informações históricas e satisfazer a necessidades legais e regulatórias. Mas a documentação é auxiliar ao desenvolvimento do produto e simplesmente é menos importante do que o próprio produto em funcionamento. Assim, devemos produzir somente a documentação necessária e suficiente para a realização do trabalho.

Um erro comum no trabalho de desenvolvimento do produto é a crença de que a documentação substitui a interação, servindo como um meio de comunicação. A documentação, na realidade, pode apenas facilitar essa interação, funcionando como seu subproduto na forma de documentos, rascunhos, desenhos, anotações etc., que podem (ou não) ser utilizados como um registro permanente. Outro equívoco comum é acreditar que a documentação pode, de forma confiável, indicar o status ou progresso e possibilitar feedback sobre o processo do trabalho e sobre o produto que foi desenvolvido até então. Na realidade, a entrega iterativa de partes do produto em funcionamento resolve essas questões de formas que simplesmente com a documentação é impossível.

Colaboração com o cliente

No desenvolvimento de produtos, em que há alta volatilidade, ambiguidade e incertezas, não pode haver "nós" e "eles" na relação cliente-time. Há simplesmente "nós", o que significa que clientes e o time que realiza o desenvolvimento do produto estão do mesmo lado, colaborando para desenvolver um produto que traga valor para esses clientes.

A colaboração entre negócios e time é crítica e envolve confiança, companheirismo, tomada de decisão conjunta, interações frequentes e rapidez na comunicação, em oposição a disputas antagônicas de contratos em que um lado busca se proteger do outro.

Responder a mudanças

Todo trabalho de desenvolvimento de produto deve balancear o planejar com o se adaptar, dependendo do nível de incerteza inerente a ele. Quando há alta

incerteza, a inspeção e a rápida adaptação predominam sobre o planejamento e a execução estrita de tarefas planejadas.

A incerteza é inerente e inevitável em processos de desenvolvimento de um produto, o que exige uma grande adaptabilidade. Mesmo assim, construir um plano nesse contexto pode ser útil, mas manter-se preso a um plano ultrapassado não funciona em favor do sucesso. Assim, somente será útil seguir o plano enquanto ele corresponder suficientemente bem à situação atual. Caso não mais corresponda, ele perde a sua validade e deve ser modificado. No desenvolvimento de um produto, praticamente tudo pode mudar em pouquíssimo tempo, como escopo, funcionalidades, tecnologia, arquitetura etc. e, assim, qualquer plano construído deve ser frequentemente adaptado para responder a essa nova realidade.

Iterações curtas de desenvolvimento do produto permitem que mudanças sejam mais rapidamente realizadas no produto.

Os Princípios Ágeis

Os doze Princípios Ágeis originais podem ser vistos a seguir, somados à minha interpretação de cada um, em que generalizo para o desenvolvimento de produtos, e não apenas software. Eles dão base ao Manifesto e foram criados por seus autores, em um momento posterior à reunião em Snowbird.

1. Nossa maior prioridade é satisfazer o cliente por meio da entrega, desde cedo e contínua, de software com valor.

O sucesso no desenvolvimento do produto está na satisfação dos clientes e usuários. Geramos, desde cedo e frequentemente, valor para os clientes e usuários a partir de entregas de incrementos do produto que atendam às suas necessidades. Esse princípio se opõe à crença tradicional de que o sucesso está em seguir um plano detalhado, entregando o que foi prometido, no prazo e dentro do orçamento previsto.

2. Mudanças de requisitos são bem-vindas, mesmo em fases tardias do desenvolvimento. Os processos ágeis utilizam a mudança em favor da vantagem competitiva para o cliente.

Aceitar a mudança como natural no processo de desenvolvimento do produto é o que nos possibilita atender às necessidades dos clientes e usuários e, assim, garantir que o que for entregue realmente traga valor. À medida que, em ciclos curtos, o produto é desenvolvido, entregue e utilizado, o feedback de clientes e usuários alimenta o seu próprio desenvolvimento. Os feedbacks que levarão a mudanças no produto ao longo do seu desenvolvimento são acolhidos como oportunidades de aumentarmos o valor entregue, e não como acontecimentos indesejáveis. Além disso, permitimos que o produto assimile as mudanças de mercado, possibilitando uma resposta rápida a elas e, com isso, o tornamos mais competitivo. Este princípio se opõe a tratarmos a mudança como uma inimiga indesejada e custosa que, na definição tradicional de gestão de projetos, deveria ser prevenida e evitada.

3. Entregar software em funcionamento com frequência, desde a cada poucas semanas até a cada poucos meses, com uma preferência por prazos mais curtos.

Entregar, com frequência, incrementos prontos e funcionais do produto geram, em cada entrega, valor para os clientes e usuários que os recebem e nos permite obter feedback sobre o que foi produzido. Assim, podemos adaptar, incrementalmente, o produto às necessidades desses clientes e usuários, reduzindo os riscos no desenvolvimento do produto. Esse princípio se opõe à prática tradicional de realizarmos poucas ou, no limite, uma entrega de valor única, apenas ao final do trabalho.

4. As pessoas do negócio e os desenvolvedores devem trabalhar em conjunto diariamente ao longo do projeto.

Pessoas do negócio e pessoas que desenvolvem o produto possuem o objetivo comum de garantir a geração de valor para os clientes e usuários. Para realizar esse objetivo, cooperam continuamente durante todo o desenvolvimento do produto, interagindo com frequência. Idealmente, trabalham lado a lado como membros de um mesmo time, ou esses papéis podem até mesmo ser exercidos pelas mesmas pessoas. Esse princípio se opõe aos cenários de antagonismo comuns em desenvolvimento de produtos, nos quais as pessoas do negócio — que frequentemente incluem os próprios clientes — e aquelas que desenvolvem o produto raramente colaboram entre si ou se comunicam diretamente. Em geral, utilizam documentos com esse fim.

5. Construa projetos em torno de indivíduos motivados. Dê-lhes o ambiente e o suporte que precisam e confie neles para realizarem o trabalho.

O produto é construído por pessoas que devem estar motivadas para fazê-lo bem. O ambiente, o suporte e a confiança necessários para realizar seu trabalho são fatores fundamentais para sua motivação. Esse princípio se opõe à crença de que o produto se constrói em torno das melhores ferramentas e processos, de que as pessoas devem ser comandadas e controladas na realização do seu trabalho, e de que as pessoas se motivam, por exemplo, a partir de metas e incentivos financeiros.

6. O método mais eficiente e efetivo de se transmitir informação para e entre uma equipe de desenvolvimento é a conversa face a face.

A melhor forma de comunicação entre membros do time que desenvolve o produto e entre esse time e o mundo externo é a comunicação face a face, que é direta, síncrona, o que permite feedback e ajustes instantâneos, e é enriquecida pela entonação de voz, olhar e linguagem corporal, entre outros fatores. Quando a comunicação presencial não é viável (em trabalho *on-line*, por exemplo), é uma boa prática fazermos o melhor uso possível da tecnologia disponível para nos aproximarmos da comunicação face a face. Esse princípio se opõe à utilização de documentos, e-mails, telefone e teleconferência, entre outros, como formas padrão de comunicação no trabalho de desenvolvimento do produto.

7. Software em funcionamento é a principal medida de progresso.

O progresso do desenvolvimento do produto ocorre à medida que partes do produto que representem valor são entregues aos clientes e usuários. Esse princípio se opõe à prática de medirmos e informarmos que houve

progresso no desenvolvimento do produto a partir do cumprimento de etapas de um plano ou da geração de artefatos que não signifiquem valor, como protótipos ou extensos documentos de planos e especificações. Esse falso progresso é, em geral, mostrado a partir de gráficos de Gantt, barras de progresso etc. indicando percentuais de completude.

8. Os processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter indefinidamente um ritmo constante.

O time que desenvolve o produto trabalha em um ritmo constante e sustentável, o que se torna possível quando esse ritmo se propaga por toda a cadeia, incluindo as diferentes partes interessadas. Quando, por exemplo, é exigido dos times um compromisso com mais trabalho do que são capazes de realizar, é comum o uso de horas extras em excesso e a decorrente sobrecarga para cumprirem seus prazos de entrega. Essas práticas podem levar a um desbalanço no fluxo de trabalho, o que acaba por gerar insatisfações no time, uma menor produtividade e uma menor qualidade no produto desenvolvido.

9. A atenção contínua à excelência técnica e a um bom desenho aumentam a agilidade.

Desenvolver o produto com alta qualidade, utilizando-se de excelência técnica e de um desenho flexível, permite que ele possa ser mais facilmente modificado e, assim, é essencial para manter a sua adaptabilidade. Esse princípio se opõe à crença de que, para obtermos agilidade no desenvolvimento do produto, a qualidade deve ser sacrificada. Na realidade, ocorre exatamente o oposto.

10. Simplicidade — a arte de se maximizar a quantidade de trabalho não feito — é essencial.

Evitamos o desperdício no desenvolvimento do produto ao não realizarmos trabalho que não é necessário. Exemplos comuns de desperdícios incluem a implementação de funcionalidades de que os clientes e usuários não precisam, a implementação de soluções desnecessariamente complexas ou prevendo casos futuros que não necessariamente se concretizarão, o planejamento com nível de detalhes maior do que podemos ter em um determinado momento e uso ou geração de artefatos desnecessários.

11. As melhores arquiteturas, requisitos e desenhos emergem de equipes que se auto-organizam.

Equipes com maior autonomia são mais eficazes. Essas equipes auto-organizadas trabalham em direção a objetivos acordados, mas têm autonomia para decidirem como realizarão esses objetivos. Assim, se tornam responsáveis e são responsabilizadas por seus resultados. Do seu trabalho, emergem, ao longo de todo o desenvolvimento do produto, a arquitetura, os requisitos e os detalhes do produto. Esse princípio, além de tratar da auto-organização, se opõe à prática comum de definirmos toda a arquitetura e requisitos do produto antes do início do seu desenvolvimento - o BDUF de que falamos anteriormente. Estes devem ser definidos progressivamente, ao longo do desenvolvimento do produto, de forma que possam emergir e evoluir de acordo com a realidade que se apresenta.

12. Em intervalos de tempo regulares, a equipe reflete sobre como se tornar mais efetiva e, então, refina e ajusta seu comportamento de acordo.

Para se tornar cada vez mais efetivo, o time regularmente inspeciona suas formas de trabalho, identifica pontos de melhoria e se adapta de acordo, promovendo a melhoria incremental contínua. É a inspeção e adaptação que o time realiza em seus processos de trabalho. A reunião de Sprint Retrospective busca implementar esse princípio no trabalho com Scrum.

2.3 Scrum é um framework

O Scrum é um framework simples e leve e não uma metodologia ou um conjunto de processos.

Um framework (ou arcabouço) é uma estrutura em torno de ou sobre a qual algo é construído (COMBLEY, 2011). Enquanto framework, o Scrum serve de base e guia para a realização do trabalho, mas não define práticas específicas e detalhadas a serem seguidas para isso.

Ou seja, Scrum não é constituído por um conjunto de receitas ou fórmulas. Scrum não prescreve, por exemplo, como o time deve desenvolver o produto, como deve facilitar seu trabalho ou remover impedimentos, nem como devem ser levantadas as necessidades de negócios ou como devem lidar com os clientes.

Scrum entende que as práticas necessárias para o sucesso no desenvolvimento do produto são muito específicas para cada contexto, não podendo ser prescritas. Não existe aquela solução ou conjunto de soluções que funcionam para todas as situações. Ao contrário, as pessoas desenvolvendo o produto, a partir das responsabilidades, eventos, artefatos e regras do Scrum, avaliam, adquirem e desenvolvem um conjunto

de práticas que melhor lhes servirão, o que é constantemente reavaliado e melhorado. Esse é um trabalho contínuo de descoberta, inspeção e adaptação.

Por ser um framework, Scrum pode funcionar bem quando combinado com ou complementado por diferentes métodos e práticas consagrados pelo mercado, que são experimentados e adaptados pelo time para seu contexto específico.

Extreme Programming (XP), por exemplo, é uma metodologia. Embora seja considerado leve, XP prescreve um conjunto mais completo e específico de práticas que devem ser seguidas para obtermos sucesso no desenvolvimento de software. Ao usar Scrum, muitos times de desenvolvimento de software pegam emprestadas diversas práticas do XP, como a programação em par, a integração contínua, o desenvolvimento guiado por testes etc.

Definição: analogia do esqueleto

Podemos imaginar que um framework se opõe a uma metodologia assim como o esqueleto está para o corpo humano. O esqueleto — ou seja, os ossos — fornece uma estrutura, mas não se movimenta sozinho. É necessário que sejam agregados a ele músculos, tendões e outros tecidos para poder se movimentar. O corpo humano, por outro lado, já possui tudo o que necessita para deslocar-se de um ponto a outro.

O framework fornece uma estrutura de trabalho, mas não é útil se aplicado sozinho. Ele precisa ser complementado por outras técnicas e práticas para que seja possível a realização do trabalho por quem o está usando. Uma

metodologia já prescreve grande parte do que é necessário para que sua aplicação seja bem-sucedida.

Scrum é um framework e não uma metodologia porque, como afirmei anteriormente, acreditamos que as técnicas e práticas necessárias para a realização do trabalho dependem do contexto em que esse trabalho é realizado e não podem ser prescritas, embora existam boas práticas reconhecidas.

Scrum é simples e leve. Com ele, buscamos não gerar nem aplicar nada que não será efetivamente útil e utilizado. Prescrevendo apenas três responsabilidades (Product Owner, Desenvolvedor e Scrum Master), cinco eventos (Sprint e reunião de Sprint Planning, de Daily Scrum, de Sprint Review e de Sprint Retrospective) e três artefatos (Product Backlog, Sprint Backlog e o Incremento), com seus respectivos compromissos (Objetivo do Produto, Objetivo do Sprint e Definição de Pronto), Scrum pode ser facilmente explicado e compreendido (neste livro, adicionei alguns elementos opcionais).

Mas, ao contrário do que pode parecer, Scrum não é fácil de ser usado. Ao longo deste livro, apresentarei algumas alternativas possíveis de técnicas já consagradas, utilizadas com sucesso por inúmeros times, e que talvez possam ajudar o seu time no uso do framework.

Curiosidade: ScrumBut

Ouvi falar pela primeira vez sobre ScrumBut em 2009, no Scrum Gathering de São Paulo. Naquela ocasião, Ken Schwaber, um dos criadores do Scrum, trouxe o assunto em uma palestra em vídeo. Um ScrumBut é um uso parcial e disfuncional do Scrum que, por essa razão,

impede os times de obterem os benefícios esperados na adoção do framework. Assim, de acordo com Schwaber, para dizer que usamos Scrum, devemos adotar todos os seus elementos.

A ideia por trás da expressão ScrumBut ("ScrumMas", em português) se origina em afirmações do tipo "nós usamos Scrum, mas...", como, por exemplo:

- "nós usamos Scrum, mas nossos ciclos duram meses";
- "nós usamos Scrum, mas somente entregamos no final do projeto";
- "nós usamos Scrum, mas é impossível parar para fazer a reunião diária todos os dias".

Foi comum a confusão entre "ScrumBut" e "ScrumButt" que, em inglês, traz uma conotação mais grosseira (e hilária, na minha opinião). A verdade é que a expressão ScrumBut praticamente caiu em desuso, principalmente porque muitos consideraram o termo pejorativo ao se referirem a adoções parciais do Scrum. Acredito que essa conotação equivocada possa ter algo a ver com isso.

2.4 Scrum se aplica a problemas complexos

Sistemas Adaptativos Complexos

Desde meados dos anos 90, conceitos de Sistemas Adaptativos Complexos (Complex Adaptive Systems, ou CAS, em inglês) começaram a ser utilizados para tentar explicar como as organizações funcionam, em um mundo cada vez mais complexo (HIGHSMITH, 2002). Diversos líderes do movimento ágil basearam suas ideias sobre

gestão em teorias que estudam esse tipo de sistema. Scrum foi criado nesse contexto, visando ao desenvolvimento de produtos complexos imersos em ambientes complexos.

Podemos ver a seguir duas das definições mais comuns para Sistemas Adaptativos Complexos:

Sistemas Adaptativos Complexos

...são sistemas que envolvem um grande número de componentes, frequentemente chamados de agentes, que se adaptam ou aprendem à medida que interagem (HOLLAND, 2006).

...consistem em uma série de componentes ou agentes, que interagem uns com os outros de acordo com conjuntos de regras que requerem que eles examinem e respondam aos comportamentos uns dos outros a fim de melhorar seu comportamento e, portanto, o comportamento do sistema de que fazem parte (STACEY, 1996).

O comportamento de Sistemas Adaptativos Complexos é emergente, e assim não pode ser previsto a partir de uma análise de causa e efeito ("se fizer isso acontece aquilo"), e nem a partir da inspeção individual de seus agentes e de suas propriedades. Ou seja, dada uma ação, não sabemos de antemão quais serão seus resultados, e nem mesmo entender e analisar os detalhes de cada um de seus agentes nos permitirá inferir o que acontecerá com o sistema como um todo.

No entanto, esses agentes do sistema interagem entre si e, a partir dessas interações, emergem padrões que aumentam a previsibilidade do comportamento do

sistema no curto prazo. São propriedades do sistema que surgem a partir das interações de suas partes. Na ausência de um controle centralizado, os agentes de Sistemas Adaptativos Complexos são auto-organizados: em frequentes ciclos de feedback, o próprio comportamento emergente do sistema o realimenta e provoca a adaptação de seus agentes e do sistema como um todo. Dessa forma, esses sistemas operam em um estado de instabilidade ordenada, ou seja, uma ordem que é emergente, imprevisível e em constante mudança. Estímulos externos também provocam essa readaptação, que causa a evolução do sistema e a emergência de novos padrões de comportamento (JAIN; MESO, 2004).

Como veremos em detalhes neste livro, os ciclos frequentes de feedback, que guiam a construção do produto e possibilitam a melhoria contínua dos processos de trabalho (inspeção e adaptação), as equipes de trabalho autogerenciadas e a alta interação necessária entre as partes envolvidas no desenvolvimento do produto são, entre outros, elementos do Scrum que podem ter seus paralelos facilmente identificados nas teorias de Sistemas Adaptativos Complexos.

Onde utilizar Scrum?

Cynefin é um modelo que se propõe a ajudar na tomada de decisões ao entendermos a relação entre causa e efeito que ocorre em um determinado sistema ou contexto conforme interagem seus agentes. Criado em 1999 por Dave Snowden, o modelo oferece cinco domínios distintos para que possamos escolher o mais adequado ao sistema ou contexto em questão e, assim, tomar decisões apropriadas (SNOWDEN; BOONE, 2007).

Pertencem aos domínios ordenados aqueles sistemas em que as suas restrições são fortes o suficiente para que o

comportamento de seus agentes seja previsível, de forma que possamos estabelecer a relação entre causa e efeito. Os domínios ordenados são divididos em Óbvio e Complicado. Se as restrições são evidentes e podemos determinar diretamente a melhor decisão, então o sistema pertence ao domínio Óbvio. Se as restrições permitem uma escolha entre opções e a melhor decisão somente pode ser determinada após uma análise, então o sistema pertence ao domínio Complicado.

Os domínios não ordenados, em que não há relação imediata aparente entre causa e efeito, são o Complexo e o Caótico. Nos sistemas do domínio Complexo, as restrições coevoluem com os agentes do sistema, de forma a habilitar essa evolução. No domínio Caótico, não há restrições.

No domínio de Desordem, é difícil de reconhecer com o que estamos lidando.

Ao apresentar o modelo Cynefin, meu objetivo é que você perceba que Scrum é adequado para os contextos de desenvolvimento de produtos. Vou analisar, em seguida, o significado de cada domínio definido no modelo e, em seguida, relacioná-los a esse tipo de trabalho.

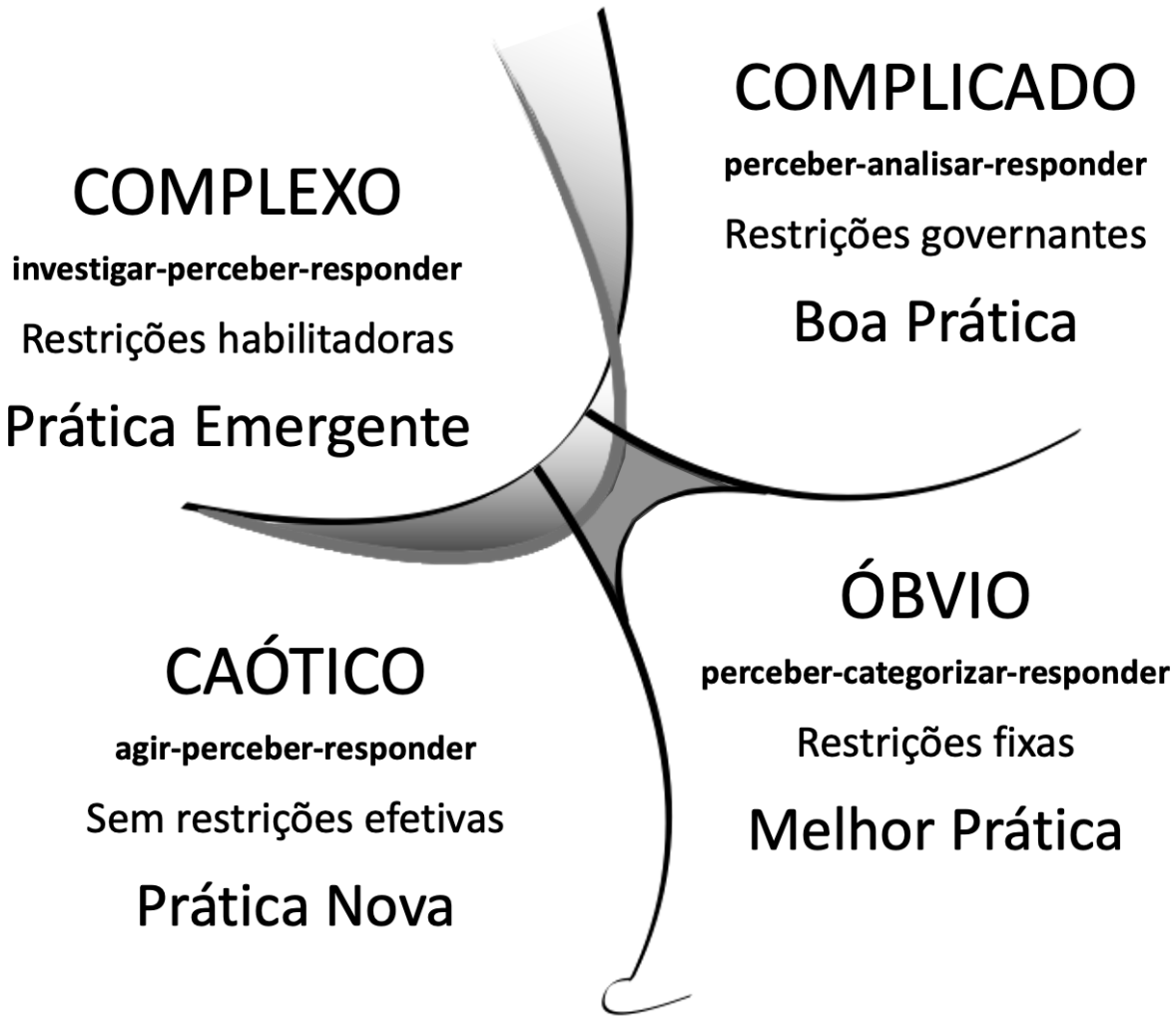


Figura 2.2: O modelo Cynefin

Domínio Óbvio

Esse é o domínio das melhores práticas, onde há estabilidade e as relações entre causa e efeito são evidentes para todos, previsíveis e repetíveis. Assim, para uma dada situação, devemos avaliar seus fatos (perceber) para definir de que tipo de situação se trata (categorizar) e, então, aplicar aquela que é a melhor prática conhecida para ela (responder). Esse domínio era chamado de "simples" no modelo Cynefin original, posteriormente modificado pelo seu autor.

Para um tipo de trabalho que predominantemente opera nesse domínio, podemos usar conhecimento do passado para escolher a melhor abordagem e, assim, utilizá-la para executar o trabalho.

A troca de uma lâmpada, por exemplo, pertence a esse domínio, uma vez que existe uma sequência conhecida de ações a cumprir para realizar esse trabalho. O trabalho de realizarmos a digitação dos dados dos produtos a serem vendidos em nossa loja virtual também caracteriza um contexto do domínio Óbvio.

Domínio Complicado

Esse é o domínio das boas práticas, onde existe a relação entre causa e efeito, mas nem todos podem vê-la. Diferentemente dos contextos do domínio Óbvio, pode haver várias respostas viáveis para um mesmo problema e especialistas podem ser necessários para investigar e escolher a que melhor se aplica. Logo, para uma dada situação, devemos avaliar seus fatos (perceber) para definir qual das possíveis soluções é a mais adequada (analisar) e, então, aplicá-la (responder).

Para um tipo de trabalho que predominantemente opera nesse domínio, podemos realizar análises suficientes dos problemas e escolher, entre as possíveis soluções, aquelas que julguemos mais adequadas.

O trabalho de especialista de um mecânico ao consertar o seu carro, por exemplo, pertence ao domínio Complicado. O mecânico vai avaliar a situação do carro para escolher entre as possíveis soluções e, então, resolver o problema.

Domínio Complexo

Esse é o domínio da emergência, onde a relação entre causa e efeito somente se torna evidente em retrospecto e não é reprodutível. Em outras palavras, para qualquer ação de seus agentes, não é possível prevermos o que vai acontecer como consequência. Apenas poderemos saber após o ocorrido. Assim, para uma dada situação, devemos investigar por meio de experimentos (investigar) para entender seus impactos (perceber) e, então, promover o que funciona e abandonar o que não funciona (responder). Em outras palavras, realizamos uma série de experimentos em paralelo para então escolher aquele que trouxer o melhor resultado.

Um teste multivariado opera no domínio Complexo. Em um exemplo desse tipo de teste, os usuários de uma mesma página de internet são divididos em grupos. Cada grupo tem acesso a uma versão dessa mesma página, cada uma com elementos diferentes. Aquela versão que gerar melhores resultados (mais conversões, por exemplo) prevalecerá. É também no domínio Complexo que operam os Sistemas Adaptativos Complexos, descritos anteriormente.

Domínio Caótico

Esse é o domínio das práticas novas, onde as relações entre causa e efeito são impossíveis de serem determinadas, pois mudam constantemente. Não há restrições (ou padrões), apenas turbulência, de modo que qualquer prática utilizada será completamente nova. Quando possível, devemos agir rapidamente de modo a restabelecer a ordem (agir), para entendermos onde existe a estabilidade (perceber) e, então, trabalhar para trazer a situação do caos para a complexidade (responder), que é um contexto mais gerenciável.

As ações necessárias em uma situação de crise, como o resgate de reféns, por exemplo, caracterizam um trabalho no domínio Caótico. Projetos com foco em pesquisa inovadora, no qual o que descobrimos em um momento pode mudar todo o trabalho a ser realizado em seguida, são exemplos de trabalhos que operam no domínio caótico.

Domínio da Desordem

Nesse domínio, múltiplas perspectivas se apresentam e não é possível determinar qual dos outros quatro domínios é o predominante. Assim, nada sabemos sobre a relação entre causa e efeito, e não podemos determinar qual a melhor forma de trabalharmos. O domínio da Desordem é representado pela parte central da figura do modelo Cynefin.

Na Desordem, as pessoas competem de forma destrutiva por soluções que estejam em sua zona de conforto, ou seja, em algum dos outros quatro domínios. A saída para essa situação é quebrar o problema em partes e definir o domínio de cada uma delas — Óbvio, Complicado, Complexo ou Caótico — para então agir de acordo.

Scrum e o Cynefin

Diferente do Scrum, a gestão tradicional de *comando-e-controle* trata o trabalho como pertencente aos domínios ordenados, ou seja, Óbvio ou Complicado. Nesses domínios, podemos estabelecer a relação entre causa e efeito, e assim as boas práticas conhecidas (ou a melhor prática) devem ser aplicadas para cada situação.

Como já destaquei na seção *Scrum é ágil*, neste capítulo, esses métodos tradicionais não são eficientes em contextos de mudança e imprevisibilidade e falham justamente por tratar o trabalho de desenvolvimento de

produtos como um processo previsível. Esse trabalho, na realidade, geralmente se posiciona em domínios não ordenados.

Nos sistemas que operam no domínio Complexo, lançamos mão de múltiplos experimentos em paralelo, de forma que emerjam a inovação, a criatividade e novos padrões. A tolerância a falhas é um aspecto essencial dessa aprendizagem empírica. Scrum, no entanto, não opera exatamente nesse domínio. Apesar de estarem presentes vários aspectos do domínio Complexo, o trabalho iterativo no Scrum produz incrementos entregáveis do produto em sequência, ao invés de produzir múltiplos candidatos a entrega em paralelo, que competiriam pela possibilidade de entrega.

Scrum também não funciona para trabalhos que operam no domínio Caótico. Para que o framework seja utilizado, é necessário estabelecermos horizontes de curto prazo nos quais os objetivos a serem realizados sejam bem definidos e estáveis, alinhados a um objetivo de mais longo prazo. No domínio Caótico, no entanto, os objetivos e a direção podem mudar a qualquer momento.

Realizei um treinamento com Dave Snowden no primeiro semestre de 2019 e tive a oportunidade de conversar com ele sobre o assunto. Segundo Snowden, Scrum opera na região liminar entre o domínio Complexo e o domínio Complicado (você pode ver, na figura do modelo Cynefin, a região sombreada entre os domínios). Scrum de fato trata a implementação de incrementos do produto em cada ciclo como um experimento, já que esse produto será modificado em ciclos seguintes a partir do feedback recebido. No entanto, Scrum se desloca do domínio Complexo em direção ao Complicado ao se focar em apenas um experimento por vez.

2.5 Scrum é embasado no empirismo

É típico se adotar a abordagem de modelagem definida (teórica) quando os mecanismos básicos pelos quais um processo opera são razoavelmente bem compreendidos. Quando o processo é complicado demais para a abordagem definida, a abordagem empírica é a escolha apropriada (OGUNNAIKE; RAY, 1994).

Scrum entende que, por ser complexo e possuir um alto grau de incerteza, o desenvolvimento de produtos deve ser realizado de forma empírica. Por essa razão, Scrum está embasado na teoria de controle de processos empíricos (SCHWABER; SUTHERLAND, 2017), conforme definida por Ogunaike e Ray (1994).

Na abordagem empírica do Scrum, trabalhamos em ciclos sucessivos de feedback, aprendendo tanto sobre os modos de produção utilizados quanto sobre o produto que está sendo desenvolvido, à medida que experimentamos, verificamos o que é válido e o que não é e adaptamos o que for necessário de acordo.

Ao contrário do que pregam métodos tradicionais, não é viável definir as características e comportamentos do produto com grande nível de detalhes no início do seu desenvolvimento. Embora trabalhemos para realizar um objetivo ou visão de produto suficientemente bem definida, as características, comportamentos e seus detalhes vão mudar ao longo desse trabalho conforme acontecem mudanças no ambiente e melhor compreendemos as necessidades dos clientes e usuários.

Scrum busca maximizar a habilidade do time que desenvolve o produto em responder rapidamente a essas mudanças. Ao mesmo tempo, o time buscará tornar-se

cada vez mais produtivo, avaliando quais práticas serão mantidas e quais serão modificadas ou acrescentadas.

Dessa forma, podemos afirmar que o Scrum baseia-se na transparência (a possibilidade de enxergar), inspeção (uma vez enxergando, capacidade de avaliar) e adaptação (uma vez avaliando, capacidade de mudar) frequentes tanto do produto quanto dos processos de seu desenvolvimento para que, pela realização de melhorias incrementais contínuas em ambos, busquemos sempre gerar o maior valor possível para os clientes e usuários.

> **Os pilares do Scrum**
>
> **Transparência**, **inspeção** e **adaptação** são considerados os pilares empíricos do Scrum.

Discussão: analogia da construção civil?

Desde quase o começo da história do desenvolvimento de software, propósito original do Scrum, a comparação com a construção civil foi largamente utilizada para descrever esse tipo de trabalho. São, no entanto, trabalhos de naturezas muito distintas.

Embora tenham evoluído ao longo dos tempos, projetos de construção civil existem há eras na história da humanidade e, em linhas gerais, sua forma de execução se manteve a mesma: uma longa fase de definições e especificações no início que tem como saída um plano, seguida de sua fase de execução.

Os métodos tradicionais de desenvolvimento de software buscaram algo similar com o modelo em cascata e suas fases sequenciais de levantamento e análise de requisitos, especificação, desenvolvimento e testes.

Ainda hoje, é comum usarmos as expressões "engenharia" ou "engenheiro de software", "arquitetura" ou "arquiteto de software", e até mesmo "construção de software", todas provindas da analogia com a construção civil.

Mas esse modelo faz sentido apenas para trabalhos que possuam um nível razoavelmente baixo de incertezas. Dada a natureza empírica do trabalho de desenvolvimento de software, compará-lo com projetos de construção civil não faz sentido. A analogia simplesmente não funciona.

2.6 Scrum é iterativo e incremental

Scrum é iterativo. O produto é desenvolvido em ciclos ou iterações, que se repetem sucessivamente, sem intervalos entre eles. Os ciclos têm, em princípio, o seu tamanho fixo e exatamente o mesmo formato bem estabelecido, composto por uma sequência definida de eventos.

A figura a seguir é uma representação simplificada de um ciclo do Scrum, indicando a sucessão de elementos presentes no ciclo e também indicando a sucessão de ciclos.

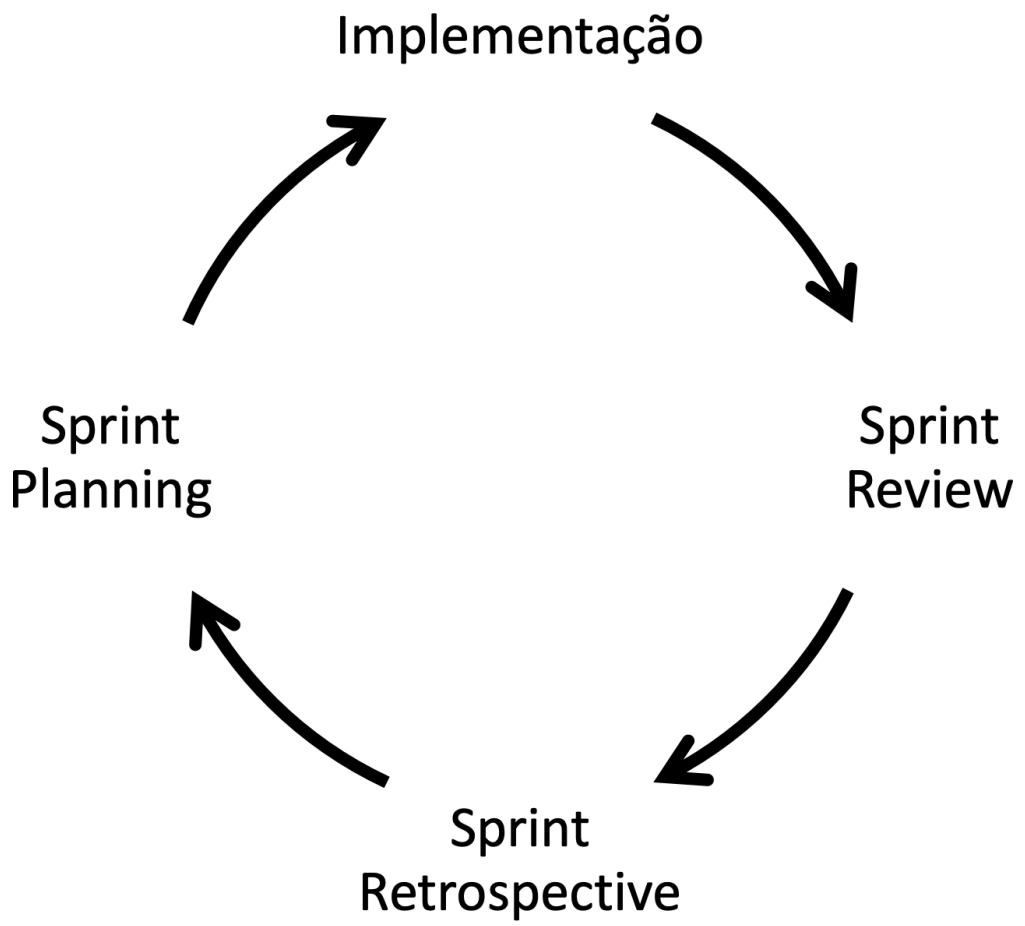


Figura 2.3: Sucessão de elementos de uma iteração no Scrum

Cada ciclo é como se fosse um pequeno projeto autocontido, com todas as atividades necessárias para implementarmos uma parte do produto estável e funcionando. Cada incremento no produto, implementado em cada ciclo, modifica e se soma ao que já temos pronto do produto até o momento. São funcionalidades prontas de ponta a ponta, de forma a facilitar a obtenção de feedback sobre o que foi produzido e possibilitar a entrega desse resultado do ciclo para seus usuários, caso decidamos fazê-lo. O feedback obtido sobre as partes prontas do produto alimenta a definição do que será realizado nos ciclos seguintes.

Definição: analogia do bolo

Imagine que dois times de confeiteiros vão assar um bolo, cada um utilizando um método diferente. Ambos os bolos possuem várias camadas (veja a figura a seguir).

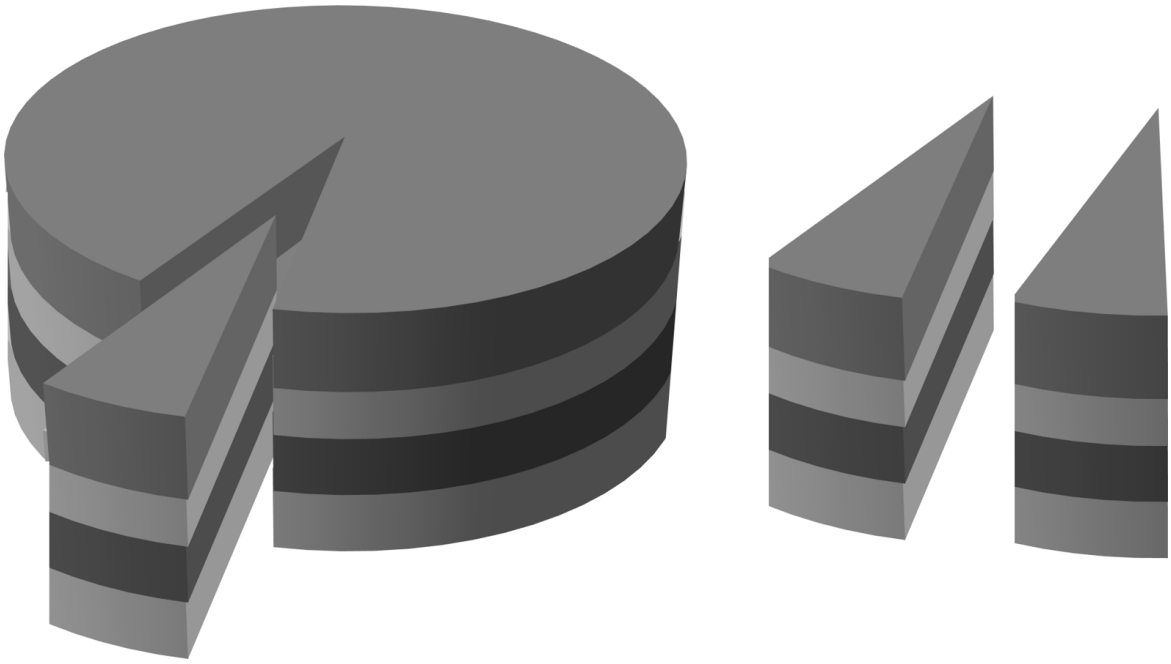


Figura 2.4: Analogia do bolo: o produto com Scrum é desenvolvido fatia a fatia

O primeiro time vai produzir o bolo camada por camada. Os membros do time misturam os ingredientes da primeira camada do bolo e, então, a assam. Em seguida, misturam os ingredientes da segunda camada, assam e colocam essa camada sobre a primeira. Fazem o mesmo com a terceira, quarta e quinta camadas, colocando cada uma sobre a anterior. Por fim, cobrem o bolo de glacê, que é a parte visível para quem comerá o bolo. Somente aí uma fatia do bolo pode ser cortada para ser consumida.

O segundo time tem um objetivo ousado. Ele deve permitir que o bolo seja consumido o mais rápido

possível e não apenas no final. Para atingir esse objetivo, o time utiliza uma forma de trabalho bem diferente: eles produzem o bolo fatia por fatia, em fatias bem finas. Cada uma dessas fatias possui apenas um pedacinho de cada uma das camadas, sendo cada pedacinho apenas o suficiente para sustentar o pedacinho acima, tudo então coberto pelo glacê. Assim, o bolo já pode ser consumido desde a sua primeira fatia.

É claro que o uso do segundo método é bastante incomum para a produção de um bolo. Mas pode ser a forma mais adequada para o desenvolvimento de novos produtos. Diferentemente do trabalho com métodos tradicionais, com Scrum o time de desenvolvimento do produto não produz camadas completas do produto, uma após a outra, desde arquitetura, banco de dados etc. até chegar na interface do usuário no caso de software. Pelo contrário, o time desenvolve o produto em incrementos. Cada um desses incrementos é uma fatia fina do produto pronta para ser utilizada, com apenas o necessário de cada uma das camadas do produto para funcionar.

Cada fatia inclui, então, uma parte fina de cada camada até a interface, que é a parte visível para o usuário — o glacê. Essa entrega de fatias funcionando do produto gera valor para os clientes e usuários desde cedo e possibilita o seu feedback, reduzindo os riscos no desenvolvimento do produto.

2.7 Os valores do Scrum

Além dos Valores e Princípios Ágeis, Scrum também tem o seu próprio conjunto de valores a serem seguidos. Eles complementam as regras e a estrutura do Scrum,

descrita a partir de suas responsabilidades, artefatos e eventos. Assim, os valores do Scrum representam os pilares para todo o trabalho realizado pelas pessoas que trabalham no desenvolvimento do produto.

Os cinco valores do Scrum são: compromisso, foco, abertura, respeito e coragem (SCHWABER; SUTHERLAND, 2020). Explico esses valores em detalhes a seguir:

- **Compromisso**: os membros do time se comprometem a buscar os objetivos acordados, visando a gerar valor ao satisfazer as próximas necessidades de negócios mais importantes. Eles se comprometem a trabalhar de forma colaborativa sobre as prioridades definidas, convergindo em cada ciclo de desenvolvimento do produto para algo pronto e com qualidade. Eles se comprometem a inspecionar e adaptar o que já está pronto, para poderem entregar mais valor. Os membros do time também se comprometem a seguir as regras do framework Scrum e a buscar continuamente melhorar sua forma de trabalhar, tornando-se cada vez mais efetivos como um time;

- **Foco**: os membros do time trabalham em ciclos curtos de tamanho fixo, o que os ajuda a manterem seu foco no trabalho planejado para o ciclo de trabalho corrente. Seu foco, no entanto, está muito mais em valor a ser gerado do que em tarefas a serem cumpridas. Assim, em cada ciclo, os membros do time mantêm seu foco em objetivos de valor claros e realizáveis, que visam a manter seus esforços nas necessidades de negócios mais importantes em cada momento. Eles planejam de forma contínua e gradual, mantendo um maior foco no que está mais próximo. O time conta com o trabalho de um facilitador, que remove

impedimentos que atrapalham e bloqueiam o seu trabalho. Esse facilitador também protege o time de distrações e interferências externas nocivas para que possa manter o foco no seu trabalho. Os times mais produtivos trabalham em apenas um produto de cada vez, evitando a multitarefa;

- **Abertura**: a abertura é necessária para que o time possa realizar a inspeção e adaptação. Assim, o time e as partes interessadas concordam em serem abertos sobre o trabalho e sobre os aprendizados e sobre os desafios que se apresentam ao realizá-lo. O time está aberto a receber feedback frequente de seus próprios membros, o que permite melhorar sua forma de trabalhar, cria transparência sobre os problemas e estimula a busca por soluções. O time também está aberto a receber feedback de partes interessadas sobre o que produziu, o que cria transparência sobre as mudanças necessárias e permite melhorarem continuamente o produto. O time sente-se seguro para planejar o trabalho que acredita que pode ser realizado e mantém visível o progresso desse trabalho, e assim pode tomar medidas de correção quando se fizerem necessárias. O time também interage, sempre que necessário, com quem necessitar para esclarecer suas dúvidas e questões e para ter impedimentos ao seu trabalho removidos o mais cedo possível;

- **Respeito**: os membros do time se respeitam e são respeitados por outros como pessoas adultas, capazes e responsáveis. São respeitados o bem-estar e o direito das pessoas que realizam o trabalho a terem uma vida privada. Os membros do time, pessoas de negócios e clientes colaboram e compartilham responsabilidades, buscando juntos o

sucesso no desenvolvimento do produto. Eles respeitam as opiniões uns dos outros, ouvem uns aos outros e buscam entender os diferentes pontos de vista. O time também mostra respeito às partes interessadas ao apresentar, com frequência, partes prontas e funcionais do produto, possibilitando seu feedback e oferecendo uma noção real de progresso. As decisões técnicas dos membros do time são respeitadas e eles respeitam as decisões de negócios com relação à ordem definida para o trabalho a ser realizado;

- **Coragem**: os membros do time têm coragem para fazer a coisa certa, para trabalhar sobre os problemas difíceis e para se manifestarem quando houver algo errado. Eles têm coragem para aceitar a mudança como parte natural do processo de desenvolvimento do produto. A organização tem coragem para confiar no time e deixá-lo livre para fazer o trabalho necessário para realizar os objetivos acordados. O time tem coragem para falhar e criar transparência sobre os problemas e, assim, aprender com essas falhas e problemas o mais cedo possível. Ele tem coragem para submeter a feedback e entregar com frequência as partes do produto criadas, tem coragem para remover os impedimentos ao seu trabalho e para trabalhar para realizar as mudanças necessárias na organização que o ajudem a se tornar mais produtivo e a organização, mais ágil.

Capítulo 3
Como é o Scrum?

Conteúdo

1. Como é o Scrum?
2. Visão geral.
3. Introdução.
4. Início do trabalho de desenvolvimento.
5. Planejamento do Sprint.
6. Desenvolvimento do produto.
7. Encerramento do Sprint.
8. Entregas.
9. Final do desenvolvimento do produto.

Como é o Scrum?

Neste capítulo, juntos passearemos pela linha do tempo no trabalho de um time no desenvolvimento de um produto com Scrum. No entanto, não pretendo ser prescritivo nem mostrar todas as possíveis alternativas. A ideia aqui é dar um exemplo genérico do que se pode passar quando utilizamos o framework para o trabalho de um time, citando as suas responsabilidades, eventos, artefatos e compromissos, adicionados a mais alguns elementos sugeridos como opções.

3.1 Visão geral

O desenvolvimento de um produto é iniciado para atender a uma oportunidade, problema, necessidade ou objetivo de negócios, representados pelo Objetivo do

Produto, seja de um cliente específico, interno ou externo, de um grupo de clientes ou do mercado.

Fazem parte do Time de Scrum: Scrum Master, que é um líder-servidor ou facilitador, Product Owner, responsável pelas decisões sobre o produto para maximizar o valor gerado, e Desenvolvedores, que realizam o trabalho de construção do produto propriamente dito.

Antes de qualquer trabalho de implementação começar, é comum serem realizadas algumas definições e preparativos básicos, buscando apenas o mínimo suficiente e necessário para o início do trabalho. É nesse momento, geralmente, que é criada uma versão inicial da lista de itens a serem implementados, chamada de Product Backlog.

O trabalho de desenvolvimento do produto com Scrum acontece em uma sucessão de ciclos, em cada qual os Desenvolvedores produzem partes do produto prontas, que podem ser entregues para seus usuários, incrementando e modificando o que foi produzido até então. Esses ciclos são chamados de Sprints. O desenvolvimento do produto com Scrum acontece Sprint após Sprint. Assim, ao terminar um Sprint, se inicia imediatamente o seguinte, sem intervalos entre eles. Para criar ritmo e disciplina, os Sprints têm sempre a mesma duração, que é de, no máximo, um mês cada. A preferência é por durações mais curtas, para levar à convergência frequente em algo pronto e para obter feedback.

Cada Sprint se inicia com a reunião de Sprint Planning, na qual Product Owner e Desenvolvedores definem um objetivo de valor para aquele Sprint, o Objetivo do Sprint. A partir dos itens de maior ordem do Product Backlog, eles planejam o trabalho a ser implementado dentro do

próprio Sprint para realizar esse objetivo, criando o Sprint Backlog. A Scrum Master pode facilitar a reunião, conforme necessário.

Após essa reunião, os Desenvolvedores iniciam o trabalho de implementação propriamente dito, que inclui tudo que é necessário para implementar partes prontas do produto. Pronto significa que o trabalho foi realizado de acordo com a Definição de Pronto e, idealmente, o Incremento ou Incrementos prontos resultantes podem ser entregues. Em cada um desses dias de trabalho, eles realizam uma reunião curta chamada de Daily Scrum, para se organizar acerca do que farão durante o dia de trabalho até a próxima Daily Scrum.

No último dia do Sprint, é realizada a reunião de Sprint Review, para que o Time de Scrum possa obter feedback dos clientes e demais partes interessadas sobre o que foi produzido no Sprint. Esse feedback alimentará o Product Backlog com o que poderá ser feito em Sprints futuras. Em seguida, o Time de Scrum realiza a reunião de Sprint Retrospective, visando à melhoria contínua da sua forma de trabalhar. No dia útil seguinte, um novo Sprint se inicia.

3.2 Introdução

O **Time de Scrum** é um grupo pequeno de pessoas responsável por todas as atividades relativas ao desenvolvimento do produto. O Time de Scrum é autogerenciado e, assim, define o que será feito, quem fará o quê, como será feito e quando, sempre alinhado ao contexto da organização e ao framework Scrum. O Time de Scrum é multifuncional, de forma a possuir, entre seus membros, todos os conhecimentos e

habilidades necessários para realizar o trabalho de desenvolvimento do produto, de ponta a ponta. Dessa forma, idealmente eliminamos as dependências externas e aumentamos a capacidade de geração de valor para os usuários do produto.

Time de Scrum

O Time de Scrum é formado por: Product Owner, Scrum Master e um número de Desenvolvedores.

Os **Desenvolvedores** realizam o trabalho de desenvolvimento do produto propriamente dito, com a adição, ajustes e modificação de características e comportamentos desse produto, realizando objetivos de valor acordados com o Product Owner.

O **Scrum Master** é um facilitador do trabalho e um líder que serve ao Time de Scrum em seu dia a dia, ensinando seus membros a assumirem a responsabilidade sobre seu trabalho, a crescer em autonomia e a se autogerenciarem. Ele atua tanto junto aos Desenvolvedores quanto junto ao Product Owner, além de facilitar as suas interações. Ele está presente e age como um facilitador nos eventos do Scrum, quando necessário.

Para realizar esse trabalho, o Scrum Master é forte em *soft skills*, ou seja, em competências comportamentais e pessoais. O Scrum Master também ensina Scrum ao Time de Scrum e a quaisquer pessoas na organização que considere necessário. Ele é o responsável por garantir que os **impedimentos** que o Time de Scrum encontre em seu trabalho sejam removidos, atuando sempre que necessário como um agente de mudança na organização. Esses impedimentos geram para o Time de Scrum o risco

de não se realizarem os objetivos e devem ser removidos o mais rapidamente possível.

O **Product Owner** é o responsável por maximizar o valor resultante do trabalho com Scrum. Ele é único para o produto e para o time que o desenvolverá. Ele trabalha diretamente com os **clientes**, ou seja, aqueles que solicitam ou contratam o desenvolvimento do produto, e com quaisquer outras partes interessadas que possam contribuir para o entendimento e definição do produto ao longo do seu desenvolvimento. O grupo de partes interessadas também inclui os próprios **usuários** do produto, que receberão, ao longo do trabalho, partes prontas do produto para serem utilizadas e, sobre elas, oferecerão **feedback** e possibilitarão a coleta de **métricas de uso**.

Para guiar o trabalho de desenvolvimento do produto, o Product Owner cria ou se alinha a um **Objetivo do Produto** definido para o produto a ser desenvolvido. O Objetivo do Produto, comumente referido como "visão do produto", representa o propósito ou motivação central para o trabalho de desenvolvimento do produto e deve ser compreendido por todas as partes interessadas, gerando alinhamento entre elas. O Product Owner comunica e mantém o alinhamento a esse Objetivo do Produto ao longo do desenvolvimento do produto.

A partir do Objetivo do Produto, o Product Owner pode opcionalmente criar um plano de como, naquele momento, espera que o produto evolua ao longo do tempo, chamado de ***roadmap*** do produto. Este é geralmente expresso por uma linha do tempo com os objetivos de negócio esperados para momentos determinados no futuro, estabelecidos em marcos desse *roadmap*. O *roadmap*, quando utilizado, é frequentemente atualizado para refletir o entendimento

mais atual sobre a possível evolução do produto. O artefato *roadmap* do produto não é parte do Scrum.

O **Product Backlog** é uma lista ordenada, incompleta e dinâmica de itens que representam o que o Product Owner acredita, em cada momento, que será produzido. Ele serve de entrada para todo e qualquer trabalho realizado pelos Desenvolvedores nos ciclos de desenvolvimento. O Objetivo do Produto é o compromisso do Product Backlog e dele faz parte, ou seja, o Time de Scrum trabalha sobre o Product Backlog para realizar o Objetivo do Produto.

O Product Owner é o único responsável por gerenciar o Product Backlog, trabalho que inclui definir o produto pouco a pouco, atualizando essa lista ao longo de todo o desenvolvimento do produto, e a ordenando para que, em cada ciclo, o mais importante para maximizar o valor seja implementado. O Product Owner realiza esse trabalho ou delega parte dele para outras pessoas, no entanto, mantendo-se sempre responsável por seus resultados.

Os itens do alto do Product Backlog são os mais importantes naquele momento para maximizar o valor e, por essa razão, possuem mais detalhes, suficientes para possibilitar que sejam implementados primeiro. Os itens mais abaixo têm gradativamente menos detalhes e vão evoluir quando e se o seu momento chegar.

O Product Backlog pode ser longo, contendo desde itens pequenos e bem detalhados até itens grandes e vagos. Mas ele também pode ser curto e conter apenas uma quantidade de itens necessária para o trabalho mais próximo (essa é a minha versão preferida, pois evita o apego ao que já foi definido). Seja como for, o Product Backlog evoluirá ao longo de todo o desenvolvimento do

produto e será frequentemente modificado com a adição, subtração, detalhamento, divisão, reordenamento e modificação de seus itens.

Para guiar o trabalho, preferimos utilizar a perspectiva dos usuários em cada um dos itens do Product Backlog, a partir das **User Stories**. Uma User Story traz uma representação simples e concisa para um item do Product Backlog. A ideia é que, para cada um desses itens representados por User Stories, Desenvolvedores, Product Owner e, talvez, clientes e partes interessadas conversem e detalhem a característica ou comportamento a ser implementado no produto. A User Story, embora seja uma boa prática, não é parte do framework Scrum. Cabe ao Time de Scrum definir a forma que melhor lhe serve para representar os itens do Product Backlog.

A **Definição de Pronto** (*Definition of Done*, em inglês) é um entendimento formal compartilhado entre Desenvolvedores e Product Owner sobre que atividades ou condições são necessárias para que qualquer item do Product Backlog seja considerado pronto ao ser implementado e, assim, passe a fazer parte do produto. A Definição de Pronto é a mesma para todos os itens do Product Backlog, ao longo do desenvolvimento do produto, mas ela evolui com o tempo de acordo com a necessidade.

Os eventos do Scrum — o próprio ciclo de desenvolvimento, ou Sprint, e as reuniões de Sprint Planning, de Daily Scrum, de Sprint Review e de Sprint Retrospective — possuem uma duração máxima ou fixa definida, chamada de **timebox**. Os timeboxes são importantes, pois evitam o desperdício, limitando o tempo em que um objetivo deve ser realizado, além de

ajudar a criar um ritmo ou uma regularidade na realização do trabalho.

Podemos ver o ciclo completo do Scrum na figura a seguir (SCHWABER; BEEDLE, 2002; SCHWABER, 2004).

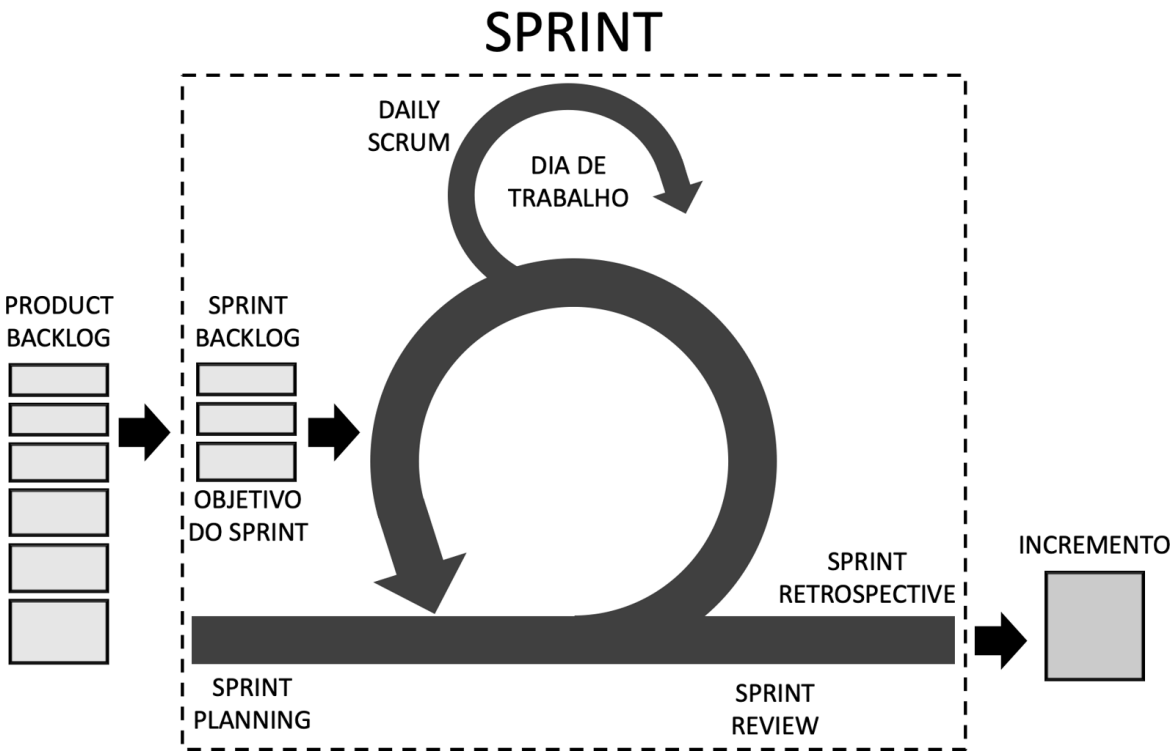


Figura 3.1: O ciclo do Scrum

3.3 Início do trabalho de desenvolvimento

O desenvolvimento do produto será realizado de forma incremental nos **Sprints**, as iterações ou ciclos de desenvolvimento no Scrum. Antes do primeiro Sprint, pode existir um curto trabalho inicial de preparação, sempre dentro do que é estritamente suficiente e necessário. A definição do Objetivo do Produto, a criação de um Product Backlog inicial, a Definição de Pronto e a

formação ou escolha do Time de Scrum são exemplos comuns dessas atividades.

No entanto, esse e quaisquer outros trabalhos de base são preferivelmente realizados progressivamente ao longo dos primeiros Sprints, nos quais mesmo assim já será gerado um valor utilizável.

Pode ser necessário, por exemplo, tomar decisões estruturais básicas para o produto, de forma a reduzir os riscos de decisões tardias que invalidariam o que já foi produzido. Pode ser também necessário criar ou adaptar uma infraestrutura que servirá de suporte para o desenvolvimento do produto. A ideia dessas atividades é gerar apenas o mínimo necessário e suficiente para reduzir os riscos, mas sem engessar o desenvolvimento do produto, nem gerar grandes desperdícios.

> Cuidado com longas definições de arquiteturas, de requisitos ou de jornadas de usuários sem a implementação nem a entrega de partes funcionais do produto! No desenvolvimento ágil de produtos, entregamos valor o mais cedo e frequentemente possível, produzindo em cada vez apenas o suficiente e necessário para isso.

Quanto à definição do Time de Scrum, esperamos que uma organização que utilize Scrum procure trabalhar com times estáveis. Nesse contexto, ao decidir quem participará do desenvolvimento de um novo produto, a melhor pergunta a ser respondida deixa de ser "de que pessoas precisamos para realizar esse trabalho", e passa a ser "qual dos nossos times tem os conhecimentos e habilidades mais adequados para o desenvolvimento desse produto". Ou seja, ao invés de formar um novo

Time de Scrum para cada novo desenvolvimento, buscamos escolher qual o melhor Time de Scrum para o trabalho em questão.

Nos casos em que é necessário formar um novo Time de Scrum, o processo varia de organização para organização. Entre diferentes possibilidades, pode haver um departamento responsável pela seleção de pessoal ou podemos, por exemplo, partir de um Product Owner, de um Scrum Master ou de um ou mais Desenvolvedores iniciais que escolherão o resto do time.

3.4 Planejamento do Sprint

O Sprint se inicia com a reunião de **Sprint Planning**, na qual Desenvolvedores e Product Owner planejam o trabalho a ser realizado no próprio Sprint. Os Desenvolvedores selecionam um conjunto de itens a partir do alto do Product Backlog, conforme ordenados pelo Product Owner, que julgam ser capazes de implementar na duração do Sprint e estabelecem juntos com o Product Owner um objetivo a ser realizado a partir da implementação desses itens, chamado de **Objetivo do Sprint**. Os itens selecionados constituem apenas uma previsão daquilo que será implementado.

Ao realizar o planejamento, os Desenvolvedores podem utilizar a média da quantidade de trabalho que foram capazes, em conjunto, de deixar pronto nos últimos poucos Sprints para decidir quantos itens farão parte do plano desse Sprint que se inicia. Essa média da quantidade de trabalho realizado por Sprint é chamada de **Velocidade**. No primeiro Sprint, no entanto, os Desenvolvedores ainda não têm esse histórico de

trabalho já realizado e terão que julgar o quanto acreditam ser razoável trazer para o Sprint.

Embora o guia oficial do Scrum estabeleça que os itens do Product Backlog são dimensionados pelos Desenvolvedores (SCHWABER; SUTHERLAND, 2020), nenhum tipo de estimativa formal é prescrito pelo framework. No entanto, **Story Point** é uma unidade de estimativa de esforço relativo muito utilizada por Desenvolvedores para facilitar o planejamento do seu trabalho.

Além de selecionar os itens e definir um objetivo, o que realizam juntamente com o Product Owner, os Desenvolvedores também criam um plano de como o que foi selecionado será implementado. Esse plano é parcialmente definido na reunião de Sprint Planning e segue sendo definido e implementado ao longo do Sprint. Esse plano é geralmente expresso por um conjunto de tarefas, para cada item, a serem realizadas durante o Sprint. Essas tarefas são, de forma geral, as ações ou atividades de trabalho que os devs identificam como necessárias para implementar cada item selecionado. Usa-se a Definição de Pronto para guiar a definição desse plano, visando a garantir que o que for implementado no Sprint seja entregável.

O conjunto formado pelos itens selecionados, seu respectivo plano e o Objetivo do Sprint é chamado de **Sprint Backlog**, e é geralmente representado na forma de um quadro de tarefas (veja a figura a seguir). O Sprint Backlog existe apenas dentro do seu Sprint respectivo. O Objetivo do Sprint é o compromisso do Sprint Backlog, ou seja, os Desenvolvedores trabalham sobre o Sprint Backlog para realizar o Objetivo do Sprint.

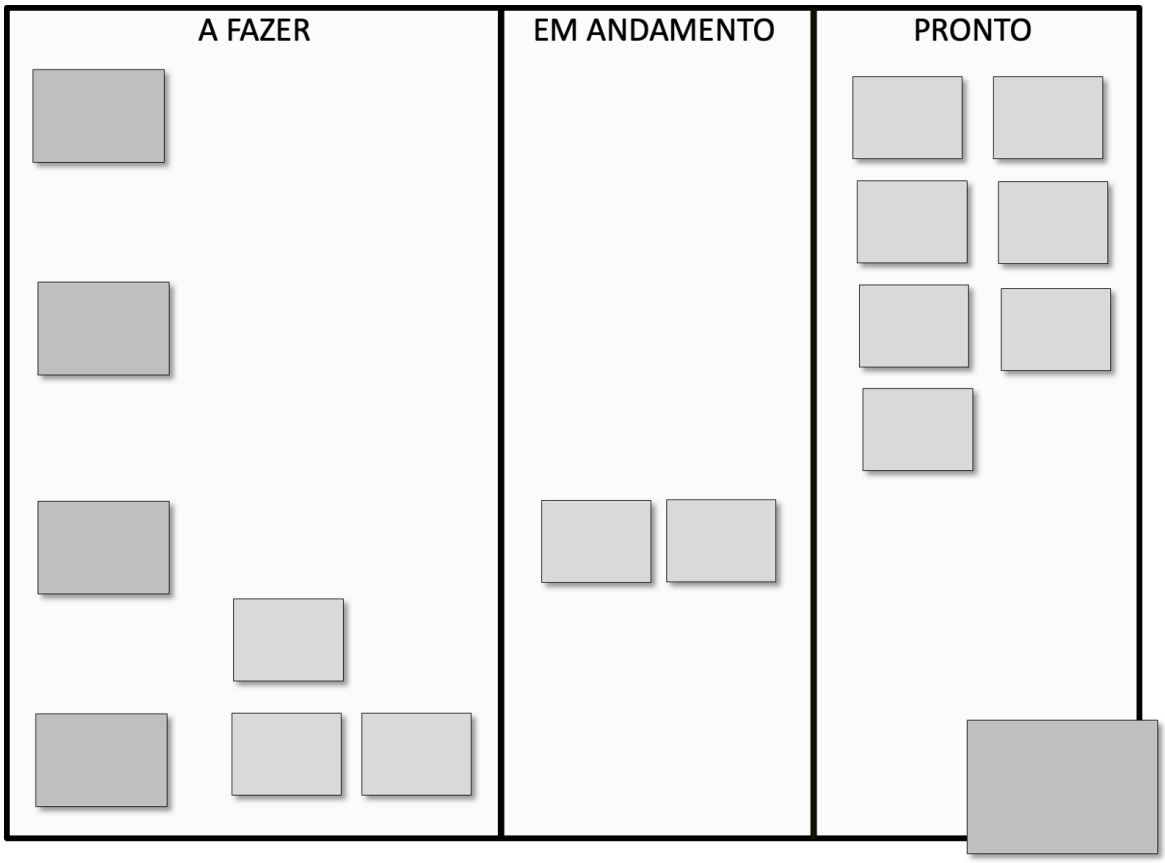


Figura 3.2: Exemplo de quadro de tarefas representando o Sprint Backlog

3.5 Implementação

Após a reunião de Sprint Planning, os Desenvolvedores iniciam o trabalho de implementação propriamente dito dos itens do Sprint Backlog. Eles começaram a definir, durante a reunião de Sprint Planning, como o trabalho será realizado. Agora, são os responsáveis por se organizar para realizar esse trabalho e por monitorar a realização do Objetivo do Sprint.

Eles iniciam o trabalho a partir do primeiro item do Sprint Backlog, que é o item mais importante para realizarem o Objetivo do Sprint. Diferentes devs realizam as diferentes

tarefas necessárias, comunicando-se e colaborando durante todo o processo. Aos poucos, vão migrando para os itens seguintes do Sprint Backlog, sempre preocupando-se em completá-los de acordo com a sua ordem.

Pare de começar e comece a terminar

Os Desenvolvedores trabalham juntos, de acordo com a ordem estabelecida, completando os itens de maior ordem primeiro e em direção aos de menor ordem, até que se encerre o tempo do Sprint. Caso não terminem tudo o que planejaram, deixam os de menor ordem de fora, mas garantindo que os de maior ordem estejam prontos. Se trabalhassem nos diversos itens ao mesmo tempo, o maior risco seria o de chegarem ao final do Sprint com a maioria ou todos os itens quase prontos e, assim, não prontos e não entregáveis.

Em cada dia do trabalho de desenvolvimento do produto, os Desenvolvedores se encontram para a reunião de **Daily Scrum**, durante no máximo 15 minutos, sempre no mesmo horário e no mesmo local. O objetivo dessa reunião de inspeção e adaptação é garantir entre eles a transparência sobre seu trabalho e planejar, informalmente, o próximo dia de trabalho, baseados no que aconteceu desde o dia anterior. Realizá-la junto ao quadro de tarefas, portanto, é uma boa prática.

A reunião de Daily Scrum é um excelente momento para que atualizem, quando adotado, o **Gráfico de Sprint Burndown**, uma ferramenta que Desenvolvedores podem utilizar para monitorar seu progresso no Sprint. Esse gráfico possui no eixo horizontal os dias de trabalho

do Sprint, e no eixo vertical a quantidade de trabalho planejado restante, que pode ser medida pela quantidade de tarefas restantes ou por algum tipo de estimativa realizada sobre as tarefas ou sobre os itens do Sprint Backlog. Os Desenvolvedores marcam no gráfico a quantidade de trabalho restante no dia corrente (veja a figura a seguir). O Gráfico de Sprint Burndown também não é prescrito pelo framework (já foi no passado).

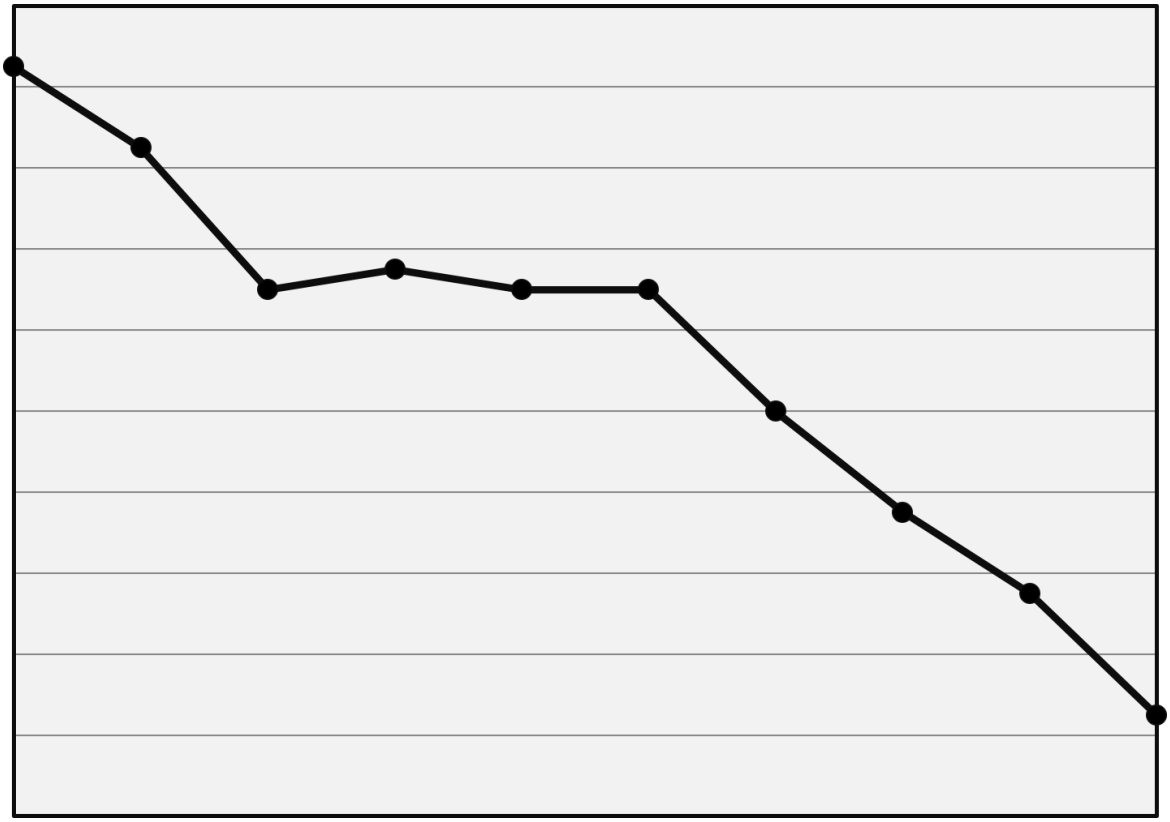


Figura 3.3: Exemplo de Gráfico de Burndown

Em seu dia a dia de trabalho, é comum Desenvolvedores se depararem com desafios. Sempre que encontram algo que os impede de realizar seu trabalho e, assim, ameaça o Objetivo do Sprint, eles avisam à Scrum Master, que inicia imediatamente o trabalho para gerir a remoção desse **impedimento**.

Uma Product Owner acessível e presente ao longo do Sprint permite que Desenvolvedores tirem dúvidas de negócios que naturalmente surgem sobre os itens que estão sendo trabalhados. Ao mesmo tempo, a Product Owner procura manter contato frequente com os clientes, usuários e demais partes interessadas, além de observar métricas de uso do produto, para entender suas reais necessidades ou objetivos de negócios mais importantes, colher seu feedback e atualizar o Product Backlog de acordo, sempre que considerar necessário.

Por vezes, um ou mais itens do alto do Product Backlog chegam à reunião de Sprint Planning carregando muitas incertezas, sem detalhes suficientes ou com uma granularidade muito grossa. Com itens assim, existe um risco significativo de que a reunião seja longa, sem no entanto ser produtiva nem efetiva. Pode ser importante, portanto, que esses itens cheguem à reunião preparados o suficiente para que os Desenvolvedores possam planejar a sua implementação. **Refinamento do Product Backlog** é o trabalho de preparação de itens do Product Backlog, que pode incluir fatiá-los em itens menores, adicionar e modificar detalhes, estimá-los e reordená-los, com o objetivo de que estejam aptos para serem implementados pelos Desenvolvedores em um Sprint.

O trabalho de refinamento é, em geral, realizado pelos Desenvolvedores em conjunto com a Product Owner. Eles decidem juntos como vão colaborar para preparar itens para implementação. O Refinamento do Product Backlog pode ser realizado na reunião de Sprint Planning. Muitos times optam, no entanto, por realizá-lo ao longo do Sprint anterior, visando a preparar itens suficientes para o Sprint seguinte. Essa interação pode acontecer em uma ou mais sessões agendadas ou sempre que os

Desenvolvedores considerarem necessário. Em um cenário mais maduro, o Product Owner pode delegar para os Desenvolvedores a responsabilidade de realizarem sessões de refinamento diretamente com clientes, usuários e demais partes interessadas.

Pode ser interessante que o Time de Scrum utilize uma **Definição de Preparado** (*Definition of Ready*, em inglês) para guiar o Refinamento do Product Backlog a ser realizado para preparar os itens que serão considerados para implementação no Sprint seguinte. A Definição de Preparado traz critérios claros que definem qual é o estado mínimo necessário em que qualquer item de Product Backlog deve estar para poder entrar no plano do que será implementado no Sprint, o Sprint Backlog. É comum essa Definição de Preparado exigir que os itens sejam pequenos o suficiente e possuam os detalhes da característica ou comportamento a ser adicionado no produto, além de outros critérios específicos do contexto. A Definição de Preparado é sempre a mesma para todos os itens do Product Backlog, mas ela pode evoluir com o tempo.

Com o uso da Definição de Preparado, Desenvolvedores e Product Owner assumem a responsabilidade conjunta de chegarem à reunião de Sprint Planning seguinte com itens suficientes do alto do Product Backlog preparados de acordo com essa definição. Assim, a Definição de Preparado pode ser vista como mais um compromisso do Product Backlog.

A Definição de Preparado não é, no entanto, parte do framework Scrum, apesar de que o conceito de haver itens preparados para o trabalho o é.

3.6 Encerramento do Sprint

Ao final do Sprint, os Desenvolvedores terão gerado, a partir do trabalho realizado no Sprint, ao menos um **Incremento** do produto.

Cada Incremento é formado por um ou mais itens prontos do Sprint Backlog, de acordo com a Definição de Pronto, que potencialmente representem valor visível e útil para os clientes e usuários do produto. Por natureza, o Incremento é pronto, então a Definição de Pronto é o compromisso de todo e qualquer Incremento gerado pelos Desenvolvedores. Por ser pronto, o Incremento é entregável, o que significa que nenhum trabalho adicional é necessário para que o Incremento possa ser entregue. O momento de entregar um ou mais Incrementos, no entanto, é uma decisão do Product Owner.

Caso, no transcorrer do Sprint, o Objetivo do Sprint tenha perdido o seu sentido, o Product Owner pode decidir pelo cancelamento do Sprint, antecipando o seu encerramento. Essa é uma situação de exceção e raramente deve ocorrer.

No último dia do Sprint, o Product Owner e os Desenvolvedores se encontram para duas reuniões consecutivas de inspeção e adaptação, ambas facilitadas pela Scrum Master. Na primeira reunião, de **Sprint Review**, o Time de Scrum faz a inspeção e adaptação do produto, enquanto que na reunião de **Sprint Retrospective**, o Time de Scrum faz a inspeção e adaptação da sua forma de trabalhar.

São convidados para a reunião de Sprint Review os clientes e demais pessoas relevantes que possam prover feedback sobre o Incremento ou Incrementos que foram

produzidos durante o Sprint. Nessa reunião, Desenvolvedores e Product Owner apresentam e demonstram para os presentes o resultado do seu trabalho no Sprint, para então obter o seu feedback. Apenas itens prontos de acordo com a Definição de Pronto fazem parte de um Incremento e são apresentados na Sprint Review. Essa reunião traz uma importante oportunidade para antecipar parte do feedback que só virá com o uso real do que foi implementado, após a sua entrega e, assim, reduzir os riscos no desenvolvimento do produto.

O feedback obtido dos clientes e demais pessoas relevantes presentes na reunião de Sprint Review é utilizado pelo Product Owner como matéria-prima para alterações no Product Backlog. Ou seja, para fornecer elementos para que os Desenvolvedores modifiquem o produto que está sendo desenvolvido, de forma a melhor atender às necessidades dos clientes e usuários.

Após a reunião de Sprint Review, os membros do Time de Scrum realizam a reunião de Sprint Retrospective, com o objetivo de se tornarem mais efetivos como um time. Nessa reunião de melhoria contínua, eles identificam o que foi bem no Sprint corrente, e que por essa razão pode ser mantido no próximo Sprint, e o que pode melhorar, buscando formas práticas e traçando planos de ação necessários para realizarem essas melhorias. Existem inúmeras técnicas para a realização de reuniões de Sprint Retrospective, sempre visando melhorar como o Time de Scrum realiza o seu trabalho.

Uma vez terminada a reunião de Sprint Retrospective, está encerrado o Sprint. Um novo Sprint se inicia imediatamente após o término da anterior, logo no dia útil seguinte.

3.7 Entregas

Uma vez que o Incremento do produto ou a soma de Incrementos do produto represente valor suficiente, é importante que chegue a usuários o mais rápido possível.

Por meio de entregas frequentes, o Time de Scrum pode obter feedback e métricas de uso do produto, desenvolvendo um produto que será utilizado e trará valor. Com cada entrega de valor para o usuário final, o Time de Scrum provê valor de negócios a seus clientes.

A estratégia de entregas a ser utilizada é definida pela Product Owner. Ela decide quando ou com que frequência as entregas serão realizadas e quem vai recebê-las. No melhor caso, seriam realizadas entregas frequentes para o usuário final, mas pode ser mais viável que usuários intermediários recebam e utilizem o produto com uma maior frequência para, assim, prover feedback e métricas.

Podemos realizar opcionalmente uma reunião de **Release Planning** para planejar a entrega seguinte. Essa reunião, que não faz parte do Scrum, pode acontecer em algum momento antes do início dos trabalhos para a entrega correspondente, provavelmente durante o Sprint anterior ou, para a primeira entrega, antes do primeiro Sprint.

Na reunião de Release Planning, caso realizada, Product Owner e Desenvolvedores podem estabelecer um plano de entrega, que geralmente contém uma data para a entrega (ao menos, aproximada), um objetivo de negócios a ser realizado, que podemos chamar de **Objetivo da Entrega**, e um conjunto de itens selecionados a partir do alto do Product Backlog visando a realizar esse objetivo. Essa reunião não substitui as

reuniões de Sprint Planning que serão realizadas para cada um dos Sprints correspondentes à entrega.

O progresso em direção à data da entrega e, portanto, à realização do Objetivo da Entrega pode ser inspecionado Sprint a Sprint. Uma ferramenta muito utilizada com esse propósito é o **Gráfico de Release Burndown**. Esse gráfico possui, no eixo X, o momento final dos Sprints correspondentes à entrega e, no eixo Y, a quantidade de trabalho planejado restante, que pode ser medida pela quantidade de itens do Product Backlog restantes entre os previstos para a entrega ou por estimativas realizadas sobre os itens, por exemplo.

Caso opte por utilizá-lo, o Product Owner marca no gráfico a quantidade de trabalho restante ao final de cada Sprint, momento em que o plano da entrega é revisto. Outras ferramentas, como o **Gráfico de Release Burnup**, que mostra o trabalho acumulado já realizado e o trabalho total previsto em cada momento, podem ser usadas em seu lugar.

A reunião de Release Planning, a Objetivo de Entrega, o Gráfico de Release Burndown e o Gráfico de Release Burnup não são parte do Scrum.

3.8 Final do desenvolvimento do produto

O desenvolvimento do produto pode ser contínuo, ou seja, cenário em que o Time de Scrum seguirá indefinidamente evoluindo e mantendo o produto, ou pode possuir uma duração predeterminada. Em qualquer dos casos, o desenvolvimento de um produto pode ser encerrado por diversas razões, dependendo do contexto: o produto deixou de ser utilizado, o tempo de trabalho

previsto em contrato expirou, valor suficiente já foi entregue para os clientes e usuários do produto, o orçamento para o trabalho acabou ou o contrato foi cancelado, entre outras.

Capítulo 4
De onde veio o Scrum?

Conteúdo

1. De onde veio o Scrum?
2. Um novo jogo.
    - Instabilidade embutida.
    - Times de projeto auto-organizados.
    - Sobreposição nas fases de desenvolvimento.
    - Múltiplo aprendizado.
    - Controle sutil.
    - Transferência de aprendizado.
3. Wicked problems.
4. Lean e Sistema Toyota.
    - O que é o Lean?
    - Os princípios do Lean.
    - Lean e Ágil.
        - Definir valor.
        - Identificar a cadeia de valor.
        - Criar fluxo e estabelecer a produção "puxada".
        - Buscar a perfeição.
5. Do trabalho braçal ao trabalho do conhecimento.
    - Taylorismo.
    - A crise de produtividade.
    - Trabalhadores do conhecimento.

4.1 De onde veio o Scrum?

O primeiro uso de Scrum foi liderado por Jeff Sutherland em 1993. Jeff trabalhava na Easel Corporation, uma

empresa de desenvolvimento de software de Massachusetts (Estados Unidos), e aplicou Scrum ao projeto mais crítico da organização. Foram obtidos resultados excepcionais com essa nova forma de se trabalhar.

Em 1995, Jeff apresentou o primeiro time de Scrum a Ken Schwaber, que era CEO da empresa Advanced Development Methods. Ken vinha trabalhando com ideias relacionadas, focadas em abordagens empíricas para o desenvolvimento de software. Eles trabalharam juntos para criar uma definição formal de Scrum, apresentada por Ken no OOPSLA de 1995, um congresso de Orientação a Objetos (SCHWABER, 1997).

Em seguida, Mike Beedle, um dos primeiros a adotar Scrum em seu trabalho, liderou os esforços para codificar e padronizar o framework, para então publicá-lo formalmente como um padrão organizacional (BEEDLE et al., 1999).

Em 2001, Jeff Sutherland e Ken Schwaber, assim como Mike Beedle, foram signatários do Manifesto Ágil como representantes do Scrum (BECK et al., 2001). Como mencionei no capítulo *O que é Scrum?* (veja *Scrum é ágil*), esse manifesto deu início ao movimento ágil, do qual Scrum faz parte. Nesse mesmo ano, Ken Schwaber publicou, em parceria com Mike Beedle, *Agile software development with Scrum*, o primeiro livro a descrever o framework Scrum (SCHWABER; BEEDLE, 2002).

Curiosidade: Jeff Sutherland no Rio de Janeiro

Recebemos o Dr. Jeff Sutherland como *keynote speaker* para o Scrum Gathering do Rio de Janeiro de 2017.

Tive o prazer de ciceronear o Jeff em seu primeiro dia na cidade. Eu o levei para almoçar em uma autêntica churrascaria rodízio brasileira e, depois de várias taças de prosecco oferecidas pelo restaurante, seguimos para o Pão de Açúcar, ponto turístico da cidade. Por fim, encerramos o dia tomando um café na Confeitaria Colombo do Forte de Copacabana.

Conversamos muito sobre Scrum, claro, e sobre a sua trajetória. Pude perceber que Jeff, ali com seus 76 anos, é uma pessoa extremamente inteligente, mas ao mesmo tempo humilde e educada. Algo que me chamou a atenção foi que ele fez questão de dar valor a cada comida típica que ele experimentou e cada lugar novo que ele conheceu.

Recebemos o Jeff na sede da K21 para um *happy hour* muito animado. E, como já era de se esperar, o seu *keynote* no evento foi um grande sucesso.

4.2 Um novo jogo

Ken Schwaber e Jeff Sutherland se referem ao artigo *The new new product development game* (ou "O novo jogo no desenvolvimento de novos produtos") como principal fonte de inspiração para a criação do Scrum. Os autores do artigo, Hirotaka Takeuchi e Ikujiro Nonaka, o publicaram na renomada revista Harvard Business Review em meados dos anos 80 (TAKEUCHI; NONAKA, 1986).

Takeuchi e Nonaka são conhecidos por suas importantes contribuições na área de gestão de conhecimento e aprendizado organizacional. No artigo, eles estudaram

empresas do Japão e Estados Unidos, como a 3M, Xerox, Honda, Epson, NEC e Hewlett-Packard. Essas empresas estavam utilizando uma abordagem diferente para times de desenvolvimento de novos produtos e, com ela, obtendo excelentes resultados em produtos como carros, copiadoras, computadores pessoais e câmeras.

Esses times possuíam seis características fundamentais: instabilidade embutida, times de projeto auto-organizados, sobreposição nas fases de desenvolvimento, múltiplo aprendizado, controle sutil e transferência de aprendizado.

Instabilidade embutida

A alta gerência dá a partida no desenvolvimento, indicando objetivos ou estratégias amplos e desafiadores para criar um elemento de tensão, ao mesmo tempo em que dá ao time uma grande liberdade para alcançar esses objetivos.

Times de projeto auto-organizados

Times de projeto auto-organizados possuem:

- autonomia no seu dia a dia do trabalho, com pouquíssima interferência da alta gerência;
- autotranscendência, ou seja, uma busca interminável por um limite inalcançável, em que o time estabelece objetivos cada vez mais altos durante o processo de desenvolvimento;
- fertilização cruzada, ou seja, o time possui uma variedade de especializações, comportamentos e personalidades, e a interação entre seus membros promove a criação de novas ideias e conceitos, a distribuição do conhecimento e a consideração do que é melhor para o grupo.

Sobreposição nas fases de desenvolvimento

Embora seus membros iniciem o projeto com diferentes horizontes de tempo, pois o trabalho de cada um pode começar em fases diferentes do projeto, o time acaba por criar um ritmo comum de trabalho, já que devem sincronizar o seu passo para alcançar um prazo limite comum. Como consequência, eles começam a trabalhar como um só, e assim os indivíduos e o todo se tornam inseparáveis, criando um pulso comum. Esse pulso funciona como uma força que move o time adiante.

A sobreposição nas fases de desenvolvimento possibilita um melhor tratamento de gargalos que surgirem no fluxo de trabalho, oferece maior velocidade e flexibilidade, reforça a responsabilidade compartilhada e a cooperação, estimula o envolvimento e compromisso, aguça o foco na resolução de problemas, encoraja a tomada de iniciativas, possibilita o desenvolvimento de habilidades diversificadas e intensifica a sensibilidade com relação às condições de mercado.

Múltiplo aprendizado

Por ficarem expostos a diversas fontes de informação, os membros do time adquirem um amplo conhecimento e habilidades diversas, gerando um time versátil capaz de resolver uma miríade de problemas rapidamente.

Esse aprendizado se dá em duas dimensões: em múltiplos níveis, isto é, o aprendizado individual, o aprendizado do grupo e o aprendizado da organização; e em diversas funções, nas quais especialistas são encorajados a acumular experiências em áreas fora de

sua especialidade. O múltiplo aprendizado permite ao time responder rapidamente às mudanças de mercado.

Controle sutil

Embora os times de projeto tenham um alto grau de liberdade, eles não trabalham sem qualquer controle. A alta gerência estabelece pontos de verificação para prevenir que a instabilidade, ambiguidade e tensão levem ao caos. Ela evita, no entanto, o tipo de controle rígido que prejudica a criatividade e espontaneidade.

Esse controle sutil é exercido das seguintes formas:

- selecionando as pessoas certas para o projeto, enquanto monitora as dinâmicas do grupo e adiciona ou remove membros quando necessário;
- criando um ambiente aberto de trabalho;
- encorajando os engenheiros a interagirem com os clientes e fornecedores;
- estabelecendo um sistema de avaliação e recompensa baseado no desempenho do grupo, e não no individual;
- gerenciando as diferenças de ritmo ao longo do processo de desenvolvimento;
- tolerando erros, buscando que sejam cometidos o mais cedo possível e que sempre se aprenda com eles;
- encorajando fornecedores a também se auto-organizarem e envolvendo-os desde cedo no projeto.

Transferência de aprendizado

Além do acúmulo de conhecimento entre diferentes níveis e funções, observamos a transferência desse conhecimento para outras divisões da organização e para outros projetos de desenvolvimento de novos produtos.

Essa transferência ocorre ao designarmos indivíduos-chave para projetos subsequentes, ou ao criarmos padrões a partir de atividades do projeto.

A transferência de conhecimento funciona bem quando o ambiente externo é estável. No entanto, mudanças no ambiente podem fazer com que essas lições aprendidas sejam impraticáveis. Nesses cenários, desaprender pode ajudar o time a se manter alinhado à realidade de fora de seu ambiente e a realizar melhorias incrementais.

Definição: o nome "Scrum"

Scrum não é uma sigla ou um acrônimo. Assim, não devemos escrever SCRUM, com as letras maiúsculas, ou S.C.R.U.M., com um ponto após cada letra. Scrum é, na verdade, um termo do jogo de rúgbi. Como, então, esse nome veio parar em um framework ágil? No artigo *The new new product development game*, Takeuchi e Nonaka tratam de times de alto desempenho de desenvolvimento de novos produtos em empresas americanas e japonesas nos anos 80. Os autores comparam sua forma de trabalhar a times de rúgbi, que passam a bola entre seus membros, para trás e para frente, mas se movem em direção ao objetivo comum como um só bloco.

Essa visão de time se opõe à prática tradicional de desenvolvimento de produtos, em que o trabalho progride em fases sequenciais. Nesse cenário, em que as funções são altamente especializadas e segmentadas, cada um faz a sua parte e passa o trabalho para a fase seguinte.

Scrum, abreviação de *scrummage*, é uma formação em que os times se posicionam para a recolocação da bola

em jogo no rúgbi. Os membros dos times opostos se agrupam uns contra os outros, na tentativa de ganhar a posse da bola. A partir dessa analogia com o rúgbi, o nome foi atribuído ao framework Scrum.

4.3 Wicked problems

> *Se Scrum fosse aplicado ao desenvolvimento de software, seria mais ou menos assim: (...) forme o time escolhendo cuidadosamente uma pessoa de cada uma das [fases tradicionais de desenvolvimento]. (...) Você então dá a eles uma descrição do problema a ser resolvido e (...) os desafia, dizendo que deverão produzir o sistema em, digamos, metade do tempo e dinheiro e que deve ter o dobro da performance de outros sistemas. Em seguida, você lhes diz que como eles farão isso é da conta deles* (DEGRACE; STALL, 1990).

Como mencionei na seção anterior, Jeff Sutherland e Ken Schwaber apontam para o artigo *The new new product development game*, publicado em 1986 por Takeushi e Nonaka, como a principal fonte de inspiração direta para a criação do framework Scrum (TAKEUCHI; NONAKA, 1986).

No entanto, foi o livro *Wicked problems, righteous solutions*, de 1990, que pela primeira vez trouxe a ideia de aplicarmos o conjunto de práticas descritas pelos dois autores japoneses ao desenvolvimento de software. E foi nesse mesmo livro, do qual destaquei o trecho que abre

esta seção, que seus autores, DeGrace e Stahl, batizaram essa nova forma de trabalhar de "Scrum".

O livro, na realidade, não oferece um método minimamente detalhado nem utilizável de como colocar essas ideias em prática. Coube a Sutherland e Schwaber criarem o framework.

DeGrace e Stahl explicam por que o modelo em cascata (descrito em *Scrum é ágil*, no capítulo *O que é Scrum*) não funciona para o desenvolvimento de software, e oferecem possíveis alternativas. Scrum aparece como uma das alternativas.

O livro gira em torno da ideia de que software é um tipo de problema que não pode ser claramente definido nem detalhado antes que ofereçamos soluções a ele (um *wicked problem*). Em outras palavras, cada tentativa de criarmos uma solução modifica a própria definição do problema.

Para os meus alunos, eu tento demonstrar esse conceito perguntando o seguinte: *o que quase certamente acontece quando colocamos uma nova funcionalidade diante de um usuário?* Ele provavelmente responde algo como: *agora que estou vendo isso funcionando, vejo que não deveria ser assim... que não preciso disso... que preciso daquilo...* etc. Ou, no mínimo, o usuário se comporta de um modo bem diferente do que esperávamos. Assim, não é possível descrever antecipadamente, em detalhes, as funcionalidades que o software deverá ter para resolver os problemas propostos. Como eu já trouxe no capítulo *Por que Scrum*, a definição do produto emerge ao longo do seu desenvolvimento e uso. Ou, como sempre gosto de dizer, **o uso define o produto**.

Jeff Sutherland faz referência ao livro *Wicked problems, righteous solutions* em pelo menos dois artigos escritos por ele nos primórdios da história do Scrum. Ele destaca essa obra como de grande influência na introdução do Scrum na Easel Corporation.

Em sua própria interpretação, Jeff busca resumir as conclusões do livro sobre as razões por que métodos tradicionais de desenvolvimento de software não funcionam:

- requisitos não são completamente compreendidos antes do início do projeto;
- usuários só sabem exatamente o que querem após ver uma versão inicial do produto;
- requisitos mudam frequentemente durante o processo de construção do software;
- novas ferramentas e tecnologias tornam as estratégias de desenvolvimento imprevisíveis.

Jeff ainda destaca o modelo *Tudo-de-uma-vez* (*All-at-once*) para descrever o Scrum como uma abordagem em que o time realiza simultaneamente todo o trabalho necessário para implementar partes do produto funcionando de ponta a ponta.

4.4 Lean e Sistema Toyota

> *Scrum é embasado no empirismo e em lean thinking.*
> Guia do Scrum (SCHWABER; SUTERLAND, 2020).

O que é o Lean?

O Sistema Toyota de Produção (STP) foi criado nos anos 1950 pela empresa japonesa Toyota Motor Corporation, com a ambiciosa missão de competir com as gigantes Ford e General Electric (GE), dos Estados Unidos. Essas empresas dominavam o mercado mundial, produzindo automóveis da forma mais barata possível, utilizando-se da produção em massa.

No cenário do Japão no pós-guerra, com uma economia devastada e um mercado interno muito limitado, a Toyota necessitava de alternativas. A empresa concluiu que deveria produzir pequenos volumes de diferentes modelos, utilizando a mesma linha de montagem (LIKER, 2003; WOMACK et al., 1992).

Nos anos 1990, os autores norte-americanos James P. Womack e Daniel T. Jones divulgaram, a partir de seus livros, o Sistema Toyota de Produção para o mundo. O sistema japonês foi generalizado e rebatizado para *Lean Production* (ou Produção Enxuta), com a justificativa de que nele usamos menores quantidades de tudo em comparação com a produção em massa. O termo *Lean Thinking* (ou Mentalidade Enxuta) também foi cunhado para estender os conceitos da manufatura para os negócios de forma geral (WOMACK; JONES, 1998; WOMACK et al., 1992).

Tomo a liberdade de chamar, nas seções a seguir, tanto o Sistema Toyota de Produção quanto o *Lean Production* e o *Lean Thinking* apenas de "Lean", exceto quando pretendo fazer alguma distinção entre eles.

Os propósitos do Ágil e do Lean são diferentes. Enquanto o Lean foi originalmente concebido para o trabalho industrial, o Ágil foi criado no contexto do desenvolvimento de software e depois estendido para outras áreas de trabalho criativo. No entanto, Lean é

apontado por diversos autores como fonte de inspiração para uma parte significativa das ideias do Ágil e do Scrum (COCKBURN, 2007; HIGHSMITH, 2004). Como vamos ver adiante, em *Lean e Ágil*, há de fato vários pontos em comum.

Os princípios do Lean

Os cinco princípios do Lean são:

- **definir valor** — especificar o que significa valor na perspectiva de seus clientes, e não do produtor. Ou seja, o produto é criado para atender às necessidades dos clientes;
- **identificar a cadeia de valor** — para cada produto (ou família de produtos), identificar todos os passos que são realizados para que possamos oferecer aos clientes o produto final. Os passos que não geram valor para os clientes e que não são identificados como necessários constituem desperdício e serão eliminados;
- **criar fluxo** — os passos que geram valor fluem continuamente, sem interrupções nem fronteiras entre departamentos, ou entre a organização e fornecedores;
- **estabelecer a produção "puxada"** — os produtos finais são gerados a partir das necessidades dos clientes, e entregues quando eles necessitam. A produção, assim, não é "empurrada" pelo produtor, mas sim "puxada" pelos clientes. Cada passo da cadeia de valor "puxa" do passo anterior apenas o que é necessário e quando é necessário;
- **buscar a perfeição** — a organização busca sempre a perfeição, eliminando os desperdícios, aprendendo

e melhorando continuamente seus processos.

Lean e Ágil

Para ajudar o leitor a compreender as influências do Lean no Ágil, traço a seguir um paralelo entre os princípios de ambos, além de destacar como o Scrum trata deles.

Definir valor

No Lean, o cliente é tratado como parte integral do processo de produção. O valor é definido pela perspectiva desse cliente, de forma a atender às suas necessidades. O produto que ele deseja é produzido e entregue quando ele necessita.

O Valor Ágil *colaboração com o cliente mais do que negociação de contratos* e o Princípio Ágil *as pessoas de negócio e os desenvolvedores devem trabalhar em conjunto diariamente ao longo do projeto* estabelecem que a contribuição dos clientes na produção de valor é essencial para o sucesso do trabalho.

No Scrum, a Product Owner é a principal responsável por interagir frequentemente com os clientes e usuários e definir, de forma incremental, um produto que lhes signifique valor. A Product Owner ordena o trabalho a partir das funcionalidades que ela acredita que gerarão o maior valor para esses clientes e usuários. Entregas frequentes são realizadas, feedback é obtido em cada ciclo de desenvolvimento e em cada entrega, e o produto é ajustado de acordo.

Identificar a cadeia de valor

Os passos na produção que não geram valor para os clientes constituem desperdício — chamados de *muda*, em japonês — e devemos buscar eliminá-los. Evitar o

desperdício, uma das principais preocupações do Lean, permeia praticamente todos os Valores e Princípios Ágeis.

Em particular, o Princípio Ágil *simplicidade — a arte de se maximizar a quantidade de trabalho não feito — é essencial* defende a redução do desperdício pela aplicação da simplicidade, ou seja, a geração, seja como acessório ou como produto do trabalho, de somente o que é realmente necessário e suficiente.

Podemos definir a cadeia de valor no Scrum como todas as atividades realizadas para que possamos oferecer aos clientes um ou mais Incrementos do produto prontos ao final de um ciclo de desenvolvimento (Sprint), de acordo com a Definição de Pronto. Entre essas atividades, aquelas que não agregam valor ao produto são consideradas desperdício.

No entanto, ainda assim podem ser necessárias ou inevitáveis. Essas atividades não podem ser eliminadas, ao menos naquele momento. São chamadas de *muda* tipo 1 no Lean. Documentações burocráticas, exigidas por contrato, podem ser exemplos desse último tipo de desperdício. Aquelas que são desperdício e são desnecessárias devem ser eliminadas imediatamente. Essas são chamadas de *muda* tipo 2 no Lean.

O time trabalha sempre a partir dos itens de maior ordem em cada momento, mais importantes para a realização de um objetivo. Assim, ao chegarem ao final de um prazo dado — uma data de entrega ou mesmo o final de um Sprint — os itens de maior ordem estarão prontos, eventualmente restando apenas os menos importantes para o objetivo. A ordem desses itens é continuamente revista e atualizada a partir do feedback dos clientes e usuários sobre o que foi produzido, garantindo que a implementação de itens de pouca

importância para o objetivo seja postergada e, muitas vezes, não realizada, o que ajuda a evitar o desperdício.

Nas práticas ágeis, a busca pela maximização da comunicação dentro do time e entre os membros do time e os clientes, por meio das entregas frequentes e das reuniões programadas, pretende propiciar inspeções frequentes dos processos de trabalho e do produto em construção. Assim, com o feedback obtido por essas inspeções, buscamos permitir a adaptação dos processos e do produto com as melhorias, que dessa forma reduzem o desperdício. Outra questão é a valorização dos indivíduos e de suas interações mais do que processos e ferramentas, pois o foco no uso excessivo de ferramentas e de processos complexos gera desperdícios.

Já o Princípio Ágil *software* (ou produto) *em funcionamento é a principal medida de progresso* é uma resposta a uma das grandes fontes de desperdício no desenvolvimento de um produto, que é a geração de funcionalidades ou artefatos (como documentos, planos e relatórios) que não criam qualquer valor para os clientes.

Podemos afirmar também que o Princípio Ágil *a atenção contínua à excelência técnica e a um bom desenho aumentam a agilidade* pretende reduzir o desperdício buscando evitar produtos mal projetados e realizados por indivíduos sem o conhecimento e as habilidades necessários.

Criar fluxo e estabelecer a produção "puxada"

Lean define que os passos necessários para a geração de valor devem fluir continuamente, um após o outro, sem paradas, filas, acúmulos em lotes ou fronteiras entre

departamentos. Visando a entregar para os clientes o que eles precisam e quando precisam, a produção é "puxada" a partir de suas necessidades, e daí cada passo "puxa" o anterior na cadeia de valor. É o chamado ***just-in-time*** ou JIT.

Diferentemente do sistema em lote, em que cada passo do processo produz um lote de itens e os "empurra" para o seguinte, no JIT cada passo produz apenas o que é necessário para suprir o que foi requisitado pelo seguinte, reduzindo assim os estoques.

De forma semelhante, no desenvolvimento de um produto com Scrum, Incrementos do produto prontos são implementados por times multidisciplinares em iterações curtas, uma após a outra, que objetivam entregas frequentes. Os Incrementos são entregues para suprir as necessidades de negócios mais importantes dos clientes em cada momento, e a descoberta do que produzir é um trabalho contínuo de interação do Product Owner e Desenvolvedores do produto com esses clientes e seus usuários.

Kanban (palavra que significa "sinalização" em japonês) é um sistema do Lean usado para auxiliar na criação do fluxo utilizando-se da visibilidade. Os operadores usam sinais visuais no fluxo de produção para determinar que uma ação deva ser tomada, por exemplo, a reposição de uma certa quantidade de peças que servem de entrada em um processo.

Sinais visuais comuns nesse sistema são os cartões de sinalização com diferentes cores (e significados), e até mesmo recipientes vazios, indicando que devem ser preenchidos novamente (GROSS; MCINNIS, 2003). Não devemos confundir *kanban* com Kanban (com "K"

maiúsculo), um método baseado no Lean criado para a gestão do trabalho do conhecimento.

O Sprint Backlog do Scrum é frequentemente representado na forma de um quadro de tarefas. Embora não seja um *kanban*, esse quadro traz alguns de seus aspectos, criando transparência sobre o fluxo de trabalho do time.

Além do *muda* (não adição de valor), duas importantes categorias de desperdício devem ser igualmente evitadas. Estas não foram observadas pelo *Lean Production*, mas são parte integrante do Sistema Toyota de Produção (LIKER, 2003).

A primeira categoria é o *muri*, definido como uma sobrecarga nas pessoas ou equipamentos além dos limites naturais, de forma que pode resultar em problemas de segurança, de saúde, de vida útil reduzida e problemas de qualidade, por exemplo. A outra categoria de desperdício é o *mura*, definido como irregularidades, flutuações ou desbalanços no ritmo de produção. Estes ocorrem porque problemas internos, como gargalos no fluxo de produção, componentes defeituosos ou faltantes ou problemas de manutenção, fazem com que, em alguns momentos, haja mais trabalho do que pessoas ou máquinas podem fazer e, em outros, haja falta de trabalho (HAMPSON, 1999; LIKER, 2003).

Uma interpretação incorreta do Lean o leva a ser utilizado para intensificar o trabalho a um ponto em que o esgotamento do trabalhador pode tornar-se um sério problema, ameaçando todo o sistema de produção. Por essa razão, Lean chega a ser chamado de "gerência pelo estresse".

Nessa interpretação, gestores aumentam a pressão sobre o sistema produtivo além da sua capacidade, reduzindo recursos e prazos ou aumentando a quantidade de trabalho, por exemplo. A ideia é que, dessa forma, o estresse gerado em trabalhadores e máquinas faça com que o sistema se fragmente e, assim, permita a identificação de pontos de melhoria e reestruturação. Pretende-se também forçar o uso da criatividade, da autorregulação e do trabalho em equipe para resolver os problemas decorrentes desse aumento de pressão, possibilitando a realização do mesmo trabalho com menos recursos.

Esse tipo de abordagem leva essas empresas a praticarem os desperdícios *muri* (sobrecarga nas pessoas ou equipamentos além dos limites naturais) e *mura* (desbalanços no ritmo de produção) mencionados anteriormente e, assim, a enfrentarem seus problemas decorrentes.

Para combater esses desperdícios, o Sistema Toyota de Produção traz o conceito de *heijunka* (ou "nivelamento", em japonês), fundamental e essencial para o estabelecimento do fluxo contínuo, mas muitas vezes esquecido pelos defensores do *Lean Production*. O *heijunka* é obtido a partir do balanceamento e nivelamento do fluxo de produção, com ritmos e níveis de produção constantes e sustentáveis, evitando assim a formação de gargalos.

Assim, com a prática do *heijunka*, o *mura* é automaticamente eliminado (LIKER, 2003). O *heijunka* somente pode ser realizado dentro das capacidades normais do processo, dos trabalhadores e das máquinas, evitando assim o *muri* (HAMPSON, 1999).

No desenvolvimento de produtos, é comum a gestão forçar um ritmo acelerado e eventualmente insustentável para o trabalho dos times, principalmente em momentos críticos, como antes de um prazo de entrega. Esse comportamento resulta principalmente em práticas como as horas extras e a redução da qualidade.

É também comum Scrum ser incorretamente utilizado para tal fim, o que pode ser acentuado por algumas características do framework, tais como a alta transparência sobre o trabalho e os ciclos curtos, com sua consequente possibilidade de frequência alta de entregas. Na realidade, a produção realizada em um ritmo constante e sustentável é essencial para o trabalho com Scrum, e é defendida pelo Princípio Ágil: *os processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter indefinidamente um ritmo constante*.

Assim, com a prática do Scrum, entendemos que devemos buscar um balanceamento do fluxo de trabalho similar ao *heijunka*, para evitar os desperdícios *muri* e *mura*. Scrum possui alguns mecanismos que, de forma simplificada, buscam atingir esse balanceamento. Entre eles, podemos destacar:

- o Sprint, que é um ciclo de desenvolvimento que se repete em formato ao longo de todo o trabalho e possui sempre a mesma duração. Essa duração fixa e constante ajuda a estabelecer um ritmo regular para o trabalho dos Desenvolvedores do produto, uma cadência com a qual se acostumam o próprio Time de Scrum, clientes e outras partes interessadas, que entendem com que frequência verão os resultados do trabalho e poderão oferecer feedback;

- o trabalho ordenado, a multidisciplinaridade e a responsabilidade sobre os resultados compartilhada entre os Desenvolvedores do produto. Eles possuem, entre si, todos os conhecimentos e habilidades necessários para o desenvolvimento do produto. Ao mesmo tempo, apesar de esses conhecimentos e habilidades estarem distribuídos de forma distinta entre os diferentes Desenvolvedores — podendo haver, por exemplo, especialistas — a responsabilidade sobre cada tarefa é de todos. Ao trabalhar de forma ordenada, uma Desenvolvedora, em vez de abrir um novo item de trabalho de menor ordem que possua tarefas dentro de suas áreas de conhecimento, busca auxiliar os Desenvolvedores no cumprimento de itens de trabalho de maior ordem, mesmo que fora de suas áreas de conhecimento. Dessa forma, distribuímos melhor esse conhecimento entre os Desenvolvedores e eliminamos os gargalos na produção progressivamente.

Lean traz também o conceito de *jidoka*, que significa "automação com um toque humano" em japonês. O *jidoka* estabelece que os próprios trabalhadores devem realizar o controle de qualidade, paralisando imediatamente a produção caso algum defeito seja encontrado para buscarem a causa raiz, resolvê-la e evitar que o problema volte a acontecer. A qualidade, assim, está embutida no próprio processo de produção.

Da mesma forma, os times multidisciplinares do Scrum são responsáveis pela qualidade do produto. Essa responsabilidade significa que a habilidade de testar ou garantir a qualidade está dentro do próprio time, não existindo times externos de qualidade.

Buscar a perfeição

O *kaizen* ("mudar para melhor", em japonês) é a busca da perfeição por meio de etapas infinitas e frequentes de melhorias contínuas, prática essencial do Lean. Uma das chaves para viabilizarmos a realização do *kaizen* é o *hansei*, que significa "reflexão profunda" em japonês.

O *hansei* é uma autorreflexão implacável e obrigatória, utilizada para identificar e reconhecer os erros cometidos ou melhorias que precisam ser feitas, para então criarmos um plano de ação que coloque as melhorias em prática. O *hansei* parte da crença de que há sempre algo que pode ser melhorado. Seu objetivo nunca é apontar culpados, mas sim tornar os problemas visíveis para ajudar o time a realizar melhorias (LIKER, 2003).

No Ágil, a melhoria contínua é expressa no Princípio Ágil *em intervalos de tempo regulares, a equipe reflete sobre como se tornar mais efetiva e, então, refina e ajusta seu comportamento de acordo*. Um dos importantes mecanismos de melhoria contínua no Scrum são as reuniões de Sprint Retrospective, que são basicamente a aplicação do *hansei*.

Definição: os Cinco Porquês

Para descobrirmos as raízes dos problemas, podemos fazer uso no Lean de uma ferramenta chamada de *Os Cinco Porquês*. Nessa técnica, uma vez que um problema ocorre, perguntamos "por quê?" iterativamente cinco vezes, sempre questionando a resposta anterior. Assim, inicialmente perguntamos o porquê do problema e obtemos uma causa imediata. Essa causa é questionada perguntando novamente "por quê?", para que obtenhamos outra causa, que por sua vez é questionada novamente e, assim, sucessivamente até que tenhamos perguntado "por quê?" cinco vezes, quando então

pretendemos ter chegado à raiz do problema. A partir daí, propomos ações de melhoria (LIKER, 2003; OSONO et al., 2008).

Essa técnica pode ser usada por times ágeis, já que descobrir a causa raiz dos problemas, para então resolvê-los, é uma prática essencial do Ágil, na qual buscamos evitar o desperdício e realizar as melhorias contínuas.

4.5 Do trabalho braçal ao trabalho do conhecimento

Taylorismo

Frederick Winslow Taylor foi um inventor e engenheiro norte-americano que viveu entre a segunda metade do século XIX e o início do século XX. Ele, que inicialmente foi um operário, propôs a racionalização do trabalho por meio da aplicação da ciência à administração. Sua proposta central era substituir a execução do trabalho de produção a partir apenas da prática e da experiência, como era feito na época, por técnicas formais e padronizadas, embasadas em estudos. Essas técnicas, ao determinarem a melhor forma que os operários deveriam realizar cada tarefa do seu trabalho, levariam a um aumento de produtividade. Taylor, que se definia em seu cartão de negócios como "consultor para a gestão", possivelmente inventou os termos "consultor", "gerente" e "gestão" no contexto do trabalho (DRUCKER, 1999).

Por suas ideias, Taylor é considerado o pai da Administração Científica. Seu trabalho teve uma importância enorme no desenvolvimento das bases da produção em massa. Somado aos trabalhos de Max

Weber (Teoria da Burocracia) e Henri Fayol (Gestão Administrativa), foi determinante para o aumento de dezenas de vezes na produtividade do trabalhador industrial durante o século XX.

O propósito maior da teoria de Taylor é o aumento da eficiência do trabalho braçal repetitivo. Ou seja, seu foco está na otimização da realização das tarefas para gerar mais resultados. Com esse objetivo, ele propôs o estudo minucioso de tempos e movimentos dos operários na execução de seu trabalho em linhas de produção. A partir da observação, foi possível detectar os movimentos desnecessários e a ociosidade para então reduzir o desperdício e consequentemente os custos, além de simplificar, otimizar e, finalmente, definir técnicas e padrões ótimos para a execução de cada tarefa (TAYLOR, 1911).

Uma das premissas da teoria de Taylor é a especialização funcional, ou seja, o trabalhador executar uma e apenas uma etapa do processo de produção, visando a fazê-lo de forma ótima.

Taylor também acreditava que o trabalhador não gosta naturalmente do seu trabalho e é motivado principalmente pelo dinheiro. Assim, uma vez apropriadamente treinado, ele deveria ser recompensado diretamente por quanto ele produz, reforçando ainda mais a especialização. Outra consequência desse pensamento é a noção de que o trabalhador deveria ser supervisionado e controlado de perto, para discipliná-lo e garantir que o trabalho está sendo executado como foi determinado. Taylor estabelece, dessa forma, o papel do gerente. Essa clara separação entre o trabalho de pensar (definir e planejar) e o trabalho braçal (executar) traz ao gerente a responsabilidade de saber mais sobre o trabalho que o próprio trabalhador.

O Taylorismo, como é chamada a aplicação dessas ideias, constitui ainda hoje parte importante das bases da mentalidade e do funcionamento da maioria das organizações, mesmo aquelas cuja natureza nada tem a ver com o trabalho repetitivo braçal.

A crise de produtividade

> *A contribuição mais importante que a gestão precisa fazer no século XXI é (...) aumentar a produtividade do trabalho do conhecimento e do trabalhador do conhecimento* (DRUCKER, 1999).

Como consequência das ideias de Taylor e seus desdobramentos, houve um crescimento espetacular na produtividade mundial ao longo do século XX. No entanto, apesar do surgimento exponencial de novas tecnologias, esse crescimento caiu vertiginosamente, década após década, a partir dos anos oitenta. Essa desaceleração pode significar que gerações seguintes perderão qualidade de vida frente a seus pais (MORIEUX, 2015).

Yves Morieux, executivo do Boston Consulting Group, afirma em entrevista que uma das principais razões para essa diminuição drástica no crescimento está no uso de métodos tradicionais de gestão - em destaque, o Taylorismo - a um contexto em que não mais se aplicam, levando à geração de um enorme desperdício (MORIEUX, 2017).

As hierarquias e estruturas complicadas, os processos burocráticos, os controles e as métricas consequentes simplesmente são incompatíveis com um tipo de

trabalho cada vez mais presente nos dias de hoje, que chamamos de trabalho do conhecimento.

Trabalhadores do conhecimento

Foi no livro *Landmarks of Tomorrow*, de 1959, que Peter Drucker utilizou o termo "trabalho do conhecimento" pela primeira vez. Drucker, considerado por muitos o pai da Administração Moderna, considerava ultrapassada a gestão tradicional de pessoas baseada nas ideias de Frederick Taylor, além de inadequada para os "trabalhadores do conhecimento", uma vez que seus princípios, regras, práticas e procedimentos foram criados originalmente para trabalhadores braçais pouco ou nada qualificados (DRUCKER, 1959).

Nesse mesmo livro, Drucker também critica a organização das empresas por especialização funcional, ou seja, quando áreas ou departamentos são divididos de acordo com o tipo de trabalho que realizam. Ao criticar a divisão funcional, Drucker afirma que "nunca tentamos organizar pessoas de diferentes habilidades e diferentes conhecimentos, fazendo diferentes contribuições, em um esforço conjunto", ou seja, em times multifuncionais.

Quem são, então, os trabalhadores do conhecimento? Diferentes de trabalhadores manuais, que produzem bens, eles utilizam e criam conhecimento para a realização de seu trabalho, que geralmente envolve a resolução de problemas e o uso de pensamento criativo. Eles sabem mais sobre seu trabalho do que quaisquer outras pessoas na sua organização, inclusive seus chefes. Inovar e aprender continuamente é parte fundamental do seu dia a dia, assim como ensinar.

Exemplos de trabalhadores do conhecimento incluem profissionais de tecnologia em geral, médicos,

pesquisadores, educadores, advogados, juízes, arquitetos, engenheiros, diplomatas, entre tantos outros.

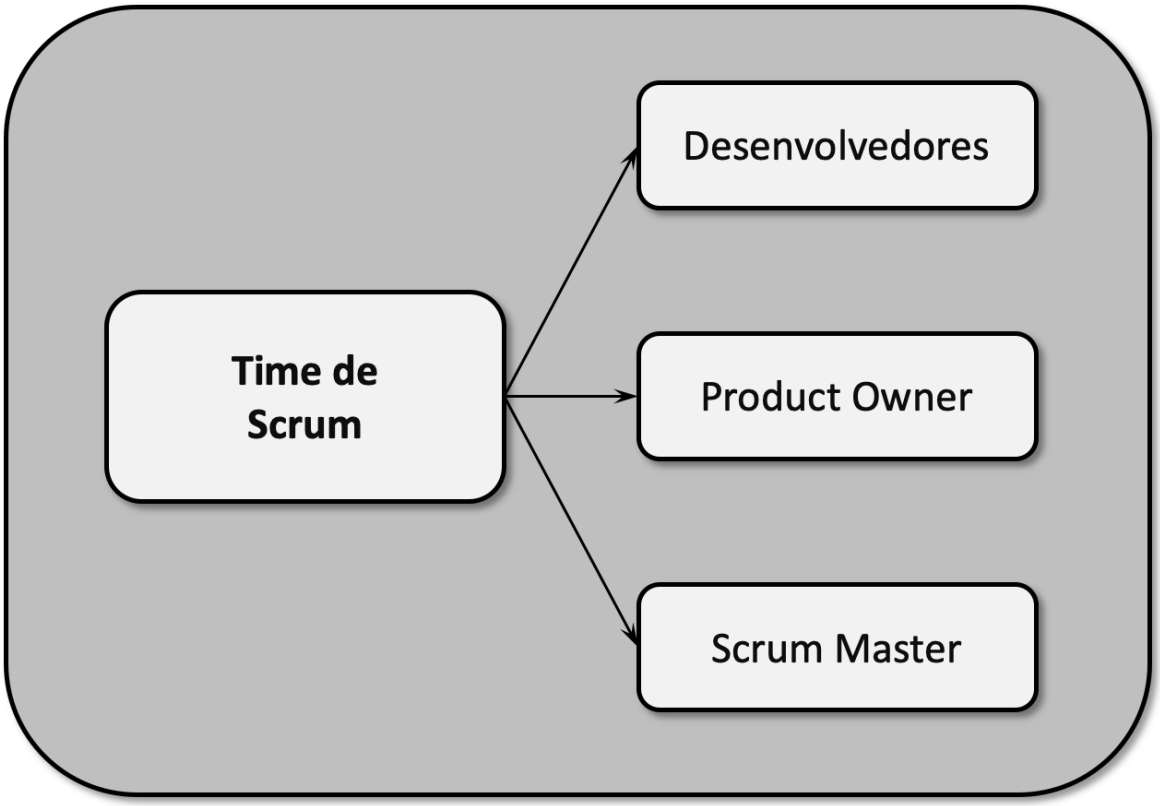
Os trabalhadores do conhecimento funcionam muito mais como associados do que como subordinados, apesar de necessitarem de direção. Ao possuírem esse alto grau de autonomia, eles não devem ser gerenciados, mas sim liderados. Eles também são responsáveis por sua própria produtividade, que primariamente é mais uma questão de qualidade e resultado do que de tarefas realizadas ou de saídas produzidas. São donos de seus meios de produção - seu conhecimento - e por essa razão possuem alta mobilidade. Como consequência, são vistos como ativos, e não como custos pela organização.

O trabalhador do conhecimento é motivado, acima de tudo, por desafios, por objetivos maiores em que acredita e por resultados não financeiros (DRUCKER, 1999).

Scrum se coloca em oposição à aplicação das ideias e práticas do Taylorismo para o trabalho do conhecimento. O framework foi criado, portanto, para esses trabalhadores do século XXI: os trabalhadores do conhecimento.

O nome da minha empresa, originalmente "Knowledge21", evoca justamente as definições de Drucker.

Time de Scrum



Capítulo 5
Sobre o Time de Scrum

Conteúdo

1. O que é o Time de Scrum?
    - Definição.
2. O que faz o Time de Scrum?
3. Como é o Time de Scrum?
    - Multidisciplinar.
        - Polinização cruzada.
        - Aprendizado.
    - Autogerenciado.
    - Pequeno.
    - Orientado a feedback.

4. Papéis externos.

5.1 O que é o Time de Scrum?

Definição

O Time de Scrum é a unidade fundamental do Scrum. É um time pequeno de pessoas, formado por um e apenas um Scrum Master, um e apenas um Product Owner e um número de Desenvolvedores. Somente essas três responsabilidades são definidas em um Time de Scrum.

> **O QUE É UM TIME?**
>
> Um time ou uma equipe é um grupo de pessoas que trabalham juntas e colaboram em busca de objetivos comuns.

O Time de Scrum, ou seja, Desenvolvedores, Product Owner e Scrum Master, enquanto membros de um mesmo time, colaboram lado a lado, em seu dia a dia de trabalho, em busca do objetivo comum de obterem o sucesso no desenvolvimento do produto a partir da geração de valor para seus clientes e usuários. **Não há hierarquias dentro do Time de Scrum.** São todos igualmente responsáveis e responsabilizados pelos resultados do trabalho.

O Time de Scrum é:

- multidisciplinar, de forma a possuir entre seus membros todos os conhecimentos e habilidades necessários para realizar o trabalho de desenvolvimento do produto, de ponta a ponta;

- autogerenciado, pois seus membros decidem o que será feito, quem fará o quê, como será feito e quando será feito, e executam o trabalho com autonomia, propriedade e responsabilidade pelos resultados;
- um conjunto suficientemente pequeno, de forma que seus membros se comuniquem efetivamente entre eles, se autogerenciem e sejam capazes de produzir valor frequentemente para os seus clientes e usuários.

Curiosidade: galinhas e porcos

Uma galinha e um porco estão juntos quando a galinha diz: "Vamos abrir um restaurante!"

O porco reflete e então diz: "Como seria o nome desse restaurante?"

A galinha diz: "Presunto com Ovos!"

O porco diz: "Não, obrigado, eu estaria comprometido, mas você estaria apenas envolvida!"

A fábula da galinha e do porco esteve presente desde o primeiro livro publicado sobre Scrum (SCHWABER; BEEDLE, 2002) e seguiu nas edições do guia oficial do Scrum de 2009 e 2010 (SCHWABER, 2009; SCHWABER; SUTHERLAND, 2010). Era, assim, parte da definição oficial do Scrum.

Ela foi utilizada para explicar que os membros do Time de Scrum estão realmente comprometidos, e por essa razão são chamados de "porcos". O Product Owner é o "porco" do Product Backlog. Os Desenvolvedores são os "porcos" do trabalho no Sprint. O Scrum Master é o

"porco" do processo do Scrum. Todos os outros são chamados de "galinhas", porque estão apenas envolvidos. A principal mensagem é que "galinhas" não podem dizer a "porcos" como eles devem fazer seu trabalho.

A fábula do porco e da galinha ainda existe no folclore do Scrum e já esteve muito presente na linguagem dos seus praticantes, mas foi perdendo força desde que foi retirada da edição do guia de 2011 (SCHWABER; SUTHERLAND, 2011) e, portanto, da definição do framework. Segundo ouvimos na época, os autores observaram que a fábula estava sendo usada de forma incorreta, potencialmente aumentando as barreiras entre o Time de Scrum e outras partes interessadas ao invés de ajudar no entendimento das responsabilidades. Além disso, muitos consideravam depreciativos os termos "porco" e "galinha".

5.2 O que faz o Time de Scrum?

O Time de Scrum é integralmente responsável pelas atividades necessárias para o desenvolvimento do produto. Essas atividades incluem a colaboração com clientes e demais partes interessadas, verificação, manutenção, operação, experimentação, pesquisa e o próprio trabalho de desenvolvimento, além de quaisquer outras atividades que se façam necessárias (SCHWABER; SUTHERLAND, 2020).

No Time de Scrum, os Desenvolvedores são os responsáveis por realizar o trabalho de desenvolvimento do produto de ponta a ponta, realizando em cada Sprint um objetivo de valor definido junto ao Product Owner a

partir da implementação de um ou mais Incrementos do produto.

O Product Owner, como parte do Time de Scrum, é o responsável por garantir e maximizar o valor do produto gerado a partir do trabalho realizado pelos membros do Time de Scrum, de forma a satisfazer as necessidades de clientes e usuários.

O Scrum Master é o responsável por promover e apoiar o entendimento e o uso do Scrum, por facilitar e potencializar o trabalho do Time de Scrum como um todo e por estimulá-lo a melhorar continuamente seus processos e práticas.

O Scrum não proíbe que uma mesma pessoa atue em mais de uma responsabilidade, embora eu recomende evitar essa prática sempre que possível. O que eu desaconselho fortemente é que a mesma pessoa exerça as responsabilidades de Scrum Master e Product Owner no mesmo Time de Scrum (veja a seção *Presente*, no capítulo *Scrum Master*).

5.3 Como é o Time de Scrum?

Multidisciplinar

O Time de Scrum forma um conjunto multidisciplinar ou multifuncional de pessoas, que, juntas, possuem todos os conhecimentos e habilidades necessários para realizar o trabalho de desenvolvimento do produto, de ponta a ponta. Assim, seus membros, em conjunto, são capazes de executar quaisquer tipos de trabalho que se façam necessários para a implementação das funcionalidades

do produto, criando um ou mais Incrementos entregáveis em cada Sprint (veja o capítulo *Incremento*).

Esse trabalho provavelmente inclui, além de todas as tarefas da construção do produto propriamente dita, o uso de métodos de garantia de qualidade, a criação de um ou mais tipos de documentação, a definição e o entendimento de detalhes de negócios, a ordenação dos itens de trabalho para gerar o próximo maior valor e a própria aplicação do framework Scrum e de diferentes técnicas ágeis, entre outros.

O Product Owner continuamente toma as decisões sobre a definição do produto, o que inclui ordenar o que será implementado. Ele possui, portanto, os conhecimentos e habilidades para tal. O Scrum Master tem as responsabilidades de facilitar as interações entre os membros do Time de Scrum, de ensinar Scrum a quem for pertinente e de atuar como um agente de mudanças, entre outras. Deve, portanto, possuir os conhecimentos e habilidades para fazê-lo. Entre os Desenvolvedores de um Time de Scrum, não há distinção de responsabilidades. Em conjunto, eles têm a responsabilidade de implementar um ou mais Incrementos prontos, de acordo com a Definição de Pronto, a partir do que foi estabelecido juntamente com o Product Owner para cada Sprint (veja o capítulo *Compromisso: Definição de Pronto*). Os Desenvolvedores, portanto, possuem em conjunto os conhecimentos e habilidades necessários para realizar esse trabalho.

Para fazer parte de um Incremento, cada item do Sprint Backlog deve estar pronto no Sprint, de acordo com a Definição de Pronto. Por essa razão, a Definição de Pronto é determinante para a definição do trabalho a ser realizado e, assim, está diretamente ligada aos

conhecimentos e habilidades que possuem os membros do Time de Scrum.

Repare que o tamanho recomendado para o Time de Scrum é pequeno, com dez membros ou menos (veja a seção *Pequeno*, neste capítulo). Ainda assim, todos os conhecimentos e habilidades necessários para desenvolver o produto existem entre os membros desse time.

A multidisciplinaridade do Time de Scrum, no entanto, não implica na multidisciplinaridade do indivíduo. Essa definição fica clara quando tratamos do Product Owner e do Scrum Master, que possuem responsabilidades bem definidas. Mas o mesmo não ocorre entre os diferentes Desenvolvedores, que dividem o trabalho com igual responsabilidade. O Scrum, no entanto, não exige que cada Desenvolvedor possua todos os conhecimentos e habilidades necessários para desenvolver o produto - isso seria utópico - mas sim que eles os possuam em conjunto. É natural que as pessoas possuam suas especialidades, conhecimentos e áreas de interesse específicos.

Polinização cruzada

O trabalho com Scrum traz a responsabilidade sobre a realização do Objetivo do Sprint para o conjunto de Desenvolvedores, e não para determinados indivíduos. Dessa forma, Desenvolvedores são constantemente estimulados a trabalhar juntos, trocar conhecimentos, buscar aprender o que se faça necessário e, assim, desenvolver habilidades secundárias.

Essa "polinização cruzada" reduz a dependência nas especialidades de determinados Desenvolvedores, diminuindo o risco de filas e esperas ameaçarem a

realização do Objetivo do Sprint. A ausência ou a iminente sobrecarga de um indivíduo em um determinado momento, por exemplo, tem menos chances de se tornar impedimento para o trabalho do conjunto, pois normalmente haverá outros ao menos minimamente capazes de realizar o trabalho ou de ajudar na sua realização.

O compartilhamento de conhecimentos também reduz as chances de que, em alguns momentos, Desenvolvedores fiquem sem ter o que fazer e, assim, procurem trabalhar em itens de menor ordem dentro de sua zona de conforto, ou até mesmo se vejam obrigados a trabalhar em mais de um time ao mesmo tempo. Um especialista que só saiba lidar com controle de qualidade, por exemplo, enfrentaria esse problema, já que só teria trabalho em momentos específicos do Sprint.

A partir da polinização cruzada, é comum também os Desenvolvedores aprenderem técnicas de facilitação com o Scrum Master, e habilidades de negócio e conhecimento sobre o produto com o Product Owner. É igualmente comum tanto o Scrum Master quanto o Product Owner adquirirem algum conhecimento técnico, ainda que básico, em sua interação com os Desenvolvedores.

Aprendizado

De forma geral, esperamos que uma parte significativa dos conhecimentos e habilidades necessários já exista entre os membros do Time de Scrum desde o momento de sua escolha ou da formação do time. Ou seja, buscamos atribuir um novo produto a ser desenvolvido a um Time de Scrum que seja capaz de realizar o trabalho. No entanto, novos desafios que exigem novos aprendizados surgem naturalmente ao longo do trabalho,

e isso ocorre tanto para os Desenvolvedores quanto para o Product Owner e para o Scrum Master.

A forma como conhecimentos e habilidades são adquiridos e compartilhados entre os membros do Time de Scrum varia muito de time para time. Podemos, no entanto, classificar as atividades que promovem esse aprendizado em formais e informais.

Muitas organizações contratam treinamentos formais para seus colaboradores, mas há geralmente um orçamento limitado para esse fim. É comum, por outro lado, os próprios colaboradores da organização promoverem, periódica ou eventualmente, sessões de treinamento para seus colegas. Estas sessões ocorrem geralmente em torno de assuntos em que esses indivíduos são especialistas ou sobre os quais aprenderam recentemente, seja por conta própria ou por meio de treinamentos contratados (ARMONY, 2010).

Típico entre Desenvolvedores de software, o *Coding Dojo* é outro método formal de treinamento técnico para esse tipo de trabalho. Por seu aspecto lúdico, a prática do dojo ganha ares informais, embora seja uma forma extremamente disciplinada e efetiva de cultivarmos habilidades para o desenvolvimento de software.

Coding Dojo

Coding Dojo é um encontro em que um grupo de desenvolvedores de software se reúne para trabalhar em um desafio de programação. Eles estão lá para se divertir e se engajar em atividades direcionadas a melhorar suas habilidades, proporcionando feedback (tanto sobre o código produzido quanto sobre as técnicas usadas), que fomenta o aprendizado.

Para mais informações, acesse: http://codingdojo.org/.

O treinamento informal se dá a partir do compartilhamento de conhecimento com colegas de trabalho ou oriundo de outras pessoas de fora do time ou da organização. Essa aquisição de conhecimentos pode acontecer no dia a dia de trabalho ou, por exemplo, com a participação em feiras e eventos. Por meio da "polinização cruzada", que mencionei anteriormente, tanto os conhecimentos e habilidades recém-adquiridos quanto os já existentes são disseminados entre os diferentes membros do Time de Scrum, em especial entre os Desenvolvedores.

Oriunda do Extreme Programming, a prática da programação em par tem se mostrado uma forma muito efetiva de treinamento informal no trabalho de desenvolvimento de software. A prática equivalente do trabalho em par pode ser estendida a diversas outras áreas. Entre diversos benefícios, ao se colocarem juntos sobre uma mesma tarefa alguém menos ou não capacitado em uma determinada matéria e outro mais capacitado, o segundo ensina ao primeiro enquanto ambos executam o trabalho. Outro benefício é a maior qualidade no produto gerado a partir da construção

conjunta. Essa prática pode acontecer em tempo integral, de forma programada durante algumas horas por dia ou eventualmente.

Autogerenciado

Scrum foi concebido de forma a estimular a autonomia de seus times. Ao contrário do que esperamos de times no trabalho tradicional, não há gerentes ou qualquer outro tipo de agente externo dizendo para os membros do Time de Scrum o que será feito, quem fará o quê, como será feito e quando será feito. O Time de Scrum é autogerenciado, o que significa que seus membros decidem o que for necessário para entregarem valor. No entanto, seus membros estão inseridos no contexto de uma organização, assim essa autogestão está alinhada com as regras e objetivos dessa organização, inclusive no que diz respeito ao uso do framework Scrum, e visa a gerar valor para seus clientes.

O Product Owner decide, em colaboração com os Desenvolvedores, o que será construído e quando será construído. Os Desenvolvedores decidem quem fará cada parte do trabalho e como o farão, e não há ninguém pressionando nem cobrando informações sobre o andamento de cada tarefa. Assim, com Scrum, o microgerenciamento dá lugar ao autogerenciamento.

Para facilitar esse autogerenciamento, Scrum estimula a transparência, oferecendo pontos-chaves nos quais o andamento do trabalho ou seus resultados são inspecionados. Um exemplo são as reuniões de Daily Scrum, nas quais os próprios Desenvolvedores inspecionam seu andamento na realização do Objetivo do Sprint; outro são as reuniões de Sprint Review, em que os resultados do trabalho realizado no Sprint recebem feedback de clientes e partes interessadas.

Com Scrum, também utilizamos ferramentas que aumentam a transparência sobre o andamento do trabalho, como o Sprint Backlog e gráficos que podem indicar o progresso ou o trabalho restante (veja o capítulo *Burndown e Burnup: acompanhando o trabalho*).

Scrum oferece objetivos claros a serem realizados no curto prazo, traduzidos em cada Sprint na forma do Objetivo do Sprint, que por sua vez é alinhado a um objetivo de mais longo prazo, o Objetivo do Produto. Esses objetivos comuns entre Product Owner e os Desenvolvedores fornecem o alinhamento necessário para o autogerenciamento acontecer. A responsabilidade compartilhada dos Desenvolvedores sobre esses objetivos, em oposição à responsabilidade individual sobre as suas partes, é outro elemento essencial, pois estimula a preocupação com o todo. No Time de Scrum, não há o "meu" e o "seu", mas sim o "nosso".

A presença tanto dos Desenvolvedores quanto do Product Owner na reunião de Sprint Review reforça o senso de responsabilidade sobre os resultados de seu trabalho, já que lá estarão frente a frente com clientes e partes interessadas para os quais o trabalho está sendo realizado. Além disso, como mencionei anteriormente, não há hierarquias entre os membros do Time de Scrum, não há subtimes e nem qualquer tipo de liderança imposta entre os Desenvolvedores. Caso existissem, levariam a cargas diferenciadas de responsabilidade e prejudicariam o autogerenciamento.

Para que essa maior autonomia se torne possível, é imprescindível que os Desenvolvedores realizem o trabalho de forma colaborativa entre eles e com o Product Owner, maximizando a transparência e a comunicação. Essa é uma das grandes justificativas para o Time de Scrum ser pequeno em número de membros.

O Time de Scrum conta ainda com o apoio de um facilitador, o Scrum Master, para estimular os seus membros a criarem transparência e a tomarem responsabilidade sobre seus problemas e sobre suas soluções, aumentando sua autonomia. O Scrum Master ainda, entre outras coisas, remove impedimentos que atrapalhem o seu trabalho, promove mudanças organizacionais necessárias para empoderá-los, ensina o uso de Scrum e facilita seu acesso aos meios necessários para realizar esse trabalho.

Definição: elementos para o autogerenciamento

Scrum oferece alguns elementos que considero essenciais para o autogerenciamento, que são os seguintes:

- a definição de objetivos claros de valor e realizáveis no curto prazo (o Objetivo de cada Sprint), alinhados a objetivos de mais longo prazo (como o Objetivo do Produto ou outros objetivos intermediários);
- responsabilidade compartilhada sobre esses objetivos entre os membros do Time de Scrum;
- alta transparência no trabalho em si e em seu progresso real, a partir da geração de partes prontas do produto e da exposição a feedback;
- times pequenos, que realizam o trabalho em conjunto e se comunicam de forma efetiva;
- a presença e atuação de um facilitador (o Scrum Master), que tem como um de seus objetivos primários estimular a autonomia do time.

Pequeno

O Time de Scrum é suficientemente pequeno para que se mantenha ágil. Ao mesmo tempo, ele é suficientemente grande para conseguir gerar algo útil e de valor em cada Sprint.

A recomendação oficial do Scrum é que o tamanho do Time de Scrum seja de dez ou menos membros, mas que o tamanho ótimo pode ser bem menor (SCHWABER; SUTHERLAND, 2020). Esses números incluem o Scrum Master e o Product Owner, que podem também exercer a responsabilidade de Desenvolvedor.

Na realidade, ao considerarmos o número de membros de um Time de Scrum, esperamos que:

- os seus membros sejam capazes de se comunicarem efetivamente para se autogerenciarem. O número de canais de comunicação cresce exponencialmente com o número de pessoas envolvidas. Assim, times grandes têm maiores dificuldades de comunicação e coordenação, o que dificulta e até pode impossibilitar o autogerenciamento;
- os seus membros possuam, em conjunto, todos os conhecimentos e habilidades necessários para produzirem um ou mais Incrementos do produto prontos, de acordo com a Definição de Pronto, em cada Sprint. Esse trabalho se traduz em funcionalidades prontas de ponta a ponta, que representam valor para os clientes e usuários, em vez de partes de funcionalidades, camadas ou componentes;
- os seus membros sejam capazes de produzir valor visível suficiente em cada Sprint, para que possam gerar oportunidades de entrega e obter, na reunião

de Sprint Review, feedback dos clientes e demais partes interessadas sobre o que foi produzido.

Quando o produto é muito grande, podemos utilizar, de forma coordenada, mais de um Time de Scrum trabalhando no mesmo produto, compartilhando os mesmos Objetivo do Produto, Product Backlog e Product Owner. Nesse caso, cada um desses times tem seu tamanho definido de acordo com o estabelecido nesta seção.

Curiosidade: mudanças no tamanho recomendado

O artigo original de Scrum, apresentado em 1995 em um congresso (SCHWABER, 1997), afirmava que o time a desenvolver o produto deveria ter entre três e seis membros. Já o primeiro livro publicado sobre Scrum (SCHWABER; BEEDLE, 2002) definia o tamanho desse time em sete mais ou menos dois membros, ou seja, entre cinco e nove membros. O livro trazia como referência um famoso artigo do psicólogo cognitivo George A. Miller, que defendia que o número de objetos que um ser humano médio é capaz de manter na sua memória de curto prazo é de sete mais ou menos dois (MILLER, 1956).

O Guia de Scrum original, de 2009, que definia o Time de Scrum como formado por Scrum Master, Product Owner, e Time de Desenvolvimento, manteve essa recomendação de sete mais ou menos dois membros para que o Time de Desenvolvimento estivesse em seu tamanho ótimo (SCHWABER, 2009).

Em 2011, os criadores do Scrum se deram conta de que times menores podem ser bem-sucedidos. Assim, o número recomendado de membros para o Time de

Desenvolvimento mudou para não menos que três e não mais que nove (SCHWABER; SUTHERLAND, 2011).

Em 2020, o Time de Desenvolvimento deixou de existir e o Time de Scrum passou a ser formado pelos Desenvolvedores, pelo Scrum Master e pelo Product Owner. Dessa forma, a recomendação oficial passou a ser dirigida ao Time de Scrum como um todo e os números mudaram, indicando apenas um limite sugerido de dez membros, conforme descrevi anteriormente (SCHWABER; SUTHERLAND, 2020).

Orientado a feedback

Na Sprint Review, os membros do Time de Scrum interagem com clientes e demais pessoas relevantes sobre os resultados do trabalho realizado no decorrer do Sprint. O principal objetivo da reunião é obter feedback dessas pessoas sobre o Incremento ou Incrementos implementados. Desenvolvedores e Product Owner apresentam e demonstram o que está pronto, de acordo com a Definição de Pronto, estimulam que os demais presentes experimentem as novas funcionalidades, fazem e recebem perguntas, oferecem respostas e tomam notas.

As pessoas externas presentes na reunião são preferencialmente as mais adequadas para prover feedback sobre o que foi produzido no Sprint, como clientes ou usuários diretos dos itens que foram implementados. Dessa forma, o feedback obtido na reunião é valioso, pois serve de matéria-prima para a atualização do Product Backlog para Sprints seguintes, permitindo a adaptação do produto de forma a gerar o próximo maior valor.

Os Desenvolvedores podem também buscar, durante o Sprint, feedback antecipado do próprio Product Owner, de clientes e de outros sobre o que já está pronto.

O feedback mais poderoso, no entanto, vem das métricas do uso do produto que são coletadas pelos Desenvolvedores e definidas em colaboração com o Product Owner. Ao possibilitar que o produto chegue frequentemente aos usuários, tornamos possível medir como esses usuários estão utilizando as partes do produto que já estão em suas mãos. Dessa forma, os Desenvolvedores e o Product Owner entendem como podem modificar e evoluir o produto para melhor atender às necessidades dos usuários, e entendem o que pode e o que não deve valer a pena construir em seguida, e assim alimentam o Product Backlog. Como já afirmei anteriormente, **o uso define o produto**.

5.4 Papéis externos

Ao longo do livro, chamo de **partes interessadas** aquelas pessoas que têm algum interesse relevante em questão no desenvolvimento do produto, como, por exemplo, aquelas que estejam diretamente envolvidas ou contribuindo para esse desenvolvimento e as que sejam diretamente afetadas ou defendam interesses que podem ser afetados por seus resultados. Exemplos podem incluir clientes, patrocinadores, executivos, gestores, departamentos da organização (jurídico e marketing, por exemplo), órgãos reguladores e associações de classes. Além de, claro, o próprio Time de Scrum.

Chamo de **clientes** apenas as pessoas, grupos, sejam internos ou externos à organização, ou organizações que

solicitam ou contratam o desenvolvimento do produto, e recebem de algum modo o valor gerado, uma vez que o que é produzido lhes vai sendo entregue ou a seus usuários. Chamo de **usuários** aqueles que, de fato, recebem e usam o produto. Clientes podem também ser usuários.

Não há, no Scrum, a definição do papel do gerente de projetos. Quando Scrum é utilizado na execução de um projeto, questões como escopo, prazo, qualidade, risco, processos e gestão do trabalho não ficam sob a responsabilidade de um único papel centralizador. Ao contrário, elas são distribuídas entre o Product Owner, os Desenvolvedores e o Scrum Master, cada qual em suas responsabilidades. Assim, não devemos e não é possível mapear o papel do gerente de projetos tradicional para as responsabilidades do Scrum.

Questões funcionais como a formação de equipes, contratações, promoções, demissões, férias, planos de carreira etc. são muito específicas ao contexto de cada organização, assim, observamos diferentes soluções para cada uma delas e não há um tratamento definido pelo framework Scrum. Outras funções gerenciais como geração de contratos, precificação, gestão de portfólio etc. também estão fora do escopo do Scrum.

Capítulo 6
Desenvolvedores

Conteúdo

1. Quem são os Desenvolvedores?
2. O que fazem os Desenvolvedores?
    - Resolvem problemas dos usuários e de negócios.
    - Implementam Incrementos entregáveis prontos.
    - Planejam, coordenam e executam seu trabalho com autonomia.
    - Identificam e informam impedimentos ao Scrum Master.
        - Sinalização de impedimentos.
        - Prevenção de impedimentos.
3. Como são os Desenvolvedores?
    - Motivados.
        - Motivação 3.0.
        - Teoria da Fixação de Objetivos.
        - Teoria das Características do Trabalho.
        - Teoria ERC.
    - Colaborativos.
    - Orientados à excelência.

6.1 Quem são os Desenvolvedores?

Os Desenvolvedores do produto formam um grupo responsável por realizar o trabalho de desenvolvimento de ponta a ponta, com a adição, ajustes e modificação de características e comportamentos desse produto. A partir da ordem dos itens definida pelo Product Owner, os

Desenvolvedores implementam, em cada Sprint, um ou mais Incrementos do produto prontos (veja o capítulo *Incremento*), de acordo com a Definição de Pronto (veja o capítulo *Compromisso: Definição de Pronto*), e que significam valor visível e entregável para os clientes e usuários do produto.

> Lembro que, neste livro, considero **DESENVOLVIMENTO DO PRODUTO** toda e qualquer atividade necessária no trabalho de implementação do produto, de forma a adicionar e modificar funcionalidades, características e comportamentos. Esse é o trabalho dos Desenvolvedores do produto. Não confunda com o trabalho de escrita de código realizado por desenvolvedores de software, que por sua vez pode ser parte do trabalho de desenvolvimento de um produto dependendo do contexto.

Os Desenvolvedores do produto, juntamente do Scrum Master e com o Product Owner, compõem o Time de Scrum. A recomendação oficial do Guia do Scrum é de que o tamanho do Time de Scrum, incluindo os Desenvolvedores, o Scrum Master e o Product Owner, seja de dez ou menos membros, mas que o tamanho ótimo pode ser bem menor (SCHWABER; SUTHERLAND, 2020).

Os Desenvolvedores gerenciam o seu trabalho de desenvolvimento do produto, balizados pelo framework Scrum. São eles que, ainda que sujeitos aos limites dados pelo contexto do cliente e organizacional, determinam tecnicamente como cada Incremento do produto será desenvolvido, planejam esse trabalho e acompanham seu próprio progresso. Para tal, os Desenvolvedores, em conjunto, têm propriedade e

autoridade sobre suas decisões e, ao mesmo tempo, são responsáveis e responsabilizados por seus resultados.

Para realizar esse trabalho, os Desenvolvedores:

- resolvem problemas dos usuários e de negócios ao realizar o desenvolvimento do produto, conforme estabelecidos junto ao Product Owner;
- implementam um ou mais Incrementos do produto entregáveis em cada Sprint, trabalhando juntos de forma sustentável e possibilitando a entrega frequente de partes do produto prontas para os clientes e usuários do produto;
- planejam, coordenam e executam seu trabalho de desenvolvimento do produto com autonomia, alinhados a objetivos definidos junto ao Product Owner;
- identificam e informam ao Scrum Master os impedimentos que obstruem seu trabalho, ou previnem-se deles, quando possível.

Os Desenvolvedores são:

- motivados, uma vez que têm os estímulos adequados e possuem tudo o que for necessário para realizarem seu trabalho;
- colaborativos, interagindo continuamente entre eles e dividindo as responsabilidades durante todo o trabalho; além de buscar a colaboração com o Product Owner, clientes e partes interessadas durante o Sprint, sempre que necessário, para definir, detalhar, obter feedback e ter dúvidas esclarecidas com relação ao que está sendo ou que será implementado;

- orientados à excelência, buscando aprender e melhorar continuamente, realizando seu trabalho com consciência e qualidade.

Curiosidade: Desenvolvedores, Time de Desenvolvimento, Time, time

Os primeiros artigos escritos sobre o Scrum definiam, de forma livre, o grupo responsável por realizar o trabalho de desenvolvimento do produto apenas como "time" (SCHWABER, 1997; BEEDLE ET AL., 1999; SUTHERLAND, 2004). Da mesma forma o fazia o primeiro livro publicado sobre o Scrum, que ainda gerava confusão ao usar o termo "Time de Scrum" com esse mesmo fim (SCHWABER; BEEDLE, 2002).

O artigo original, de 1995, ainda definia que um "Time de Projeto de SCRUM" deveria ser formado para cada nova entrega. Curiosamente, esse time incluiria um ou mais times de desenvolvimento, que continham desenvolvedores de software, documentadores e pessoal de controle de qualidade. Esse "Time de Projeto de SCRUM" também incluiria um time de gerência, contendo pessoas que seriam afetadas pela entrega e liderado por um gerente de produto (SCHWABER, 1997). O formato foi desde cedo abandonado.

O primeiro guia oficial do Scrum, de 2009, já definia o "Time", com inicial maiúscula, como papel do framework (SCHWABER, 2009). O nome "Time de Desenvolvimento", que existiu por quase uma década, foi introduzido apenas na edição de 2011 (SCHWABER; SUTHERLAND, 2011). No guia de 2020, o conceito de "Time de Desenvolvimento" e, portanto, de time dentro do time, deixou de existir, e aqueles que desenvolvem o produto

passaram a ser chamados de Desenvolvedores (SCHWABER; SUTHERLAND, 2020).

6.2 O que fazem os Desenvolvedores?

Resolvem problemas dos usuários e de negócios

Ao apenas cumprir tarefas que lhes são passadas, especializando-se em uma parte do processo de desenvolvimento de um produto, o trabalhador não consegue, a partir desse trabalho fragmentado, identificar que valor está gerando.

Os Desenvolvedores do produto, no entanto, trabalham orientados a resolverem problemas reais dos usuários finais, de forma a gerarem valor de negócio. Em outras palavras, buscam suprir necessidades ou permitir o aproveitamento de oportunidades. Como são diretamente responsáveis por essa geração de valor, os Desenvolvedores buscam obter feedback e entender como os usuários reais estão utilizando as partes do produto que já lhes foram entregues. Juntamente com o Product Owner, eles então ajustam o produto de acordo para melhor resolver os problemas e entendem o que é mais importante a se fazer em seguida.

O Product Owner ordena os itens do Product Backlog visando aos próximos problemas de negócios mais importantes a serem resolvidos, a partir da resolução de problemas dos usuários. Em cada Sprint, os Desenvolvedores e o Product Owner escolhem, a partir dos itens do alto do Product Backlog, quais serão implementados e estabelecem um Objetivo do Sprint que reflete o problema a ser resolvido a partir da

implementação desses itens (veja o capítulo *Compromisso: Objetivo do Sprint*). Os Desenvolvedores colaboram ao longo do Sprint, compartilhando a responsabilidade de realizar esse objetivo comum. Assim, eles se mantêm alinhados e focados em produzir os resultados de valor, e não simplesmente trabalham para cumprir tarefas. Quaisquer tarefas realizadas pelos Desenvolvedores no decorrer do Sprint estão, direta ou indiretamente, alinhadas a esse objetivo. Além disso, cada Objetivo de Sprint está alinhado e realiza parte do objetivo mais amplo, o Objetivo do Produto (veja o capítulo *Compromisso: Objetivo do Produto*).

Idealmente, os Desenvolvedores vão além de apenas se alinhar a definições realizadas pelo Product Owner. A mentalidade de separarmos quem define de quem executa o trabalho, ou quem pensa de quem faz, é infelizmente ainda muito presente nas organizações. Os Times de Scrum mais efetivos, no entanto, são aqueles em que os Desenvolvedores se envolvem e se engajam integralmente em desde o entendimento dos problemas a serem resolvidos até a definição, implementação e validação de soluções, trabalhando lado a lado com o Product Owner, clientes, usuários e demais partes interessadas durante todo o processo.

Implementam Incrementos entregáveis prontos

Os Desenvolvedores do produto fazem parte de um time multidisciplinar, o Time de Scrum. Esse time não trabalha apenas sobre camadas específicas, componentes ou partes de funcionalidades do produto, cenário em que dependeria de outros times ou pessoas para conseguir gerar valor visível para o usuário final. O Time de Scrum possui, entre seus membros, os conhecimentos e

habilidades necessários para realizar todo o trabalho de desenvolvimento do produto, de ponta a ponta.

Dependências externas, que não estão sob o controle dos Desenvolvedores, constituiriam um risco muito alto para o trabalho realizado durante o Sprint. Se fosse necessário esperar pelo trabalho de pessoas ou times externos para poderem realizar a implementação de um ou mais itens, os Desenvolvedores poderiam não conseguir realizar o Objetivo do Sprint. Um conjunto multidisciplinar reduz esses riscos, já que é capaz de realizar todo o trabalho sem, a princípio, depender de ninguém.

Assim, a partir da ordem dos itens definida pelo Product Owner, os Desenvolvedores implementam, em cada ciclo de desenvolvimento, funcionalidades prontas que constituem um ou mais Incrementos prontos. Pronto, de acordo com a Definição de Pronto, significa que já podem ser entregues e utilizados pelos usuários do produto. Dessa forma, os Desenvolvedores possibilitam, em cada ciclo como um mínimo, a entrega para os usuários do produto, ainda que a decisão de quando entregar pertença ao Product Owner.

Ao realizarem esse trabalho, não há papéis ou responsabilidades definidas entre os Desenvolvedores. Embora seja natural e comum cada um ter individualmente habilidades, conhecimentos, interesses ou áreas de foco primários, todos são igualmente responsáveis pelo desenvolvimento do produto e pela realização dos objetivos acordados. Esperamos, portanto, que dirijam seus esforços para onde forem mais úteis em cada momento, na medida do que for prático e possível.

Os Desenvolvedores dedicam integralmente o seu tempo útil ao trabalho planejado para o Sprint. Outras atividades, além daquelas que se fizerem necessárias

para o desenvolvimento do produto, são evitadas, pois desviam o seu foco e prejudicam seus resultados, podendo ameaçar a realização do Objetivo do Sprint.

Planejam, coordenam e executam seu trabalho com autonomia

Os Desenvolvedores formam, em conjunto com o Product Owner e o Scrum Master, um time autogerenciado, o Time de Scrum. Como parte desse time, os Desenvolvedores possuem autonomia para planejar, coordenar e executar o seu trabalho em cada Sprint.

Na reunião de Sprint Planning, os Desenvolvedores colaboram e negociam com o Product Owner para decidirem o que acreditam que será realizado no decorrer do Sprint que se inicia. Ou seja, a partir da ordem dos itens definida pelo Product Owner, são eles que têm a última palavra com relação a quantos itens farão parte do plano do Sprint, o Sprint Backlog (veja o capítulo *Sprint Backlog*).

Para entender sua capacidade de trabalho em cada Sprint e, assim, auxiliá-los com o planejamento, os Desenvolvedores podem usar estimativas para a implementação de cada item do Product Backlog. Essas estimativas, quando usadas, são sempre realizadas pelos próprios Desenvolvedores, e nunca pelo Product Owner, Scrum Master ou qualquer outra pessoa (veja o capítulo *Story Points: estimando o trabalho*).

Os Desenvolvedores também definem, porém em conjunto com o Product Owner, um Objetivo do Sprint realista, que guiará seu trabalho durante o Sprint, e buscarão realizá-lo por meio da implementação dos itens selecionados.

Repare que o trabalho planejado para o Sprint é apenas uma **PREVISÃO**. Embora deem o seu melhor para completar esse trabalho, o foco principal dos Desenvolvedores é a realização do Objetivo do Sprint. É comum e perfeitamente normal eles não completarem todos os itens selecionados, desde que trabalhem juntos e na ordem definida, buscando garantir assim que, se sobrarem itens, serão os menos importantes para a realização do Objetivo do Sprint.

É importante que os Desenvolvedores possuam o espaço necessário para criar e desenvolver, dentro de sua capacidade e ritmo, seus próprios meios para realizar o Objetivo de cada Sprint. Por serem aqueles que realizam o desenvolvimento do produto propriamente dito, de ponta a ponta, eles evitam dependências externas e têm autonomia para planejar como implementarão cada um desses itens selecionados. Esse plano, em geral, é criado quebrando os itens em tarefas e, possivelmente, estimando essas tarefas.

No decorrer do Sprint, os Desenvolvedores realizam o trabalho para transformar cada item do Sprint Backlog em uma funcionalidade pronta do produto, de acordo com a Definição de Pronto, visando a realizar o Objetivo do Sprint. Eles possuem autonomia para executar esse trabalho da forma que considerarem mais apropriada, organizando, dividindo e coordenando suas tarefas, acompanhando o progresso e buscando suas próprias soluções para os problemas que surgirão pelo caminho.

Entretanto, destaco que, ao falarmos de autonomia, estamos tratando dos Desenvolvedores em conjunto, e não do indivíduo. Ou seja, a ideia não é que cada um

faça o que ache melhor, mas sim o que o conjunto acredite que deva ser feito. Além disso, essa maior autonomia também não deve ser confundida com anarquia ou falta de controle. Ela se dá dentro dos limites estabelecidos pelo Scrum, pelo contexto organizacional, pelas necessidades dos clientes e usuários e pela ordem de implementação definida durante o planejamento de cada ciclo para realizar o Objetivo do Sprint que negociaram com o Product Owner.

Identificam e informam impedimentos ao Scrum Master

Costumo definir "impedimento" como *um obstáculo ou barreira que dificulta significativamente ou impede que o trabalho dos Desenvolvedores seja realizado, bloqueando um ou mais itens do Sprint Backlog de forma a ameaçar o Objetivo do Sprint*. De acordo com essa definição, a resolução de um impedimento não está ao alcance dos Desenvolvedores, está fora do seu contexto de trabalho ou lhes tomará muito tempo. Assim, conseguimos diferenciar impedimentos de problemas do dia a dia, que os próprios Desenvolvedores podem resolver, sem que ameacem o Objetivo do Sprint.

Os Desenvolvedores buscam proteger-se e prevenir-se de impedimentos em seu trabalho. Uma vez que encontram um impedimento que não pôde ser evitado, ele é imediatamente informado ao Scrum Master que, então, toma as ações necessárias para removê-lo o mais rapidamente possível. Dessa forma, os Desenvolvedores não esperam até a reunião de Daily Scrum seguinte para comunicar impedimentos ao Scrum Master, mas sim o fazem ativamente durante o seu trabalho.

Sinalização de impedimentos

Além da comunicação verbal, uma boa prática que pode ser usada pelos Desenvolvedores é a sinalização visual do impedimento no quadro de tarefas que representa o seu Sprint Backlog, junto ao item ou à tarefa afetada.

Essa sinalização não substitui a ação de informarem o impedimento imediatamente ao Scrum Master, mas ajuda tanto o Scrum Master quanto os Desenvolvedores a não negligenciarem e a melhor lidarem com o impedimento.

Um exemplo típico desse tipo de sinalização é a marcação do item ou tarefa impedida com uma cor diferente, seja colando outra nota adesiva sobre o item ou tarefa (quando os Desenvolvedores utilizam um quadro de tarefas físico), mudando a cor do próprio item ou tarefa (no caso de um quadro virtual), ou usando algum outro tipo de marcação. A figura a seguir mostra um exemplo de como podemos sinalizar um impedimento em uma tarefa.

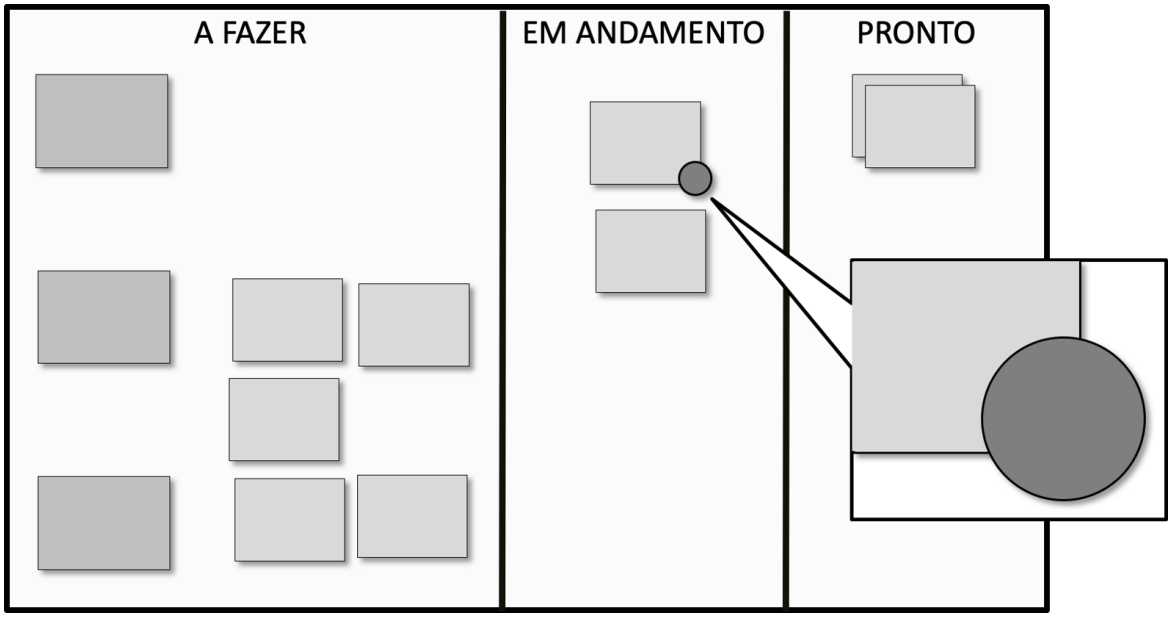


Figura 6.1: Exemplo de sinalização de um impedimento no quadro de tarefas

Prevenção de impedimentos

Em geral, impedimentos podem evidenciar problemas sérios na estrutura da organização, na qualidade dos processos ou na qualidade dos produtos desenvolvidos. Assim, é importante que os Desenvolvedores não entendam os impedimentos como um fator natural e aceitável do processo.

Impedimentos também geram desperdícios ao levarem a um consumo indevido de energia dos Desenvolvedores. Esse esforço se dá na identificação do impedimento como tal, o que pressupõe um esforço inicial para resolvê-lo, e passa pela interrupção do trabalho, pela sinalização do impedimento ao Scrum Master e pelas mudanças de contexto decorrentes de trocas do item em execução. Os impedimentos também aumentam a quantidade de trabalho em processo, já que os Desenvolvedores iniciam novos itens enquanto os impedidos ficam em espera, e assim se alargam os prazos e a quantidade de itens prontos ao final do Sprint é reduzida.

Considerando esses efeitos extremamente negativos dos impedimentos, é saudável que os Desenvolvedores, antes de tudo, busquem evitar a própria ocorrência do impedimento. Para que seja possível fazê-lo, os Desenvolvedores podem utilizar alguns importantes recursos, como os exemplos descritos a seguir (PIMENTEL, 2009):

- **um verdadeiro engajamento em realizar os objetivos** — quanto menor for o engajamento e, portanto, a motivação dos Desenvolvedores acerca do Objetivo do Sprint, mais facilmente um problema cuja resolução está ao seu alcance será entendido como impedimento e aceito como tal. Assim, esse

problema poderá, mesmo que involuntariamente, ser usado como razão para atrasos ou mesmo para a não realização do trabalho;

- **o uso da dinâmica da iteração e seu *timebox* a seu favor** — solicitações externas ou iniciativas dos próprios Desenvolvedores que desviam o foco de seu trabalho do Objetivo do Sprint também podem transformar-se em impedimentos. Assim, é importante que os Desenvolvedores sejam capazes de questionar se a urgência da interrupção é real, ou se ela pode esperar o próximo Sprint, em que o trabalho necessário será devidamente planejado. O Objetivo do Sprint pode ficar seriamente ameaçado, por exemplo, quando um pedido da alta gerência fizer com que Desenvolvedores parem o que estão fazendo para atendê-lo, abandonando o planejamento. O mesmo problema ocorre quando, no decorrer de um Sprint, Desenvolvedores resolvem testar ou implantar novas ferramentas para melhorar sua produtividade, mas gastam tempo demais nessas atividades não planejadas. Lembro aqui que, caso o Objetivo do Sprint perca o sentido, o Sprint provavelmente será cancelado (veja *O Sprint pode ser cancelado?*, no capítulo *Sprint*);
- **o acionamento preventivo do Scrum Master** — assim que identificam um impedimento iminente, Desenvolvedores acionam o Scrum Master para ajudá-los a rejeitar esse impedimento antes mesmo que ele se concretize, protegendo as atividades que estão sendo executadas. Por exemplo, logo que identificarem ser futuramente necessário o envolvimento pontual de uma pessoa ou equipe externa de difícil disponibilidade, os Desenvolvedores podem imediatamente solicitar uma ação do Scrum

Master para se antecipar ao problema e garantir a disponibilidade dessa pessoa ou equipe;

- **a identificação de dependências em tempo de planejamento** — ao identificarem, em tempo de planejamento (Daily Scrum, Sprint Planning ou planejamento de uma entrega), dependências internas e externas que poderão impactar o seu trabalho durante o Sprint, os Desenvolvedores podem buscar meios de evitá-las ou antecipar o envolvimento e a ação de atores que possam ajudar a endereçá-las. Por exemplo, não recomendamos colocar para implementação uma funcionalidade que depende de uma ferramenta ou de uma pessoa externa que somente poderá se tornar disponível no decorrer do Sprint.

É importante, portanto, que os Desenvolvedores busquem se proteger e se prevenir dos impedimentos, tratando-os não somente como algo que prejudica suas pequenas tarefas do dia a dia, mas sim como algo que pode prejudicar e até mesmo comprometer os seus objetivos.

6.3 Como são os Desenvolvedores?

Motivados

A falta de motivação no trabalho é historicamente relacionada a baixos rendimentos, faltas, atrasos, tédio, frustração, insatisfação e ineficiência entre trabalhadores (MOTTA, 1991).

Ao buscarmos fatores que influenciam a motivação dos Desenvolvedores, devemos levar em conta que os seres

humanos são complexos por natureza e, assim, são em cada instante influenciados por suas necessidades, desejos, aspirações, preferências, problemas, frustrações e toda a sorte de emoções. O ser humano, portanto, possui momentos bons e momentos ruins na realização do seu trabalho.

Em qualquer time, a motivação no trabalho é essencial para gerar compromisso de seus membros com o valor a ser gerado e com a qualidade do produto de seu trabalho. A motivação também pode ajudar a manter uma estabilidade na composição do time, já que pode levar a uma menor rotatividade de pessoal, ajudando a não se perder o conhecimento e ritmo de trabalho adquiridos.

Os Princípios Ágeis apontam explicitamente o ambiente, o suporte e a confiança necessários para realizar o trabalho como essenciais para a motivação do indivíduo no trabalho. Outro fator que pode ser entendido como motivador, também apontado pelos Princípios Ágeis, é a existência de um ritmo sustentável de trabalho, para o qual evitamos a prática de horas extras e a aceleração forçada do ritmo de trabalho, por exemplo, diante da perspectiva de atraso em uma entrega prevista.

O Scrum Master, enquanto uma liderança motivadora, também exerce influência sobre a motivação dos Desenvolvedores, facilitando a auto-organização e encorajando-os à auto-observação, autoavaliação e autorreforço (veja *O líder do time autogerenciado*, em *Como é o Scrum Master?*, no capítulo *Scrum Master*).

Destaco, a seguir, algumas teorias sobre motivação no trabalho que trazem fatores motivacionais relevantes a times. Começo com a Motivação 3.0 (PINK, 2009), mais recente e alinhada com o nosso trabalho, e sigo com

algumas teorias mais clássicas e reconhecidas (Teoria de Fixação de Objetivos, Teoria das Características do Trabalho e Teoria ERC), usando como base a minha dissertação de mestrado de anos atrás (ARMONY, 2010).

Motivação 3.0

Daniel Pink, em seu livro *Drive: the surprising truth about what motivates us*, compartilha o resultado de estudos que mostram que a motivação do trabalhador pode funcionar de um jeito muito diferente do que imaginamos.

Quando o trabalho requer apenas habilidades básicas, mecânicas, como uma série de passos com um resultado único, os bônus funcionam conforme esperado: quanto mais se paga, melhor o desempenho. Mas, quando o trabalho requer habilidades cognitivas, tomada de decisão ou criatividade, mesmo que um mínimo, uma recompensa maior surpreendentemente leva a um desempenho pior (ARIELY, 2008).

A melhor forma de utilizar o dinheiro é pagar o suficiente para que o dinheiro não seja mais um problema, e assim o trabalhador passa a pensar no trabalho em si. A partir daí, os principais motivadores são autonomia, maestria e propósito:

- autonomia: o desejo natural de dirigirmos nosso próprio trabalho e nossas vidas. Enquanto que o controle pode levar a uma maior conformidade, a autonomia leva a um maior engajamento;
- maestria: o desejo de nos tornarmos cada vez melhores em algo que nos importa;
- propósito: o anseio de realizarmos o nosso trabalho conectados a algo significativo, a serviço de algo

> maior que nós mesmos. É parte disso entendermos o porquê de nosso trabalho e sermos capazes de reconhecer qual a nossa contribuição.

Pink, dessa forma, define o que ele chama de *Motivação 3.0*. Nesse novo paradigma, a motivação intrínseca é muito mais importante do que a motivação extrínseca para aumentar o desempenho e a satisfação no trabalho.

É interessante observarmos como o Scrum se conecta bem a esses três elementos. A autonomia está diretamente presente no trabalho de times autogerenciados. A maestria é estimulada a partir da melhoria contínua, fator fundamental do Scrum, e destacada no Princípio Ágil *a atenção contínua à excelência técnica e a um bom desenho aumentam a agilidade*. Também é estimulada no time multidisciplinar, em que um Desenvolvedor acaba aprendendo com o outro. O propósito está sempre claro ao implementarmos, em ciclos curtos, algo de ponta a ponta que visa a gerar valor para clientes e usuários. Esse foco no propósito é reforçado pelo feedback frequente, pelas entregas contínuas e pelos objetivos de valor estabelecidos como parte do trabalho, como o Objetivo do Sprint e o Objetivo do Produto.

Scrum é desenhado para esse tipo de trabalho - o trabalho criativo. De acordo com essa teoria, autonomia, maestria e propósito, portanto, são vistos como os principais elementos motivadores para os Desenvolvedores.

Teoria da Fixação de Objetivos

De acordo com essa teoria, a fixação de objetivos desafiadores, claros e realizáveis, o compromisso com esses objetivos e a disponibilidade de feedback

mostrando o progresso na realização desses objetivos são poderosos mecanismos motivacionais (LOCKE, 1996), tanto para indivíduos quanto para grupos (LOCKE; LATHAM, 2006).

Podemos entender os objetivos no Scrum como o Objetivo do Sprint, estabelecido para cada ciclo, e o Objetivo do Produto. Adicionalmente, podemos estabelecer objetivos intermediários, como os para as entregas.

O Princípio Ágil *software em funcionamento é a principal medida de progresso* (que podemos adaptar para *produto em funcionamento*) nos indica que os principais feedbacks sobre o progresso dos Desenvolvedores em direção aos seus objetivos são aqueles oferecidos por clientes e demais partes interessadas sobre Incrementos do produto implementados, somados às métricas coletadas sobre o uso real do produto por seus usuários.

Os Gráficos de Burndown ou Burnup de Trabalho também fornecem feedback visual para o trabalho dos Desenvolvedores (veja o capítulo *Burndown e Burnup: acompanhando o trabalho*). No entanto, diferentemente do que prega essa teoria, eles indicam o progresso no cumprimento de um plano (o Sprint Backlog) mais do que o progresso na realização de objetivos.

Teoria das Características do Trabalho

Segundo essa teoria, o indivíduo pode se motivar em seu trabalho se ele percebe esse trabalho como compensador ou importante, acredita que é diretamente responsável por seus resultados e é capaz de determinar se esses resultados foram ou não satisfatórios (HACKMAN et al., 1975).

Para chegar a esses estados psicológicos, é importante que determinadas características estejam presentes no trabalho do indivíduo. A primeira delas é a pessoa necessitar de uma variedade de habilidades e conhecimentos para executar seu trabalho, o que ocorre naturalmente em um time multidisciplinar. Outra dessas características é executar uma parte inteira e identificável do trabalho do início ao fim, característica importante do Scrum em que os Desenvolvedores produzem, em cada Sprint, funcionalidades prontas e de ponta a ponta.

O indivíduo deve também perceber seu trabalho como tendo um impacto significativo sobre a vida ou o trabalho de outras pessoas. As métricas de uso consequente das entregas frequentes e o feedback provindo das reuniões de Sprint Review, que realimentam o trabalho, somados à autorregulação entre os Desenvolvedores, lhes trazem uma noção sobre a sua própria eficácia e desempenho, mais uma característica necessária. Por fim, a autonomia na realização do trabalho, outra dessas características, é natural em um time autogerenciado.

Teoria ERC

Segundo essa teoria, o ser humano é motivado por três categorias de necessidades que devem ser satisfeitas, ordenadas da seguinte forma: as necessidades de existência, as de relacionamento e as de crescimento (ALDERFER, 1969; ALDERFER et al., 1974).

Podemos caracterizar como necessidades de existência dos Desenvolvedores as condições básicas de trabalho, como salários em dia, um ambiente de trabalho adequado, uma infraestrutura suficiente e relações de trabalho respeitosas.

As necessidades de relacionamento do indivíduo podem ser realizadas a partir da intensa comunicação e do bom relacionamento entre os Desenvolvedores, que levam ao compartilhamento de pensamentos e sentimentos relevantes.

Já as necessidades de crescimento do indivíduo podem ser atendidas a partir da perspectiva de desenvolvimento pessoal, da possibilidade de utilizar a criatividade em seu trabalho e na resolução de problemas, de ser produtivo e de completar tarefas relevantes.

A Teoria ERC é uma evolução da conhecida Teoria das Necessidades de Maslow.

Discussão: avaliações e recompensas individuais?

As avaliações relativas ao trabalho realizado e as recompensas correspondentes, quando dirigidas ao indivíduo, estimulam-no a trabalhar, acima de tudo, como indivíduo. Ao recompensarmos os trabalhadores individualmente, o que estamos incentivando é a competição entre eles em vez da colaboração para realizarem os objetivos comuns (HACKMAN, 1987). E essa competição não necessariamente é pelo melhor trabalho, mas muitas vezes por uma maior visibilidade diante do gerente que está julgando. Por exemplo, ao definirmos os bônus a partir da contribuição individual de cada Desenvolvedor em um Time de Scrum, cada um pode passar a trabalhar em detrimento do outro, minimizando a colaboração e, assim, estaremos reduzindo a efetividade do time como um todo.

Quando dirigidas ao conjunto, as avaliações e recompensas podem estimular seus membros a trabalharem como um verdadeiro time, que busca junto

um objetivo comum. Por essa razão, existe uma ampla preferência por que times ágeis sejam avaliados e recompensados como um time, e não a partir da contribuição individual de cada um.

Temos visto, no entanto, algumas abordagens de avaliação cruzada entre Desenvolvedores funcionando muito bem. Nelas, cada um avalia seus colegas de acordo com parâmetros estabelecidos pelo próprio time. O conjunto das avaliações realizadas, então, pode servir de base para o sistema de recompensas, que pode também somar-se ou ser cruzado com uma avaliação externa do conjunto.

> Na K21, utilizamos o Merit Money como parte de nosso sistema de bônus. É um sistema colaborativo de bonificação do Management 3.0 que incentiva o apoio e o reconhecimento entre os membros do time, com foco em ações e comportamentos que contribuam com a melhoria do time como um todo (MARINHO, 2018). Com o Merit Money, comportamentos como colaboração, comunicação, capacidade de dar e receber feedback, foco etc. são avaliados pelos colegas. Repare, no entanto, que não avaliamos os resultados individuais. Esses, acreditamos, são alcançados em conjunto e são consequências dos comportamentos desejados.

Colaborativos

Em seu dia a dia de trabalho, os Desenvolvedores de um Time de Scrum colaboram entre si continuamente, em iguais condições e compartilhando as responsabilidades. Independente das áreas de conhecimento e de atuação

primária de cada um, não há entre eles papéis ou responsabilidades definidas, nem hierarquias ou lideranças designadas. Todos trabalham em conjunto, implementando itens do Sprint Backlog a partir daquele de maior ordem, buscando assim realizar o Objetivo de cada Sprint e o Objetivo do Produto.

Em qualquer momento no Sprint, os Desenvolvedores buscam, sempre que necessário, comunicar-se e colaborar com pessoas que ajudem a esclarecer, detalhar, tomar decisões ou obter feedback antecipado sobre itens do Sprint Backlog, ou seja, aqueles escolhidos para implementação. Com frequência, eles buscam o próprio Product Owner para tal. No entanto, dependendo do contexto e da disponibilidade, os Desenvolvedores podem buscar diretamente clientes e demais partes interessadas.

Para tornar essa colaboração possível, essas pessoas idealmente se colocam suficientemente acessíveis e disponíveis. Caso contrário, os Desenvolvedores podem ter que paralisar a implementação de um ou mais itens enquanto aguardam por essas interações, o que se torna um impedimento. Outra possível consequência é que os Desenvolvedores tomem decisões sobre o produto sem ter informações suficientes para isso.

Por vezes, o Scrum Master intervém para facilitar esse acesso. O Scrum Master também está presente, quando necessário, para estimular a colaboração e cuidar da qualidade das interações entre os Desenvolvedores, entre eles e a Product Owner, e entre eles e demais pessoas nos momentos em que isso se fizer necessário.

Na reunião de Sprint Planning, os Desenvolvedores e Product Owner colaboram para esclarecer, refinar e preparar os itens que serão implementados no Sprint. No

decorrer do Sprint, Desenvolvedores, Product Owner e, possivelmente, clientes e outras partes interessadas podem antecipar esse trabalho de preparação e realizar sessões de Refinamento do Product Backlog, preparando itens para o Sprint seguinte (veja o capítulo *Refinamento do Product Backlog*). Os Desenvolvedores colaboram com o Product Owner, clientes e demais partes interessadas na reunião de Sprint Review para obterem feedback sobre os resultados do trabalho realizado durante o Sprint. Por fim, a Sprint Retrospective reúne Desenvolvedores, Product Owner e Scrum Master para, de forma colaborativa, buscarem melhorias na sua forma de trabalhar.

Discussão: há papéis ou responsabilidades específicas entre os Desenvolvedores?

Os Desenvolvedores são integralmente responsáveis por implementar, em cada Sprint, ao menos um Incremento do produto pronto, de acordo com a Definição de Pronto. Para realizar esse trabalho, há entre eles todas as habilidades e conhecimentos necessários. Não existem, no entanto, papéis ou responsabilidades definidas entre os Desenvolvedores. Cada um deles é, simplesmente, um "Desenvolvedor".

Como exemplo, o trabalho de desenvolvimento do produto obrigatoriamente inclui todo e qualquer tipo de atividade necessária para garantir sua qualidade. Assim, de acordo com a definição acima, as habilidades para realizar essas atividades devem estar presentes entre os Desenvolvedores, seja em um ou mais indivíduos, bem definidas ou espalhadas entre eles. No entanto, mesmo quando essas habilidades se concentram em Desenvolvedores específicos, a responsabilidade pela qualidade dos Incrementos implementados não recai

apenas sobre eles. Ainda que, por herança de métodos tradicionais, esses Desenvolvedores fossem anteriormente caracterizados como "controle de qualidade" ou "testadores", esses papéis não existem no Scrum e a responsabilidade é compartilhada entre todos os Desenvolvedores.

O mesmo princípio é aplicado aos outros conhecimentos e habilidades necessários para o desenvolvimento do produto. Esses estão todos presentes entre os Desenvolvedores de um Time de Scrum, seja em um ou em mais de um deles, mas nunca concentrados em papéis definidos focados em tratar do assunto. Lembro que as únicas responsabilidades definidas pelo framework Scrum são Scrum Master, Product Owner e Desenvolvedor.

Não há também times separados realizando quaisquer desses trabalhos, seja com antecedência ou posteriormente aos Desenvolvedores. Todo e qualquer trabalho necessário para o desenvolvimento do produto é realizado iterativa e incrementalmente pelos Desenvolvedores.

Orientados à excelência

> *A atenção contínua à excelência técnica e a um bom desenho aumenta a Agilidade* - Princípio Ágil.
>
> *Em intervalos de tempo regulares, a equipe reflete sobre como se tornar mais efetiva e, então, refina e ajusta seu comportamento de acordo* - Princípio Ágil.

Os Desenvolvedores possuem, entre eles, pessoas tecnicamente qualificadas que trabalham com

consciência e implementam Incrementos de alta qualidade. Com esse trabalho, visam a atender às necessidades de seus clientes e usuários da melhor forma possível, e não simplesmente cumprir tarefas.

A falta de qualificação das pessoas e a falta de compromisso com a qualidade podem levar ao acúmulo de problemas técnicos e de soluções malfeitas ou temporárias. Não existe mágica: não há método ou processo que faça com que maus profissionais produzam bons resultados.

Os Desenvolvedores, no entanto, são orientados à excelência. Ainda que se considerem suficientemente qualificados para construírem o produto de ponta a ponta, eles buscam continuamente aprender como podem melhor realizar o trabalho, como podem fazê-lo com mais qualidade e como podem melhor atender às necessidades de seus clientes e usuários, tornando-se mais efetivos. Diversas práticas do Scrum, como a reunião de Sprint Retrospective, reforçam essa ideia de melhoria contínua.

Capítulo 7
Product Owner

Conteúdo

1. Quem é o Product Owner?
2. O que faz o Product Owner?
    - Gerencia a definição do produto.
    - Gerencia a relação dos clientes com o produto.
    - Assegura o Objetivo do Produto.
    - Gerencia as entregas do produto.
    - Participa dos eventos do Scrum.
    - Colabora com os Desenvolvedores durante o Sprint.
3. Como é o Product Owner?
    - Único.
    - Disponível para o trabalho com Scrum.
    - Competente para a definição do produto.

7.1 Quem é o Product Owner?

O Product Owner, também chamado de P.O., é a pessoa responsável por garantir e maximizar o valor do produto gerado a partir do trabalho do Time de Scrum, de forma a satisfazer as necessidades de clientes e usuários. O Product Owner é apenas uma pessoa. Ele possui a autoridade para tomar as decisões com relação à definição do produto, mas sempre alinhadas com o Objetivo do Produto. Com Scrum, o produto é definido de forma incremental, ao longo de todo o seu desenvolvimento.

O Product Owner, juntamente com o Scrum Master e com os Desenvolvedores do produto, compõe o Time de Scrum.

Ao realizar o seu trabalho, o Product Owner:

- gerencia a definição do produto, o que inclui inserir, detalhar, remover, ordenar e trazer transparência e clareza para os itens do Product Backlog, ao longo de todo o desenvolvimento do produto, de forma a maximizar o resultado do trabalho do Time de Scrum. Ele pode realizar esse trabalho sozinho ou delegar, mesmo que em parte, aos Desenvolvedores ou a outras pessoas, mantendo-se, no entanto, responsável pelos resultados;

- gerencia a relação dos clientes e demais partes interessadas com o desenvolvimento do produto, entendendo quem são essas pessoas a influenciar as decisões sobre o produto, balanceando suas necessidades, comunicando-se com elas para ajudar a descobrir essas necessidades, e influenciando-as para maximizar sua colaboração para o desenvolvimento do produto;

- assegura o Objetivo do Produto, definindo-o junto a negócios, aos clientes do produto e às demais partes interessadas ou alinhando-se ao já estabelecido, comunicando-o a todos os envolvidos e garantindo que o trabalho seja feito de forma a realizá-lo;

- gerencia as entregas do produto, definindo seus objetivos e a melhor estratégia para a realização dessas entregas;

- participa do Sprint nos eventos do Scrum, realizando com os Desenvolvedores o planejamento do Sprint na reunião de Sprint Planning, a apresentação e

obtenção de feedback dos clientes e demais partes interessadas na reunião de Sprint Review e o processo de melhoria contínua na reunião de Sprint Retrospective;

- colabora com os Desenvolvedores, sempre que necessário, mostrando-se disponível para esclarecer dúvidas e tomar decisões quanto aos detalhes do produto, e para, com eles, refinar e preparar o Product Backlog para a implementação, conforme necessário e possível.

O Product Owner é:

- único para um Time de Scrum, pois deve haver apenas um foco de decisões sobre o produto para os Desenvolvedores;
- disponível para colaborar com os Desenvolvedores, para estar presente nas reuniões do Scrum em que sua presença é obrigatória, para interagir com os clientes e demais partes interessadas, e para manter e ordenar o Product Backlog;
- competente para a definição do produto, com conhecimento e poder suficientes para tomar decisões rápidas e adequadas, visando otimizar o valor do trabalho realizado pelo Time de Scrum.

Discussão: de onde vem o Product Owner?

Escolher como Product Owner o próprio cliente do produto ou alguém designado por ele é uma opção adotada por muitas organizações e defendida por vários

especialistas em Scrum. Nesse cenário, o Product Owner definirá o produto a ser desenvolvido a partir de suas próprias necessidades ou de quem ele representa diretamente, ou seja, de quem espera ter seus problemas resolvidos. Naturalmente, essa escolha busca garantir que o Product Owner possua o domínio do negócio e do contexto, e que tome decisões de produto que favoreçam o próprio cliente.

A prática mostra, no entanto, que essa não é a melhor opção em muitos casos. Na realidade, já vi muitos fracassos acontecerem como decorrência dessa escolha, especialmente quando tratamos de um cliente externo.

A partir da nossa experiência na K21, defendemos que o Product Owner deve ser, em princípio, alguém próximo aos times que estão desenvolvendo o produto. Desse modo, no caso de um cliente externo, a melhor escolha seria alguém do próprio fornecedor.

Para justificar essa opinião, eu argumento que Product Owner:

- é uma responsabilidade que exige grande conhecimento da função, e raramente alguém do cliente saberá exercê-la. Na maioria das vezes, o cliente simplesmente não tem as habilidades e os conhecimentos necessários, que obrigatoriamente incluem saber definir o produto de forma iterativa e incremental, além de dominar o próprio framework Scrum. Esse problema é mais comum e mais grave do que parece. No entanto, muitos defendem que, para resolver essa questão, o Scrum Master poderia ensinar o que for necessário ao Product Owner, talvez com o apoio de treinamentos específicos. Esse investimento de tempo e dinheiro em uma formação tão complexa e tão específica, que engloba uma

série de habilidades, conhecimentos, práticas e, muitas vezes, uma grande mudança de cultura, somente vale a pena se a pessoa treinada como Product Owner passar a atuar com essa responsabilidade de forma permanente. Pensando dessa forma, pode fazer sentido formar como Product Owner alguém da área solicitante, no caso de um cliente interno;

- é uma responsabilidade que exige descobrir e enxergar os problemas existentes, em vez de partir das soluções desejadas. Uma visão externa, quase de um consultor, torna esse trabalho mais viável. O cliente, no entanto, está imerso no contexto de seus problemas e geralmente não os identifica com clareza, raramente sendo capaz de saber o que realmente precisa, ou entender que é um processo de descoberta. Por conta disso, não é nada incomum o cliente trazer soluções "prontas" para problemas que ele não entendeu ainda, fazendo com que os Desenvolvedores realizem implementações que não entregam muito valor. Além disso, que cliente você conhece que sabe priorizar? Tudo é prioridade, não? Assim, o Time de Scrum pode fazer exatamente o que o cliente pediu, mas esse cliente, ao final, não ficará satisfeito;
- é uma responsabilidade que exige disponibilidade e colaboração. Mesmo que esse Product Owner do cliente possua os conhecimentos e habilidades necessários, é bem possível que ele não tenha tempo disponível para atuar com essa responsabilidade, pois provavelmente estará mais envolvido em outras questões de seu próprio dia a dia de trabalho no cliente. Além disso, sua relação com os Desenvolvedores poderá se converter em uma

relação de cobrança, e não de colaboração como esperamos de membros de um mesmo Time de Scrum. O resultado disso pode ser um Product Owner que pouco interage com os Desenvolvedores, que não confia neles o suficiente para delegar, que cobra deles em vez de colaborar, que possivelmente não está presente nas reuniões do Scrum em que sua presença é necessária (e, muitas vezes, sequer é desejado lá pelos Desenvolvedores), e que não consegue dedicar tempo para pensar no produto e refinar o Product Backlog;

- é uma responsabilidade que exige equilíbrio e política. Quando o produto possui vários clientes ou diferentes partes interessadas, que podem até mesmo pertencer a diferentes organizações, fica claro o desvio na escolha de um Product Owner entre eles. Somente pode haver um Product Owner no Time de Scrum e, assim, o cliente que for escolhido como tal poderá tomar decisões que o favoreçam em detrimento dos outros.

Por todos esses motivos, recomendo fortemente, para a grande maioria dos casos, que o Product Owner se origine do lado de quem está desenvolvendo o produto, especialmente no caso de um cliente externo. Deve ser alguém formado para tal, com as habilidades e conhecimentos necessários. Ele realizará seu trabalho a partir do contato frequente com os clientes do produto e com as demais partes interessadas, balanceando suas necessidades, mas tomando decisões alinhadas com os objetivos da sua própria organização.

Importante ressaltar que, para ser de fato um Product Owner, essa pessoa deve ter a autoridade para tomar as decisões necessárias sobre o produto. Ou seja, mesmo que não seja alguém do cliente, o Product Owner é

sempre quem de fato tem a responsabilidade pela definição do produto.

7.2 O que faz o Product Owner?

Gerencia a definição do produto

Ao longo de todo o trabalho de desenvolvimento do produto, o Product Owner mantém contato frequente com os clientes e demais partes interessadas para identificar e entender os problemas mais importantes a serem resolvidos com o produto. Com o apoio dos Desenvolvedores, o Product Owner busca continuamente medir e entender como os usuários estão utilizando o produto que lhes está sendo entregue de forma incremental, e o que melhor pode atender às suas próximas necessidades mais importantes. Com isso, ele decide que características e comportamentos serão implementados em cada momento, inserindo-as como itens em uma lista ordenada, chamada de Product Backlog.

O Product Owner é o responsável pela gestão do Product Backlog e possui a autoridade sobre ele, ou seja, tem a palavra final do que será feito. Ele atualiza, ordena e reordena seus itens, conforme considera necessário. Ele também remove aqueles itens que não são mais necessários e que, por essa razão, não deverão mais ser implementados. Ele garante a transparência e a clareza do Product Backlog aos Desenvolvedores, que devem ser capazes de saber o que está por vir e influenciar as decisões.

O Product Owner colabora com os Desenvolvedores para tomar as decisões quanto ao produto e pode delegar para eles, mesmo que em parte, esse trabalho de gestão do Product Backlog. Quanto maior a maturidade dos Desenvolvedores, em conjunto, e quanto maior seu domínio do negócio, mais desse trabalho o Product Owner poderá delegar para eles. Dependendo do nível de delegação, eles poderão desde apoiar efetivamente o trabalho do Product Owner na definição do produto, oferecendo suas ideias e sua perspectiva quanto ao que será implementado, até tomar diretamente decisões importantes sobre o produto.

O Guia do Scrum, na edição de 2020, abre ainda a possibilidade de que o Product Owner delegue o trabalho de gestão do Product Backlog a outras pessoas (SCHWABER; SUTHERLAND, 2020). No entanto, recomendo fortemente que esse trabalho fique restrito a membros do Time de Scrum.

Independente de quanto e de para quem o Product Owner delegue o trabalho de gestão do Product Backlog, ele permanece responsável por seus resultados.

O Product Owner e os Desenvolvedores garantem que os próximos itens ordenados pelo próprio Product Owner estarão preparados para serem implementados, Sprint após Sprint, com detalhes suficientes e com granularidade fina. Eles podem realizar, em conjunto, esse trabalho de refinamento. Alternativamente, o Product Owner pode delegar aos Desenvolvedores completa ou parcialmente a responsabilidade de interagirem diretamente com clientes e demais partes interessadas para fazê-lo.

Os Desenvolvedores influenciam também diretamente as decisões do Product Owner oferecendo uma visão do que

é tecnicamente possível, do esforço de implementação de cada item do Product Backlog e de que trabalho técnico adicional poderá ser necessário. Cabe aos Desenvolvedores identificarem itens de trabalho técnico que se façam necessários — como a atualização de algum software de apoio, o teste de uma nova ferramenta ou uma melhoria de base de alguma parte do produto, por exemplo — e sugerir ao Product Owner sua inserção no Product Backlog. Em princípio, o Product Owner decide, após buscar junto aos Desenvolvedores o entendimento do que seja necessário entender, se esses itens serão realmente implementados e quando.

Em seu trabalho de gestão da definição do produto, o Product Owner garante que tudo o que será implementado está alinhado com o Objetivo do Produto estabelecido.

É comum que o Product Owner desenvolva uma estratégia mais formal para o desenvolvimento do produto visando a realizar o Objetivo do Produto. A estratégia, assim como o próprio desenvolvimento do produto, evolui de forma iterativa e incremental, e o ajuda a tomar as decisões necessárias quanto ao que será implementado.

Um formato possível de estratégia de desenvolvimento do produto é o *roadmap* do produto, que descreve o futuro esperado do produto em alto nível. Em um *roadmap*, apontamos uma data provável para cada entrega, e o objetivo a ser realizado em cada uma dessas entregas ou marcos (veja a seção *Objetivos intermediários* no capítulo *Compromisso: Objetivo do Produto*).

Discussão: Product Owner é proxy do cliente?

Ao contrário do que observamos em inúmeras organizações, o Product Owner **não é**:

- o *proxy* dos clientes (SCHWABER, 2011) e nem de ninguém;
- um canal de comunicação ou uma ponte para o(s) cliente(s);
- o representante do(s) cliente(s);
- um tomador de pedidos ou de *tickets* do(s) cliente(s).

O Product Owner, na realidade, é aquele que possui a responsabilidade final quanto à definição do que será implementado. Ele possui autoridade e conhecimento das técnicas necessárias para fazê-lo de forma a satisfazer às necessidades dos clientes. Ele entende que não será capaz de atingir esse objetivo caso simplesmente atenda a seus desejos e vontades.

O cliente, de forma geral, não sabe o que precisa e, assim, o Product Owner é responsável por trabalhar junto aos Desenvolvedores para descobrir, definir e validar soluções junto aos usuários, clientes e partes interessadas ao longo de todo o trabalho.

Gerencia a relação dos clientes com o produto

Além dos clientes, as partes interessadas no produto são seus usuários, patrocinadores e quaisquer pessoas ou grupos que tenham algum tipo de influência nas definições do produto ou que tenham interesse direto, seja no seu andamento, no seu sucesso ou em seus impactos. As partes interessadas, dependendo do contexto, podem incluir, por exemplo, a área de *marketing*, a área jurídica, agentes reguladores externos, associações de classe etc.

O Product Owner faz a gestão dos clientes do produto e demais partes interessadas em sua relação com o produto que está sendo desenvolvido, trabalho em que pode contar com o suporte dos Desenvolvedores, do Scrum Master e de pessoas externas ao Time de Scrum. Essa gestão inclui:

- identificar corretamente quem são os clientes e as demais partes interessadas, ou seja, as pessoas que são relevantes ao desenvolvimento do produto;
- identificar, entender e levar em consideração as necessidades e interesses dos diferentes clientes e demais partes interessadas com relação ao produto;
- gerenciar as expectativas dos clientes e demais partes interessadas com relação ao produto, garantindo que tenham a visibilidade necessária;
- balancear e gerenciar conflitos entre as necessidades e expectativas dos diferentes clientes e demais partes interessadas;
- ajudar os clientes e demais partes interessadas a descobrir e entender o que lhes trará maior valor em cada momento do trabalho, influenciando suas opiniões ao defender a visão do que acredita que será melhor para o produto;
- obter a colaboração direta dos clientes e demais partes interessadas sempre que se mostrar necessária, em situações como, por exemplo, em que o Scrum Master necessita de ajuda na remoção de impedimentos organizacionais ou relacionados ao ambiente do cliente, ou em que os Desenvolvedores precisam de interagir com eles para detalhar itens que serão implementados em seguida;
- comunicar-se frequentemente e proativamente com os clientes e demais partes interessadas para realizar esse trabalho.

Assegura o Objetivo do Produto

O Product Owner é o responsável por garantir o entendimento claro do Objetivo do Produto. Ele comunica o Objetivo do Produto a todos os envolvidos, assegurando que ele seja compreendido por todos e durante todo o trabalho de desenvolvimento do produto. O Product Owner é o responsável por manter o alinhamento do Time de Scrum, clientes e demais partes interessadas ao Objetivo do Produto, gerenciando o Product Backlog de forma a garantir que todo o trabalho realizado pelos Desenvolvedores ajude a realizar esse objetivo.

O Objetivo do Produto responde de forma curta e direta à pergunta: "Por que esse produto está sendo desenvolvido?" Ou, ainda melhor, "Que problema será resolvido com o desenvolvimento desse produto?". Ele serve de guia para o trabalho do Time de Scrum e alinha o entendimento e as expectativas quanto ao produto entre o Time de Scrum e as diferentes partes interessadas.

O Objetivo do Produto pode ser definido fora do contexto do Scrum, alinhado a alguma estratégia da organização. Em muitos cenários, no entanto, o próprio Product Owner é o responsável por estabelecer o Objetivo do Produto junto a pessoas de negócios, aos clientes e a demais partes interessadas do produto antes do início do seu desenvolvimento. Ele pode também buscar, quando possível, envolver os Desenvolvedores nas atividades necessárias para chegarem ao Objetivo do Produto, pois isso aumenta seu senso de propriedade sobre o que será desenvolvido.

Gerencia as entregas do produto

A gestão da estratégia de entregas do produto é uma atribuição do Product Owner. Seguindo os Princípios Ágeis, ele define a melhor estratégia para a realização das entregas de Incrementos do produto desde cedo e frequentemente, para possibilitar o feedback rápido e, assim, reduzir os riscos do desenvolvimento do produto. Ele modifica essa estratégia quando julga necessário.

A estratégia de entregas é condicionada por questões técnicas e de negócios. Por um lado, as entregas são alinhadas a estratégias de negócios da organização e do próprio produto, de forma a contribuir com elas. Ao mesmo tempo, o Product Owner considera questões técnicas apontadas por Desenvolvedores e, em muitos casos, pelos próprios clientes ou usuários, para decidir como e quando essas entregas serão realizadas. Existem diferentes possibilidades para a estratégia de realização das entregas de um produto. Recomendo a leitura do capítulo *Release Planning (adicional)*.

Participa dos eventos do Scrum

O Product Owner, obrigatoriamente, participa das reuniões de Sprint Planning, Sprint Review e Sprint Retrospective.

Na reunião de Sprint Planning, o Product Owner colabora e negocia com os Desenvolvedores para decidirem o que será realizado durante o Sprint que se inicia. Ou seja, a partir da ordem dos itens definida pelo próprio Product Owner e a partir da capacidade de trabalho dos Desenvolvedores, eles decidem juntos quantos itens estarão previstos para serem implementados nesse Sprint, a partir do item de maior ordem, e qual será o

Objetivo do Sprint que guiará os Desenvolvedores em seu trabalho de implementação dos itens selecionados.

Ainda na reunião de Sprint Planning, enquanto os Desenvolvedores planejam parcialmente como implementarão esses itens selecionados (em geral, quebrando-os em tarefas), surge um número de questões sobre o trabalho a ser realizado. Assim, é importante que o Product Owner, caso não esteja presente nessa parte da reunião, ao menos se coloque disponível e acessível para esclarecer essas dúvidas.

Na reunião de Sprint Review, Product Owner e Desenvolvedores buscam feedback de clientes e demais partes interessadas sobre o Incremento ou Incrementos que foram implementados durante o Sprint. Esse feedback servirá de matéria-prima para o Product Owner decidir o que será feito em futuros Sprints, realimentando o Product Backlog. A interação com clientes e demais partes interessadas que ocorre na Sprint Review é o resultado da colaboração entre eles, o Product Owner e os Desenvolvedores, e seu formato varia de Time de Scrum para Time de Scrum.

Na reunião de Sprint Retrospective, os membros do Time de Scrum buscam identificar pontos de melhorias na sua forma de trabalhar, e traçam respectivos planos de ação a serem postos em prática já no Sprint seguinte. O Product Owner, enquanto membro do Time de Scrum, é parte integral dessa reunião.

Colabora com os Desenvolvedores durante o Sprint

Durante o Sprint, o Product Owner se coloca disponível e acessível, permitindo assim aos Desenvolvedores que esclareçam dúvidas, troquem ideias ou solicitem

decisões rápidas sobre itens do Sprint Backlog, conforme necessário.

Eles interagem para refinarem e prepararem itens no alto do Product Backlog para o Sprint seguinte. Essas interações podem acontecer na reunião de Sprint Planning e em sessões de Refinamento do Product Backlog, ao longo do Sprint (veja o capítulo *Refinamento do Product Backlog*).

Porém, como mencionei anteriormente, o Product Owner pode delegar, completa ou parcialmente, o trabalho de refinamento e detalhamento do produto para que os Desenvolvedores o realizem diretamente com clientes e outras partes interessadas. Quanto maior for a maturidade dos Desenvolvedores no exercício do detalhamento do produto, maior será esse nível de delegação. E, quanto maior for a delegação, menor será a necessidade da disponibilidade do Product Owner para os Desenvolvedores durante o Sprint. De uma forma ou de outra, o Product Owner evita bloquear o trabalho dos Desenvolvedores à espera de sua presença e evita que eles precisem tomar decisões sem ter as informações necessárias.

Os Desenvolvedores podem, conforme acharem necessário, buscar o feedback do Product Owner no decorrer do Sprint sobre itens em andamento ou já prontos. Esse feedback pode ajudá-los a melhorar a sua entrega e a realização do Objetivo da Sprint.

O Product Owner, no entanto, ansioso por ver partes prontas durante o Sprint ou por obter informações sobre o andamento do trabalho, pode exercer pressões sobre os Desenvolvedores que violam a sua autonomia e são prejudiciais aos resultados do trabalho. Diante de problemas como esses, o Scrum Master trabalha para

ensinar ao Product Owner como ele melhor pode realizar seu trabalho, sempre em colaboração com o resto do Time de Scrum.

Interferências frequentes ou relevantes do Product Owner no planejamento já realizado para o Sprint também são prejudiciais, pois desviam o foco dos Desenvolvedores e, assim, podem comprometer a realização do trabalho. Por essa razão, os Desenvolvedores, amparados pelo Scrum Master, rejeitam quaisquer alterações no Sprint Backlog que, durante o Sprint, modifiquem ou possam colocar em risco o Objetivo do Sprint já acordado.

7.3 Como é o Product Owner?

Único

A responsabilidade de Product Owner é de apenas uma pessoa em um Time de Scrum. Ou seja, os Desenvolvedores se relacionam com um e apenas um Product Owner.

A existência de mais de um Product Owner interagindo com os Desenvolvedores potencialmente geraria dúvidas e conflitos sobre de quem é o poder de decisão sobre o produto em desenvolvimento. Quando houvesse discordâncias, por exemplo, os Desenvolvedores não saberiam quem ouvir e, assim, o processo decisório sobre o produto se tornaria desnecessariamente lento e confuso.

De forma similar, a existência de Product Owners substitutos, que compareceriam com alguma frequência quando o Product Owner não pudesse estar presente, ou um revezamento de Product Owners poderiam confundir

os Desenvolvedores e o próprio trabalho de definição do produto. Como na situação anterior, as expectativas e interpretações diversas de diferentes Product Owners poderiam inviabilizar a compreensão e a implementação de determinados itens e, assim, a realização do Objetivo do Sprint.

Ausências do Product Owner por doença ou férias, no entanto, são situações infrequentes ou de exceção. Nesses casos, dificilmente há alguma solução diferente de simplesmente substituí-lo provisoriamente, mesmo que com o próprio trabalho dos Desenvolvedores. Dessa forma, o uso de substitutos acontece apenas quando for estritamente necessário.

Não existem, ao mesmo tempo, um Product Owner do cliente e outro da organização que está desenvolvendo o produto, trabalhando como um representante ou como um canal de comunicação. Nem existem também, ao mesmo tempo, um Product Owner de negócios e um Product Owner técnico. Ambas são disfunções que vejo com frequência em clientes. O Product Owner é único: ou ele é uma pessoa escolhida no cliente, ou ele é uma pessoa escolhida no fornecedor.

O Product Owner também não é um comitê ou um departamento da organização. Grupos de pessoas, mesmo que tomem decisões únicas e centralizadas, em geral demoram um tempo demasiadamente longo para chegar a elas, e seus membros podem ser acessados individualmente e expor opiniões distintas. Com um Product Owner único, os Desenvolvedores enxergam apenas um foco para a tomada de decisões sobre o produto e para o esclarecimento de dúvidas.

O Product Owner é a única pessoa que detém a responsabilidade por gerenciar o Product Backlog. Suas

decisões sobre o que será implementado no produto devem ser respeitadas, de forma que ninguém além dele pode mudar a ordem para a implementação por ele estabelecida.

Para realizar seu trabalho, no entanto, o Product Owner é aconselhado, influenciado e conta com o apoio dos Desenvolvedores, que melhor fazem esse trabalho quanto mais domínio do negócio de que trata o produto possuírem. O Product Owner também interage com frequência com clientes, demais partes interessadas e com quem mais julgar necessário.

Para o desenvolvimento de produtos maiores, pode haver a necessidade de existirem vários Times de Scrum trabalhando juntos. Nesses casos, ainda assim haverá apenas um Product Owner, comum a esses times, ordenando um Product Backlog único.

Definição: Product Owner único

O Product Owner é único e, assim, **não** existem:

- um Product Owner do time e outro Product Owner do cliente;
- um Product Owner técnico e um Product Owner de negócios;
- um proxy Product Owner e um Product Owner de verdade;
- um comitê de Product Owners.

Disponível para o trabalho com Scrum

O trabalho do Product Owner envolve principalmente:

- estar presente na reunião de Sprint Planning. O Product Owner define, junto com os Desenvolvedores, qual o Objetivo do Produto a ser realizado no Sprint e qual o trabalho a ser feito para realizarem esse objetivo;
- colocar-se acessível e disponível para tomar decisões e esclarecer dúvidas de Desenvolvedores sobre o produto durante todo o Sprint, sempre que solicitado;
- interagir com os Desenvolvedores para que, juntos, preparem itens para o Sprint seguinte;
- estar presente na reunião de Sprint Review para, junto com os Desenvolvedores, obter feedback dos clientes do produto e demais partes interessadas sobre o trabalho realizado no Sprint;
- estar presente na reunião de Sprint Retrospective para ajudar o Time de Scrum a melhorar seu trabalho e a se tornar mais produtivo;
- interagir frequentemente com os clientes e demais partes interessadas ao longo de todo o trabalho para entender suas necessidades de negócios, obter seu feedback sobre o trabalho já entregue e atualizar o Product Backlog com as novas informações.

Todas essas atividades podem demandar do Product Owner uma dedicação considerável de tempo. Além disso, não há nenhuma regra no Scrum que impeça o Product Owner de trabalhar, ao mesmo tempo, em mais de um Time de Scrum, seja no desenvolvimento de apenas um ou de mais produtos. No entanto, dependendo da complexidade e tamanho dos produtos, essa pode se tornar uma tarefa muito difícil para um Product Owner centralizador. Um Product Owner ocupado, ausente e que não tem disponibilidade

suficiente para realizar seu trabalho, pode trazer consequências desastrosas para o trabalho.

Diante disso, lembro que, em cenários de maior maturidade, o Product Owner pode delegar o trabalho de gestão do Product Backlog, mesmo que parcialmente, para os Desenvolvedores. Assim, em um cenário de Scrum em escala, delegar pode ser a solução para viabilizar o trabalho de um Product Owner com múltiplos times.

Competente para a definição do produto

O Product Owner define o que será implementado para o produto, Incremento a Incremento, visando a otimizar o valor do trabalho realizado pelo Time de Scrum. Para tomar as decisões necessárias, não esperamos que o Product Owner saiba de antemão tudo sobre o produto em desenvolvimento. Afinal, os detalhes do produto emergem ao longo do seu desenvolvimento. Eles são "alvos em movimento", já que o Product Owner define o que implementar em seguida a partir do próprio uso do produto e do feedback frequente dos seus clientes, usuários e demais partes interessadas.

Como afirmei na seção *Gerencia a definição do produto*, neste capítulo, o Product Owner está empoderado para tomar as decisões que considere as mais adequadas, em cada momento, com relação ao que será implementado, desde que alinhadas com o Objetivo do Produto. Assim, ele não é um intermediário para os clientes do produto, mas sim toma decisões com o propósito de atender às necessidades desses clientes. O Product Owner assume integralmente essa responsabilidade, pois entende que, se simplesmente obedecesse às vontades dos clientes e, assim, atuasse como um tomador de pedidos, ele não

seria capaz de maximizar o valor gerado pelo Time de Scrum.

Em muitos contextos, no entanto, não é incomum o Product Owner não possuir, desde o início, um grande conhecimento sobre o negócio em que o produto está inserido. Nesses casos, o Product Owner constrói o conhecimento necessário de forma gradual, ao longo do trabalho, definindo continuamente o que deve ser implementado visando a gerar o próximo maior valor. Um cenário em que isso é comum ocorre, por exemplo, no desenvolvimento do produto para um cliente externo, quando o Product Owner provém do fornecedor.

De todo modo, esperamos que o Product Owner procure estar sempre preparado para tomar, com agilidade, decisões suficientemente bem direcionadas, antecipando-se às possíveis dúvidas sobre o que será implementado a seguir, consultando-se com quem for adequado, quando for adequado, e contando com o feedback que logo receberá para realizar ajustes de acordo.

O framework Scrum promove, de forma inerente, a mitigação dos riscos de grandes desperdícios e de consequentes prejuízos provindos de decisões equivocadas do Product Owner. A partir das reuniões de Sprint Review, realizadas em cada Sprint, e a partir do próprio uso do que é entregue aos usuários, a qualidade das decisões do Product Owner torna-se visível rápida e frequentemente.

Capítulo 8
Scrum Master

Conteúdo

1. Quem é o Scrum Master?
2. O que faz o Scrum Master?
    - Promove e apoia o uso do Scrum.
    - Facilita o trabalho do Time de Scrum.
    - Garante a remoção de impedimentos.
        - Natureza dos impedimentos e sua resolução.
    - Promove as mudanças organizacionais necessárias.
3. Como é o Scrum Master?
    - Liderança a serviço do Time de Scrum e da organização.
        - O líder-servidor.
        - O facilitador hábil.
        - O facilitador de eventos.
        - O líder do time autogerenciado.
    - Competente em soft skills.
    - Presente.

8.1 Quem é o Scrum Master?

O Scrum Master é responsável por promover e apoiar o entendimento e o uso do Scrum. Utilizando-se, entre diferentes técnicas, de seu conhecimento de Scrum e de sua habilidade para lidar com pessoas e com grupos, o Scrum Master trabalha para facilitar e potencializar o trabalho do Time de Scrum. Ele ajuda o Product Owner e os Desenvolvedores do produto a buscarem

continuamente melhorar seus processos e práticas, de forma a se tornarem mais eficientes e eficazes na realização do seu trabalho e a tornarem o seu trabalho mais prazeroso.

O Scrum Master vela pela qualidade das interações entre os membros do Time de Scrum, e entre eles e pessoas externas. Para tal, ele atua como um facilitador e ensina aos envolvidos o que for necessário para promover interações que maximizem os resultados do trabalho do Time de Scrum. O Scrum Master, juntamente com o Product Owner e com os Desenvolvedores, compõe o Time de Scrum. Ele trabalha em prol do Time de Scrum como um todo, incluindo o Product Owner. É um erro comum focar apenas nos Desenvolvedores e deixar o Product Owner desassistido.

Para realizar esse trabalho, o Scrum Master:

- promove e apoia o uso do Scrum, trabalhando para que o framework seja compreendido e adequadamente utilizado pelo Time de Scrum;
- facilita o trabalho do Time de Scrum em seu dia a dia e nos eventos do Scrum, visando aumentar a sua autonomia e habilitar seus membros a trabalharem como um time no desenvolvimento do produto, comunicando-se efetivamente entre eles, buscando continuamente melhorar seus processos de trabalho e realizando esse trabalho com responsabilidade, qualidade e produtividade;
- garante a remoção dos impedimentos que atrapalham o trabalho dos Desenvolvedores e, quando possível, ajuda a prevenir que aconteçam;
- promove as mudanças organizacionais necessárias para que o Time de Scrum possa realizar seu

trabalho com efetividade.

O Scrum Master é:

- uma liderança a serviço do Time de Scrum, facilitando as interações entre seus membros conforme necessário, os apoiando e os habilitando a desenvolverem plenamente seu potencial e habilidades, a gerarem mais valor e a se tornarem mais autônomos;
- competente em *soft skills*, ou seja, ele possui competências comportamentais e pessoais como comunicação, facilitação e habilidades políticas. Ele é também corajoso, proativo e autoconfiante para realizar as mudanças necessárias, remover impedimentos e proteger o trabalho dos Desenvolvedores;
- presente, na medida do possível, durante o trabalho dos Desenvolvedores, para observar, identificar, ajudar a criar transparência e a resolver problemas, para remover impedimentos e para proteger os Desenvolvedores de interrupções nocivas.

Curiosidade: ScrumMaster ou Scrum Master?

O Scrum Master era chamado originalmente de "ScrumMaster", com as duas palavras juntas. Assim consta nos primeiros artigos desde a primeira definição do papel (BEEDLE ET AL., 1999; SUTHERLAND, 2001; SCHWABER, 2003; SUTHERLAND, 2004), nos primeiros livros sobre o Scrum (SCHWABER; BEEDLE, 2002; SCHWABER, 2004; 2007) e nas duas primeiras edições do Guia do Scrum (SCHWABER, 2009; 2010).

Com a conturbada saída de Ken Schwaber da Scrum Alliance em 2009 e a consequente criação da Scrum.Org, o nome foi modificado para "Scrum Master", em duas palavras separadas. Já apareceu assim na edição do guia de 2011 (SCHWABER; SUTHERLAND, 2011), quando Ken Schwaber e Jeff Sutherland, os criadores do Scrum, passaram a escrevê-lo em conjunto.

O que eu ouvi sobre as razões dessa mudança foi que "Certified ScrumMaster", a certificação oferecida pela Scrum Alliance, está registrada por essa entidade. Por essa razão, Ken Schwaber modificou o nome para "Scrum Master", de modo a evitar problemas de direitos na criação da sua própria certificação na Scrum.Org, a "Professional Scrum Master". Aqui no livro, a partir desta edição, eu escolhi adotar o nome definido oficialmente no Guia do Scrum, "Scrum Master".

8.2 O que faz o Scrum Master?

Promove e apoia o uso do Scrum

O Scrum Master age para que os valores, práticas e regras do Scrum sejam entendidos e seguidos por todo o Time de Scrum. Ele ensina o Ágil, o Scrum e a abordagem empírica para os Desenvolvedores e para o Product Owner. Ele fica atento a desvios na prática do Scrum e toma medidas que considere necessárias para estimular o time a corrigir esses desvios. É responsabilidade do Scrum Master promover a realização de todos os eventos do Scrum e o respeito aos horários e aos *timeboxes*. Ele também estimula os Desenvolvedores

a atualizarem o Sprint Backlog e a monitorarem a realização do Objetivo do Sprint.

O Scrum Master promove a autonomia do Time de Scrum, ensinando seus membros a se autogerenciarem em torno do Objetivo do Produto e do Objetivo de cada Sprint e, assim, aumentarem gradualmente a sua responsabilidade e capacidade de resolver seus próprios problemas e de chegar a suas próprias soluções. O Scrum Master também os estimula a se desenvolverem como um grupo multidisciplinar e a praticarem a melhoria contínua em sua forma de trabalhar, a fim de que sejam capazes de maximizar o valor gerado por eles.

Ele atua, sempre que julgar necessário, para que outras pessoas da organização entendam e apoiem o trabalho do Time de Scrum. Ao mesmo tempo, é ele que ensina ao Time de Scrum a operar em um ambiente em processo de transformação, muitas vezes hostil ao uso de Scrum.

O Scrum Master estimula o Product Owner a aprender técnicas de gestão do Product Backlog e de definição de objetivos, assegurando-se de que ele seja capaz de atualizar e organizar esse Product Backlog de modo a maximizar o valor gerado.

Ele também estimula o Product Owner e os Desenvolvedores a prepararem os itens do Product Backlog, interagindo entre eles e, de acordo com o contexto, com clientes, usuários e outras partes interessadas, de forma que cheguem à reunião de Sprint Planning com os itens de maior ordem preparados para entrarem em implementação.

Um bom Scrum Master, no entanto, não age como um fiscal dos processos, papel em que forçaria os envolvidos a usarem o Scrum corretamente. Ele também não realiza

partes do trabalho por eles para que o uso "correto" do Scrum aconteça. Na realidade, um dos importantes objetivos do Scrum Master é tornar-se cada vez menos necessário, aumentando a autonomia do Time de Scrum. Com esse propósito, o Scrum Master ensina e habilita os Desenvolvedores e o Product Owner a utilizarem o Scrum, de forma que cada vez menos dependam dele para que os valores, práticas e regras do Scrum sejam seguidos e usados.

Como exemplo, um comportamento disfuncional comum ocorre quando o Scrum Master avisa todos os dias aos Desenvolvedores quando é hora de realizarem a sua reunião diária. Ao agir dessa forma, no entanto, em vez de habilitá-los e torná-los independentes, o Scrum Master pode estar levando os Desenvolvedores a sempre dependerem dele para que a reunião ocorra. O Scrum Master fará melhor seu trabalho se for capaz de mostrar e convencer os Desenvolvedores sobre a importância de manterem a disciplina da realização da reunião e, assim, criarem o hábito e se organizarem para fazê-lo.

Facilita o trabalho do Time de Scrum

O Scrum Master facilita o trabalho do Time de Scrum, conforme necessário, para que se tornem cada vez mais efetivos. Ele promove interações significativas e com qualidade entre os Desenvolvedores e entre eles e o Product Owner, para as quais são essenciais a boa comunicação, a transparência e as relações de trabalho construtivas.

Essa facilitação pode ocorrer tanto nas reuniões de Sprint Planning, Sprint Review e Sprint Retrospective, em que a presença do Scrum Master é obrigatória, quanto no dia a dia de trabalho do time.

Recomendo fortemente que times novos ou com pouca experiência com Scrum contem com a facilitação do Scrum Master na Daily Scrum, ainda que sua presença na reunião não seja prescrita pelo framework. Também recomendo a facilitação do Scrum Master para as sessões de refinamento do Product Backlog (veja o capítulo *Refinamento do Product Backlog (adicional)*), em que haverá interações entre os Desenvolvedores, Product Owner e, possivelmente, outras partes interessadas.

Uma parte significativa desse trabalho de facilitação envolve a observação ativa dos comportamentos individuais e em conjunto, além da leitura de cenários. Por essa razão, na medida do possível, é importante que o Scrum Master maximize sua presença no dia a dia de trabalho dos Desenvolvedores.

Para mais informações sobre o papel do facilitador e técnicas de facilitação, veja *O facilitador hábil* e *O facilitador de eventos* na seção *Liderança a serviço do Time de Scrum* deste capítulo.

Garante a remoção de impedimentos

Impedimentos ameaçam o trabalho na busca de realizar o Objetivo do Sprint, pois dificultam significativamente ou obstruem o trabalho dos Desenvolvedores. O Scrum Master é o responsável pela resolução desses impedimentos, removendo-os ele mesmo ou mobilizando as pessoas e os recursos necessários para tal.

Um impedimento tipicamente acontece em um trabalho que já foi iniciado pelos Desenvolvedores e, assim, gera uma espera sobre um ou mais itens mais importantes naquele momento. Por essa razão, uma vez identificado por Desenvolvedores, o impedimento é imediatamente sinalizado para o Scrum Master, que por sua vez toma

ações rápidas e efetivas para a sua remoção. É importante que as soluções adotadas atuem diretamente sobre as possíveis causas raízes em vez de utilizarmos soluções paliativas.

O Scrum Master estimula os Desenvolvedores a identificarem se há impedimentos por vir. Assim, quando um ou mais Desenvolvedores se antecipam a um impedimento, o Scrum Master trabalha imediatamente para ajudar a evitar que ele aconteça.

No entanto, devemos saber distinguir problemas e questões do dia a dia de impedimentos no trabalho, sobre os quais atua o Scrum Master. A resolução de problemas está ao alcance e no contexto de atuação dos Desenvolvedores. Por essa razão, são tratados por eles próprios, ainda que possam contar com a ajuda do Scrum Master.

Natureza dos impedimentos e sua resolução

De acordo com sua natureza, os impedimentos podem ser classificados em organizacionais, básicos, administrativos ou de nível de serviço (PIMENTEL, 2009):

Organizacionais — ocorrem quando a própria estrutura ou funcionamento da organização os impõe aos Desenvolvedores. Podem acontecer em pelo menos três situações distintas:

- os Desenvolvedores necessitam, mas não obtêm a ajuda ou intervenção de outra pessoa, equipe ou área da organização. Nesses casos, o Scrum Master usa o acesso e influência que construiu e está construindo na organização para buscar, o mais rapidamente possível, o apoio necessário para os Desenvolvedores realizarem seu trabalho e monitorar se esse apoio foi realmente obtido.

Esse tipo de impedimento se configura, por exemplo, quando o equipamento de trabalho de um Desenvolvedor apresenta defeito, mas a equipe de suporte não resolve o problema em tempo hábil;

- outra pessoa, equipe ou área da organização intervém no trabalho de Desenvolvedores e demanda sua ajuda, desviando seu foco do Objetivo do Sprint. O Scrum Master tem o importante papel de proteger os Desenvolvedores de interferências externas que coloquem em risco o Objetivo do Sprint. De forma geral, o modo mais efetivo de realizar essa proteção em ambientes hierárquicos é utilizando política e o ensino do Scrum e de seus benefícios.

  Esse tipo de impedimento ocorre, por exemplo, quando um membro da organização em uma posição hierarquicamente superior aborda diretamente um Desenvolvedor para realizar alguma tarefa externa ao desenvolvimento do produto;

- questões de cunho político ou burocrático presentes na organização obstruem o trabalho dos Desenvolvedores. Como um agente de mudanças, o Scrum Master trabalha para modificar as práticas organizacionais em benefício da eficiência e eficácia dos Desenvolvedores em seu trabalho. Ele, por exemplo, pode se unir a Scrum Masters de outros times para realizar as mudanças necessárias na organização.

  A adoção de métodos ou práticas pesadas que conflitem com os Valores e Princípios Ágeis, como as que geram uma produção excessiva de documentação, constitui um exemplo desse tipo de impedimento;

Básicos — são impedimentos que ocorrem pela falta de um ou mais elementos essenciais à realização do trabalho em algum determinado momento. O Scrum Master trabalha para garantir que os Desenvolvedores possuam acesso aos meios necessários para realizarem suas tarefas.

Esse tipo de impedimento inclui, por exemplo, a falta de ferramentas ou equipamentos necessários para a realização do trabalho, a inexistência de um ambiente adequado ou de elementos essenciais nesse ambiente (como cadeiras ou mesas, por exemplo), e até mesmo a impossibilidade de acesso a dados ou informações necessárias;

- **administrativos** — são impedimentos provenientes de ocorrências administrativas, como absenteísmo (atrasos, licenças, folgas, faltas por motivos diversos etc.), demissões e restrições de horários de trabalho. Essas questões estão muitas vezes ligadas a áreas em que atua o Scrum Master, como políticas organizacionais ou problemas intraequipe que podem gerar desmotivação;

- **nível de serviço** — aqui são identificados impedimentos oriundos de problemas na sustentação de algum serviço em operação. O Scrum Master trabalha para criar transparência sobre o problema e monitora sua resolução junto à equipe que tem a atribuição de resolvê-lo, sendo muitas vezes necessário estimular o envolvimento dos Desenvolvedores nesse trabalho.

Exemplos desse tipo de impedimento incluem, para o trabalho de desenvolvimento de software, os erros em ambientes de produção e problemas com o servidor de

aplicações, com o servidor de integração ou com ferramentas diversas de apoio.

Promove as mudanças organizacionais necessárias

O Scrum Master atua como um agente de mudanças na organização em que está inserido o Time de Scrum, liderando e planejando a adoção de Scrum onde necessário. Ou seja, ele trabalha no entorno do Time de Scrum e em outras partes da organização para promover mudanças necessárias, de forma que o Time de Scrum possa realizar seu trabalho com mais produtividade e efetividade.

Esse trabalho pode envolver, por exemplo, ensinar Scrum para os membros do próprio Time de Scrum e para pessoas de quem eles dependam para realizar seu trabalho, de forma que possam ter interações com qualidade e colaborar em vez de gerar impedimentos.

No entanto, o mais importante, mas ao mesmo tempo o mais difícil e desafiador, pode ser o trabalho de promover as mudanças culturais necessárias para que o uso de Scrum seja efetivo. Para a maioria das organizações, a adoção da mentalidade ágil, do Scrum e da abordagem empírica implica em uma profunda mudança em como elas entendem e lidam com o trabalho, com as pessoas que realizam esse trabalho e com seus clientes e usuários (veja a introdução do capítulo *Por que Scrum?* e as seções *Scrum é ágil* e *Scrum é embasado no empirismo* do capítulo *O que é Scrum?*).

Em seu dia a dia, ao identificar um impedimento ao trabalho dos Desenvolvedores, o Scrum Master vai em busca de sua causa raiz. Quando esse impedimento tem origem no próprio funcionamento da organização, o Scrum Master atuará como um agente para promover as

mudanças necessárias e, assim, evitar que o impedimento ocorra novamente. Ele pode identificar, por exemplo, que é necessário modificar alguma regra ou estrutura ineficiente que gera desperdícios ou atrapalha o trabalho do Time de Scrum, e trabalhará para tal.

Um Scrum Master efetivo é persistente e perseverante, e não aceita que não consegue modificar a organização em que trabalha. Por mais difícil e moroso que esse trabalho possa parecer, promover as mudanças necessárias é uma parte essencial de suas atribuições.

Ele é capaz de transitar entre o Time de Scrum e a organização que o contém, construindo relacionamentos com pessoas-chave da organização que têm influência sobre o trabalho do Time de Scrum. A partir do bom relacionamento, da percepção política e da habilidade em realizar perguntas, ele obtém acesso a informações importantes e as utiliza para alinhar as necessidades e objetivos do Time de Scrum e os da organização (veja *O líder do time autogerenciado* em *Como é o Scrum Master?*, neste capítulo).

O Scrum Master pode atuar juntamente com outros Scrum Masters para realizar o trabalho aqui descrito, formando, por exemplo, uma comunidade responsável por essa atuação.

8.3 Como é o Scrum Master?

Liderança a serviço do Time de Scrum e da organização

Podemos afirmar que o Scrum Master exerce um tipo de liderança que serve ao Time de Scrum, com o intuito de

potencializar o seu trabalho, e à organização como um todo (SCHWABER; SUTHERLAND, 2020) para torná-la mais ágil. Para melhor descrevermos a natureza dessa liderança e trazermos algumas práticas que podem ajudá-lo nesse trabalho, utilizamo-nos de ideias já reconhecidas e aplicadas. O Guia Oficial do Scrum colocava abertamente, da edição de 2011 à edição de 2017, a liderança-servidora como o modelo a ser seguido (SCHWABER; SUTHERLAND, 2011, 2013, 2017). Essa prescrição foi atenuada na edição de 2020, mas segue útil como recomendação. Eu trago ainda os modelos do facilitador hábil, do facilitador de eventos e o de líder do time autogerenciado, que frequentemente nos são úteis em nosso trabalho na K21.

O líder-servidor

No modelo criado por Robert K. Greenleaf, um líder-servidor serve em primeiro lugar, assegurando-se que as necessidades de maior prioridade dos liderados estão sendo atendidas (GREENLEAF; SPEARS, 2002). Diferente da liderança tradicional, em que as pessoas trabalham para servir o líder, o líder-servidor trabalha para servir às pessoas. Seu status de líder deriva de sua livre aceitação pelos liderados em resposta à sua atuação como servidor. A liderança-servidora busca envolver os liderados no processo decisório, é fortemente apoiada em um comportamento ético e cuidador e promove o crescimento dos outros enquanto melhora o bem-estar e a qualidade da vida organizacional (SPEARS, 2010).

O Scrum Master serve aos Desenvolvedores, ao Product Owner e à organização como um todo. Enquanto líder-servidor, ele coloca as necessidades dos Desenvolvedores e do Product Owner em primeiro lugar e os encoraja, os apoia e os habilita a desenvolverem plenamente seu potencial e habilidades, a

desempenharem cada vez melhor e a se tornarem mais autônomos, ao mesmo tempo em que estão inseridos em um ambiente organizacional que favoreça esses resultados.

São reconhecidas dez características essenciais de líderes-servidores (SPEARS, 2010):

- saber ouvir: ouvir intencionalmente os liderados, entender o que está e o que não está sendo dito, buscar identificar seus objetivos e ajudá-los a esclarecer esses objetivos;
- empatia: empatizar com os liderados, de modo que se sintam aceitos e reconhecidos pelo que são. O líder-servidor assume sempre a boa intenção de seus liderados, não os rejeitando como pessoas mesmo quando considere certos comportamentos ou resultados não aceitáveis;
- reparação: o líder-servidor tem o potencial de reparar o indivíduo e sua relação com os outros. Essa reparação é uma importante força transformacional e integradora;
- consciência: a consciência geral e, em especial, a autoconsciência levam o líder-servidor a ter uma visão sistêmica e a estar alerta para perceber e entender situações envolvendo ética, poder e valores;
- persuasão: influenciar, convencer ou construir consenso para promover decisões na organização, em vez de utilizar autoridade, coerção ou qualquer tipo de poder formal;
- conceitualização: encarar os problemas ou a organização sob uma perspectiva que vai além da realidade do seu dia a dia e dos objetivos de curto prazo, abrangendo também, de forma balanceada,

uma visão conceitual mais ampla e, assim, definir um futuro a ser perseguido;
- antecipação: prever os resultados mais prováveis de uma situação ou de uma decisão, a partir do entendimento das lições do passado, das realidades do presente e de intuição;
- guarda: comprometer-se com servir, antes de mais nada, às necessidades de seus liderados e ao bem comum, atuando como um guardião dos mesmos;
- compromisso com o crescimento das pessoas: nutrir o crescimento pessoal e profissional dos liderados dentro da organização, por acreditar que eles têm um valor intrínseco maior que suas contribuições no trabalho;
- construção de comunidade: identificar formas de construir uma convivência em comunidade entre os liderados, estimulando um senso de pertencimento e o compartilhamento de normas sociais e de objetivos comuns.

O facilitador hábil

Na definição clássica de Roger Schwarz (2002), facilitação de um grupo é: *um processo pelo qual uma pessoa cuja escolha é aceitável para todos os membros do grupo, que é suficientemente neutra e que não possui autoridade considerável no processo decisório do grupo, diagnostica e intervém para ajudar o grupo a melhorar como ele identifica e resolve problemas e toma decisões, para aumentar a efetividade do grupo*. A partir dessa definição, Schwarz criou o modelo do "facilitador hábil".

Podemos utilizar o modelo criado por Schwarz para descrever e justificar algumas características que esperamos do Scrum Master no seu trabalho junto aos Desenvolvedores e ao Product Owner - que, somados ao próprio Scrum Master, constituem o Time de Scrum. Em

particular, adoto aqui a abordagem em que o facilitador ensina o time a melhorar seus processos de trabalho e a depender cada vez menos dele, chamado por Schwarz de "facilitação desenvolvimentista".

O facilitador habilita o time a aumentar sua efetividade. A principal atribuição do Scrum Master enquanto facilitador é ajudar o Time de Scrum a aumentar a sua efetividade. Para tal, ele ajuda a habilitar os Desenvolvedores e o Product Owner a melhorarem:

- **seus processos**: como trabalham juntos, ou seja, como se comunicam, como é o seu processo criativo, como identificam e resolvem problemas, como lidam com conflitos e como se relacionam com seu entorno;
- **sua estrutura**: características relativamente estáveis importantes para o funcionamento do grupo, como a existência de objetivos claros a serem realizados, de uma composição do time adequada para a realização de suas tarefas, de tarefas motivadoras, de valores e crenças consistentes com um time efetivo, de normas ou acordos explícitos derivados desses valores e crenças, e de tempo suficiente para realizar seu trabalho e para melhorar sua forma de trabalhar;
- **seu contexto**: como o contexto no qual o time está inserido (por exemplo, sua organização ou setor) influencia a sua efetividade, como a existência de objetivos organizacionais claros, de uma cultura organizacional que apoie o uso do Scrum e de práticas de gestão que tornem o time mais efetivo, da premiação de comportamentos do time (e não do indivíduo) consistentes com os objetivos do time, de informações necessárias para realizar o trabalho, de feedback sobre o seu trabalho, de acesso a

treinamento e consultoria que se façam necessários para possibilitar a resolução de problemas, aumentar seus conhecimentos e desenvolver habilidades necessárias para realizar seu trabalho, de acesso ao material e tecnologia necessários e de um ambiente físico adequado.

O Scrum Master, enquanto facilitador, intervém diretamente no processo, estrutura e contexto do Time de Scrum para ajudá-lo a se tornar mais efetivo. Mas, ao mesmo tempo, o Scrum Master estimula o Time de Scrum a refletir sobre como pode melhorar seu processo, estrutura e contexto, e ensina-o a desenvolver as habilidades necessárias para atuar neles por si só, pois a responsabilidade de aumentar sua efetividade pertence ao próprio time.

Essas mudanças, no entanto, só podem ocorrer se o Time de Scrum possuir a autoridade necessária para realizá-las e se todos os seus membros compartilharem a responsabilidade por essas mudanças. Embora o Time de Scrum não possua controle direto sobre seu contexto, ele pode influenciá-lo e provocar mudanças que o possibilitem a se tornar mais efetivo.

O facilitador é neutro. O Scrum Master, enquanto facilitador, é uma pessoa suficientemente neutra, que tem como objetivo aumentar a responsabilidade e capacidade do grupo de resolver seus próprios problemas. Assim, ele não interfere diretamente no conteúdo das discussões do grupo. Para que essa neutralidade seja possível, o Scrum Master preferencialmente não acumula a responsabilidade de Desenvolvedor ou de Product Owner, que têm suas opiniões e interesses e, assim, são invariavelmente parciais.

Uma situação de tomada de decisão poderia expor rapidamente a parcialidade do Scrum Master que também é Desenvolvedor, debilitando sua capacidade de atuação como um facilitador. Imagine quando os Desenvolvedores se dividem em opiniões diferentes e esse facilitador-Desenvolvedor participa em uma das opiniões. Será que a outra parte, que tem opinião contrária à do facilitador, vai lidar bem com isso?

Situações de conflito envolvendo o facilitador-Desenvolvedor, que invariavelmente ocorrerão, podem ser ainda mais ainda problemáticas, já que ele inevitavelmente será parcial e impossibilitado de atuar como facilitador para ajudar a resolver a questão.

O facilitador habilita o time a aumentar sua autonomia. Uma parte importante do trabalho do facilitador é habilitar os membros do Time de Scrum a realizarem, em conjunto, suas próprias escolhas e decisões a partir do uso de informações suficientes. Ao realizarem essas escolhas livres e informadas, eles naturalmente tornam-se comprometidos com elas e, ao mesmo tempo, as revisam e as mantêm atualizadas. O trabalho do Scrum Master, enquanto facilitador, é chave para possibilitar o autogerenciamento.

Um elemento importante para possibilitar as escolhas livres e informadas é a transparência. Para assegurá-la, o Scrum Master evita fazer arranjos com determinados membros do Time de Scrum ou com terceiros, seja para ocultar informações de uns ou de outros, seja para dissimular comportamentos. Ao contrário, o Scrum Master ajuda os membros do Time de Scrum a garantirem a transparência das informações disponíveis.

De forma geral, o time é mais efetivo se é internamente comprometido com suas escolhas. No entanto, mesmo

que o Scrum Master esteja realizando uma facilitação adequada e mesmo que haja informações suficientes disponíveis, as decisões do time são livres e, dessa forma, podem ser ruins. E, ainda que o Scrum Master acredite que uma determinada escolha não é apropriada, sua intervenção não é, a princípio, recomendável. Seu papel, nesse caso, é o de estimular o time a refletir sobre as consequências de suas decisões e ajustá-las de acordo. Ao intervir com suas próprias opiniões ou, pior, tomar a decisão pelo time, o Scrum Master reduz tanto a autonomia do time quanto sua própria credibilidade enquanto facilitador.

Para aumentar a autonomia do Time de Scrum e desenvolver sua efetividade, as intervenções do Scrum Master, enquanto facilitador, buscam sempre reduzir a dependência do Time de Scrum no próprio Scrum Master. Dessa forma, podemos afirmar que o Scrum Master trabalha em direção a se tornar cada vez menos necessário, inclusive habilitando os membros do Time de Scrum a atuarem como facilitadores eles mesmos.

O facilitador não é intermediário ou representante. O Scrum Master também não atua como um intermediário entre os outros membros do Time de Scrum, ou entre o Time de Scrum e pessoas externas a ele. Ao contrário, ele estimula os membros do Time de Scrum a desenvolverem as habilidades de se comunicarem e lidarem diretamente com quem se fizer necessário.

O Scrum Master também não atua como representante do Time de Scrum ou de qualquer de seus membros diante de outros membros ou de pessoas externas. Como consequência, não é o Scrum Master, por exemplo, que apresenta ao resto da organização os resultados do

trabalho ou o desempenho do Time de Scrum ou de algum de seus membros.

Ele não tem autoridade para utilizar qualquer informação obtida a partir da facilitação para influenciar decisões externas sobre membros do Time de Scrum como, por exemplo, bonificações ou punições. Caso o fizesse, a confiança dos outros membros do Time de Scrum no Scrum Master ficaria comprometida, assim como sua neutralidade e, consequentemente, sua efetividade como facilitador. Esse tipo de informação, caso necessária, é fornecida diretamente pelos próprios membros do Time de Scrum.

Da mesma forma, o Scrum Master não atua como intermediário entre Desenvolvedores e Product Owner ou entre diferentes Desenvolvedores. Ele, ao contrário, estimula que se comuniquem diretamente.

O facilitador é um especialista nos processos, não no conteúdo. Como um especialista no processo, o Scrum Master sabe quais elementos mais contribuem para tornar o Time de Scrum mais efetivo.

Uma parte importante desses elementos são os próprios conhecimentos e habilidade de facilitação. Como parte de seu trabalho, o Scrum Master ensina ao time a utilizar elementos de facilitação em suas interações, aumentando assim a autonomia do time.

Outra parte importante dos processos de trabalho é o próprio framework Scrum, que deve ser corretamente compreendido e utilizado pelo Time de Scrum. O Scrum Master então ensina as regras, responsabilidades, eventos e artefatos do Scrum ao time como um todo, sempre que necessário, e a pessoas da organização que

precisem desse conhecimento para serem capazes de apoiar o trabalho do time.

Portanto, embora não interfira diretamente no conteúdo das discussões do grupo, o Scrum Master atua junto ao Time de Scrum no nível de processos, trazendo à luz esses processos e ensinando-os aos envolvidos sempre que julgar necessário.

O facilitador de eventos

Michael Wilkinson (2004), no livro *The secrets of facilitation: the S.M.A.R.T. guide for getting results with groups*, define uma sessão facilitada como: *um encontro altamente estruturado no qual o facilitador guia os participantes por meio de uma série de passos predefinidos para que cheguem a um resultado que é criado, compreendido e aceito por todos os participantes*. Ainda segundo o autor, técnicas de facilitação são apropriadas sempre que um grupo necessita de criar uma compreensão comum e de aceitação para obter sucesso em chegar a um determinado resultado.

Além de facilitar o dia a dia de trabalho do Time de Scrum, o Scrum Master tem a importante função de atuar como um facilitador nos eventos do Scrum. Ele estimula os envolvidos a participarem ativamente das discussões, ajuda-os a manter o foco nos objetivos do evento e ajuda-os a chegarem a conclusões compartilhadas.

Podemos interpretar qualquer um dos eventos do Scrum como uma sessão facilitada, que possui um caminho determinado a ser percorrido e um resultado específico a ser realizado. O resultado esperado da reunião de Sprint Planning, por exemplo, é um plano para o Sprint, ou seja, o Sprint Backlog e o Objetivo do Sprint. O resultado

esperado da reunião de Sprint Review é o feedback dos clientes e demais partes interessadas sobre o Incremento ou Incrementos do produto implementados pelos Desenvolvedores durante o Sprint.

As características de um facilitador de eventos descrito por Wilkinson são bastante similares às do "facilitador hábil", descrito anteriormente. De acordo com o autor, o facilitador de eventos não interfere no conteúdo das discussões, nem oferece soluções ou suas próprias opiniões. Ele, na realidade, serve ao grupo guiando os participantes por meio de passos predefinidos para que alcancem resultados específicos, utilizando seu conhecimento de dinâmicas de grupo e dos passos do processo.

O facilitador cria um ambiente em que os participantes se sentem à vontade para contribuir. Em geral, ele responde a questionamentos com perguntas e sabe realizar as perguntas certas, que estimularão os participantes a chegarem a suas próprias conclusões.

O facilitador procura prevenir comportamentos disfuncionais que possam ocorrer durante a sessão, ajudando o grupo, por exemplo, a estabelecer suas regras básicas de conduta. Ele reconhece e lida com comportamentos disfuncionais quando ocorrem, buscando soluções a partir de suas reais causas. O facilitador também ajuda o grupo a manter seu foco nos resultados da reunião e a construir um consenso em torno das soluções geradas.

Ao final, os resultados da sessão facilitada devem ter sido criados, compreendidos e aceitos por todos os participantes.

O líder do time autogerenciado

Os trabalhos de Cummings (1978), Manz & Sims (1987) e Druskat & Wheeler (2003), a partir de pesquisas realizadas sobre a prática, principalmente na indústria, identificam um tipo de liderança frequentemente emergente em times autogerenciados. Eu utilizei essas referências em minha dissertação de Mestrado (Armony, 2010).

Times autogerenciados são adotados em diferentes áreas de trabalho. Embora possa parecer paradoxal, a prática mostra que times autogerenciados muitas vezes possuem líderes. Sua função primária, no entanto, é a de facilitar o autogerenciamento, e não a de gerenciar a equipe. O líder de um time autogerenciado, portanto, tem o importante papel de *liderar o time em direção a que o time lidere a si mesmo*.

O Scrum Master possui muitas das características de um líder de times autogerenciados. Esse tipo de líder facilita o autogerenciamento principalmente ao encorajar o time à auto-observação e autoavaliação, de forma que o time monitore, torne consciente e avalie seu desempenho; e do encorajamento do time ao autorreforço, de forma que seus membros demonstrem entre si que valorizam um bom trabalho feito por um colega e valorizam o alto desempenho do time.

Essas ações de encorajamento são mais efetivas quando o líder consegue criar uma relação de confiança e de atenção com os membros do Time de Scrum. Essa relação facilita o levantamento de informações que determinem os tipos de ações que devem ser encorajadas.

Para serem efetivos, líderes de times autogerenciados realizam tanto funções voltadas internamente quanto voltadas externamente ao time, atuando na organização

que o contém e atravessando constantemente a fronteira entre os dois lados. O líder efetivo é capaz de construir relacionamentos tanto com membros do time quanto com outros indivíduos ou unidades da organização que podem afetar o desempenho do time (como a alta gerência, áreas de apoio e outros grupos) e de buscar entender ambas as perspectivas.

Nesse trabalho, a pessoa líder de um time autogerenciado não tem poder nem dá ordens: ela na verdade obtém acesso a informações-chaves de ambos os contextos por meio do bom relacionamento, da percepção política e da habilidade em realizar perguntas. Ela utiliza essas informações-chaves para alinhar as necessidades do time com as da organização e vice-versa, de forma a fazer com que tanto o time quanto a organização se comportem de modo a facilitar a efetividade de ambos.

Esse trabalho envolve, por exemplo, evitar que uma decisão tomada pelo time, que de fato cabe ao time, seja rejeitada pela organização. Isso também permite ao líder promover o poder de ação e de decisão do time, o que o leva a assumir uma maior responsabilidade por seus resultados.

Competente em soft skills

O indivíduo que atua bem como um Scrum Master possui competências comportamentais e pessoais que lhe permitem realizar o seu trabalho. Essas competências são chamadas de *soft skills*.

A facilitação do trabalho do Time de Scrum, parte importante do trabalho do Scrum Master, inclui promover a detecção e transparência dos problemas e assegurar a

boa relação entre os Desenvolvedores e entre esses e o Product Owner.

O Scrum Master utiliza-se de boa comunicação e de voz ativa, mas sem comandar, para realizar esse trabalho. Ele também usa de suas habilidades de negociação e de política para estimular a resolução de conflitos e promover a tomada de decisão do Time de Scrum em direção ao consenso. Um Scrum Master que não tenha o perfil adequado pode encontrar dificuldades para atuar em suas responsabilidades.

Um Scrum Master autoconfiante, corajoso e proativo atua como um agente de mudanças na organização e, utilizando-se de persistência e perseverança, promove as mudanças necessárias para que o Time de Scrum possa entregar valor para seus clientes.

O Scrum Master faz uso dessas mesmas habilidades para se articular em diferentes níveis organizacionais com o propósito de realizar a remoção de impedimentos e de defender os Desenvolvedores (e ensiná-los a se defenderem) de interferências externas e de mudanças que possam afetar o Objetivo do Sprint.

A seguir, podemos ver exemplos de *soft skills* desejáveis em um Scrum Master:

- habilidade de comunicação;
- saber ouvir;
- flexibilidade e adaptabilidade;
- capacidade de negociação e habilidade política;
- capacidade de resolução de problemas e pensamento crítico;
- habilidade de ensinar, por meio de *coaching* ou *mentoring*;
- habilidade de facilitação;

- colaboratividade;
- autoconfiança;
- autoconsciência;
- paciência;
- persistência e perseverança;
- sociabilidade;
- gestão de seu tempo;
- atitude positiva.

Discussão: o Scrum Master necessita de conhecimento técnico?

As habilidades necessárias para o trabalho técnico - como, por exemplo, a programação de computador no desenvolvimento de _software - e as habilidades necessárias para o trabalho do Scrum Master são muito diferentes. Concordo, no entanto, que o conhecimento técnico pode ajudar o Scrum Master a se inserir mais rapidamente no contexto de trabalho dos Desenvolvedores, além de facilitar sua leitura de cenários em cada situação.

O que vejo na prática, na realidade, é que um bom Scrum Master não realmente necessita desse conhecimento para realizar o trabalho. Utilizando-se de *soft skills*, ele vai gradualmente aprender a linguagem dos Desenvolvedores, entender os termos importantes de seu trabalho, se entrosar e conquistar o seu respeito.

Ao evitar um Scrum Master técnico, queremos também evitar um problema comum nesses cenários: ele usar esse conhecimento para interferir nas discussões (e, portanto, no conteúdo) do trabalho dos Desenvolvedores, prejudicando sua autonomia. Eu mesmo, tendo um

background técnico forte no desenvolvimento de software, tive que lidar com essa questão com cuidado no meu início de carreira como Scrum Master.

Presente

O Scrum Master presente toma ciência imediatamente de obstáculos que estejam impedindo o trabalho dos Desenvolvedores e pode rapidamente tomar ações para removê-los, antes que ameacem o Objetivo do Sprint. O Scrum Master atento a desvios na prática do Scrum pode agir para corrigi-los, ensinando e habilitando os Desenvolvedores e o Product Owner a usarem o Scrum corretamente.

O Scrum Master que acompanha o trabalho do Time de Scrum é capaz de detectar comportamentos disfuncionais, conflitos ou problemas de relacionamento entre os Desenvolvedores ou entre os Desenvolvedores e o Product Owner, e atuar o mais rápido possível para ajudar a resolver esses problemas. Ele, presente, pode proteger os Desenvolvedores de interferências externas que ameacem o Objetivo do Sprint, o que pode acontecer em qualquer momento. Além disso, ele atua como um facilitador nos eventos do Scrum sempre que necessário, e não há como fazê-lo sem estar neles presente.

Em suma, o Scrum Master idealmente se faz presente sempre que o Time de Scrum necessitar dele, o que pode significar um trabalho em tempo integral. Infelizmente, é comum Scrum Masters trabalharem em tempo parcial ou com mais de um time ao mesmo tempo. Nesses casos, o Scrum Master pode ser necessário em momentos em que não estará presente. Desse modo, em vez de economizar dinheiro, a organização vai desperdiçar a oportunidade

de ter plenamente alguém especializado em potencializar o trabalho do Time de Scrum.

De forma geral, o Scrum Master em tempo integral é importante para times que trabalham juntos há pouco tempo, para times que estão começando a utilizar Scrum, para times jovens, para times inseridos em organizações hierárquicas e burocráticas e para times disfuncionais, ou seja, aqueles que enfrentam problemas graves na forma como operam.

Discussão: o Scrum Master também pode ser...?

Vejo muitas adoções de Scrum em que a pessoa que cumpre o Scrum Master também trabalha parte de seu tempo como Desenvolvedor. Embora essa divisão de trabalho não seja proibida, a atuação dupla pode sacrificar a neutralidade do Scrum Master e, assim, dificultar seu trabalho como facilitador.

Não é incomum que as sugestões ou decisões desse Scrum Master acabem por ter, para os outros Desenvolvedores, um peso maior do que suas próprias opiniões e que, assim, invariavelmente terminem por acatá-las. Esse tipo de situação pode ficar mais evidente em momentos de crise.

Quando esse cenário ocorre, o Scrum Master está fazendo o oposto do que deveria ser seu trabalho: habilitar os Desenvolvedores, em conjunto, a aumentarem sua autonomia. Além disso, principalmente devido à falta de tempo e diferenças no tipo de trabalho, as atividades de remoção de impedimentos, facilitação e atuação como agente de mudanças organizacionais podem ser prejudicadas quando o Scrum Master compartilha de uma atuação técnica.

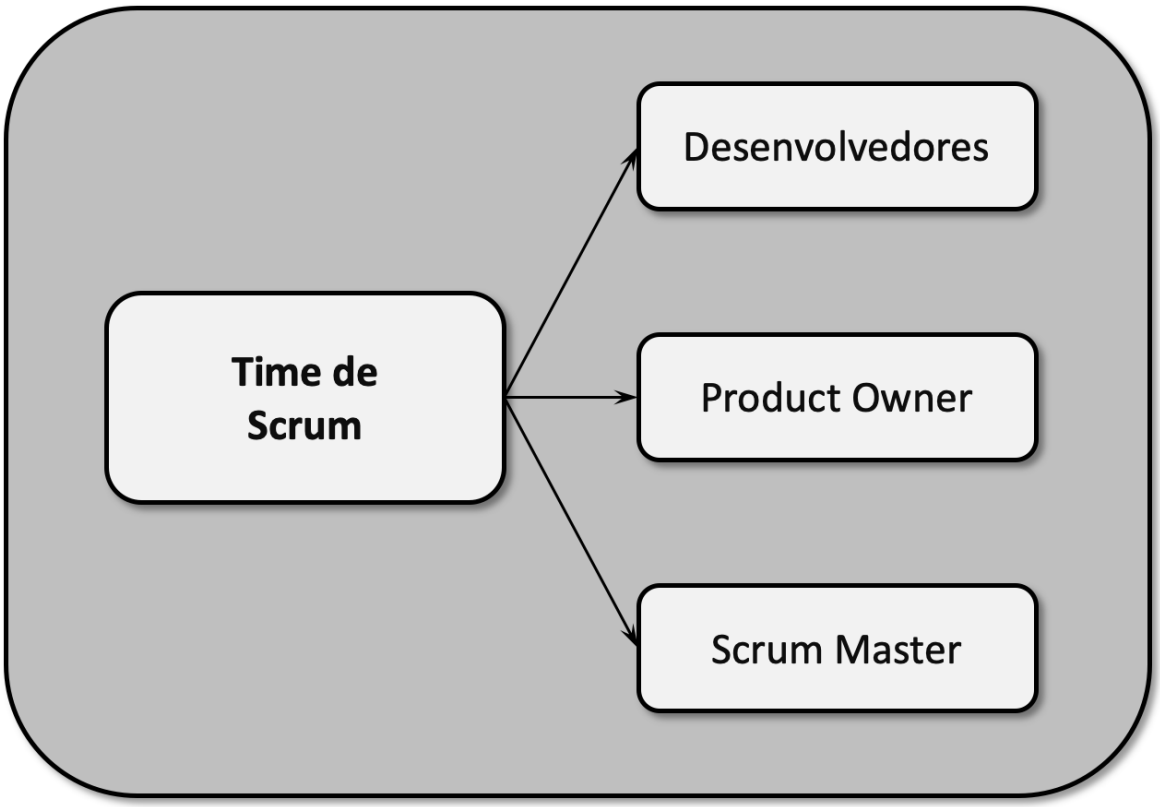
Mesmo assim, esse muitas vezes pode ser o ponto de partida mais viável em sua organização. Ao não entender bem o que é um Scrum Master, a gestão não pode aceitar a contratação de alguém em tempo integral e acabar empurrando suas responsabilidades para um Desenvolvedor.

Se você estiver nessa situação, a solução não é recusar a adoção de Scrum. Entenda bem suas responsabilidades, identifique as limitações que a jornada dupla está produzindo e, enquanto agente de mudança, trabalhe dentro da organização para pouco a pouco viabilizar o seu trabalho e de outros Scrum Masters em tempo integral.

No entanto, recomendamos fortemente que o Scrum Master não acumule as responsabilidades do Product Owner em um mesmo Time de Scrum. Essa recomendação constava como regra nas duas primeiras versões do guia oficial do Scrum (SCHWABER, 2009, 2010), mas foi retirada em seguida (SCHWABER; SUTHERLAND, 2011) para tornar o framework menos prescritivo.

Caso as responsabilidades de Scrum Master e Product Owner se concentrem em um único indivíduo, o conflito de interesses fica exacerbado. Não há ninguém para facilitar a relação entre o Product Owner e os Desenvolvedores. Não há ninguém para defender os Desenvolvedores e educar o Product Owner se, disfuncionalmente, o próprio Product Owner buscar interferir no trabalho durante o Sprint. Nesses casos, a concentração de poder pode deixar esse indivíduo com características próximas às de um gerente de projetos tradicional, exercendo comando e controle sobre os Desenvolvedores.

Artefatos e compromissos do Scrum



Capítulo 9
Sobre os artefatos e os compromissos do Scrum

Conteúdo

1. O que são os artefatos e os compromissos do Scrum?
    - Definição.
2. Quais são os artefatos e compromissos do Scrum?
    - Artefatos e compromissos do Scrum.

9.1 O que são os artefatos e os compromissos do Scrum?

Definição

Um artefato no Scrum é criado pelo Time de Scrum para representar trabalho a ser implementado, valor a realizar ou valor realizado. Os artefatos do Scrum oferecem informações essenciais para o alinhamento entre os membros do Time de Scrum e, dependendo do artefato, entre eles e outras partes interessadas. Essa transparência permite a todos aqueles que estão inspecionando o artefato utilizarem a mesma base para a adaptação (SCHWABER; SUTHERLAND, 2020).

O Scrum define o uso de três artefatos: o Product Backlog, o Sprint Backlog e o Incremento. Cada artefato do Scrum carrega em si um compromisso, que é um objetivo ou um conjunto de critérios a ser cumprido pelo Time de Scrum a partir do uso ou da criação do próprio artefato. Os compromissos são estabelecidos em um alto nível, o suficiente para não determinarem como eles próprios serão realizados. Dessa forma, buscam estimular o empirismo. O Time de Scrum mantém seu foco no compromisso de cada artefato e, dessa forma, os têm como referência durante a execução de seu trabalho.

Cada artefato oferece informações suficientes de forma que o progresso na realização de seu compromisso possa ser compreendido. O compromisso do Product Backlog é o Objetivo do Produto, o do Sprint Backlog é o Objetivo do Sprint e o do Incremento é a Definição de Pronto. Adiciono ainda a Definição de Preparado como mais um compromisso do Product Backlog.

Curiosidade: compromissos

O guia do Scrum de 2020 introduziu o conceito de *compromisso*. Com essa adição, Scrum definiu oficialmente o Objetivo do Produto - que substitui o conceito de visão de produto, já utilizado há tempos pela comunidade - e melhor enquadrou os já conhecidos Objetivo do Sprint e a Definição de Pronto, agora classificados como compromissos e diretamente ligados a artefatos.

9.2 Quais são os artefatos e os compromissos do Scrum?

Artefatos e compromissos do Scrum

Os artefatos do Scrum são:

- o Product Backlog, que é uma lista ordenada, dinâmica e incompleta de itens que serve como a única fonte para o trabalho dos Desenvolvedores;
- o Sprint Backlog, que é um plano que contém o trabalho que os Desenvolvedores do produto acreditam que vão realizar durante o Sprint para a implementação de um ou mais Incrementos do produto, e que define o valor a ser realizado;
- o Incremento, que é o resultado pronto da implementação de um ou mais itens do Sprint Backlog pelos Desenvolvedores do produto durante o Sprint, e é utilizável por clientes e usuários do produto, potencialmente representando valor para eles.

Os compromissos do Scrum são:

- o Objetivo do Produto, que expressa o propósito ou motivação central para o trabalho do Time de Scrum no desenvolvimento do produto, a ser satisfeito a partir da implementação de itens do Product Backlog. O Objetivo do Produto é o compromisso do Product Backlog;
- o Objetivo do Sprint, que expressa um propósito claro definido para um Sprint, a ser realizado ou satisfeito a partir da implementação de itens do seu Sprint Backlog correspondente. O Objetivo do Sprint é o compromisso do Sprint Backlog;
- a Definição de Pronto, que é um entendimento formal compartilhado entre os Desenvolvedores e o Product Owner sobre o que é necessário para que considerem que qualquer item de um Sprint Backlog ou qualquer Incremento está pronto.

Como opção a se adicionar ao Scrum, sugiro:

- a Definição de Preparado, que é um entendimento formal compartilhado entre Product Owner e os Desenvolvedores do produto sobre o estado em que um item do Product Backlog deve estar para ser considerado preparado o suficiente para entrar no Sprint Backlog e, portanto, para ser implementado. Entendo que a Definição de Preparado pode ser vista como mais um compromisso do Product Backlog, uma vez que seus itens devem cumpri-la para poderem entrar em um Sprint e ser implementados.

Capítulo 10
Product Backlog

Conteúdo

1. O que é o Product Backlog?
    - Definição e uso.
    - Gestão do Product Backlog.
2. Como é o Product Backlog?
    - O Product Backlog.
        - Ordenado.
        - Planejável.
        - Emergente.
        - Gradualmente detalhado.
    - Os itens do Product Backlog.

10.1 O que é o Product Backlog?

Definição e uso

O Product Backlog é um artefato do Scrum na forma de uma lista dinâmica e incompleta de itens que refletem o que acreditamos, no momento atual, que será implementado para o produto visando realizar o Objetivo do Produto (veja o capítulo *Compromisso: Objetivo do Produto*). Essa lista é ordenada, o que significa que no alto da lista estão seus itens mais importantes para a realização do Objetivo do Produto e que sempre dali são selecionados e retirados os próximos itens para implementação. O Objetivo do Produto também faz parte do Product Backlog.

O Product Backlog é um artefato vivo e, assim, está em constante evolução. À medida que o produto vai sendo desenvolvido, entregue e utilizado, mais vamos conhecendo e entendendo as necessidades emergentes dos seus usuários, e o produto evoluirá de acordo com isso. O Product Backlog é então modificado com a adição, subtração, reordenamento e refinamento de seus itens, que terão a granularidade e o nível de detalhes de acordo com quão próximos estiverem de ser implementados. Dessa forma, os itens do topo do Product Backlog são de granularidade mais fina, representam mais detalhes e podem possuir uma estimativa com maior precisão do esforço necessário para sua implementação. Na outra ponta, os itens da parte de baixo do Product Backlog são de granularidade mais grossa, representam menos detalhes e possuem estimativas mais grosseiras.

O Product Backlog é a única fonte de trabalho para os Desenvolvedores, que da mesma forma são os únicos a consumirem seus itens para realizar o desenvolvimento do produto. Ninguém pode forçá-los a trabalhar a partir de outra fonte.

Gestão do Product Backlog

O Product Owner é o responsável pela gestão efetiva do Product Backlog, definindo o seu conteúdo, realizando sua ordenação e garantindo sua disponibilidade para os Desenvolvedores do produto. Ele pode, no entanto, delegar esse trabalho para os Desenvolvedores e até mesmo para terceiros (veja, no capítulo *Product Owner*, a seção *Gerencia a definição do produto*). No entanto, o Product Owner se mantém responsável pelos resultados.

O trabalho de gestão do Product Backlog inclui a interação frequente com clientes e com demais partes

interessadas, visando a realizar o Objetivo do Produto e a maximizar o valor gerado. Ou seja, muitos contribuem para a definição do produto e influenciam as decisões sobre o que deve e o que não deve ser implementado no produto. Mas apenas o Product Owner possui autoridade sobre o Product Backlog e, assim, apenas ele pode tomar ou delegar essas decisões.

10.2 Como é o Product Backlog?

O Product Backlog

A analogia do horizonte, que eu trouxe no capítulo *Por que Scrum?*, descreve bem a natureza do Product Backlog (veja *Planejar utilizando apenas o nível possível de detalhes*, em *Redução do desperdício*). Essa analogia nos mostra que o Product Backlog é ordenado, planejável, emergente e gradualmente detalhado.

Ordenado

Os itens do Product Backlog são ordenados de acordo com a ordem em que serão implementados e, consequentemente, entregues. Essa ordenação é, em princípio, realizada pelo Product Owner. Ela tem como propósito permitir que realizemos incrementalmente o Objetivo do Produto, enquanto satisfazemos as necessidades de seus usuários e geramos valor de negócio para os clientes do produto.

A partir da ordenação, as funcionalidades que os usuários mais necessitam em cada momento chegam às suas mãos. Com o seu uso, obtemos feedback e métricas que nos permitem adicionarmos ao Product Backlog itens

com os ajustes e novas funcionalidades, ordenados para maximizar a geração de valor.

Na reunião de Sprint Planning, o Product Owner e os Desenvolvedores selecionam um número de itens a partir do topo do Product Backlog para serem implementados no Sprint que se inicia. Como vemos na figura a seguir, os itens abaixo destes poderão ser implementados em Sprints seguintes. Os itens mais abaixo no Product Backlog poderão ser implementados em Sprints futuros.

Quanto mais alto no Product Backlog o item estiver, mais chances de ser implementado ele terá, e mais cedo isso poderá acontecer. Por outro lado, quanto mais baixo no Product Backlog o item estiver, menor a sua ordem de implementação e, assim, menores são suas chances de ser realizado, já que mudanças podem fazê-lo perder o sentido ou ser significativamente modificado.

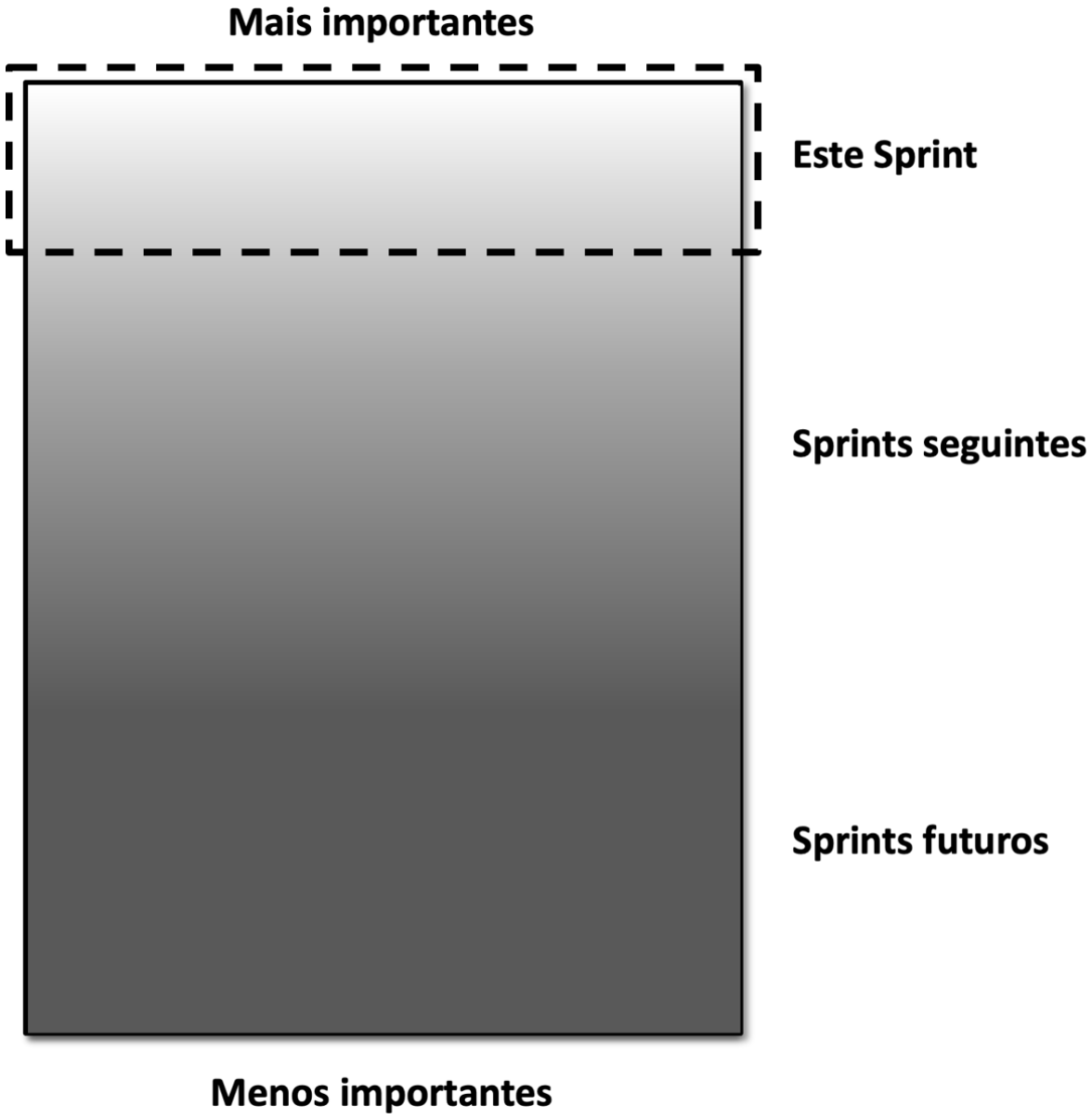


Figura 10.1: O Product Backlog é ordenado

Discussão: ordenado ou priorizado?

Na edição de 2011 do guia oficial do Scrum, os termos "priorizar", "priorizado" e "priorização", relativos ao Product Backlog, foram substituídos por "ordenar", "ordenado" e "ordenação" (SCHWABER; SUTHERLAND, 2011). James Coplien (2011) escreveu um artigo

explicando as razões a pedido dos criadores do Scrum, Ken Schwaber e Jeff Sutherland.

De acordo com Coplien, priorizar uma lista significa arranjar seus itens de acordo com a importância relativa entre eles. Dessa forma, ao priorizarmos itens do Product Backlog, realizamos comparações dois a dois, do tipo "esse item é mais importante do que aquele" ou "esse item tem maior valor (ou maior retorno) do que aquele". Ou, de forma parecida, comparamos grupos de itens, atribuindo a eles etiquetas como "alta prioridade", "média prioridade" e "baixa prioridade" ou similares. São, portanto, comparações locais, sem um olhar mais amplo que consideraria o Objetivo do Produto e o seu contexto. A priorização, portanto, banaliza o complexo trabalho do Product Owner de determinar a ordem em que os itens do Product Backlog serão implementados e entregues.

Para ordenar os itens do Product Backlog, no entanto, o Product Owner utiliza uma visão mais holística, visando a maximizar o valor gerado. Ao ordenar, em vez de priorizar, ele entende esse valor como o resultado do uso do produto no tempo e considera o que consegue enxergar desse futuro. Assim, essa ordenação é diretamente influenciada por importantes questões como riscos, oportunidades, dependências externas, o conhecimento disponível para implementar esses itens e mudanças nas condições do mercado.

A ordenação dos itens do Product Backlog é um trabalho contínuo, que acontece ao longo de todo o trabalho de desenvolvimento do produto. Para determinar a ordem dos itens, o Product Owner considera diversos fatores, entre eles:

- o valor de negócio e o custo de implementação de cada item;
- as interdependências entre os itens e as dependências de fatores externos;
- os riscos associados a cada item;
- a incerteza sobre o comportamento do usuário;
- questões técnicas trazidas pelo time ou times;
- urgências do mercado;
- questões burocráticas ou políticas;
- o conhecimento atual dos Desenvolvedores sobre tecnologias ou regras necessárias;
- a estratégia de negócios dos clientes;
- a estratégia de negócios da organização.

O trabalho de ordenação não é simples e requer conhecimento do negócio, das oportunidades e riscos do mercado, da concorrência, da capacidade de implementação do conjunto de Desenvolvedores, de técnicas de validação e invalidação de hipóteses e de uma série de outros fatores. O Product Owner busca gradualmente o conhecimento e a assistência necessários para realizar esse trabalho.

Planejável

Um Product Backlog planejável permite que os Desenvolvedores e o Product Owner sejam capazes de, a partir das informações contidas no Product Backlog, realizar o planejamento do trabalho para o Sprint que se inicia, além de vislumbrar, em mais alto nível, o que poderá ser realizado em Sprints futuros.

De forma geral, o Product Backlog ser planejável significa que ele permite:

- estabelecer o escopo mais provável do Sprint atual. Ou seja, o que é mais provável que esteja dentro dos

Incrementos do produto resultantes de um Sprint ou, em mais alto nível, dentro de um conjunto de Incrementos a serem implementados a seguir;

- definir quando o próximo ou próximos Incrementos do produto a serem implementados poderão ser entregues para os clientes, ou seja, a data da entrega, dado um escopo provável;
- calcular a capacidade provável de implementação do conjunto de Desenvolvedores, em geral a partir da média do que eles foram capazes de implementar nos Sprint recentes, também chamada de Velocidade (veja em *Velocidade*, no capítulo *Story Points: estimando o trabalho*);
- monitorar o progresso do trabalho em direção a uma data planejada (final do Sprint ou uma data de entrega, por exemplo), dado o trabalho e o tempo restantes. Gráficos de Acompanhamento do Trabalho são ferramentas que podem ser usadas com esse propósito (veja no capítulo *Burndown e Burnup: acompanhando o trabalho*).

Emergente

O Product Backlog é uma lista continuamente em evolução e, assim, nunca está terminado ou completo. À medida que os Desenvolvedores trabalham sobre os itens do Product Backlog escolhidos para o Sprint, transformando-os em partes em funcionamento do produto, itens são adicionados, removidos, modificados, refinados e reordenados. Esse é um trabalho que ocorre continuamente enquanto o Time de Scrum estiver trabalhando sobre o produto e, potencialmente, enquanto o produto existir.

O Product Backlog evolui em um processo contínuo de descoberta, refletido na inspeção e adaptação do produto realizada a partir do feedback de clientes e demais partes interessadas sobre os Incrementos do produto implementados e mostrados a eles, e a partir de métricas de uso do que já foi entregue para seus usuários até então.

Gradualmente detalhado

O Product Backlog é progressivamente refinado ao longo de todo o trabalho de desenvolvimento do produto, de forma que seus itens possuam detalhamento e granularidade adequados.

Itens do topo do Product Backlog são os próximos a serem implementados e, assim, devem representar detalhes suficientes e necessários para que sejam elegíveis para serem implementados. Esses itens possuem granularidade fina, ou seja, exigem um menor esforço de implementação.

Apenas itens de granularidade fina podem ser implementados. Na K21, tratamos de fatias, em referência à Analogia do Bolo (veja *Scrum é iterativo e incremental*, no capítulo *O que é Scrum?*). Cada fatia representa valor para o usuário, algo que ele pode utilizar diretamente. Assim, apenas fatias finas de funcionalidade podem fazer parte do Sprint Backlog de um Sprint.

Definição: fatias finas

Fatiar fino os itens a serem implementados traz diversas vantagens. Entre elas, posso citar:

- possibilita uma melhor ordenação, já que fatias que se originam de diferentes itens originalmente maiores podem ter prioridades bem diferentes entre elas;
- permite, com mais de uma fatia, resolver mais rapidamente a parte mais importante de um problema, de ponta a ponta;
- permite que a resolução de um problema, de ponta a ponta, caiba mais facilmente em um Sprint;
- como consequência, podemos entregar mais valor mais cedo;
- traz uma menor complexidade associada à funcionalidade a ser implementada;
- como consequência, permite uma mais rápida e melhor compreensão, tanto do time quanto de clientes e usuários;
- ainda como consequência, possibilita a realização de melhores estimativas;
- representam hipóteses que necessitam de menor esforço de implementação e, assim, geram menos desperdício se invalidadas.

Itens mais abaixo no Product Backlog possuem granularidade gradativamente mais grossa e representam menos detalhes. À medida que um item vai se aproximando de sua implementação, ele sobe no Product Backlog, pode evoluir em itens menores e os detalhes que serão necessários para sua implementação são esclarecidos e elaborados.

Discussão: qual o melhor Product Backlog?

Na figura a seguir, apresento três possíveis formatos para o Product Backlog.

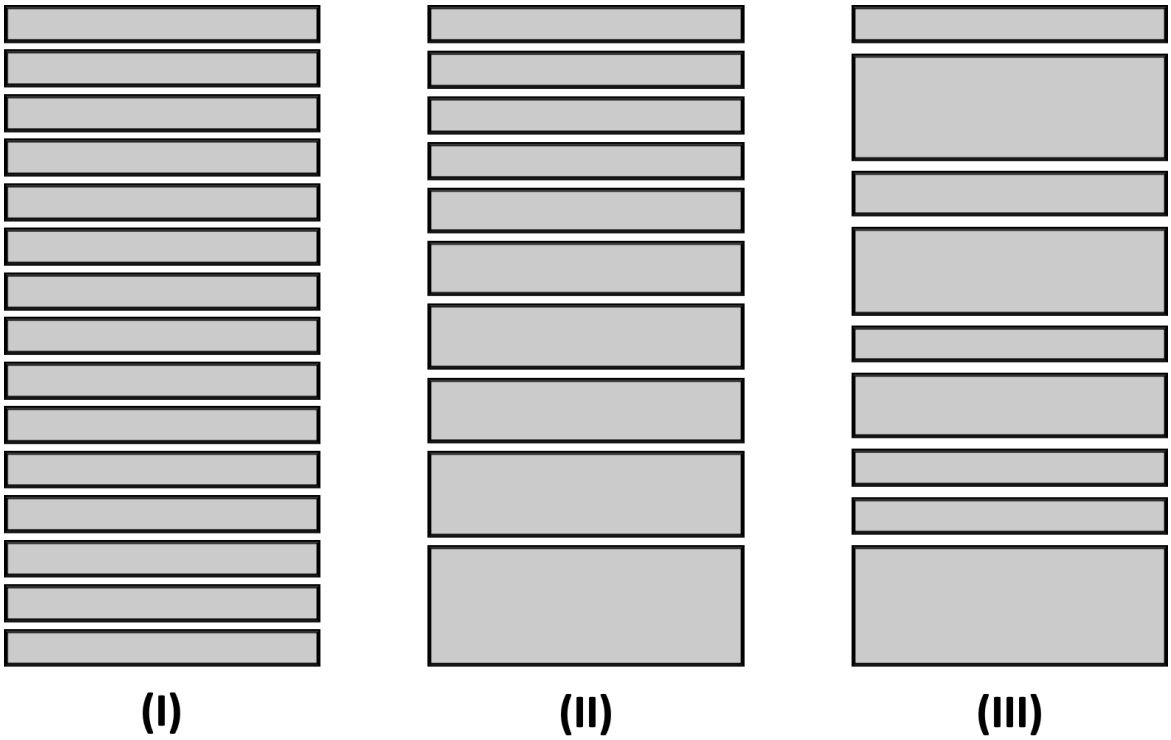


Figura 10.2: O Product Backlog é gradualmente detalhado

No Product Backlog (I), todos os itens têm granularidade fina e carregam detalhes. Mais detalhes do que o suficiente e necessário constituirão desperdício, já que, pela natureza emergente do Product Backlog, os detalhes são conhecidos e modificados à medida que o produto é desenvolvido.

No Product Backlog (III), um item do alto tem granularidade grossa, o que significa que colocá-lo para ser implementado traria riscos, pois significa um tempo mais longo para fazê-lo, com uma ordenação pobre e com uma maior incerteza associada. Para corrigir esse problema, será necessário fatiar esse item em um ou mais alguns poucos itens pequenos e detalhados, que carregarão as partes mais importantes e manterão a alta ordem. O restante será movido, em um único item grande e menos detalhado, mais para baixo no Product Backlog, para talvez ser implementado no futuro.

Além desse item de granularidade grossa no alto, o Product Backlog (III) traz diversos outros mais abaixo que possuem granularidade mais fina e trazem mais detalhes do que o necessário, o que novamente pode constituir desperdício, já que há uma grande chance de que os detalhes mudem até o momento de sua implementação. Refinar esse Product Backlog se faz necessário.

O Product Backlog (II) é o que considero o ideal, pois é gradualmente detalhado, ou seja, possui detalhes suficientes e necessários, de acordo com a ordem do item.

Os itens do Product Backlog

Os itens do Product Backlog definem o que os Desenvolvedores implementarão para o produto. Esses itens de trabalho podem ser expressos na forma de características, comportamentos, funcionalidades, experimentos, melhorias, correções de problemas (SCHWABER; SUTHERLAND, 2017), pesquisas que se façam necessárias, objetivos de negócios dos clientes e demais partes interessadas e necessidades dos usuários.

Recomendo que os itens do Product Backlog sejam representados como comportamentos ou características do produto sob a perspectiva dos usuários, na forma de User Stories.

Definição: User Stories

User Stories foram inventadas pelos criadores da metodologia Extreme Programming (JEFFRIES et al., 2000), e podem ser utilizadas para representar cada um dos itens do Product Backlog.

Uma User Story é uma descrição concisa e simples de uma característica ou comportamento do produto sob o ponto de vista de um usuário do produto. A User Story também é, no Scrum, um convite para conversas entre os Desenvolvedores e o Product Owner (ou outras partes interessadas) para estabelecer os detalhes do que será implementado para criar essa característica ou comportamento no produto.

O uso de User Stories, dessa forma, estimula a comunicação sobre o que deve ser feito a partir de conversas, em vez do uso de documentação para esse mesmo fim. Elas também ajudam os Desenvolvedores a manter seu foco nos usuários finais e em quais benefícios são esperados que eles obtenham, em vez de trabalharem orientados a tarefas técnicas.

Um formato comumente utilizado para User Stories estabelece quem é o usuário, o que o usuário poderá fazer com essa característica ou comportamento e qual valor o usuário obterá com ela, como pode ser visto a seguir:

Eu, *[QUEM]*, quero *[O QUÊ]* tal que *[POR QUÊ]*

Recomendo a leitura do capítulo *User Stories: representando o trabalho*.

O Scrum define que cada item do Product Backlog possui atributos como uma descrição, ordem e tamanho, sendo que esse último nem sempre é utilizado (veja *No estimates: estimativas são desperdício* no capítulo *Story Points: estimando o trabalho*). Esses atributos variam de acordo com o contexto.

O Product Backlog é uma lista linear, com itens em sequência e sem subitens, ordenada de forma que jamais

haja dois itens com a mesma ordem.

O Product Backlog possui alta visibilidade para o Time de Scrum e é facilmente reordenável. Existem softwares especializados que permitem o trabalho com o Product Backlog, mas uma planilha ou até mesmo notas adesivas em um quadro branco podem funcionar muito bem (veja a figura a seguir).

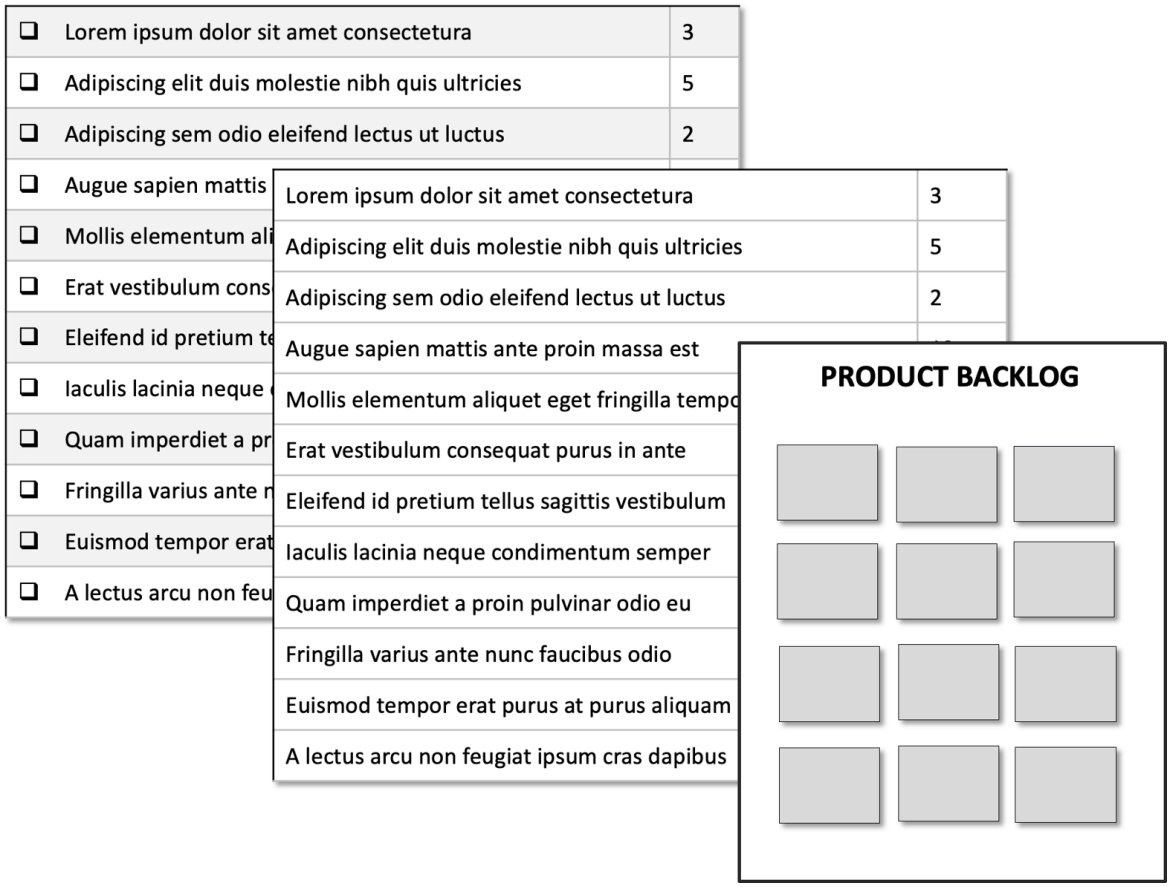


Figura 10.3: Três exemplos de formatos de Product Backlog

Discussão: lista de erros?

O Product Backlog é a única fonte de trabalho a ser realizado no produto pelos Desenvolvedores.

Esse trabalho também inclui as correções de erros encontrados no produto. Ou seja, erros em funcionalidades já implementadas deverão dar origem a itens de correção, adicionados ao Product Backlog na ordem em que serão implementados, com relação aos outros itens do Product Backlog.

Para o desenvolvimento de software, por exemplo, evitamos o uso tão comum de ferramentas de *bug tracking*, pois elas se sobreporiam ao uso do Product Backlog.

Capítulo 11
Compromisso: Objetivo do Produto

Conteúdo

1. O que é o Objetivo do Produto?
    - Definição e uso.
    - Como é criado o Objetivo do Produto?
    - Como é realizado o Objetivo do Produto?
2. Como é o Objetivo do Produto?
    - O Objetivo do Produto.
    - Teste do Elevador.
    - Caixa da Visão do Produto.
    - É/Não é/Faz/Não faz.
3. Objetivos intermediários.
    - Objetivo da Entrega.
        - O que é o Objetivo da Entrega?
        - Como é o Objetivo da Entrega?
    - Roadmap do produto.
        - O que é o roadmap do produto?
        - Como é o roadmap do produto?

11.1 O que é o Objetivo do Produto?

Definição e uso

O Objetivo do Produto expressa o propósito ou motivação central para o trabalho do Time de Scrum no desenvolvimento do produto, a ser satisfeito a partir da implementação de itens do Product Backlog. É um objetivo ou necessidade de alto nível definida pela

perspectiva do negócio ou pela perspectiva de seus usuários.

O Objetivo do Produto oferece uma direção comum a todos, permitindo que o plano para realizá-lo, na forma do Product Backlog, emerja de forma iterativa e incremental e seja frequentemente inspecionado e adaptado. O Objetivo do Produto fornece contexto, alinhamento, orientação, motivação e inspiração para o trabalho de desenvolvimento do produto. É o objetivo maior a ser realizado pelo Time de Scrum, que se traduz na satisfação dos clientes (PICHLER, 2010).

O Objetivo do Produto é o compromisso do Product Backlog, e dele faz parte, assim como o Objetivo do Sprint é o compromisso do Sprint Backlog de um Sprint.

Definição: produto

O guia do Scrum de 2020 introduziu (finalmente) uma definição formal para *produto*. O produto é *um veículo para entregar valor. Ele possui limites bem definidos, partes interessadas conhecidas e usuários ou clientes bem definidos. Um produto pode ser um serviço, um produto físico ou algo mais abstrato* (SCHWABER; SUTHERLAND, 2020).

O produto é o meio pelo qual o Time de Scrum, motivado e direcionado pelo Objetivo do Produto, gera valor para seus clientes, a partir da implementação de itens do Product Backlog. Para produzir esse valor, o produto é desenvolvido, Sprint após Sprint, Incremento após Incremento, e entregue com frequência a seus usuários finais, que o utilizam. Enquanto houver um Time de Scrum executando Sprints sobre o produto, ele estará em

contínuo desenvolvimento, ou seja, criação, modificação, evolução e manutenção.

Embora o produto seja desenvolvido de forma colaborativa com todo o Time de Scrum, o Product Owner é, em última instância, o responsável pelo produto e por seus resultados.

Scrum foi inicialmente criado para o desenvolvimento de produtos de software. No entanto, já há tempos que o framework não se restringe a essa finalidade. Em 2011, o Guia do Scrum teve praticamente todas as referências a software retiradas e o framework foi generalizado para diferentes tipos de produtos (SCHWABER; SUTHERLAND, 2011).

Na prática, vi ao longo do tempo o uso do Scrum se estendendo a diversos tipos de propósitos dentro do que chamamos de trabalho criativo. Dentre esses, o desenvolvimento de diferentes tipos de hardware (incluindo carros de corrida), a criação de campanhas de marketing e até mesmo o desenvolvimento e a execução de serviços.

O Objetivo do Produto é claro o suficiente para que tanto os Desenvolvedores do produto quanto o Product Owner e as demais partes interessadas estejam alinhadas em torno dele, entendendo seu significado e o impacto esperado a partir dele.

O Objetivo do Produto pode ser compartilhado por mais de um Time de Scrum, trabalhando no desenvolvimento de um mesmo produto.

Como é criado o Objetivo do Produto?

O Objetivo do Produto é estabelecido antes que o trabalho de desenvolvimento do produto se inicie. Apesar de normalmente ele se manter relativamente estável, ele é modificado sempre que isso se mostrar necessário.

Em princípio, o Product Owner é o responsável direto pela criação e pela atualização do Objetivo do Produto, bem como por sua gestão e compartilhamento. Desenvolvedores, clientes e outras pessoas relevantes também podem estar diretamente envolvidos na criação e refinamento desse objetivo.

Existem casos, no entanto, em que sua definição vem de fora do Time de Scrum, mas mesmo assim ele se mantém sob a responsabilidade do Product Owner. Em qualquer caso, o Objetivo do Produto está sempre alinhado a objetivos da organização.

Durante o trabalho de desenvolvimento do produto, o Product Owner e os Desenvolvedores garantem continuamente que os itens do Product Backlog estejam alinhados com o Objetivo do Produto, e trazem visibilidade caso identifiquem que ele não é mais válido ou que há problemas com ele.

Curiosidade: visão de produto e Objetivo do Produto

O guia do Scrum de 2020 introduziu o Objetivo do Produto como o compromisso do Product Backlog no Scrum (SCHWABER; SUTHERLAND, 2020), que faz parte do próprio Product Backlog. Até então, o Objetivo do Produto não existia oficialmente no framework e era senso comum utilizar o termo "visão do produto" para suprir essa lacuna.

Eu vejo a definição de mercado de "visão de produto" como um sinônimo para o Objetivo do Produto no Scrum.

Como é realizado o Objetivo do Produto?

O Time de Scrum, como um todo, é responsável pela realização do Objetivo do Produto e possui todas as habilidades e conhecimentos necessários para isso. Com esse fim, eles desenvolvem o produto, Sprint após Sprint, Incremento após Incremento, a partir de um plano incompleto e dinâmico, o Product Backlog.

Em cada Sprint, os Desenvolvedores implementam itens retirados do alto do Product Backlog para realizar o Objetivo do Sprint, que por sua vez representa um incremento à realização do Objetivo do Produto.

À medida que o produto vai sendo desenvolvido, entregue e utilizado, aprendemos melhor sobre como realizar o Objetivo do Produto e o Product Backlog evolui de acordo.

11.2 Como é o Objetivo do Produto?

O Objetivo do Produto

O Objetivo do Produto é expresso dentro do Product Backlog. Ele é uma proposta de valor que visa a oferecer direção e promover o alinhamento entre as diversas partes interessadas em torno do que esperam como resultado do trabalho.

Pensando tanto nesse propósito quanto no Princípio Ágil da simplicidade (veja a seção *Scrum é ágil* do capítulo *O*

que é Scrum?), preferimos algum artefato simples, conciso, direto e claro para representar o Objetivo do Produto, contendo apenas o suficiente e necessário. Documentos de dezenas ou até centenas de páginas, mapas ou *canvas* complicados recheados de informações e detalhes do que será o produto tendem a gerar dificuldade de entendimento, interpretações diferentes e um consequente desalinhamento, portanto não servindo bem para esse propósito.

O Objetivo do Produto inclui, como um mínimo, uma descrição dos clientes ou dos usuários do produto e qual valor será gerado para eles.

Existem diversas técnicas para criarmos uma visão de produto, que portanto podem ser aproveitadas para o Objetivo do Produto. Em seguida, apresento algumas dessas técnicas que considero úteis.

Teste do Elevador

Geoffrey Moore, no seu livro *Crossing the chasm* (2001), apresenta um modelo interessante para a visão do produto, o chamado "Teste do Elevador" ou *Elevator Pitch*. A ideia original é que seja possível explicar o que é o produto para um potencial investidor durante a subida de um elevador, ou seja, em um tempo bastante curto, de forma a convencê-lo a investir no produto.

Jim Highsmith adaptou o modelo para a visão do produto no trabalho com o Ágil. Aqui, apresento essa versão com pequenas modificações:

PARA (usuário mais importante),

que (problema que o usuário mais importante enfrenta),

o (nome do produto) **é um** (categoria do produto)

que (benefício-chave, razão convincente para utilizar).

Ao contrário de (alternativa primária competidora),

nosso produto (diferenciação com relação à alternativa primária).

Um exemplo de aplicação para um site de relacionamentos pode ser visto em seguida:

PARA jovens solteiros em busca de um relacionamento sério,

que estão insatisfeitos com encontros malsucedidos e a oferta de numerosas opções pouco criteriosas,

o LoveFinder **é um** site promotor de encontros amorosos

que os ajuda a encontrar sua alma gêmea.

Ao contrário de sites de encontros populares,

nosso produto reúne pretendentes compatíveis entre si, de acordo com o perfil, história de vida e preferências pessoais.

Outro exemplo, agora para um aplicativo móvel de viagens, pode ser visto a seguir:

Para turistas usuários de smartphone,

que estão cansados de que lhes ofereçam sempre as mesmas viagens óbvias e sem graça,

o MyTrip **é um** aplicativo móvel de viagens

que sugere roteiros diários flexíveis de acordo com seu perfil.

Ao contrário de guias de viagens com roteiros predefinidos,

nosso produto elabora trajetos personalizados e adaptáveis.

Vamos a um último exemplo, dessa vez para um treinamento nosso:

Para gestores sem experiência com o trabalho ágil,

que têm dificuldade em apoiar os seus times,

o treinamento de Gestão 3.0 da K21 é um treinamento com certificação internacional

que ensina a mentalidade ágil para gestão e diversas ferramentas para uso imediato.

Ao contrário dos treinamentos da concorrência,

nosso treinamento, além de divertido e prático, traz experiências reais.

O "Teste do Elevador" é um enunciado de visão do produto curto e efetivo que informa, de forma priorizada, quem são os usuários mais importantes do produto, qual

problema desses usuários será resolvido, um nome e categorização que fornecem um posicionamento do produto, uma razão convincente para utilizar o produto e a diferenciação do produto diante da principal alternativa existente.

Para **usuário mais importante**, colocamos o usuário considerado mais importante no momento, aquele em que estamos nos focando principalmente. Caso o foco do desenvolvimento do produto mude para outro usuário, a visão é revisada e reescrita de acordo.

Para o **problema enfrentado**, eu acredito que criamos uma maior empatia ao descrevermos a dor do usuário, e não apenas um desejo ou necessidade. Repare que, no exemplo do site de relacionamentos, falamos de uma insatisfação do usuário, e não algo como *que querem receber boas opções para encontros*.

Escolha um **nome** relevante para o produto e uma **categoria** que ajude a localizar melhor qual o tipo de produto. Como guia, imagine em que categoria de uma loja de aplicativos de celular esse produto estaria relacionado ou, talvez, em que departamento estaria disponível caso fosse vendido em uma loja de departamentos.

O **benefício-chave** é a principal razão pela qual o usuário utilizaria o produto, ou seja, o valor central que o produto traz para ele.

A **alternativa primária** pode ser o nome de um produto concorrente direto ou a forma como o usuário resolve atualmente o problema, por exemplo, *...ao contrário do trabalho manual*. A **diferenciação primária** define como esse produto se diferencia da alternativa primária competidora. Em vez de listar os aspectos negativos da

alternativa primária junto a ela, liste aqui os aspectos positivos do produto em comparação a ela.

Caixa da Visão do Produto

A criação da Caixa da Visão do Produto é uma atividade lúdica, em que os participantes constroem uma caixa para o produto a ser desenvolvido como se, hipoteticamente, ele fosse ser vendido nas prateleiras de uma loja. Ao criar a caixa, os participantes colaboram para gerar um alinhamento, ainda em alto nível, sobre o que será o produto.

A caixa carregará algumas características importantes para a visão.

A parte da frente da caixa tem o objetivo de atrair o "comprador", ao passear pela loja e ver a caixa na prateleira. Para tal, desenhamos a frente com um nome atrativo, uma imagem relacionada relevante e alguns tópicos curtos e diretos para vender o produto.

A parte de traz da caixa, uma vez que o possível comprador já está com a caixa em suas mãos, vai ajudá-lo na decisão de compra. Assim, consta nessa parte uma descrição mais detalhada dos atributos do produto, em um parágrafo ou em tópicos, e requisitos ou restrições de uso (veja a figura a seguir).

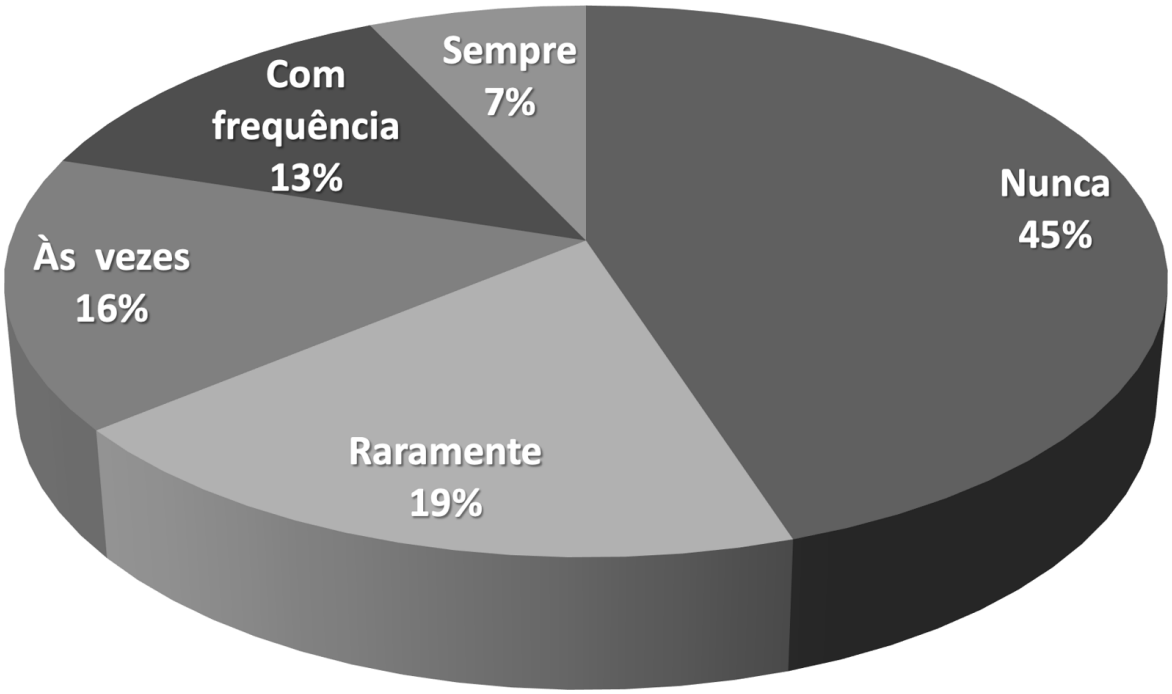


Figura 11.1: Caixa da Visão do Produto

A atividade de criação da caixa é geralmente realizada pelo Product Owner em colaboração com os Desenvolvedores, que ficam com um artefato concreto e ao alcance de suas mãos para consulta durante todo o trabalho.

Geralmente, como Objetivo do Produto eu costumo utilizar a Caixa da Visão do Produto em conjunto com o Teste do Elevador, de forma a complementá-lo.

Uma versão mais moderna e adaptada da Caixa de Visão de Produto pode ser realizada sobre uma cartolina ou uma folha de *flipchart*, que ficará visível em uma parede no local de trabalho. Imaginamos aqui que o produto será vendido em uma loja de aplicativos de celular e, assim, a página de venda do produto contará com uma imagem, um resumo que deve vender o produto e mais detalhes, que podem aparecer ao desdobrarmos a folha.

É/Não é/Faz/Não faz

Essa é uma técnica que criei para realizar exercícios sobre as responsabilidades do Scrum em minhas aulas. Um grande amigo, Paulo Caroli, esteve presente em uma dessas aulas e adaptou a técnica para a definição de um Produto Mínimo Viável em seu trabalho. Hoje, essa técnica faz parte da cultura da minha empresa e a utilizamos bastante no nosso dia a dia como uma forma de facilitar o alinhamento sobre diversos temas. Alguns usos incluem a definição de missão ou visão, de um papel ou de alguma nova iniciativa.

Dividimos uma folha grande ou um quadro em quatro quadrantes: é, não é, faz, não faz, e utilizamos notas adesivas (veja a figura a seguir). Um grupo de pessoas colabora em torno desse quadro. Para a visão de produto, caso possível, esse grupo é idealmente formado por Product Owner, clientes, outras partes interessadas e Desenvolvedores.

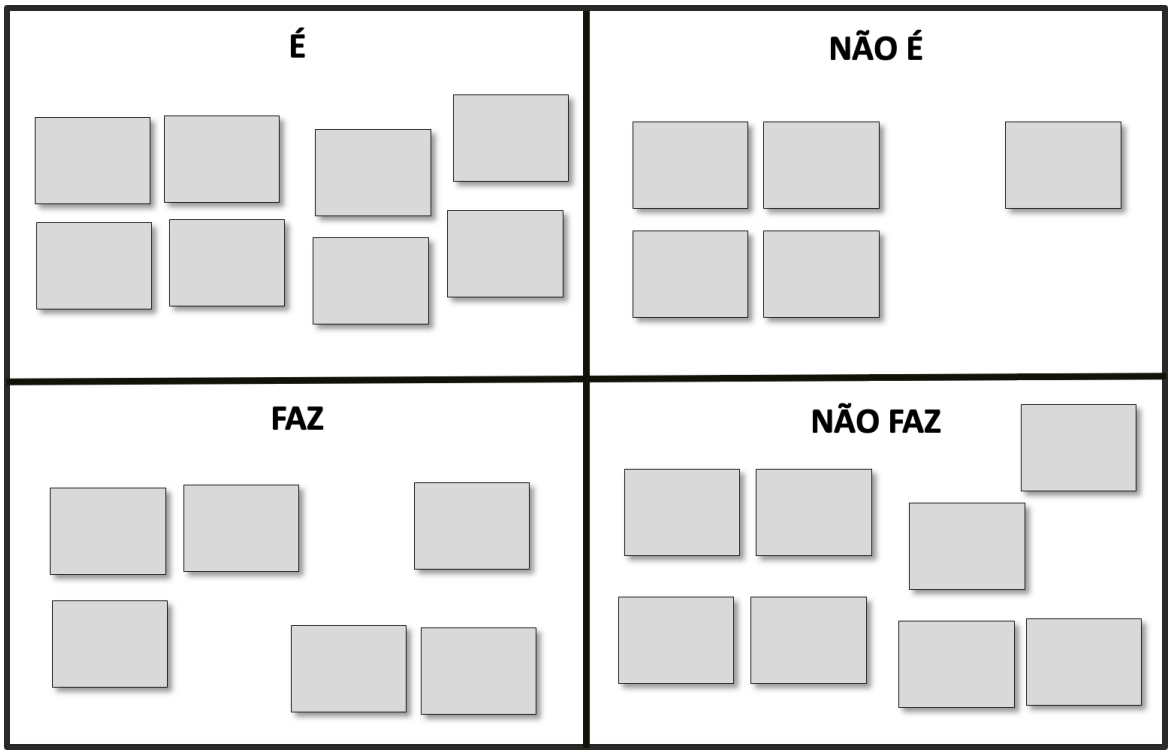


Figura 11.2: Quadro do É/Não é/Faz/Não faz

Colaborativamente, os participantes escrevem em notas adesivas:

- **"É"** - o que esperam que o produto seja, em suas características e atributos;
- **"Não é"** - o que esperam que o produto não seja. Aqui, buscamos delimitar a definição do produto, listando aquelas características e atributos que, pelo que ele é, poderíamos imaginar erradamente para o produto;
- **"Faz"** - o que esperam que o produto será capaz de realizar. Aqui, buscamos preferivelmente listar os problemas que o produto vai resolver, mas também podemos listar suas principais funcionalidades em alto nível;
- **"Não faz"** - o que esperam que o produto não vai realizar. Aqui, buscamos delimitar as ações do produto, listando os problemas (ou funcionalidades

de alto nível) que, pelo que ele faz, poderíamos imaginar erradamente que o produto resolverá.

Geralmente, para o Objetivo do Produto usamos essa técnica em conjunto com o Teste do Elevador, de forma a complementá-lo.

11.3 Objetivos intermediários

Scrum define apenas dois objetivos, que são compromissos a serem cumpridos no desenvolvimento de um produto: o Objetivo do Produto e os Objetivos de Sprints. Não é incomum, no entanto, utilizarmos objetivos intermediários entre eles.

Os membros do Time de Scrum, em seu trabalho, continuamente realizam esses objetivos claros e factíveis, com os quais se comprometem. Para tal, eles implementam desde os itens de maior ordem em direção aos de menor ordem.

Objetivos de Entrega e objetivos expressos em marcos de um *roadmap* do produto são exemplos de objetivos intermediários que podemos utilizar no trabalho de desenvolvimento de um produto com Scrum. Nesse caso, como podemos ver na figura a seguir, os Desenvolvedores implementam as funcionalidades em um Sprint para realizarem o Objetivo do Sprint; um conjunto de Objetivos de Sprint leva à realização de um objetivo intermediário, como um Objetivo de Entrega ou um marco do *roadmap* do produto; e os objetivos intermediários, em conjunto, levam à realização do Objetivo do Produto.

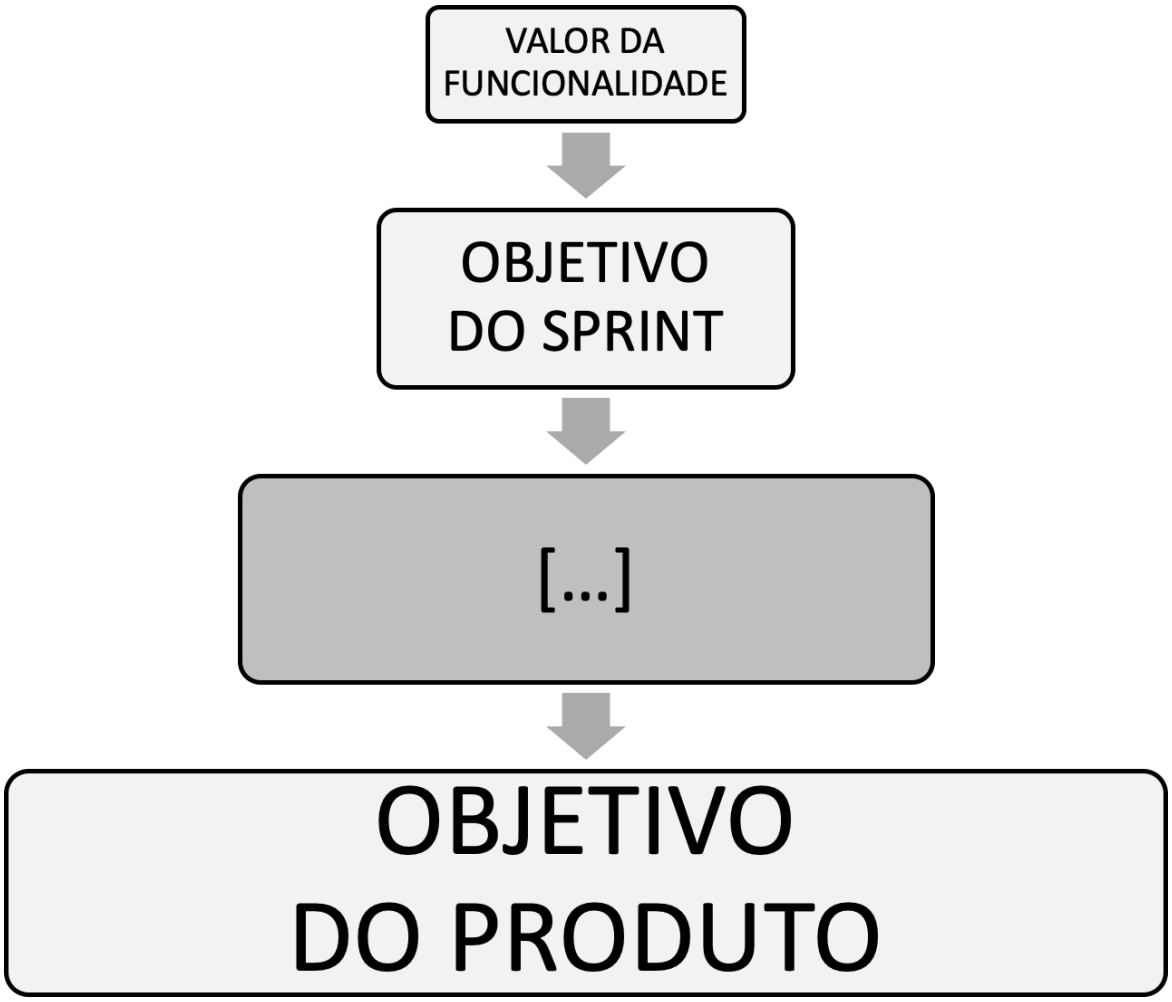


Figura 11.3: Do valor da funcionalidade ao Objetivo do Produto

Podemos entender que o conjunto formado pelos Objetivos do Sprint e pelos objetivos intermediários representa uma estratégia evolutiva para realizar o Objetivo do Produto, por meio do trabalho do Time de Scrum.

A execução dessa estratégia de produto existe sob a responsabilidade do Product Owner e é, assim como o próprio desenvolvimento do produto, iterativamente definida, detalhada, reavaliada e modificada de acordo com o feedback obtido sobre o produto e mudanças em seu contexto.

Os objetivos, no Scrum, visam prover foco e direção para o trabalho, incentivando o autogerenciamento. Eles funcionam ao mesmo tempo como guias e motivadores. Esperamos, dessa forma, que esses objetivos expressem o valor que a partir deles será gerado. Taxas de produtividade ou quantidades de trabalho produzido não são boas escolhas para objetivos, portanto.

Os objetivos geralmente são definidos pelo Product Owner ou por outras partes interessadas e ajustados com os Desenvolvedores. Também podem ser definidos inteiramente em conjunto com eles. Por serem aqueles que os realizarão, quanto maior for a participação dos Desenvolvedores nesse processo, mais realistas e factíveis serão os objetivos, e mais comprometidos os Desenvolvedores estarão com eles.

Objetivo da Entrega

O que é o Objetivo da Entrega?

O Objetivo da Entrega geralmente existe quando trabalhamos com uma entrega planejada. Podemos considerá-lo como é o compromisso do plano da entrega, que é formado pela data da entrega, por um conjunto de itens selecionados e pelo próprio Objetivo da Entrega. É um objetivo ou necessidade de alto nível, expresso pela perspectiva do negócio ou do usuário, a ser realizado por meio do trabalho dos Desenvolvedores para uma entrega. Caso utilizado, o Objetivo da Entrega existe para cada entrega planejada e é estabelecido antes do início do trabalho correspondente à entrega, possivelmente durante uma reunião de Release Planning (veja o capítulo *Release Planning*).

Embora não seja um artefato oficial do framework, o guia oficial do Scrum já o definiu em edições mais antigas,

afirmando que "...as reuniões de Sprint Review e de Planning são usadas para inspecionar o progresso em direção ao Objetivo da Entrega..." e "o Objetivo do Sprint é um subconjunto do Objetivo da Entrega" (SCHWABER; SUTHERLAND, 2010).

Assim, cada Sprint que termina, previsto para aquela entrega, realiza um Objetivo de Sprint que é um incremento à realização do Objetivo da Entrega. E cada entrega realizada representa um incremento à realização do Objetivo do Produto.

O Objetivo da Entrega também pode representar um marco no *roadmap* do produto, casos em que cada um desses marcos corresponde a uma entrega. Veja no exemplo da figura a seguir um *roadmap* de produto para um sistema de software de vendas pela internet.

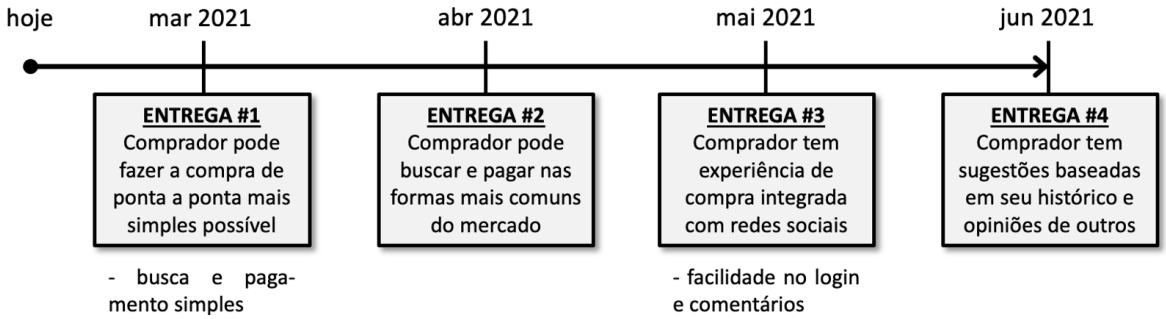


Figura 11.4: Roadmap do produto com Objetivos de Entrega

Assim como o Objetivo do Produto, o Objetivo da Entrega fornece contexto, orientação, motivação e inspiração para todo o trabalho de desenvolvimento do produto até o momento da entrega correspondente.

Como é o Objetivo da Entrega?

Não existem formatos prescritos para os Objetivos de Entrega. Esses objetivos são necessidades a serem atendidas em uma dada janela de tempo, e essas

necessidades podem ser descritas a partir da perspectiva de seus usuários ou de seus clientes.

Um formato que sugiro neste livro para os Objetivos de Entrega, similar ao que sugiro para o Objetivo do Sprint, é:

> *(Por meio da entrega / Até* ***< QUANDO >****),* ***< QUEM >*** *poderá/será capaz de* ***< O QUÊ >***

Onde **< QUANDO >** se refere à data da entrega ou, alternativamente, a um identificador da entrega (número ou nome, por exemplo); **< QUEM >** pode ser uma categoria de usuários, um cliente ou uma parte interessada específica; e **< O QUÊ >** define qual é a necessidade a ser atendida. Podemos ver dois exemplos a seguir para produtos distintos, o primeiro pela perspectiva do usuário e o segundo pela perspectiva do negócio:

> *Por meio da entrega #3, os compradores serão capazes de ter uma experiência de compra integrada com redes sociais*.
>
> *Até o final do primeiro semestre, a loja poderá oferecer seus produtos para os compradores principais*.

Roadmap do produto

O que é o roadmap do produto?

O *roadmap* do produto é um plano em alto nível de como acreditamos que o produto vai evoluir ao longo do tempo, em termos de objetivos a serem realizados, até

um momento futuro determinado, que pode ser o final previsto para o trabalho de desenvolvimento do produto, para o projeto ou algum momento futuro relevante. Assim, ele é útil apenas quando é possível imaginarmos, em alto nível, como o produto evoluirá no futuro.

No trabalho com Scrum, o Product Owner é geralmente responsável por criar, manter e comunicar o *roadmap* do produto. O Product Owner geralmente cria o *roadmap* do produto em uma ou mais sessões de trabalho em que podem estar presentes diferentes partes interessadas e os Desenvolvedores.

O *roadmap* do produto facilita o diálogo entre o Time de Scrum, a organização, os clientes e as demais partes interessadas sobre a evolução do produto, indicando a todos o que pretendemos realizar em alto nível, ao longo do trabalho futuro. Ele é também útil na estratégia da organização, uma vez que ajuda a coordenar o desenvolvimento e entregas de produtos relacionados, as atividades necessárias para essas entregas e outras atividades estratégicas que impactam ou são impactadas por elas.

Como é o roadmap do produto?

O *roadmap* do produto é geralmente representado por uma linha do tempo com marcos, cada um contendo uma data exata ou aproximada no futuro, e um objetivo do produto a ser realizado até a data (veja a figura a seguir).

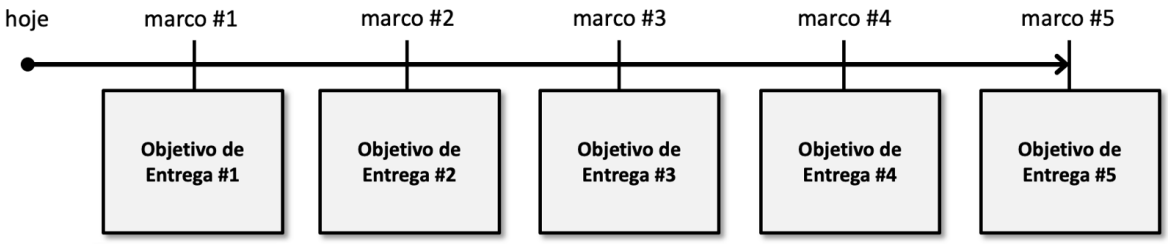


Figura 11.5: Roadmap do produto

Cada um desses objetivos pode ser entendido como um objetivo intermediário entre o Objetivo do Produto e os Objetivos dos Sprints realizados para o seu marco do *roadmap* correspondente. Assim, *roadmap* do produto com Scrum mostra o caminho incremental para a realização do Objetivo do Produto.

Quando os marcos do *roadmap* coincidem com entregas, podemos chamar o objetivo intermediário expresso em cada marco de Objetivo da Entrega. No entanto, quando a frequência de entregas é mais alta - em cada Sprint ou em entregas contínuas, por exemplo - pode não fazer sentido estabelecer Objetivos de Entrega. Nesses casos, o objetivo expresso em cada marco desse *roadmap* ainda servirá de referência como um incremento à realização do Objetivo do Produto.

Cada marco pode também conter objetivos parciais, sejam técnicos, de usuário ou de negócios. É também aceitável haver mais de um objetivo principal definidos por marco, em geral direcionados a diferentes classes de usuários, clientes ou demais partes interessadas.

Capítulo 12

Compromisso: Definição de Preparado (adicional)

Conteúdo

1. O que é a Definição de Preparado?
    - Definição e uso.
    - O trabalho de preparação.
    - Disfunções no uso da Definição de Preparado.
2. Como é a Definição de Preparado?
    - A Definição de Preparado.
    - Definição de Preparado compartilhada.

12.1 O que é a Definição de Preparado?

Definição e uso

> *Os itens do Product Backlog que podem ser Prontos pelo Time de Scrum dentro de um Sprint são considerados preparados para seleção em um evento de Sprint Planning - Guia do Scrum (SCHWABER; SUTHERLAND, 2020).*

A Definição de Preparado é utilizada para garantir que os itens que podem ser selecionados na reunião de Sprint Planning para implementação estejam preparados segundo um critério bem definido. É um entendimento formal compartilhado entre Product Owner e os Desenvolvedores do produto sobre o estado em que um item do Product Backlog deve estar para ser considerado

preparado para entrar no Sprint Backlog e, portanto, para ser implementado.

Embora haja menção a itens "preparados" desde a edição do Guia do Scrum de 2011 (SCHWABER; SUTHERLAND, 2011), a Definição de Preparado não é parte oficial do Scrum. Seu uso é uma prática emergente e, portanto, não é obrigatório.

Entendo a Definição de Preparado como um compromisso do Product Backlog, uma vez que seus itens devem cumpri-la para poderem ser implementados em um Sprint.

Mesmo que o item, no momento da reunião de Sprint Planning, esteja no alto do Product Backlog, os Desenvolvedores e o Product Owner optam por não o adicionar ao Sprint Backlog se não estiver preparado de acordo com a Definição de Preparado até o final da reunião.

O trabalho de preparação

Não é incomum Times de Scrum realizarem o trabalho de preparação dos itens apenas durante a Sprint Planning. Essa prática pode levar a um Sprint mal planejado, já que detalhes demais são deixados para serem discutidos apenas naquela reunião. A reunião pode tornar-se longa, cansativa e ineficiente, originando um Sprint Backlog mal formulado e colocando em risco todo o trabalho do Sprint.

Diversos times optam, no entanto, por antecipar esse trabalho de preparação, realizando-o ao longo do Sprint anterior, em sessões de Refinamento do Product Backlog (veja o capítulo *Refinamento do Product Backlog*). Essas sessões são realizadas pelos Desenvolvedores em

conjunto com o Product Owner e, possivelmente, outras partes interessadas. Seu principal resultado esperado é que itens suficientes estejam preparados para ser implementados, e que sejam considerados na reunião de Sprint Planning seguinte. A Definição de Preparado, portanto, funciona como um critério de entrada para o Sprint, como podemos observar na figura a seguir.

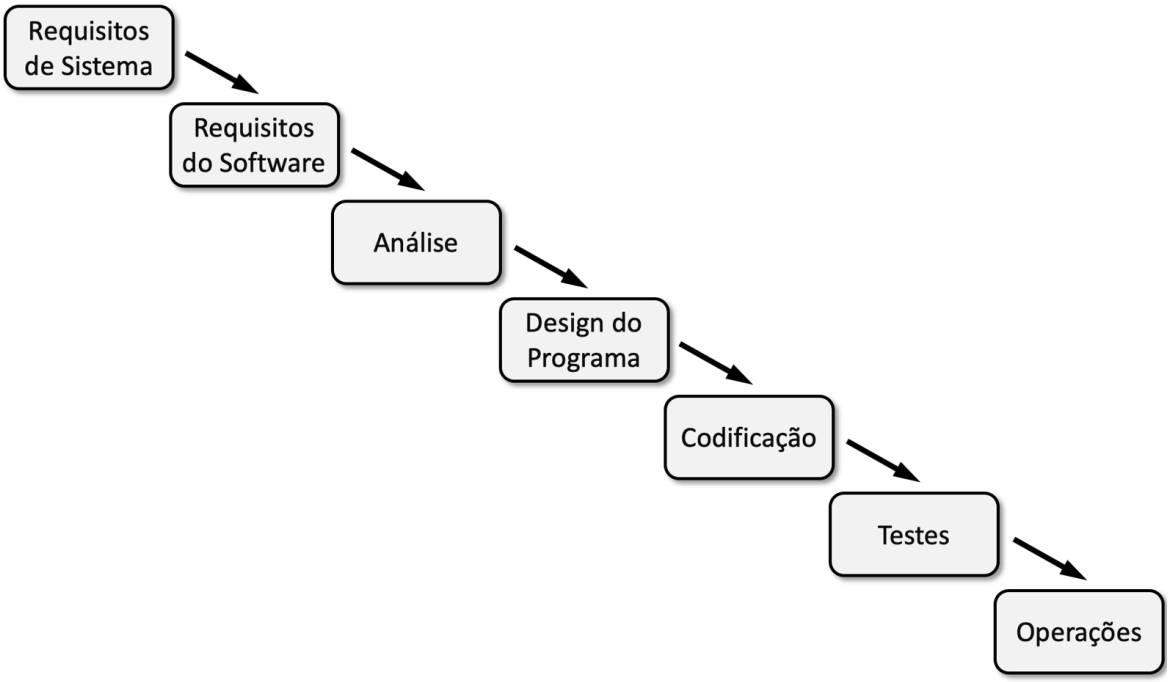


Figura 12.1: Fluxo do item: Definição de Preparado e Definição de Pronto

Podemos observar, nessa mesma figura, o fluxo de um item do Product Backlog através de um Sprint. Caso o item do Product Backlog esteja preparado de acordo com a Definição de Preparado, ele pode ser aceito para discussão na reunião de Sprint Planning e, consequentemente, pode fazer parte do Sprint Backlog. Se o item estiver pronto de acordo com a Definição de Pronto ao final do Sprint, ele sai do Sprint como parte do produto (veja o capítulo *Compromisso: Definição de Pronto*).

Disfunções no uso da Definição de Preparado

> **Preparado?**
>
> Começa a reunião de Sprint Planning. O Product Owner lê a primeira história, do alto do Product Backlog, e alguns Desenvolvedores afirmam enfaticamente:
>
> — *Essa não dá pra fazer não!*
>
> Ao que o Product Owner questiona:
>
> — *Mas por quê?*
>
> — *Não está preparado! Faltam muitos detalhes. Desse jeito que você trouxe, não dá para implementar!* - decreta um dos Desenvolvedores.

É comum a situação disfuncional em que os Desenvolvedores utilizam a Definição de Preparado para se protegerem da responsabilidade de preparar os itens. Em alguns casos, eles deixam essa responsabilidade inteiramente por conta do Product Owner. Em outros, alguma parte essencial do processo de preparar os itens está nas mãos de terceiros, que nem sempre cumprirão a parte que lhes cabe. E ainda, restrições do próprio processo de trabalho podem exigir que alguma atividade, mesmo que realizada pelos próprios Desenvolvedores, deva estar completa para que o item possa estar preparado.

A responsabilidade de existirem itens suficientes preparados no momento de sua escolha para o Sprint Backlog é conjunta. Product Owner e Desenvolvedores estão de olho no alto do Product Backlog e no Sprint seguinte que se aproxima. Eles preocupam em, juntos,

buscarem meios de preparar esses itens. Sessões de Refinamento do Product Backlog durante o Sprint anterior ajudam nisso. Esses itens então estarão preparados apenas com o mínimo suficiente e necessário para que possam ser colocados para implementação.

12.2 Como é a Definição de Preparado?

A Definição de Preparado

A Definição de Preparado tem geralmente o formato de uma lista de critérios, condições ou, ainda, passos de um processo. Veja um exemplo na figura a seguir.

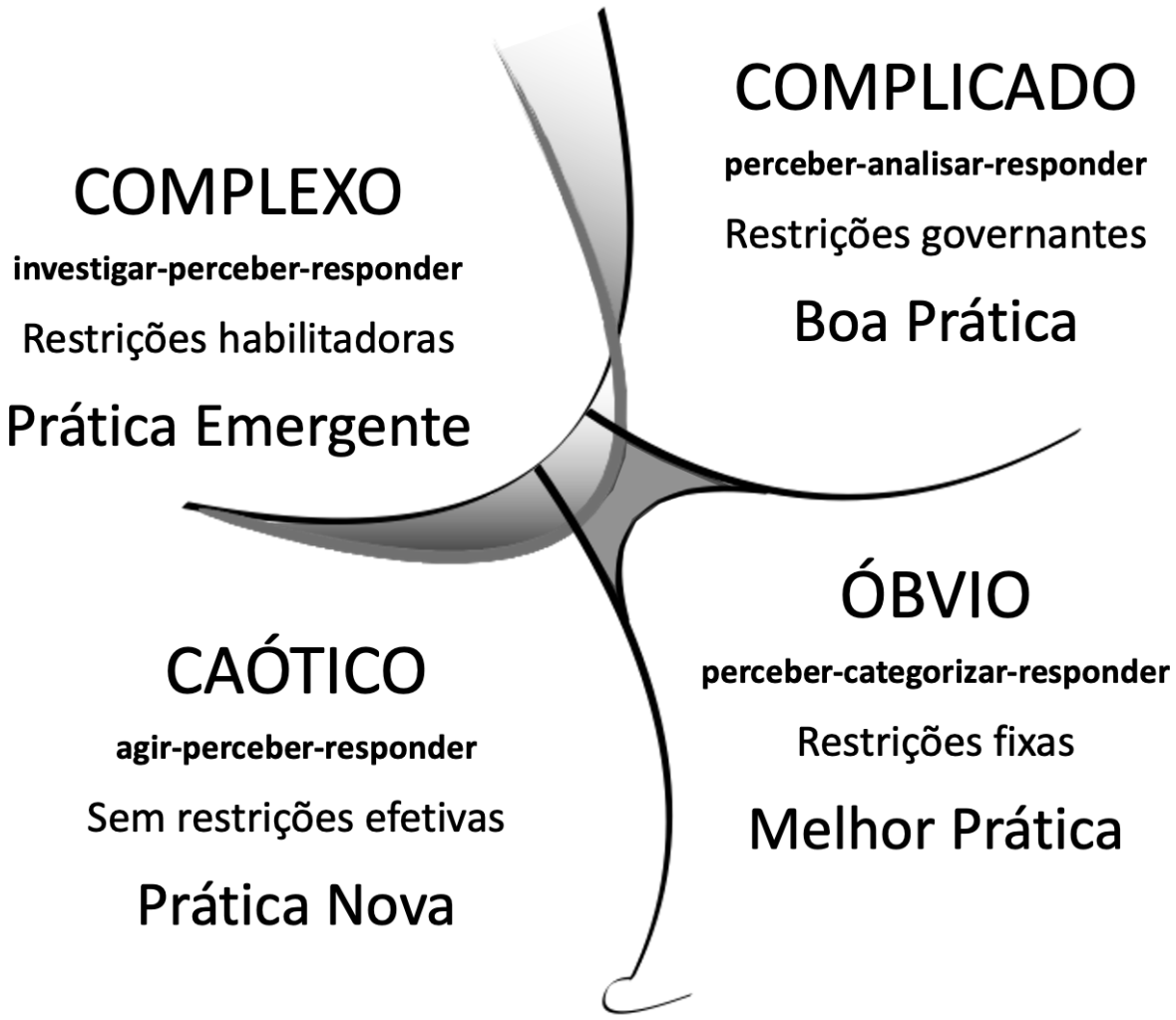


Figura 12.2: Exemplo de Definição de Preparado

É comum a Definição de Preparado conter os seguintes tópicos:

- houve um alinhamento sobre o item e seu consequente detalhamento, de forma a gerar uma compreensão compartilhada sobre o que o item representa. Esse detalhamento pode ser descrito na forma de Testes de Aceitação, discutidos, compreendidos e acordados entre Product Owner e os Desenvolvedores (veja a seção *Confirmação* no capítulo *User Stories: representando o trabalho*);

- o item foi estimado pelos Desenvolvedores (veja o capítulo *Story Points: estimando o trabalho*);
- o item é pequeno o suficiente, de acordo com algum critério estabelecido pelos Desenvolvedores (um valor máximo para sua estimativa, por exemplo).

Uma Definição de Preparado saudável exige apenas o mínimo suficiente e necessário para que os itens possam ser implementados e evita a criação de fases ou etapas no trabalho e seus consequentes gargalos.

A Definição de Preparado é criada pelos Desenvolvedores em conjunto com o Product Owner antes do início do desenvolvimento do produto, geralmente antes mesmo do primeiro Sprint. Entretanto, ela pode ser modificada e evoluir de forma a acomodar novas necessidades identificadas ao longo do trabalho.

A Definição de Preparado é criada, compreendida e compartilhada pelos membros do Time de Scrum. Dessa forma, mantê-la visível para todos eles é essencial.

Definição de Preparado compartilhada

Convenções, padrões, restrições e diretrizes da organização podem definir um mínimo a ser seguido como parte da Definição de Preparado de todos os seus Times de Scrum. Nesses casos, esse mínimo é parte da Definição de Preparado de cada um dos times, e cada um acrescenta suas particularidades a ela.

No caso em que múltiplos Times de Scrum trabalham sobre um mesmo produto, esses times podem criar sua Definição de Preparado em conjunto. Essa definição comum não impede que cada time acrescente mais elementos à sua Definição de Preparado.

Capítulo 13
Sprint Backlog

Conteúdo

13.1 O que é o Sprint Backlog?

Definição e uso

O Sprint Backlog é um artefato do Scrum formado pelo Objetivo do Sprint - **por quê**, por uma lista de itens escolhidos para implementação durante o Sprint, selecionados do alto do Product Backlog na reunião de Sprint Planning - **o quê**, adicionada de um plano de como esse trabalho será executado - **como**.

Na prática, o Sprint Backlog é um plano que contém o trabalho que os Desenvolvedores do produto acreditam que vão realizar durante o Sprint para a implementação de um ou mais Incrementos do produto, e que define o valor a ser realizado.

O Sprint Backlog existe apenas no contexto de seu Sprint correspondente. Dessa forma, ele é criado na reunião de Sprint Planning e deixa de existir após as reuniões de Sprint Review e Sprint Retrospective, ao final de seu Sprint.

O Sprint Backlog pertence ao Time de Scrum e é um artefato essencial utilizado pelos Desenvolvedores para organizarem o seu trabalho durante o Sprint. Por essa razão, o Sprint Backlog deve oferecer alta visibilidade para eles.

Por quê?

O Objetivo do Sprint é o compromisso do Sprint Backlog, e dele faz parte. Ele é estabelecido na reunião de Sprint Planning a partir da colaboração entre os membros do Time de Scrum. Ele é realizado pelos Desenvolvedores durante o Sprint, a partir da implementação dos itens do Sprint Backlog e da consequente geração de um ou mais Incrementos.

O Objetivo do Sprint expressa uma necessidade ou problema, uma proposta de valor ou um propósito para esse trabalho a ser realizado, sempre alinhado ao Objetivo do Produto (veja os capítulos *Compromisso: Objetivo do Sprint* e *Compromisso: Objetivo do Produto*).

O quê?

A lista de itens escolhidos para o Sprint Backlog é o resultado da colaboração entre os Desenvolvedores e Product Owner, realizada na reunião de Sprint Planning. Nessa reunião, esses itens são retirados do alto do Product Backlog e movidos para o Sprint Backlog que está sendo criado, em princípio mantendo a ordenação realizada pelo Product Owner.

Definição: User Stories

Como mencionei no capítulo *Product Backlog*, User Story é uma forma concisa e simples de representarmos cada item do Product Backlog e, portanto, do Sprint Backlog.

Recomendo fortemente o seu uso, para mantermos o foco do Time de Scrum nos usuários finais do produto.

Leia o capítulo *User Stories: representando o trabalho*.

Esses itens representam uma previsão do que os Desenvolvedores acreditam que conseguirão implementar naquele Sprint. É, portanto, um equívoco acreditar que esse plano constitui uma obrigação e que o resultado do Sprint será um fracasso caso ele não seja cumprido.

Para construir essa lista, os Desenvolvedores cooperam com o Product Owner para serem capazes de escolher uma quantidade de itens, desde o alto do Product Backlog, que acreditem que preencha sua capacidade de trabalho em um Sprint. A escolha da quantidade de itens que entrará no Sprint Backlog é uma prerrogativa dos Desenvolvedores. Com frequência, Desenvolvedores utilizam, como parâmetro para essa escolha, sua Velocidade medida, baseada em quanto foram capazes de implementar, em conjunto, em Sprints anteriores (veja *Velocidade*, no capítulo *Story Points: estimando o trabalho*).

Um Product Backlog bem ordenado tem em seu topo itens que são coerentes entre si. Por essa razão, esse conjunto de itens leva o Time de Scrum a realizar necessidades dos usuários, o que por sua vez leva à realização da necessidade de negócios mais importante

do produto naquele momento (veja *Ordenado*, em *Como é o Product Backlog?*, no capítulo *Product Backlog*). Essa necessidade, uma vez ajustada e dimensionada pelos Desenvolvedores, é expressa por meio do Objetivo do Sprint. Dessa forma, os itens para o Sprint Backlog serão implementados, do mais importante ao menos importante, visando realizar o Objetivo do Sprint.

Curiosidade: melhorias de processos no Sprint Backlog

A edição do guia oficial do Scrum de 2017 introduziu a obrigação de que o Sprint Backlog contivesse ao menos um item, de alta ordem, que tratasse de melhoria de processos, levantado na reunião de Sprint Retrospective anterior (SCHWABER; SUTHERLAND, 2017). O objetivo era garantir que a melhoria contínua acontecesse.

Essa obrigação nunca me pareceu correta, já que não a vejo como suficientemente emergente da prática comum. Observo, na realidade, que os times oferecem transparência a melhorias levantadas na Sprint Retrospective de diferentes formas, e essa é apenas uma delas.

Já na edição seguinte do guia, de 2020, essa obrigação foi retirada (SCHWABER; SUTHERLAND, 2020).

Como?

Além do Objetivo do Sprint e do conjunto de itens retirados do Product Backlog, o Sprint Backlog contém um plano de como esses itens serão implementados. A implementação dos primeiros itens do Sprint Backlog é detalhada pelos Desenvolvedores ainda na reunião de Sprint Planning, que seguem fazendo o mesmo para os

itens seguintes ao longo do Sprint. Esse plano é detalhado o suficiente para que eles possam se coordenar na realização desse trabalho.

A prática mais comum é expressarem esse plano na forma de um conjunto de tarefas correspondente a cada item do Sprint Backlog. Essas tarefas indicam o trabalho concreto que será realizado pelos Desenvolvedores, ou seja, elas representam os pequenos passos, normalmente técnicos, a serem executados para que o item correspondente esteja pronto, de acordo com a Definição de Pronto (veja o capítulo *Compromisso: Definição de Pronto*). Portanto, quando todas as tarefas de um item estão terminadas, o item está pronto, de acordo com essa definição.

As tarefas possuem uma granularidade que ajuda os Desenvolvedores a entenderem, se organizarem, distribuírem e acompanharem o que falta do trabalho a ser realizado.

Cada tarefa carrega uma indicação do seu andamento atual. Se a tarefa ainda não foi iniciada, se está em andamento ou se já está terminada. Os Desenvolvedores também podem adicionar estimativas para o tempo de implementação de cada tarefa, para que possam acompanhar seu progresso em direção ao final do Sprint. Esse acompanhamento pode ser realizado utilizando, por exemplo, o Gráfico de Sprint Burndown (veja *Gráfico de Sprint Burndown*, no capítulo *Burndown e Burnup: acompanhando o trabalho*).

O formato de tarefas, embora largamente utilizado, não é prescrito pelo Scrum.

Curiosidade: tarefas não são parte do Scrum

As edições do Guia do Scrum de 2009 e 2010 definiam, ainda que de forma confusa, o Sprint Backlog como uma lista de itens extraídos do Product Backlog durante a reunião de Sprint Planning, cada um desses itens decomposto na forma de tarefas a serem realizadas pelos Desenvolvedores (SCHWABER, 2009; SCHWABER; SUTHERLAND, 2010).

A confusão se dá porque os autores afirmavam que o Sprint Backlog é a lista de tarefas... Mas que contém itens provindos do Product Backlog.

A partir da edição de 2011, o Sprint Backlog foi redefinido como um conjunto de itens provindos do Product Backlog, somado a um plano de como esses itens serão implementados, sem prescrever o formato de tarefas ou qualquer outro formato específico para esse plano (SCHWABER; SUTHERLAND, 2011). A edição de 2020 passou a incluir no Sprint Backlog o Objetivo do Sprint (SCHWABER; SUTHERLAND, 2020).

Há ainda times que optam por sequer adicionar um plano a cada item, ficando o Sprint Backlog apenas com seu conjunto de itens. É uma prática razoavelmente comum em times que trabalham com itens em fatias extremamente finas, ou seja, de curtíssima duração de implementação.

13.2 Como é o Sprint Backlog?

O Sprint Backlog

O Sprint Backlog pode ser representado simplesmente por uma lista ordenada dos itens escolhidos para o

Sprint, cada um adicionando instruções de como o item será implementado, além do Objetivo do Sprint. Essas instruções, como mencionei anteriormente, geralmente são expressas pelos Desenvolvedores na forma de tarefas para cada item. Desse modo, um quadro de tarefas, como o da figura a seguir, é uma representação bastante comum para o Sprint Backlog. Na figura, a nota adesiva na parte inferior direita está representando o Objetivo do Sprint, mas ele pode ser expresso de outra forma.

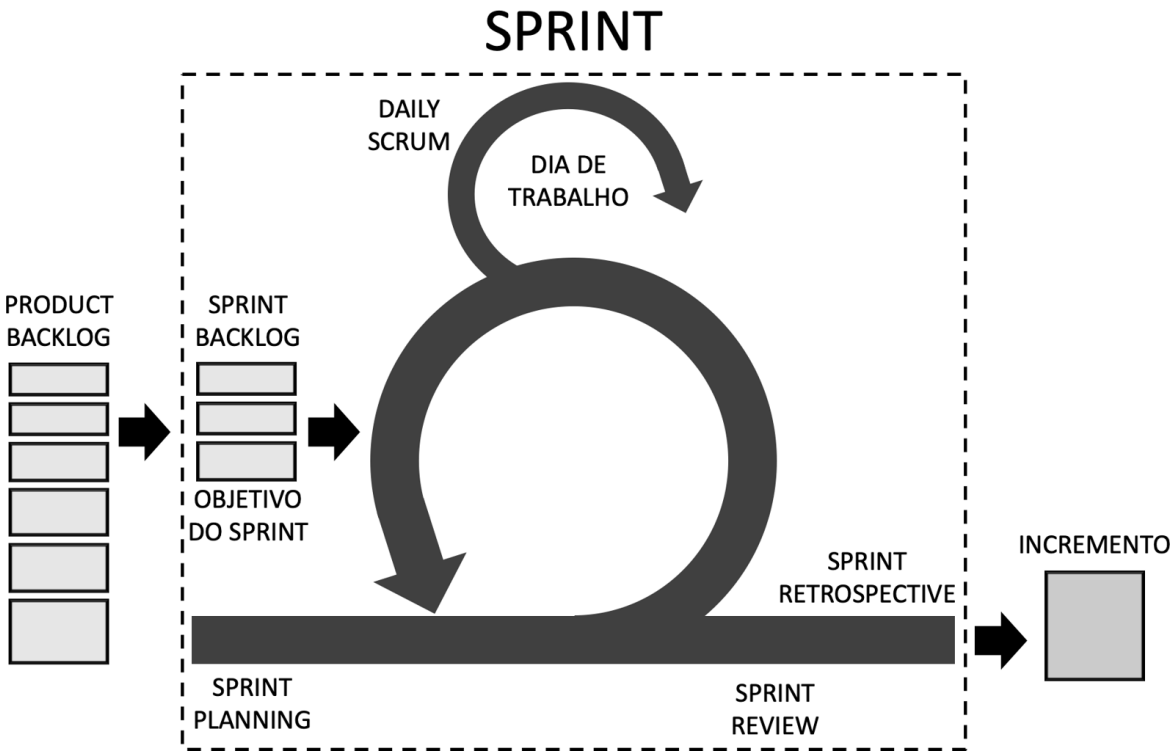


Figura 13.1: Quadro de tarefas representando o Sprint Backlog

No quadro de tarefas, os itens do Sprint Backlog estão dispostos verticalmente de acordo com a sua ordem de implementação, ou seja, serão implementados de cima para baixo. A cada item corresponde uma raia horizontal, demarcada verticalmente apenas pelo item mais abaixo ou pelo fim do quadro.

Repare, no exemplo anterior, que o quadro reflete um Sprint em andamento e que seu último item ainda não foi quebrado em tarefas. Essa quebra será realizada pelos Desenvolvedores em um momento mais próximo da sua implementação, no decorrer do Sprint.

Estão dispostas em cada raia as tarefas correspondentes àquele item, distribuídas em diferentes colunas de acordo com seu status de andamento: "a fazer", "em andamento" e "pronto" (ou quaisquer termos semelhantes). Esse comportamento está ilustrado na figura a seguir.

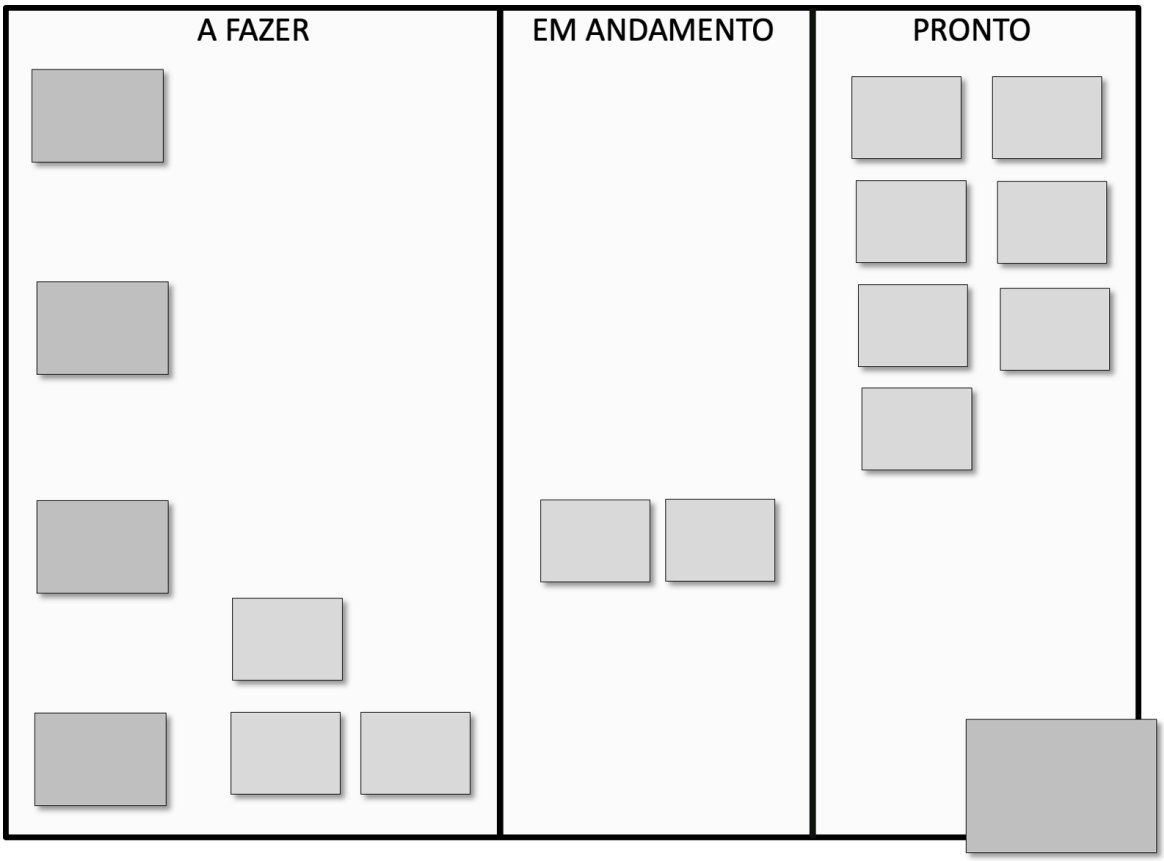


Figura 13.2: Como utilizar o quadro de tarefas

O Sprint Backlog não é estático

O Sprint Backlog pertence ao Time de Scrum, que é responsável por seu uso e manutenção. O Product Owner e os Desenvolvedores usam o melhor de seu conhecimento no momento do planejamento para criá-lo, detalhando trabalho suficiente para os primeiros dias do Sprint. Cumprir esse plano, no entanto, não é o objetivo dos Desenvolvedores. Seu foco se mantém no Objetivo do Sprint, enquanto o resto do Sprint Backlog é apenas o meio para realizá-lo.

É importante, portanto, que o Sprint Backlog reflita sempre o momento atual. Por essa razão, esperamos que ele evolua e seja atualizado no decorrer do Sprint, de forma a retratar o entendimento dos Desenvolvedores, em cada momento, sobre o trabalho a ser implementado a seguir visando realizar o Objetivo do Sprint.

Assim, o Sprint Backlog criado na reunião de Sprint Planning não é estático: ao longo do Sprint, os Desenvolvedores aprendem mais sobre o trabalho a ser realizado e seguem detalhando-o e atualizando-o de acordo. Essas mudanças incluem, por exemplo, a adição de tarefas para itens de menor ordem, a atualização do status de andamento de tarefas, reestimativas e a remoção de tarefas (isso caso a representação do trabalho em tarefas seja utilizada).

Adicionalmente, quando há algum aprendizado relevante sobre o trabalho, os Desenvolvedores, em comum acordo e em colaboração com o Product Owner, podem decidir por realizar mudanças nos próprios itens do Sprint Backlog. Eles podem adicionar novos itens, remover ou modificar os itens já existentes, desde que essas mudanças de forma alguma alterem o Objetivo do Sprint. Ao contrário, a ideia é justamente aproveitar o conhecimento adquirido para melhor realizar esse objetivo.

Instrumento de pressão?

Nem Product Owner, nem Scrum Master podem pressionar os Desenvolvedores a aceitarem, para o Sprint Backlog, mais itens do que acreditam serem capazes de implementar.

O Sprint Backlog também não deve ser utilizado por outras pessoas como instrumento de controle sobre os Desenvolvedores, nem mesmo pelo Product Owner ou pelo Scrum Master. O Sprint Backlog é um artefato para maximizar a transparência sobre o trabalho, de forma que os próprios Desenvolvedores possam acompanhar seu progresso na realização do Objetivo do Sprint, estimulando uma maior autonomia.

Trabalho em equipe e ordenado

Durante o Sprint, os Desenvolvedores trabalham a partir do item de maior ordem e em direção ao de menor ordem, visando realizar o Objetivo do Sprint. Embora seja normal trabalharem em tarefas diferentes, eles colaboram sobre os mesmos itens, buscando tornar prontos — de acordo com a Definição de Pronto — os itens de maior ordem primeiro. Dessa forma, os Desenvolvedores trabalham em equipe.

No entanto, são comuns times em que os diferentes itens ou os diferentes tipos de tarefa possuem "donos". Durante todo o trabalho, cada Desenvolvedor atua em um item diferente, em geral de acordo com suas áreas de conhecimento ou preferências (veja a figura a seguir). Esse comportamento é considerado disfuncional e traz diversos prejuízos.

Além de ir de encontro ao trabalho em equipe e ao compartilhamento de conhecimentos e

responsabilidades, essa abordagem leva à abertura da implementação de vários itens ao mesmo tempo sem, no entanto, se preocupar com seu fechamento. Há, assim, o risco de os Desenvolvedores chegarem ao final do Sprint sem muitos dos itens prontos e, ao não privilegiarem os itens de maior ordem, aumentarem o risco de não realizar o Objetivo do Sprint.

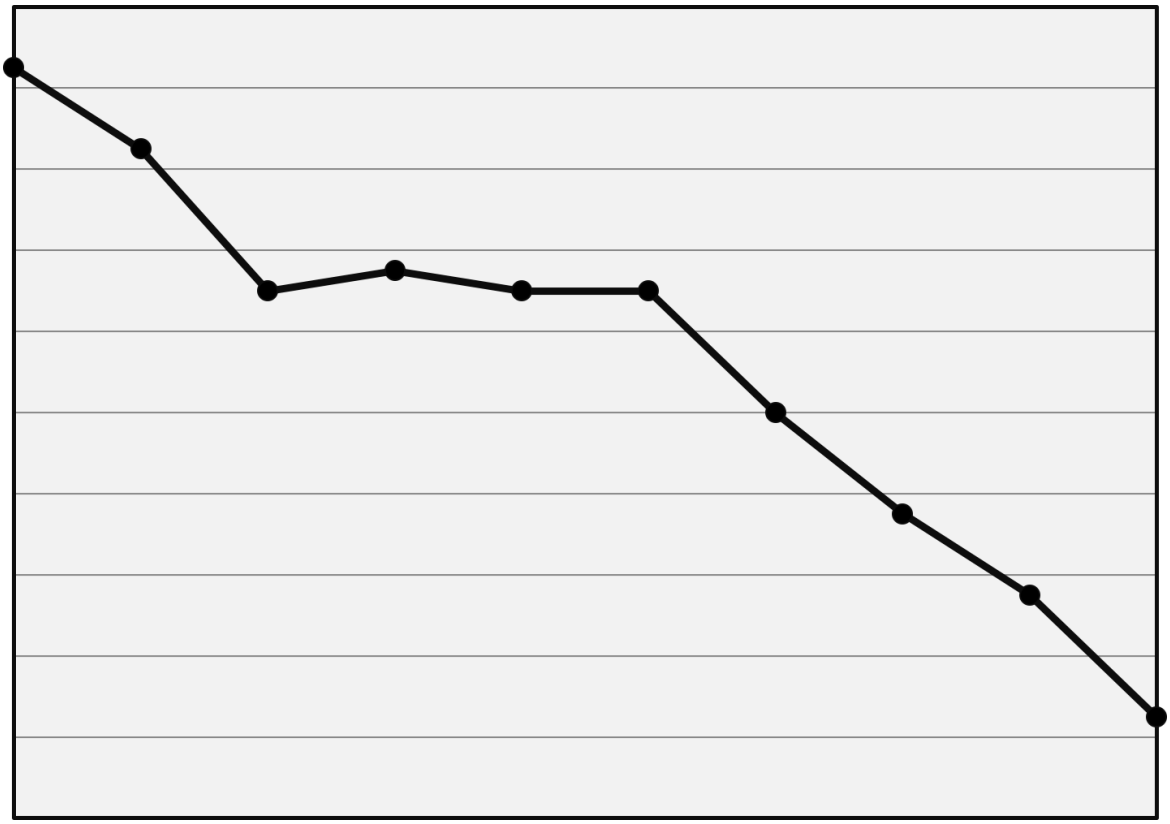


Figura 13.3: Divisão indesejável de trabalho

O método Kanban, criado por David Anderson, traz uma frase que traduz muito bem como recomendamos que esse problema seja resolvido: *pare de começar e comece a terminar!*

Quadro real x quadro virtual

Para Desenvolvedores que trabalham juntos na mesma sala, a representação do Sprint Backlog é muito mais

efetiva em um quadro de tarefas real do que em um quadro virtual de um programa de computador, que necessita de ser acessado para mostrar a informação.

O quadro real é bem visível, localizado na própria sala de trabalho dos Desenvolvedores, e notas adesivas são usadas geralmente para representar os itens e suas tarefas correspondentes. Esse quadro funciona como um "irradiador de informação", ou seja, a informação chega aos olhos de quem é importante chegar, sem haver um esforço significativo para buscá-la.

Ao escrever em uma nota adesiva para colar no quadro real, pense que os outros Desenvolvedores devem entender facilmente o que está escrito, pois seu conteúdo pertence a todos. Escreva com letra grande e legível. Utilize marcadores fortes em vez de canetas comuns, de forma que possam ser lidos bem à distância. Use notas adesivas de boa marca, com uma cola que não sai facilmente, e retire cada folha do bloco com cuidado suficiente para não a entortar, de forma que depois não caia facilmente do quadro.

O uso do quadro virtual tem sentido quando os membros do Time de Scrum trabalham *on-line*, desde locais diferentes. Para esses casos, há diversas ferramentas disponíveis no mercado, desde simples e gratuitas até complicadas e caras. Eu prefiro sempre buscar as mais simples que funcionem bem.

Capítulo 14
Compromisso: Objetivo do Sprint

Conteúdo

14.1 O que é o Objetivo do Sprint?

Definição e Uso

O Objetivo do Sprint expressa um propósito claro definido para o Sprint a ser realizado ou satisfeito a partir da implementação de itens do Sprint Backlog. É uma espinha dorsal, um tema central, um nexo ou uma coerência que conecta esses itens, oferece sentido em eles estarem reunidos e visa levar os Desenvolvedores do produto a trabalharem juntos, e não em diferentes iniciativas. O Objetivo do Sprint responde por que esse trabalho será realizado, ou seja, por que vale a pena executar o Sprint, oferecendo uma direção comum para todos.

O Objetivo do Sprint é um compromisso do Sprint Backlog de um Sprint, e dele faz parte.

Ele representa o valor que o Time de Scrum espera que seja produzido no Sprint ao implementar um ou mais Incrementos, alinhado ao Objetivo do Produto. Ele portanto representa uma necessidade ou problema, ou uma proposta de valor, descrita pela perspectiva de negócios ou do usuário, a ser realizada como resultado do trabalho implementado no Sprint.

É também válido um Objetivo de Sprint realizado a partir da execução de um experimento e da obtenção de seus resultados. Esse experimento pode ser, por exemplo, a validação ou invalidação de uma hipótese. Também considero válido um objetivo que trata apenas da preparação do experimento para que possa ser executado, caso para chegar a resultados seja necessário um tempo maior do que o Sprint.

O guia do Scrum original (SCHWABER, 2009) já alinhava o Objetivo do Sprint a objetivos de valor, afirmando que ele é "um subconjunto do objetivo da entrega" (veja a seção *Objetivo da Entrega* no capítulo *Compromisso: Objetivo do Produto*).

Curiosidade: o Objetivo do Sprint como um compromisso

O Objetivo do Sprint é mencionado no guia oficial do Scrum desde a sua primeira edição (SCHWABER, 2009). Mas foi somente no guia oficial de 2020 que o Objetivo do Sprint foi introduzido como o compromisso de um Sprint Backlog (SCHWABER; SUTHERLAND, 2020).

Se imaginarmos o Sprint como um “miniprojeto”, o Objetivo do Sprint pode ser visto como uma (mini) visão do produto a ser desenvolvido nesse “miniprojeto” (veja

o bloco *Sprint = projeto*, na seção *O que é o Sprint?* do capítulo *Sprint*).

O Objetivo do Sprint guia, dá propósito, motiva e mantém um foco comum para o trabalho dos Desenvolvedores durante o Sprint. Ele estimula e reforça a colaboração e a autonomia, uma vez que os Desenvolvedores definem como vão realizá-lo e se tornam responsáveis por fazê-lo acontecer. Ao mesmo tempo, esse objetivo representa um valor visível, produzido pelos Desenvolvedores no Sprint, sobre o qual será oferecido feedback por clientes e demais partes interessadas na reunião de Sprint Review.

Os Desenvolvedores e o Product Owner trabalham, Sprint após Sprint, satisfazendo incrementalmente o Objetivo do Produto. O Objetivo do Sprint, no entanto, é um alvo mais imediato e mais tangível, que representa um incremento à realização desse Objetivo do Produto, e conecta a ele o valor de cada item do seu Sprint Backlog correspondente. O Objetivo do Sprint pode também representar um incremento à realização de algum objetivo intermediário ao Objetivo do Produto, como um Objetivo de Entrega ou um marco de *roadmap* (veja a seção *Objetivos intermediários*, no capítulo *Compromisso: Objetivo do Produto*).

O Objetivo do Sprint é o compromisso para um Sprint Backlog assim como o Objetivo do Produto é o compromisso para o Product Backlog.

Como é criado o Objetivo do Sprint?

O Objetivo do Sprint é estabelecido e acordado entre Product Owner e Desenvolvedores durante a reunião de Sprint Planning de cada Sprint, na atividade *Por que esse Sprint tem valor?*.

O Product Owner mantém o Product Backlog ordenado. Ele chega à reunião com informações suficientes para decidir qual a próxima necessidade dos clientes ou usuários a atender, que acredita ser a parte mais importante do Objetivo do Produto naquele momento. Ele então colabora com os Desenvolvedores para estabelecer um Objetivo para o Sprint que atenda a essa necessidade, ajustado em congruência com os itens que acreditam que conseguirão realizar durante o Sprint. Ou seja, os itens são selecionados para o Sprint Backlog de acordo com o Objetivo do Sprint, e vice-versa.

Nessa colaboração, o Product Owner sabe que forçar os Desenvolvedores além de sua capacidade de trabalho não trará os resultados que ele almeja, e que Desenvolvedores são mais comprometidos quando têm liberdade de fazer escolhas e, dessa forma, poderão dar o melhor de si para entregar valor para os clientes e usuários do produto.

Uma vez definido na reunião de Sprint Planning, o Objetivo do Sprint não pode mais ser alterado. Para garantir a integridade do Sprint, o Scrum Master tem o papel de assegurar que não seja feita durante o Sprint nenhuma mudança que possa afetar o seu objetivo. No entanto, caso o seu Objetivo do Sprint perca o sentido, o Sprint pode ser cancelado pelo Product Owner (veja no capítulo *Sprint*, a seção *O Sprint pode ser cancelado?*).

Como é realizado o Objetivo do Sprint?

Os Desenvolvedores são responsáveis pela realização do Objetivo do Sprint e possuem todas as habilidades e conhecimentos necessários para isso. Com esse fim, eles implementam um ou mais Incrementos do produto ao longo do Sprint a partir do plano estabelecido no Sprint Backlog. Dessa forma, os itens selecionados para o Sprint

Backlog são implementados desde o que mais contribui para a realização do Objetivo do Sprint em direção ao que menos contribui para tal.

No entanto, o trabalho de desenvolvimento de produto é um processo empírico e, assim, carrega uma incerteza inerente e exige flexibilidade, mesmo no curto prazo (veja a seção *Scrum é embasado no empirismo* no capítulo *O que é Scrum?*). Assim, o Sprint Backlog é o melhor que os Desenvolvedores podem imaginar no momento da sua criação, na reunião de Sprint Planning. Esse entendimento, no entanto, pode mudar uma vez que iniciem o trabalho. O Objetivo do Sprint oferece essa flexibilidade necessária para o trabalho dos Desenvolvedores no Sprint.

Dessa forma, embora os Desenvolvedores façam o seu melhor para concluir todo o trabalho planejado, dentro de limites razoáveis, é natural que muitas vezes não consigam fazê-lo. Ao sobrarem itens não prontos ao término do tempo de trabalho do Sprint, esses serão sempre os de menor ordem, ou seja, os que menos contribuem para a realização do Objetivo do Sprint. Assim, o sucesso do Sprint não reside em completarem tudo o que foi planejado, mas sim em realizarem satisfatoriamente o Objetivo do Sprint a partir do que foram capazes de produzir no Sprint.

Essa flexibilidade também pode permitir que modifiquemos o plano, mas mantendo o Objetivo do Sprint. À medida que realizam seu trabalho, os Desenvolvedores aprendem cada vez mais sobre o que estão fazendo no Sprint. Com isso, pode acontecer de encontrarem uma outra forma, que seja melhor, mais viável ou com o escopo reajustado, de realizarem o Objetivo do Sprint, diferente do que foi inicialmente definido no Sprint Backlog. Quando ocorre, essa

mudança é negociada em colaboração com o Product Owner.

No entanto, nenhuma mudança durante o Sprint jamais pode modificar ou ameaçar o Objetivo do Sprint, nem sacrificar a qualidade do trabalho realizado.

Dificuldades com o Objetivo do Sprint

Em minha experiência, vejo com frequência Times de Scrum que simplesmente não utilizam o Objetivo do Sprint, ou o utilizam de forma incorreta. Muitos sequer entendem do que se trata.

Há diversas razões para isso ocorrer. Product Backlogs orientados a desejos e pedidos, e não a objetivos bem definidos de negócio, por exemplo, dificilmente servem de entrada para Sprints com objetivos claros. O mesmo ocorre com Sprints dedicados a diferentes projetos ou produtos. Um produto cujo desenvolvimento não caminha para realizar incrementalmente um Objetivo do Produto também pode levar ao mesmo problema: itens escolhidos para o Sprint Backlog com pouca relação entre eles, tornando impossível criar nexos na forma de Objetivos de Sprint.

É natural, no entanto, que pequenas modificações, correções pendentes de problemas e adições a realizar surjam e até que se acumulem no Product Backlog, Sprint após Sprint, durante o trabalho de desenvolvimento de um produto. Como consequência, pode ser muito difícil e até mesmo impossível que todos os itens selecionados para o Sprint Backlog compartilhem de um objetivo em comum.

Nesses casos, podemos esperar que, com alguma frequência, nem todos os itens do Sprint Backlog estejam

diretamente conectados à realização do Objetivo do Sprint. Esse fato de forma alguma invalida a existência de um Objetivo do Sprint ou diminui sua importância, que é a de oferecer uma espinha dorsal coerente, que mantenha o foco dos Desenvolvedores e os estimule a trabalharem juntos. Ou seja, criamos um Objetivo para o Sprint mesmo que alguns dos itens selecionados tenham pouca ou nenhuma relação com ele.

14.2 Como é o Objetivo do Sprint?

O Objetivo do Sprint

O Objetivo do Sprint é geralmente representado por uma frase curta e direta. Ele é uma e apenas uma proposta de valor, que visa manter em si o foco de todos os Desenvolvedores durante seu trabalho no Sprint. Por essa razão, não recomendo o uso do conector *e* ao formular a frase. Evitamos, portanto, algo como "ser capaz de *X* **e** de *Y*".

Ainda que seja diretamente relacionado à implementação de itens do Sprint Backlog, o Objetivo do Sprint não é uma quantidade de itens a serem implementados. Ele também não é um item ou um composto de itens selecionados do Product Backlog, nem mesmo os de maior ordem. Tampouco é a uma frase longa contendo todos os itens do Sprint Backlog por extenso, como "implementar X, Y e Z".

Em seguida, podemos ver alguns exemplos de Objetivos de Sprint:

- *o Comprador mais importante poderá comprar um dos livros pré-selecionados*;

- *a Livraria será capaz de realizar vendas básicas dos livros pré-selecionados para o comprador mais importante*;
- *o Administrador poderá fazer um login mais seguro*;
- *seremos capazes de comprovar se os consumidores preferem a nova cor principal do produto ao invés da antiga*;
- *o restaurante poderá entender o baixo número de retornantes a partir do novo mecanismo de feedback*;
- *o segmento de mercado A será capaz de comprar o serviço X*;
- *simplificar o processo de pagamento para reduzir a taxa de abandono em X%*.

O primeiro exemplo mostra a perspectiva de um usuário, o Comprador mais importante, enquanto o segundo mostra a perspectiva do negócio, representado pela Livraria. Ambas são válidas e levarão, provavelmente, ao mesmo resultado. Nos dois casos, imagino que os itens de maior ordem do Sprint Backlog tratarão do que é essencial para uma compra básica, de forma a realizar minimamente esse Objetivo do Sprint. Poderemos ter, por exemplo, a busca pelo título do livro, a inserção de dados básicos de entrega e o pagamento com cartão de crédito. Itens mais abaixo tratarão de questões relacionadas com gradativa menor importância para esse objetivo, como por exemplo, a busca pelo autor e o pagamento com débito em conta, entre outros. Ao serem implementados, estes enriquecerão a qualidade da realização do Objetivo da Sprint. Os outros exemplos ilustram Objetivos do Sprint em diferentes contextos.

Curiosidade: o Objetivo do Sprint como um objetivo de valor

O Objetivo do Sprint foi definido no primeiro livro de Scrum como *um objetivo a ser alcançado através da implementação do Product Backlog* e a principal razão para tê-lo é *dar ao time algum espaço de manobra com relação à funcionalidade* que será produzida (SCHWABER; BEEDLE, 2002).

Em suas duas primeiras edições, o guia oficial do Scrum afirmava adicionalmente que Objetivo do Sprint é *uma declaração que oferece orientação para o time quanto ao porquê de ele estar construindo o incremento* e que *o Objetivo do Sprint é um subconjunto do Objetivo da Entrega* (SCHWABER, 2009, 2010), trazendo a sua definição mais próxima de uma proposta de valor (veja a seção *Objetivo da Entrega*, no capítulo *Compromisso: Objetivo do Produto*). Com a retirada do Objetivo da Entrega do framework, a edição seguinte (SCHWABER; SUTHERLAND, 2011) colocava que *o Objetivo do Sprint pode ser um marco em um propósito maior do roadmap do produto* (veja a seção *Roadmap do produto* , no capítulo *Compromisso: Objetivo do Produto*).

No entanto, parece que os autores escolheram acomodar uma disfunção comum na edição de 2013: alguns times definem como Objetivo do Sprint um objetivo interno. O guia de 2013 abriu espaço para esse uso, afirmando que o Objetivo do Sprint ***pode*** *ser realizado a partir da implementação do Product Backlog*, que *o Product Backlog selecionado entrega uma função coerente, que* ***pode*** *ser o Objetivo do Sprint* e, finalmente, que *o Objetivo do Sprint pode ser qualquer outra coerência que leve o Time de Desenvolvimento a trabalhar em conjunto ao invés de em diferentes iniciativas* (SCHWABER; SUTHERLAND, 2013). Essa definição foi mantida nas duas edições seguintes (SCHWABER; SUTHERLAND, 2016, 2017).

Exemplos desse uso interno disfuncional incluem:

- *automatizar o processo de verificação de qualidade com a nova ferramenta*;
- *trabalhar em par ao menos 50% do tempo*;
- *aprender como realizar Testes de Aceitação automatizados com a ferramenta Cucumber*.

Itens de aprendizado ou de melhoria interna podem sim fazer parte do Sprint, mas não devem ser o objetivo principal. O Objetivo do Sprint representa, na realidade, uma necessidade ou problema a ser resolvido para o negócio ou para os usuários, alinhada ao Objetivo do Produto, a ser realizada incrementalmente a partir da implementação dos itens selecionados para o Sprint Backlog.

A edição de 2020 afirma que o Objetivo do Sprint *comunica por que o Sprint tem valor para as partes interessadas* e que *o Objetivo do Sprint é o único objetivo para o Sprint* (SCHWABER; SUTHERLAND, 2020), trazendo de volta o propósito original do Objetivo do Sprint (e, na minha opinião, o mais correto desde sempre): uma proposta de valor.

Um formato para o Objetivo do Sprint

Um dos formatos que sugiro utilizar para o Objetivo do Sprint é o seguinte:

> *(Por meio deste Sprint,)* ***[QUEM]*** *poderá/será capaz de* ***[O QUÊ]****.*

[QUEM] pode ser uma categoria de usuários, um cliente ou uma parte interessada específica, e *[O QUÊ]* define

qual é o objetivo ou necessidade de negócios ou do usuário a ser atendida.

Scrum, no entanto, não prescreve nenhum formato para o Objetivo do Sprint. Na verdade, sequer sugere. Seja qual for o formato utilizado, o importante é que cumpra com o seu propósito principal: levar os Desenvolvedores a trabalharem juntos, colaborando em seu dia a dia, como um time deve ser, realizando Sprint após Sprint o Objetivo do Produto.

Capítulo 15
Incremento

Conteúdo

1. O que é o Incremento?
    - Definição e uso.
    - O Incremento é entregável.
    - O Incremento e a Sprint Review.
    - O Incremento e a Definição de Pronto.
2. Como é o Incremento?
    - O Incremento.
    - O que não faz parte do Incremento?

15.1 O que é o Incremento?

Definição e uso

Um Incremento é o resultado pronto, de acordo com a Definição de Pronto, da implementação de um ou mais itens do Sprint Backlog, que é utilizável por clientes e usuários do produto e potencialmente representa valor para eles (veja o capítulo *Compromisso: Definição de Pronto*). Cada Incremento gerado deixa o Time de Scrum mais próximo de realizar o Objetivo do Produto.

Em cada Sprint, os Desenvolvedores implementam os itens do Sprint Backlog, daquele de maior ordem ao de menor ordem. Ao realizar esse trabalho, eles geram um ou mais Incrementos. Ao menos um Incremento deve ser gerado como resultado de cada Sprint, obrigatoriamente. O Incremento é um artefato do Scrum.

Curiosidade: mais de um Incremento por Sprint

Até a edição do guia oficial do Scrum de 2017, o Incremento era considerado o resultado da implementação de todos os itens que chegavam prontos, de acordo com a Definição de Pronto, ao final do Sprint (SCHWABER; SUTHERLAND, 2017). Ou seja, era gerado apenas um Incremento por Sprint.

A edição de 2020 nos permite considerar como um Incremento o resultado da implementação de qualquer conjunto de itens prontos, de acordo com a Definição de Pronto, durante o Sprint, que seja utilizável e represente valor (SCHWABER; SUTHERLAND, 2020) potencialmente.

O principal propósito dessa alteração foi abrir a possibilidade de realizarmos entregas para usuários e clientes no decorrer do Sprint, permitindo-nos rapidamente obter feedback sobre o Incremento entregue para reagirmos de acordo. Esse tempo curtíssimo de inspeção e adaptação é particularmente útil para quando executamos experimentos durante o Sprint e coletamos métricas de uso.

Scrum, portanto, passou a aceitar múltiplos Incrementos por Sprint, embora ainda seja permitido implementar apenas um.

O Incremento é entregável

Enquanto pronto, o Incremento é entregável, o que significa que o Product Owner pode decidir por entregá-lo para os clientes e usuários do produto quando achar adequado, inclusive durante o Sprint em que foi produzido.

A definição de se um determinado conjunto de itens realmente representa um Incremento passa obrigatoriamente pelo julgamento de que ele, ao somar-se ao que já foi entregue até então, tem o potencial de gerar valor para usuários, clientes e outras partes interessadas. Para que seja um Incremento, o Time de Scrum também deve acreditar que ele o leva mais próximo de realizar o Objetivo do Produto.

Apesar de cada Incremento ser entregável, o Product Owner pode optar por acumular um número de Incrementos para só então fazer a sua entrega. Ou seja, "entregável" é diferente de "entregue".

Embora o ideal seja colocar as fatias do produto prontas nas mãos dos usuários com a mais alta frequência possível, há diferentes razões que podem levar o Product Owner a decidir por acumular Incrementos antes de realizar uma entrega.

Dependendo do tipo do negócio, clientes ou usuários podem não conseguir absorver as mudanças tão rapidamente ou pode haver questões políticas, burocráticas, estratégicas ou técnicas que impeçam o Product Owner de decidir naquele momento por entregar o Incremento.

Curiosidade: Incremento potencialmente entregável

Os criadores do Scrum antigamente se referiam ao Incremento como "potencialmente entregável" (SCHWABER; SUTHERLAND, 2009, 2010, 2011, 2013, 2016, 2017). Essa linguagem, que na minha opinião é um pleonasmo, mudou a partir da edição de 2020

(SCHWABER; SUTHERLAND, 2020). Eles passaram a se referir ao Incremento como útil ou utilizável e de valor. Por razões didáticas, eu opto aqui neste livro por manter o uso do termo "entregável".

O Incremento e a Sprint Review

Os Desenvolvedores e o Product Owner demonstram a soma dos Incrementos do produto prontos para os clientes e demais pessoas relevantes ao final do próprio Sprint em que foram produzidos, na reunião de Sprint Review. O principal objetivo dessa reunião é obter feedback dessas pessoas sobre o trabalho realizado, e assim ajudar a definir o que serão os próximos Incrementos, alimentando o Product Backlog.

Cada Incremento do produto representa valor visível e utilizável para os presentes na reunião. Ou seja, cada Incremento adiciona ou modifica características e comportamentos no produto que podem ser experimentados e utilizados ali mesmo, e é isso que torna esse feedback possível e valioso durante a reunião.

A Sprint Review, portanto, não é o momento para realizarmos a entrega de um ou mais Incrementos, mas sim um importante momento para recebermos feedback sobre eles.

O Incremento e a Definição de Pronto

A Definição de Pronto é um compromisso de todo e cada Incremento do produto, ou seja, os Desenvolvedores devem segui-la para gerar cada Incremento dentro de seu Sprint. Por essa razão, ela é determinante para as habilidades e conhecimentos que os Desenvolvedores,

em conjunto, devem possuir para implementar todo e cada Incremento daquele produto.

Ao seguir uma Definição de Pronto ideal, os Desenvolvedores implementarão um Incremento entregável, ou seja, que pode ser entregue a seus clientes e usuários assim que pronto. Dessa forma, idealmente nenhum trabalho adicional, seja de implementação, verificação, documentação ou quaisquer tarefas devem ser necessárias para que um Incremento do produto produzido no Sprint possa ser entregue. Quando entregar o Incremento, no entanto, é uma decisão que cabe ao Product Owner. Mesmo que já seja possível, no entanto, o Product Owner pode decidir que mudanças são necessárias ou que vai acumular alguns Incrementos do produto para só então fazer uma entrega.

Caso os Desenvolvedores de um Time de Scrum não sejam capazes de produzir Incrementos que sejam entregáveis, eles deixarão, intencionalmente, um certo trabalho por fazer após cada Sprint. Esse trabalho por fazer não faz parte da sua Definição de Pronto e, consequentemente, será sistematicamente realizado por outras pessoas ou equipes para que só então o Incremento possa ser entregue.

Esse trabalho que, de forma planejada, resta por fazer após cada Sprint reduz a capacidade de entrega, aumenta o tempo do ciclo de feedback e, por consequência, reduz a adaptabilidade e a agilidade do Time de Scrum e de sua organização como um todo. Esses problemas se acentuam ainda mais quando há refluxo, ou seja, quando há trabalho por fazer retornando para o time. Um exemplo comum ocorre quando o trabalho de verificação de erros está fora da Definição de

Pronto do time e, assim, é realizado por pessoas externas.

A relação entre a Definição de Pronto não ideal e o trabalho que resta por fazer sobre os Incrementos prontos é um forte ponto de atenção para o Scrum Master, que é um agente de mudanças em sua organização. Pela natureza de seu trabalho, os Scrum Masters buscam meios de promover, mesmo que aos poucos, que o conjunto de Desenvolvedores de cada time adquira os conhecimentos e habilidades necessários para transformar o pronto em, de fato, entregável.

15.2 Como é o Incremento?

O Incremento

Cada Incremento é composto por novas características e comportamentos para o produto e por mudanças ou melhorias no que foi produzido anteriormente. Assim, um Incremento pronto se soma e modifica todos os Incrementos já implementados anteriormente, deixando o Time de Scrum mais próximo de realizar o Objetivo do Sprint corrente e, por consequência, o Objetivo do Produto.

A natureza incremental do trabalho de desenvolvimento de um produto faz com que um item já considerado pronto jamais volte ao Product Backlog, ao Sprint Backlog, ou seja implementado novamente. Sempre que há a necessidade de mudança em alguma característica ou comportamento que já foi criada anteriormente no produto, um novo item relativo a esse trabalho é criado no Product Backlog, e esse item potencialmente fará parte de um Incremento futuro.

Itens não prontos ao final do Sprint, de acordo com a Definição de Pronto, não fazem parte de nenhum Incremento. Eles podem voltar ao Product Backlog, modificados ou não, ou até serem eliminados, sempre a critério e escolha do Product Owner.

O que não faz parte do Incremento?

Um Incremento sempre adiciona ou modifica um comportamento ou característica do produto. Por essa razão, nem todo trabalho realizado pelos Desenvolvedores durante o Sprint faz parte de algum Incremento.

Pesquisas realizadas com clientes, a criação de maquetes, protótipos, desenhos e outros tipos de artefatos intermediários, por exemplo, geram resultados que não se refletem diretamente em mudanças no produto em direção ao Objetivo do Produto. Embora possam ser passos importantes para a geração de valor, não se tornam parte do produto a partir de algum Incremento.

Um cuidado importante que devemos ter é que todo e cada Sprint obrigatoriamente tenha como saída ao menos um Incremento. Outros subprodutos do Sprint que, diferentemente dos Incrementos implementados, não significam um resultado concreto, não devem ser entendidos como uma medida de progresso real no trabalho.

Capítulo 16
Compromisso: Definição de Pronto

Conteúdo

16.1 O que é a Definição de Pronto?

Definição e uso

A Definição de Pronto é um entendimento formal compartilhado entre os membros do Time de Scrum sobre o que é necessário para que considerem que um trabalho realizado pelos Desenvolvedores está pronto. Ela é um compromisso de todo e cada Incremento do produto produzido pelos Desenvolvedores. A Definição de Pronto pode ser descrita como um conjunto de critérios a serem cumpridos, comuns para o trabalho a ser realizado em cada item do Product Backlog a ser implementado, para que possa formar parte de um Incremento. Dessa forma, quando os Desenvolvedores afirmam que um item ou um Incremento do produto está pronto, todos do Time de Scrum compreendem o que pronto significa, criando transparência.

Embora possa evoluir e se tornar mais estrita com o tempo, a mesma Definição de Pronto se aplica a todos os itens do Product Backlog e a cada Incremento do produto gerado, formado por um conjunto de itens implementados no Sprint. Ela ajuda os Desenvolvedores no momento de planejar o trabalho a ser realizado no Sprint e os guia durante a realização desse trabalho.

Pronto?

Aflita com a entrega atrasada, a gerente de um projeto de desenvolvimento de *software* pergunta ao desenvolvedor:

— *E aí, tudo pronto?*

Ao que o desenvolvedor responde:

— *Claro, está pronto!*

— *Excelente! Vamos colocar em produção agora!* — se anima a gerente.

— *Não, não! Não dá não!* — alerta o desenvolvedor.

— *Mas você não disse que está pronto?*

— *Sim, está pronto. Só... falta testar.*

— *Se falta testar, não está pronto!* — responde a gerente, indignada.

— *Hum... está quase pronto então!* — é a resposta final do desenvolvedor.

Situações em que não há uma compreensão conjunta do que "estar pronto" significa são comuns no trabalho. Com Scrum, pronto é definido e acordado entre todos, e não há o "quase pronto". Ou está pronto e faz parte de um Incremento do produto, ou não está.

Discussão: Definição de Pronto e Testes ou Critérios de Aceitação não são a mesma coisa?

Um engano comum é confundir a Definição de Pronto com os Testes de Aceitação (veja *Confirmação*, em *O que é a User Story?*, no capítulo *User Stories: representando o trabalho*) ou, como alguns autores chamam, Critérios de Aceitação. São coisas distintas.

Diferentemente dos Testes de Aceitação, a Definição de Pronto não é específica para um item ou para um grupo de itens e nada tem a ver com os seus detalhes. Não existe, portanto, a Definição de Pronto de um item ou de um Sprint. Embora possa evoluir com o tempo, ela é a mesma para todos os itens implementados do produto.

Mas há uma relação direta entre Definição de Pronto e os Testes de Aceitação. Uma boa Definição de Pronto exige, preferencialmente de forma explícita, que cada item passe em seus próprios Testes de Aceitação, acordados especificamente para o item.

A Definição de Pronto e o Incremento

A Definição de Pronto é o compromisso de todo e qualquer Incremento gerado pelos Desenvolvedores.

Uma Definição de Pronto ideal estabelece que o Incremento, resultado do trabalho dos Desenvolvedores em um Sprint, seja entregável, ou seja, que nenhum trabalho adicional é necessário para que ele possa ser entregue para os clientes e usuários do produto, se assim decidir o Product Owner (veja a seção *O Incremento e a Definição de Pronto* no capítulo *Incremento*).

Uma consequência direta é que uma boa Definição de Pronto garante que qualquer Incremento do produto tenha o nível de qualidade exigido para que sua entrega possa ser realizada.

A Definição de Pronto e o trabalho dos Desenvolvedores do produto

Os Desenvolvedores possuem, em conjunto, todas as habilidades e conhecimentos necessários para, em cada Sprint, implementar Incrementos do produto prontos, de acordo com a Definição de Pronto. Com isso, é fácil entender como a Definição de Pronto e a composição dos Desenvolvedores de um Time de Scrum são intrinsecamente relacionadas.

As atividades da Definição de Pronto são, portanto, utilizadas como referência quando os Desenvolvedores selecionam quanto trabalho preveem que conseguirão produzir no Sprint. Servem da mesma forma para quando os Desenvolvedores detalham o plano para o Sprint, quebrando os itens do Sprint Backlog em tarefas, por exemplo.

16.2 Como é a Definição de Pronto?

A Definição de Pronto

A Definição de Pronto varia de time para time, de produto para produto. Ela geralmente se parece com uma lista de critérios ou de atividades a serem cumpridas.

Na figura a seguir, vemos um exemplo de Definição de Pronto para o trabalho de desenvolvimento de um sistema de software.

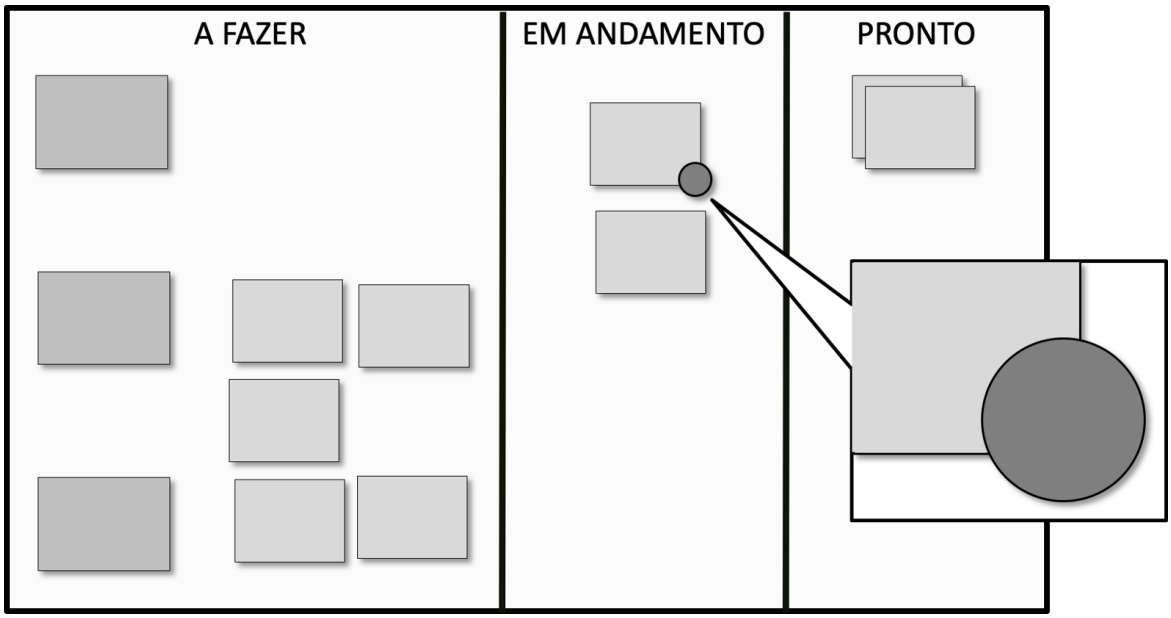


Figura 16.1: Um exemplo de Definição de Pronto

Essa lista é definida de acordo com as necessidades identificadas para o desenvolvimento do produto — das quais garantir a qualidade sempre faz parte — e em conformidade com as convenções, padrões, diretrizes e restrições organizacionais. Essas atividades ou critérios devem portanto ser cumpridos para construir o produto e incluem, por exemplo, diferentes tipos de testes que sejam considerados necessários, documentação que faça parte do produto, quaisquer integrações que devam ser realizadas etc.

A Definição de Pronto é criada antes do início do desenvolvimento do produto. Em geral, antes mesmo do primeiro Sprint. Entretanto, ela pode ser modificada e evoluir ao longo do desenvolvimento do produto, de forma a acomodar mudanças e melhorias no processo de trabalho ou novas necessidades identificadas, como a adoção de uma nova modalidade de verificação de qualidade ou de algum novo artefato que seja estabelecido como necessário.

A Definição de Pronto é criada, compreendida e compartilhada pelos membros do Time de Scrum. Dessa forma, mantê-la visível para todos eles é essencial.

Restrições à Definição de Pronto

Para a construção da Definição de Pronto é essencial que consideremos as restrições organizacionais que afetam o trabalho do Time de Scrum, sejam de negócio, de processo, tecnológicas ou culturais. Devido a essas restrições, muitos Times de Scrum trabalham com uma Definição de Pronto distante da ideal: restam atividades por fazer, que muitas vezes serão realizadas por outras pessoas ou grupos, para tornar os Incrementos de fato entregáveis.

Essa necessidade de realizarmos um trabalho além do pronto é uma disfunção e traz riscos consideráveis, já que limita a habilidade do time de entregar valor. A transferência de responsabilidade (ou *passagem de bastão*) e o refluxo geram filas e consequentes esperas, aumentando o tempo necessário para os Desenvolvedores implementarem algo que possa ser entregue aos clientes e usuários (veja a seção *Entregas Frequentes*, no capítulo *Por que Scrum?*). Exemplos dessas atividades adicionais incluem: diferentes tipos de testes ou verificações realizados por uma equipe externa, outras validações (como auditorias internas ou externas), integrações com outros produtos em desenvolvimento, homologações por parte do cliente e alguns tipos de documentação.

Na medida do possível, o Time de Scrum trabalha ao longo do tempo para reduzir a disfunção, ganhando as habilidades, conhecimentos e autoridade para transferir progressivamente esse trabalho por fazer para dentro dos Sprints. Enquanto agente de mudança, o Scrum

Master trabalha à frente dessa transformação. Assim, a Definição de Pronto vai se tornando cada vez mais estrita, à medida que o Time de Scrum consegue aproximar, de fato, o pronto do "entregável".

Uma abordagem comum, embora disfuncional, é a utilização de um "Sprint de estabilização" como forma de convergir o trabalho realizado em cada entrega. Ou seja, adicionamos um Sprint imediatamente antes da data acordada para a entrega e nele realizamos todo o trabalho, em geral de verificações e correções, que ficou por fazer nos Sprints anteriores. Como consequência, diversos problemas serão descobertos tardiamente e o produto poderá acumular problemas sérios que podem, inclusive, impedir que a entrega seja realizada.

Em situações muito particulares, esse trabalho adicional antes da entrega será sempre necessário e existirá durante todo o trabalho. Pode ser o caso, por exemplo, de produtos de alto risco em que são necessárias verificações de segurança adicionais ou auditorias externas.

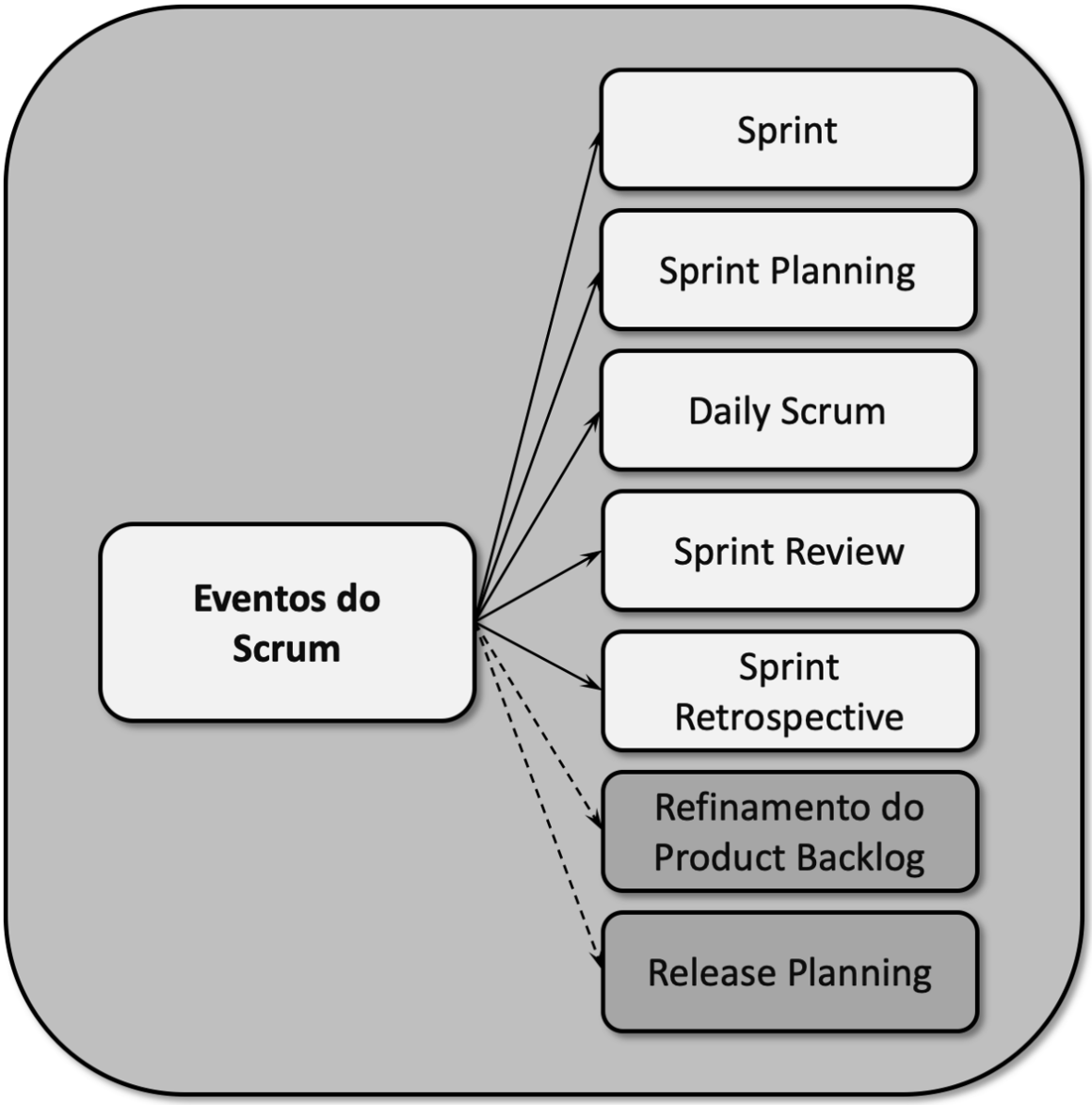
Definição de Pronto compartilhada

Convenções, padrões, restrições e diretrizes da organização podem definir um mínimo a ser seguido como parte da Definição de Pronto de todos os seus Times de Scrum. Nesses casos, portanto, esse mínimo é parte da Definição de Pronto de todos os times, e cada um acrescenta suas particularidades a sua própria Definição de Pronto.

No caso em que múltiplos Times de Scrum trabalham sobre um mesmo produto, esses times criam em conjunto sua Definição de Pronto. Essa definição conjunta

não impede que cada time possua a sua própria, acrescentando mais elementos a ela.

Eventos do Scrum



Capítulo 17
Sobre os eventos do Scrum

Conteúdo

1. O que são os eventos do Scrum?
    - Definição.

2. Quais são os eventos do Scrum?
    - Eventos do Scrum.

17.1 O que são os eventos do Scrum?

Definição

Os eventos do Scrum são as reuniões realizadas por membros do Time de Scrum, além do próprio ciclo de desenvolvimento, o Sprint, que contém todos os outros eventos.

Os eventos do Scrum criam uma rotina para os membros do Time de Scrum. A partir da regularidade imposta pelo framework para a sua realização, os eventos geram uma cadência com a qual os membros do Time de Scrum se habituam. Para reforçar essa consistência e reduzir a complexidade, o Guia do Scrum recomenda que os eventos sejam realizados sempre no mesmo horário e no mesmo local (SCHWABER; SUTHERLAND, 2020).

Ao serem obrigatórios e por cobrirem questões cruciais relativas ao planejamento, à execução e à verificação do trabalho de desenvolvimento do produto, os eventos do Scrum também visam a minimizar a necessidade de outras reuniões não previstas pelo framework.

Ao trazer objetivos claros e a transparência necessária, cada evento oferece uma oportunidade formal para o Time de Scrum inspecionar e adaptar diferentes artefatos do Scrum, promovendo, dessa forma, o empirismo. Lembre-se que os três pilares do Scrum são a transparência, inspeção e adaptação (veja mais sobre o empirismo e os três pilares na seção *Scrum é embasado no empirismo*, no capítulo *O que é Scrum?*).

As reuniões, também chamadas por muitos de cerimônias, são a Sprint Planning, a Daily Scrum, a Sprint Review e a Sprint Retrospective.

Adiciono aos eventos do Scrum uma fase curta de inicialização do trabalho, as sessões de Refinamento do Product Backlog e o planejamento de entregas (ou Release Planning), todos adicionais ao que prescreve o framework.

Definição: os timeboxes do Scrum

Com exceção dos Sprints, com duração fixa, cada evento do Scrum possui uma duração máxima definida, chamada de timebox.

O objetivo principal dos timeboxes é limitar o tempo em que um objetivo deve ser alcançado, de forma a não gerarmos desperdícios com trabalhos intermináveis. Os timeboxes ajudam a criar um ritmo ou uma regularidade no trabalho do Time de Scrum.

Utilizando como referência a duração fixa do Sprint de até um mês, o guia oficial do Scrum define a reunião de Sprint Planning como um timebox de até oito horas, a Daily Scrum de até 15 minutos, a Sprint Review de até quatro horas e a Sprint Retrospective como um timebox de até três horas (SCHWABER; SUTHERLAND, 2020). O guia sugere que, se os Sprints são mais curtos, as reuniões são geralmente mais curtas.

Até o guia de Scrum de 2011, a recomendação oficial era de que a duração máxima das reuniões fosse proporcional à duração dos Sprints, exceto pela reunião de Daily Scrum. Assim, para Sprints de duas semanas, a Sprint Planning duraria no máximo quatro horas, a Sprint Review, duas horas, a Sprint Retrospective, uma hora e

meia, e a Daily Scrum se manteria com o máximo de 15 minutos (SCHWABER; SUTHERLAND, 2011). Considero, até hoje, uma boa recomendação.

17.2 Quais são os eventos do Scrum?

Eventos do Scrum

Os eventos do Scrum são:

- o Sprint, que é o ciclo de desenvolvimento, a iteração, que se repete ao longo de todo o trabalho de desenvolvimento do produto, e que contém todos os outros eventos do Scrum, além do próprio trabalho de desenvolvimento do produto;
- a Sprint Planning, que é a reunião realizada no primeiro momento do Sprint em que o Time de Scrum planeja o trabalho a ser implementado naquele Sprint e cria o Sprint Backlog;
- a Daily Scrum, uma reunião diária, curta e objetiva, em que os Desenvolvedores coordenam suas tarefas no Sprint Backlog para até a Daily Scrum seguinte, em busca do Objetivo do Sprint;
- a Sprint Review, uma reunião realizada no último dia de cada Sprint em que o Time de Scrum obtém feedback de clientes e demais partes interessadas sobre o Incremento ou Incrementos prontos produzidos durante o Sprint, e assim pode atualizar o Product Backlog;
- a Sprint Retrospective, uma reunião de melhoria contínua que fecha o Sprint, na qual o Time de Scrum

inspeciona o Sprint que está se encerrando quanto a seus processos de trabalho, dinâmicas, pessoas, relacionamentos, comportamentos, práticas, ferramentas utilizadas e ambiente, e planeja as melhorias necessárias para o Sprint seguinte.

Como opções a se adicionar ao Scrum, sugiro:

- a Inicialização, uma fase curta inicial, prévia ao início do trabalho de desenvolvimento do produto, formada por um conjunto de atividades de preparação específicas a cada contexto e sempre dentro do mínimo necessário e suficiente;
- a Release Planning, que é a atividade formal ou informal de planejar como, quando e para quem a próxima entrega de uma fatia utilizável do produto será realizada;
- o Refinamento do Product Backlog, que é a preparação de um número de itens do alto do Product Backlog suficiente para formar o próximo Sprint Backlog, realizada a partir de atividades como o detalhamento de itens, seu reordenamento, o desmembramento de itens maiores em itens menores, a adição e a remoção de itens no Product Backlog e a adição de estimativas.

CAPÍTULO 18

Inicialização (adicional)

Conteúdo

1. O que é a Inicialização?
2. Como é a Inicialização?

18.1 O que é a Inicialização?

Ken Schwaber (2004), que com Jeff Sutherland desenvolveu e formalizou o framework Scrum, declarou em um de seus livros: *o plano mínimo necessário para se poder iniciar [o primeiro Sprint] consiste de uma visão e um Product Backlog. A visão descreve por que [o trabalho] está sendo realizado e qual é o estado final desejado*, enquanto que o Product Backlog inicial fornece itens de trabalho para esse primeiro Sprint.

O que chamo de Inicialização neste livro é o trabalho inicial de preparação estritamente necessário para o desenvolvimento do produto. No entanto, não existe no Scrum uma fase de inicialização. Esperamos, na realidade, que a maior parte desse trabalho de base seja realizada ao longo dos primeiros Sprints sem, no entanto, ocupá-los por completo. Como regra do Scrum, os Desenvolvedores devem gerar um valor utilizável em todo e qualquer Sprint, e os iniciais não são exceção. Por essa razão, mesmo quando há um volume considerável de atividades preparatórias, os Desenvolvedores implementam funcionalidades entregáveis desde o primeiro Sprint, mesmo que poucas. Gradativamente,

Sprint após Sprint, eles realizam cada vez menos dessas atividades e geram cada vez mais valor.

Não é incomum, no entanto, serem necessárias algumas atividades de preparação prévias ao primeiro Sprint. Nesses contextos, essas atividades são geralmente tratadas como uma fase de inicialização, que recebe extraoficialmente os mais variados nomes, como Pré-jogo, Sprint Zero, *Discovery*, *Inception*, entre outros.

Acho importante destacar que Sprint Zero não é um termo apropriado (embora seja o mais comum), já que os Sprints devem possuir duração fixa e cada Sprint deve gerar valor visível para os usuários, o que não é o caso.

Como curiosidade, o nome Pré-Jogo aparece no artigo seminal do *framework* Scrum, apresentado por Ken Schwaber em um congresso em 1995 (SCHWABER, 1997).

18.2 Como é a Inicialização?

O trabalho de inicialização é realizado sempre na base do mínimo estritamente necessário e suficiente, buscando não gerar desperdícios e não limitar mais do que o necessário a geração de valor nesse início.

Essas atividades de preparação são extremamente contextuais. Entre as que mais frequentemente ocorrem antes do primeiro Sprint, posso citar:

- a criação ou alinhamento a um Objetivo do Produto;
- a criação de um Product Backlog inicial, em geral apenas suficiente para o primeiro ou para os primeiros Sprints. Esse Product Backlog pode, no entanto, ser estendido para um pouco mais adiante,

desde que seus itens possuam granularidade adequada;
- a formação ou escolha do Time de Scrum, ou seja, a definição de quem é o Scrum Master, o Product Owner e os Desenvolvedores do produto;
- a definição da duração dos Sprints que, em princípio, se manterá sempre a mesma.

Lembro que, caso esse trabalho prévio exista, ele será curto, durando apenas desde alguns dias a, no máximo, algumas poucas semanas.

Dependendo de cada contexto, outras atividades de inicialização podem se somar a essas. As que listo a seguir, apenas como exemplo, são preferivelmente realizadas progressivamente ao longo dos primeiros Sprints, de forma a não postergar o início da geração de valor:

- a criação ou adaptação de uma infraestrutura básica que servirá de suporte para o desenvolvimento do produto, sempre na base do mínimo suficiente e necessário, e que evoluirá ao longo do trabalho;
- a definição de uma arquitetura inicial mínima, e que evoluirá ao longo do trabalho. Essa arquitetura inicial visa a reduzir os riscos de decisões tardias que invalidariam o que já foi produzido, mas sem correr o risco de engessar o trabalho;
- investigações de que tecnologias serão úteis ou necessárias para o desenvolvimento do produto;
- a aquisição de tecnologias e equipamentos necessários para o trabalho;
- a criação da Definição de Pronto - veja o capítulo *Compromisso: Definição de Pronto*;
- a criação da Definição de Preparado - veja o capítulo *Compromisso: Definição de Preparado (adicional)*;

- a identificação das diferentes partes interessadas que deverão ser levadas em conta no desenvolvimento do produto;
- a realização de treinamentos necessários. Para times iniciando com Scrum, esses treinamentos podem ser essenciais para seu sucesso;
- a realização de atividades de formação de time (*teambuilding*).

Capítulo 19
Sprint

Conteúdo

Resumo

- **Objetivo:** produzir valor entregável que realize o Objetivo do Sprint.
- **Quando:** durante todo o desenvolvimento do produto, um seguido do outro.
- **Duração:** fixa, de até um mês.
- **Participantes obrigatórios:** Desenvolvedores, Product Owner e Scrum Master.
- **Saídas esperadas:** um ou mais Incrementos do produto prontos, de acordo com a Definição de Pronto, que realizem o Objetivo do Sprint.

19.1 O que é o Sprint?

Definição

O Sprint é o ciclo de desenvolvimento, a iteração, que se repete ao longo de todo o trabalho de desenvolvimento do produto, um seguido do outro. Em cada Sprint, o Time de Scrum planeja o trabalho que pretende realizar naquele ciclo, implementa os itens selecionados do alto do Product Backlog, que significam valor visível e utilizável para seus usuários, coleta feedback sobre ele e busca melhorar sua forma de trabalhar.

O Sprint contém todos os outros eventos do Scrum, além do próprio trabalho de desenvolvimento do produto. Esse trabalho, enquanto existir, funciona inteiro dentro de Sprints, que acontecem um após o outro, sem paradas ou intervalos (veja a figura a seguir).

Como afirma o guia oficial do Scrum, "um novo Sprint começa imediatamente após a conclusão do Sprint anterior" (SCHWABER; SUTHERLAND, 2020). Desse modo, não há intervalos entre Sprints. Também não interrompemos um Sprint, mesmo que por alguns poucos dias, para tratar de qualquer assunto, seja relativo ao produto ou não. Em princípio, todas as atividades de trabalho que envolvem uso de tempo dos Desenvolvedores do produto são trazidas para dentro dos Sprints.

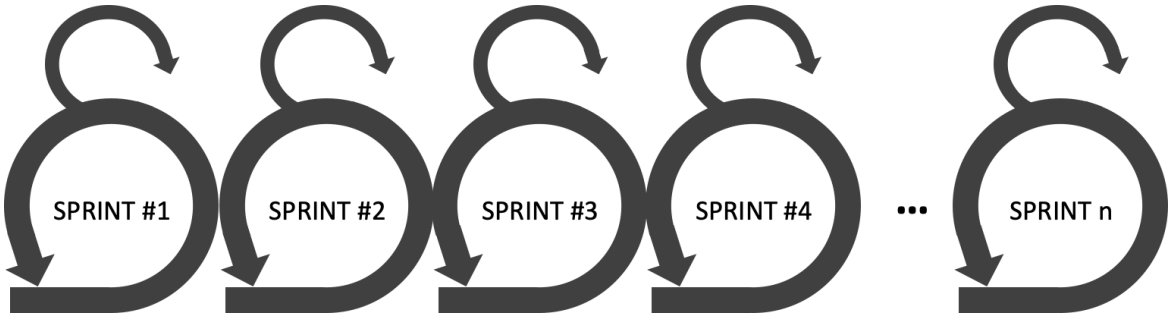


Figura 19.1: Scrum: um Sprint após o outro

Sprint = projeto

Podemos ver o Sprint como um projeto curto, um "miniprojeto", com início e fim. Nele, planejamos, implementamos e garantimos a qualidade de determinados itens visando a realização de um objetivo bem definido — o Objetivo do Sprint. Podemos dizer que esse objetivo corresponde a uma (mini)visão do produto a ser realizada nesse "miniprojeto". O "miniproduto" desse "miniprojeto" é o conjunto de Incrementos do produto implementados no Sprint.

Ao final de cada "miniprojeto", o trabalho realizado é revisado, o time colhe feedback para os próximos "miniprojetos" e levanta melhorias a realizar. Podemos afirmar que o desenvolvimento do produto com Scrum acontece, portanto, dentro de "miniprojetos", um após o outro.

Objetivo do Sprint

O Sprint possui como finalidade realizar um objetivo bem definido, negociado e acordado entre os Desenvolvedores e o Product Owner durante a reunião de Sprint Planning. É o chamado Objetivo do Sprint, que guia o trabalho dos Desenvolvedores, levando-os a trabalharem juntos, e não em diferentes iniciativas (veja o capítulo *Compromisso: Objetivo do Sprint*).

Para realizar o Objetivo do Sprint, os Desenvolvedores implementam, ao longo do Sprint, um ou mais Incrementos do produto prontos, de acordo com a Definição de Pronto, a partir dos itens do Sprint Backlog.

Durante o Sprint, o Objetivo do Sprint não pode ser modificado e os Desenvolvedores não aceitam nenhuma mudança que ameace esse objetivo. Em casos de exceção, quando o Objetivo do Sprint perde seu sentido no decorrer do ciclo, o Sprint pode ser cancelado (veja *O Sprint pode ser cancelado?*, neste capítulo).

O Objetivo do Sprint é o compromisso do Sprint Backlog.

Sprint Backlog

O Sprint Backlog é um plano constituído pelo Objetivo do Sprint (**por quê**), por uma lista de itens retirados do alto do Product Backlog para implementação durante o Sprint (**o quê**) e por um plano de como esse trabalho será executado (**como**). A partir da implementação desses itens, é gerado um ou mais Incrementos, visando à realização do Objetivo do Sprint.

Durante o Sprint, esse plano pode ser esclarecido e renegociado pelos Desenvolvedores juntamente com o Product Owner, à medida que aprendem mais sobre o seu trabalho. No entanto, o nível de qualidade não pode decrescer como resultado dessa negociação.

Incrementos

Em cada Sprint, os Desenvolvedores geram ao menos um Incremento do produto. Cada Incremento representa valor visível e utilizável e é o resultado da implementação de um ou mais itens do Sprint Backlog, que devem estar prontos, de acordo com a Definição de Pronto. O Incremento, portanto, idealmente está sempre apto a ser entregue para os seus usuários, ou seja, ele é entregável (veja o capítulo *Incremento*).

No entanto, a entrega em si é feita apenas quando o Product Owner julgar adequado. O Product Owner pode optar por entregar os Incrementos para seus usuários durante o próprio Sprint ou por acumular Incrementos até realizar a entrega, inclusive a partir dos resultados de mais de um Sprint.

Em muitos cenários, no entanto, os Desenvolvedores não são capazes de produzir Incrementos de fato entregáveis. Para tornar possível a entrega, restam atividades por fazer após o Sprint, o que reduz a capacidade de entrega do time e posterga a detecção de problemas (veja a seção *O Incremento e a Definição de Pronto* no capítulo *Incremento*).

19.2 Como é o Sprint?

O Sprint

Dentro de cada Sprint ocorrem a reunião de Sprint Planning, em seu primeiro dia, as reuniões de Daily Scrum, em cada dia do Sprint, e as reuniões de Sprint Review e de Sprint Retrospective, em seu último dia. Ocorre também, claro, o trabalho de implementação propriamente dito, além de quaisquer outras atividades ou reuniões realizadas com a participação dos Desenvolvedores. Veja como é um Sprint na figura a seguir.

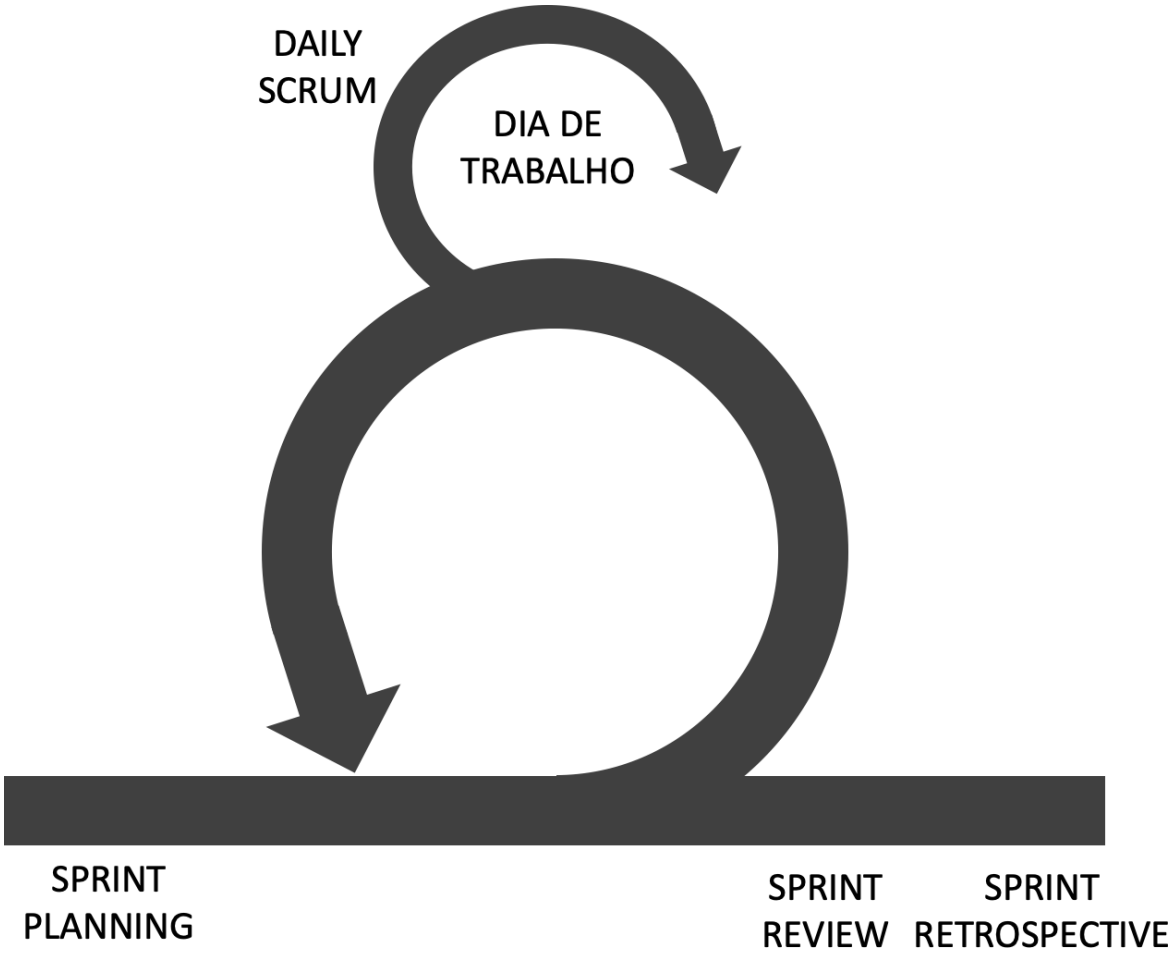


Figura 19.2: O Sprint

Duração

Os Sprints têm duração fixa e consistente, o que significa que eles não devem durar nem mais, nem menos que o acordado e que essa duração não varia de Sprint para Sprint. A ideia por trás desse conceito é a de reduzir a complexidade, criando um ritmo de trabalho para os Desenvolvedores. Essa cadência criada a partir dos eventos do Scrum e do próprio trabalho de desenvolvimento do produto, Sprint após Sprint, disciplinam tanto o próprio time quando outras partes interessadas, facilitando a realização desses eventos, a participação e oferta regulares de feedback de pessoas

externas na reunião de Sprint Review e o planejamento das entregas.

A duração consistente também traz uma maior previsibilidade ao trabalho dos Desenvolvedores e facilita o planejamento de cada Sprint, já que o time melhor entende e rastreia sua capacidade de trabalho e se torna capaz de calcular a sua Velocidade (veja a seção *Velocidade* no capítulo *Story Points: estimando o trabalho*), caso utilize essa prática.

Os Sprints têm a duração máxima de um mês.

Curiosidade: mudanças na duração dos Sprints

O primeiro artigo sobre Scrum, apresentado em um congresso em 1995, definia a duração dos Sprints em tipicamente de uma a quatro semanas, sem fixar a mesma duração entre os diferentes Sprints (SCHWABER, 1997).

Com o amadurecimento do framework, a literatura imediatamente posterior passou a determinar que os Sprints tivessem cerca de um mês (BEEDLE et al., 1999) ou obrigatoriamente trinta dias corridos de duração (SCHWABER; BEEDLE, 2002), dependendo da fonte. A duração dos Sprints foi então estabelecida como obrigatoriamente consistente entre os diferentes Sprints.

A prática do Scrum no desenvolvimento de produtos com diferentes características fez com que aceitássemos Sprints de três, de duas e, tempos mais tarde, de uma semana.

Hoje, as regras do Scrum apenas limitam a duração máxima de cada Sprint em um mês, além de manter a

obrigatoriedade na consistência entre os diferentes Sprints (SCHWABER; SUTHERLAND, 2020).

Para estabelecermos a duração dos Sprints, é mais comum o uso de semanas (ou dias corridos) do que dias úteis. A medida em semanas cria uma maior regularidade, já que o Sprint começará e terminará sempre nos mesmos dias da semana, e isso pode ajudar a dar ritmo no trabalho e facilitar a agenda dos envolvidos. Adicionalmente, sugiro que pode ser psicologicamente interessante iniciarmos o Sprint em uma segunda-feira e terminá-lo em uma sexta-feira, trazendo mais facilmente aos envolvidos um sentimento de fechamento. Já a medida da duração do Sprint em dias úteis facilita a medição da Velocidade e o seu uso para o planejamento, já que o número de dias de trabalho do Sprint não varia com o advento de feriados, mas o ganho pode ser pequeno diante do sacrifício da regularidade.

Pode eventualmente ser necessário modificar a duração do Sprint conforme as condições do trabalho e do ambiente mudam. Product Owner e Desenvolvedores devem estar de acordo com relação a essa nova duração dos Sprints, que também será fixa e constante.

Sprints mais curtos são quase sempre preferíveis. Os ciclos mais curtos de feedback potencializam a melhoria contínua, aumentam a capacidade de entrega, reforçam o foco do time e diminuem as chances de mudanças afetarem o Objetivo do Sprint. Um benefício adicional que podem trazer é que clientes e demais partes interessadas presentes nas reuniões de Sprint Review verão progressos reais no trabalho mais frequentemente. Já Sprints muito longos diminuem o foco do time, podem prejudicar o seu ritmo de trabalho, retardam o

aprendizado provindo da obtenção de feedback e podem gerar uma maior ansiedade em seus clientes.

Definição: EVDnC, Sprints de um dia

EVDnC (leia *evidence*), ou *Extreme Value-Driven Coaching*, é uma abordagem que criamos na K21 em 2013 em que os times, trabalhando bem próximos ao Product Owner ou a pessoas de negócio, executam Sprints de um dia de duração. Dessa forma, eles realizam diariamente as reuniões de abertura e fechamento do Sprint, embora em versões mais curtas. Mas o mais importante é que eles se concentram em gerar um valor visível e entregável para o usuário em cada dia de trabalho. Para que essa convergência diária em valor seja possível, o trabalho dos times deve se focar estritamente na resolução de problemas, não em funcionalidades predefinidas, e deve ser fatiado bem fino.

Ciclos de tão curta duração provocam a emergência dos problemas, tanto locais quanto sistêmicos, que impactam negativamente a capacidade de implementação e de entrega dos times, mas que geralmente ficam escondidos no dia a dia de trabalho. Esses problemas são tratados por agentes de mudança durante o processo ou, quando isso não é possível imediatamente, entram em uma lista a ser tratada em um momento posterior próximo.

Já utilizamos o EVDnC com centenas de times em nosso trabalho de Transformação Ágil. EVDnC pode ser executado por um tempo limitado (em geral, uma semana) ou continuamente.

Saiba mais em http://evdnc.org (em inglês).

Diversos fatores podem influenciar a escolha da duração dos Sprints. Apresento, em seguida, alguns deles.

Nível de maturidade

De forma geral, o uso de Sprints mais curtos demonstra uma maior maturidade dos times e do seu entorno. No entanto, costumo recomendar a times que estão em seus primeiros passos na utilização de Scrum que comecem com Sprints de duas semanas de duração. Assim, se acostumam com a cadência do trabalho e com as práticas mais essenciais do Scrum. Sprints de uma semana poderiam trazer uma sucessão de falhas maior do que o time é capaz de suportar.

No entanto, para times iniciantes que acompanhamos de perto em nosso trabalho de Transformação Ágil, não é incomum já iniciarmos o trabalho com Sprints de uma semana. Assim, aumentamos sua frequência de obtenção de feedback, principalmente a partir das reuniões de Sprint Review e de Sprint Retrospective. Esse feedback frequente permite ao time evoluir mais rapidamente ao identificar em que e como melhorar.

Embora permitidas pelo framework, recomendo fortemente que evitemos o uso de Sprints de um mês, mesmo que para times muito imaturos. O Sprint muito longo tende a esconder os problemas ao invés de estimular a melhoria contínua.

Frequência das mudanças

A frequência com que mudam as necessidades de negócios, dos usuários e suas prioridades é um fator importante na escolha da duração do Sprint. Sprints mais curtos aumentam a capacidade do Time de Scrum de responder a essas mudanças, e podem ser utilizados quando a natureza do produto ou de seus clientes e

usuários tornam as necessidades a serem atendidas mais voláteis. Sprints mais longos podem ser usados no desenvolvimento de produtos com necessidades de negócios mais estáveis, em que o Product Owner encontrará menos problemas em planejar para um tempo um pouco mais longo.

Em qualquer um dos casos, o planejamento é realizado para um período curto o suficiente para que sejam mínimas as chances do Objetivo do Sprint perder o sentido no decorrer da sua realização.

Nível de incerteza

Quanto mais curtos forem os Sprints, mais frequentemente virá o feedback dos clientes e demais partes interessadas sobre o que foi produzido. Ao mesmo tempo, o Time de Scrum será capaz de entregar esses Incrementos mais frequentemente para os usuários e obter seu feedback, em geral a partir de métricas de uso.

Quanto maior for a incerteza sobre o negócio, sobre os usuários, sobre de que problemas o produto deve tratar ou sobre como vai resolvê-los, maior será essa necessidade de feedbacks frequentes, que trazem oportunidades para correções de curso no trabalho realizado. Essa adaptação frequente diminui os riscos de desperdício do trabalho do time em um ambiente de incertezas.

Repare que esse ponto está relacionado com o anterior, mas não trata necessariamente da mesma questão.

Disponibilidade de clientes e demais partes interessadas

Diferentes pessoas, entre clientes e demais partes interessadas, podem ser convidadas para a reunião de

Sprint Review. É necessário, no entanto, que estejam presentes na reunião aqueles que têm a capacidade de trazer feedbacks valiosos para a continuação do trabalho. Uma menor disponibilidade geral dessas pessoas pode tornar mais difícil para o Time de Scrum trabalhar com Sprints muito curtos.

Capacidade de geração de valor de negócio

A duração do Sprint ideal é tal que os Desenvolvedores possam produzir, em conjunto, ao menos um Incremento do produto que signifique valor visível e utilizável para seus usuários, e que receberá feedback de clientes e demais partes interessadas durante a reunião de Sprint Review. Dependendo da natureza do produto, um Sprint muito curto pode não ser suficiente para que seja possível produzir valor visível suficiente.

Criação de ritmo de trabalho

Sprints mais curtos podem ajudar a criar um ritmo para o trabalho dos Desenvolvedores, uma cadência com a qual o próprio time e demais partes interessadas se acostumam. Esse ritmo ajuda a manterem o foco do trabalho, reforçando o seu compromisso em perseguir o Objetivo de cada Sprint e aumentando a previsibilidade das entregas.

Além disso, a alta frequência de reuniões de Sprint Retrospective possibilita ao Time de Scrum experimentar e aprender mais rapidamente com soluções que possam trazer melhorias a seus processos. Ao mesmo tempo, as sensações de sucesso e de trabalho cumprido que vêm com o desfecho frequente de Sprints é um forte motivador para o trabalho dos Desenvolvedores, reforçando ainda mais seu compromisso.

No entanto, Sprints de uma semana podem acabar ditando um ritmo forte e estressante demais para Times de Scrum menos maduros, que podem assim preferir Sprints com duração maior.

Manutenção do foco

Em Sprints muito longos, como os de um mês, o foco dos Desenvolvedores no Objetivo do Sprint tende, com o passar do tempo, a se enfraquecer. Sprints mais curtos favorecem a manutenção do foco no Objetivo do Sprint e no trabalho a ser realizado.

Custo de preparação

A prática mostra que, mesmo para Sprints curtos como os de uma semana, não é incomum a disfunção de times realizarem a reunião de Sprint Planning ocupando boa parte de um dia de trabalho. O mesmo pode ocorrer com as reuniões de Sprint Review e de Sprint Retrospective juntas. Nesses casos, o custo de iniciarmos e de encerrarmos o Sprint pode ser alto demais para que valha a pena realizarmos um Sprint tão curto. É recomendável, assim, tratarmos a disfunção antes de utilizarmos um Sprint tão curto.

Tamanho do projeto ou do produto

Se o trabalho é de curta duração, pode ser mais apropriado estabelecermos Sprints curtos, pois dessa forma o número de oportunidades de feedback é maior, o que pode reduzir os riscos inerentes à incerteza. Para um trabalho com um ou dois meses de duração, por exemplo, Sprints de uma semana podem ser mais adequados.

O Sprint pode ser cancelado?

Caso julgue que o Objetivo do Sprint perdeu o seu sentido, o Product Owner (e apenas ele) pode decidir pelo cancelamento do Sprint. Nesses casos, não faz sentido prosseguir com o Sprint até o seu final. Exemplos em que isso pode acontecer incluem mudanças na legislação ou regulamentação, ações relevantes da concorrência, crises no mercado ou quaisquer mudanças drásticas que tornem o Objetivo do Sprint obsoleto.

Escolhemos, no entanto, uma duração para os Sprints curta o suficiente para que sejam mínimas as chances de o Objetivo do Sprint perder o sentido nesse período, de forma que cancelamentos de Sprint sejam raros. Cancelamentos de Sprint são traumáticos para todo o Time de Scrum e devem ser evitados, na medida do possível. Cancelamentos frequentes indicam um comportamento disfuncional.

Assim, caso seja inevitável, o Product Owner informará aos Desenvolvedores sobre o cancelamento do Sprint. Os itens já prontos (de acordo com a Definição de Pronto) serão revistos, em uma reunião de Sprint Review antecipada. Uma reunião de Sprint Retrospective também será realizada em seguida.

Seguindo as regras do Scrum, um novo Sprint será iniciado imediatamente em seguida. No entanto, como é uma situação de exceção, observo na prática que os Desenvolvedores e Product Owner geralmente decidem como proceder de acordo com o caso. Ocorrendo o cancelamento, é comum preferirem não modificar o calendário já previsto. Desse modo, em vez de iniciar um novo Sprint com o tamanho regular, podem temporariamente violar as regras do Scrum e optar por iniciar um Sprint mais longo que contenha os dias que sobraram, ou realizar um Sprint curto com esses dias restantes ou simplesmente utilizá-los para tratar de

algum assunto relevante, de forma a manter as datas de início e término dos Sprints seguintes.

Discussão: falhar ou não falhar no Sprint?

> *O Time se comprometeu com o Objetivo do Sprint e com esses itens do Product Backlog.(...) Pronto define o que Time quer dizer quando se compromete com "aprontar" um item do Product Backlog em um Sprint* - edição original (e obsoleta) do Guia do Scrum (SCHWABER, 2009).

Os Desenvolvedores devem se comprometer com completar todos os itens escolhidos para o trabalho no Sprint? Se não completaram todos esses itens, o time falhou? Edições antigas do Guia do Scrum, como a que acabo de citar, utilizavam o verbo "comprometer" e levaram a interpretações incorretas sobre qual o objetivo dos Desenvolvedores em um Sprint.

Primeiramente, o propósito dos Desenvolvedores em um Sprint não é completar o plano, ou seja, o trabalho definido no Sprint Backlog daquele Sprint. Seu propósito é realizar o Objetivo do Sprint, o que deve ser possível sem completar tudo o que foi planejado no Sprint Backlog. Segundo, o trabalho de criação de um produto, para o qual Scrum foi concebido, é um trabalho não linear. Esse trabalho depende de inúmeros fatores, muitos deles basicamente humanos. Assim, é de se esperar que esse tipo de trabalho carregue uma grande incerteza. Não cumprir o plano exato, portanto, deveria ser a regra, e não a exceção. Dessa forma, não cumprir o plano não deve trazer o significado de falha.

Em edições seguintes do Guia do Scrum, os autores substituíram "compromisso" por "previsão", afirmando que o Sprint Backlog é uma previsão para as funcionalidades que estarão no(s) próximo(s) Incremento(s) e o trabalho necessário para implementá-lo(s) (SCHWABER; SUTHERLAND, 2017). Na edição de 2020, tanto o termo "compromisso" quanto "previsão" já não mais constam.

O Objetivo do Sprint, por outro lado, traz flexibilidade suficiente visando a que seja uma exceção os Desenvolvedores não conseguirem realizá-la, e não a regra.

Alguns autores e praticantes de Scrum, no entanto, defendem que, no momento em que os Desenvolvedores detectarem que não conseguirão realizar o Objetivo do Sprint, o Product Owner deve ser chamado para negociarem uma redução desse objetivo, ou mesmo o cancelamento do Sprint. Em minha opinião, admitir essas possibilidades é um comportamento que dá margem a transformarmos um resultado indesejado — não realizarmos o Objetivo do Sprint — em uma parte normal do trabalho. Ou seja, em vez de falhar e aprender com a falha e, assim, tornar-se capaz de definir objetivos mais adequados, o time pode, como consequência, sentir-se suficientemente confortável com o cancelamento de Sprints ou reduções do Objetivo do Sprint.

Caso seja necessário, no entanto, é um comportamento saudável os Desenvolvedores chamarem o Product Owner para esclarecerem e renegociarem o Sprint Backlog já definido de forma a melhor realizarem o Objetivo do Sprint, de acordo com o tempo restante no Sprint.

Capítulo 20
Sprint Planning

Conteúdo

1. O que é a Sprint Planning?
    - Definição.
    - Sprint Backlog.
    - Duração.
    - Preparação.
2. Como é a Sprint Planning?
    - A Sprint Planning.
        - Por que esse Sprint tem valor?
        - O que poderá ser feito nesse Sprint?
        - Como o trabalho escolhido ficará pronto?
    - Algumas opções para a reunião.
        - Sprint Planning 1 e Sprint Planning 2.
        - Planejamento intercalado de Sprint.
        - Estimativas de tarefas.
        - Planejamento just-in-time de Sprint.

Resumo

- **Objetivo:** planejar o Sprint, o ciclo de desenvolvimento, que se inicia.
- **Quando:** no primeiro dia do Sprint, iniciando-o.
- **Duração:** máxima de oito horas quando os Sprints são de um mês, em geral menos quando os Sprints são mais curtos.
- **Participantes obrigatórios:** Product Owner, Desenvolvedores e Scrum Master.
- **Saídas esperadas:** Sprint Backlog.

20.1 O que é a Sprint Planning?

Definição

O planejamento do trabalho a ser implementado no Sprint, o ciclo de desenvolvimento, é realizado na Sprint Planning. Essa reunião acontece logo no primeiro momento do primeiro dia do seu Sprint.

A reunião de Sprint Planning conta com a participação obrigatória do Product Owner, dos Desenvolvedores do produto e do Scrum Master, que atua como um facilitador. Eles podem eventualmente convidar para a reunião pessoas externas que sejam capazes de oferecer informações e recomendações importantes. Todos os Desenvolvedores participam com igual poder de opinião e decisão, quaisquer que sejam suas áreas de conhecimento ou de atuação.

Nessa reunião, os participantes criam o Sprint Backlog, que é um plano com o trabalho que os Desenvolvedores acreditam que vão realizar durante o Sprint para a implementação de um ou mais Incrementos do produto, e que define o valor a ser realizado.

O resultado desse trabalho de implementação no Sprint será ao menos um Incremento entregável, ou seja, capaz de ser entregue a clientes e usuários do produto.

Sprint Backlog

O Sprint Backlog é um artefato criado a partir da colaboração entre os Desenvolvedores e o Product Owner. Ele é formado por itens retirados do Product Backlog para serem implementados no Sprint, por um plano de como o trabalho escolhido ficará pronto e pelo Objetivo do Sprint, que estabelece por que esse Sprint

que se inicia tem valor (veja os capítulos *Sprint Backlog* e *Compromisso: Objetivo do Sprint*).

Na figura a seguir, podemos ver um exemplo de Sprint Backlog antes do início da implementação de seus itens.

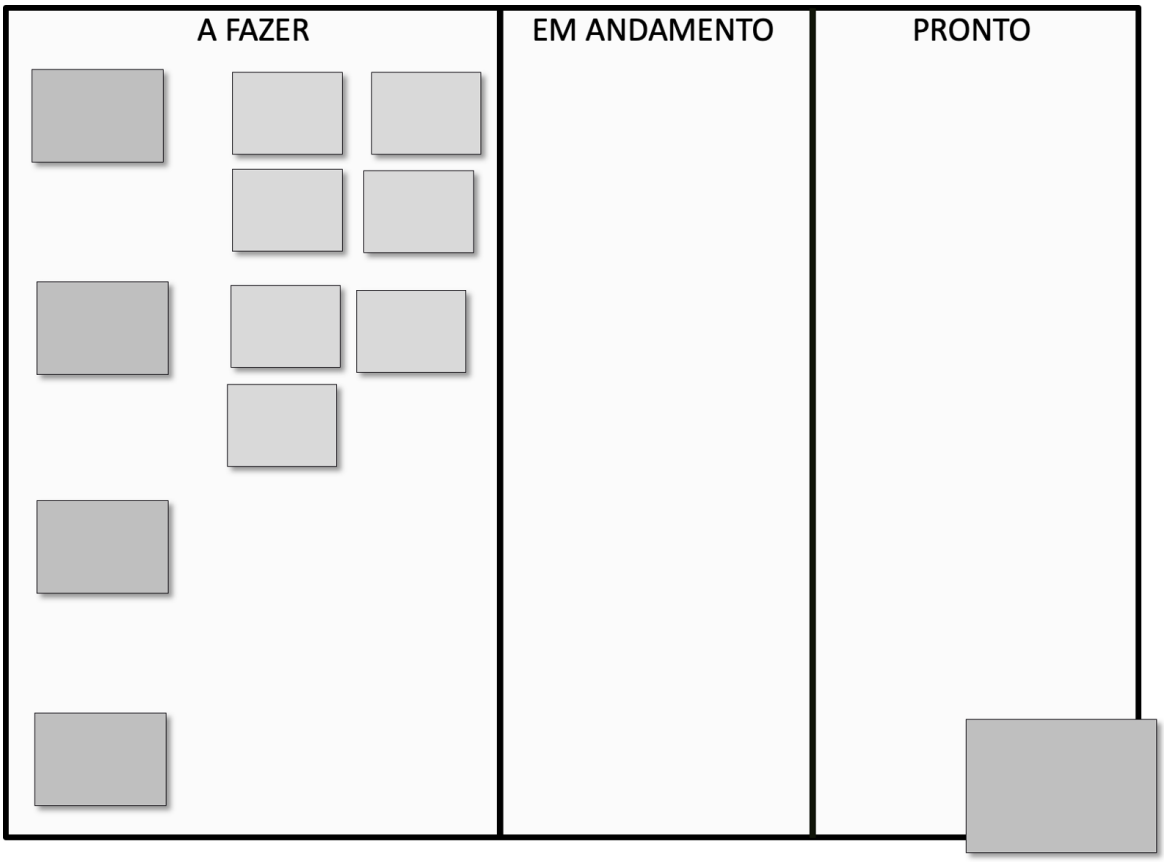


Figura 20.1: Sprint Backlog recém-criado na reunião de Sprint Planning

Repare nesse exemplo que, durante a Sprint Planning, apenas os dois primeiros itens tiveram seus detalhes de implementação definidos. O plano expresso no Sprint Backlog é incompleto e evoluirá durante o Sprint, à medida que os Desenvolvedores adquiram maior conhecimento ao executá-lo e se aproximam da implementação dos itens seguintes.

Duração

A reunião de Sprint Planning é um *timebox* de oito horas quando os Sprints duram um mês, e é geralmente mais curta quando os Sprints são mais curtos. Recomendo, no entanto, que o *timebox* da reunião não passe de 5% do tempo do Sprint, como estabelecia o Guia do Scrum original (SCHWABER, 2009). Seguindo essa recomendação, ela terá no máximo duas horas por semana de duração de um Sprint e, dessa forma, no máximo quatro horas para quando os Sprints são de duas semanas.

Ao aliarmos a prática de sessões de Refinamento do Product Backlog ao uso da Definição de Preparado, podemos reduzir significativamente a duração da reunião de Sprint Planning. No entanto, quando essas boas práticas não são usadas, a reunião tende a ficar mais longa e menos produtiva (veja a seção *Preparação* neste capítulo).

Preparação

Preparar os itens previamente à reunião de Sprint Planning pode aumentar as chances de sucesso do Sprint, ou seja, de os Desenvolvedores realizarem o Objetivo do Sprint definido e acordado. É importante, portanto, que Product Owner e Desenvolvedores colaborem para garantir um número suficiente de itens do alto do Product Backlog preparados antes da reunião de Sprint Planning. Esse trabalho de preparação pode ser realizado em sessões de Refinamento do Product Backlog durante o Sprint anterior.

É igualmente importante que os Desenvolvedores e Product Owner definam quais são os critérios mínimos para considerarem que um item está preparado para ser implementado, no que chamamos de Definição de Preparado.

Para mais informações, veja os capítulos *Refinamento do Product Backlog (adicional)* e *Compromisso: Definição de Preparado (adicional)*).

20.2 Como é a Sprint Planning?

A Sprint Planning

De forma geral, o Product Owner explica para os Desenvolvedores qual é a direção a seguir, enquanto que os Desenvolvedores estipulam quanto trabalho acreditam ser possível ser realizado no Sprint. O Objetivo do Sprint emerge dessa interação.

No entanto, apesar de não existir um formato predefinido ou obrigatório, são tratados na reunião três tópicos principais, que não necessariamente ocorrem em sequência.

No tópico "Por que esse Sprint tem valor?", o Product Owner e os Desenvolvedores definem por que esse Sprint será realizado, estabelecendo o Objetivo do Sprint.

No tópico "O que poderá ser feito nesse Sprint?", Desenvolvedores colaboram e negociam com o Product Owner sobre que itens do alto do Product Backlog acreditam que serão capazes de implementar durante o Sprint, visando a realizar o Objetivo do Sprint e, desse modo, deixá-los mais próximos do Objetivo do Produto.

No terceiro tópico, "Como o trabalho escolhido ficará pronto?", os Desenvolvedores planejam para os primeiros dias do Sprint como os itens escolhidos serão implementados, geralmente por meio da criação de tarefas e as adicionam ao Sprint Backlog.

O Scrum Master geralmente facilita essas interações entre Product Owner e Desenvolvedores.

Por que esse Sprint tem valor?

Product Owner e Desenvolvedores colaboram para estabelecer o Objetivo do Sprint, que comunica por que acreditam que o que será feito no Sprint terá valor para clientes e demais partes interessadas. Eles estabelecem um objetivo realista com que os Desenvolvedores podem se comprometer a perseguir durante o Sprint a partir da implementação dos itens escolhidos do alto do Product Backlog para o Sprint Backlog.

Frequentemente, a seleção dos itens (tópico "O que poderá ser feito nesse Sprint?") e a definição do Objetivo do Sprint ocorrem ao mesmo tempo, de forma fluida. Esse objetivo emerge como um nexo entre os itens, que foram ordenados pelo Product Owner de forma coerente entre eles, de forma a ajudar os Desenvolvedores a manterem o seu foco no que realmente interessa, a geração de valor.

O Objetivo do Sprint também pode ser negociado e estabelecido apenas após os itens terem sido escolhidos.

É comum o Product Owner trazer para a reunião uma proposta de que valor ele espera que seja gerado no Sprint. Dessa forma, ele sugere uma ideia para o Objetivo do Sprint que ele espera ver realizado, mas que será negociado e ajustado com os Desenvolvedores.

O Objetivo do Sprint deve estar definido antes do final da Sprint Planning. Uma vez terminada a reunião, ele não será mais modificado durante o Sprint. Porém, caso ele perca o sentido, o Sprint poderá ser cancelado pelo Product Owner (veja a sessão *O Sprint pode ser cancelado?*, no capítulo *Sprint*).

O que poderá ser feito nesse Sprint?

Nas atividades desse tópico, os Desenvolvedores, ao colaborarem com o Product Owner, definem o que acreditam que será implementado no Sprint que está se iniciando. Eles escolhem, a partir do alto do Product Backlog, que itens farão parte do Sprint Backlog.

> Os Desenvolvedores, em conjunto, têm a autoridade para definir quantos itens farão parte do Sprint Backlog, a partir da ordenação realizada pelo Product Owner. Essa lista de itens, no entanto, não gera um compromisso, trata-se apenas de uma previsão dos Desenvolvedores sobre o quanto acreditam que será possível implementar durante o Sprint.

O Product Owner responde a perguntas de Desenvolvedores sobre o conteúdo, propósito e importância de cada um dos itens em potencial para o Sprint, tanto individualmente quanto relacionando-o aos outros itens já descritos. Juntos, eles preparam esses itens para que possam ser implementados, detalhando e modificando-os de acordo com o necessário.

Eles podem reduzir significativamente a necessidade dessas atividades na Sprint Planning caso realizem sessões de Refinamento do Product Backlog no Sprint anterior (veja o capítulo *Refinamento do Product Backlog*). A reunião pode se tornar mais objetiva e eficiente, já que os itens chegam mais preparados para sua implementação.

A colaboração para a escolha dos itens se encerra quando os Desenvolvedores concordam que itens além dos que já foram apresentados estarão acima de quanto eles acreditam serem capazes, em conjunto, de

implementar. Essa decisão é tomada a partir da sua experiência com Sprints anteriores, que pode ser apoiada no cálculo da sua Velocidade (veja a seção *Velocidade* no capítulo *Story Points: estimando o trabalho*).

É importante o Product Owner não tentar impor mais trabalho do que os Desenvolvedores acreditam que são capazes de realizar, seja forçando a inserção de mais itens no Sprint Backlog ou impondo a redução de estimativas. É também igualmente importante que o Product Owner entenda que são os Desenvolvedores quem melhor pode prever o quanto são capazes de implementar e, portanto, essa escolha é prerrogativa deles.

Os itens selecionados são removidos do Product Backlog e passam a fazer parte do Sprint Backlog do Sprint que se inicia. Na prática, esse trabalho de seleção de itens não é necessariamente uma transposição direta de itens desde o alto do Product Backlog para o Sprint Backlog. Eles são escolhidos e negociados de forma a fazerem sentido juntos, formando um conjunto coerente, conforme mencionei anteriormente. Por essa razão, algumas mudanças na ordem original dos itens do Product Backlog poderão ocorrer.

As saídas obrigatórias desse tópico são os itens escolhidos para o Sprint Backlog.

Como o trabalho escolhido ficará pronto?

No outro tópico da reunião de Sprint Planning, os Desenvolvedores começam a decidir como trabalharão para implementar os itens escolhidos para o Sprint Backlog para que fiquem prontos, de acordo com a Definição de Pronto, de forma a realizar o Objetivo do Sprint.

Muitos times realizam na Sprint Planning o planejamento da implementação de todos os itens escolhidos. Outros preferem detalhar apenas o trabalho para os primeiros dias, e seguir planejando e executando o resto ao longo desse mesmo Sprint. Em todos os casos, o Sprint Backlog é atualizado no decorrer do Sprint, à medida que os Desenvolvedores aprendem mais sobre o trabalho que estão realizando.

Detalhar o plano para um item mais próximo da sua implementação pode reduzir significativamente o desperdício no seu planejamento, já que os Desenvolvedores o farão em um momento em que podem saber mais sobre ele. Já quando esse detalhamento é realizado inteiramente no início do Sprint, problemas a serem enfrentados podem ser identificados precocemente e, então, endereçados imediatamente, reduzindo os riscos de surpresas perigosas emergirem apenas durante o Sprint.

O guia de Scrum atual não prescreve um formato para esse plano, embora afirme que *é frequentemente realizado ao decompor itens do Product Backlog em itens de trabalho menores, de um dia ou menos* (SCHWABER; SUTHERLAND, 2020). O Guia do Scrum de 2009, no entanto, prescrevia que esse plano fosse expresso na forma de tarefas a serem realizadas durante o Sprint (SCHWABER, 2009). Essa é ainda a forma mais comum de planejamento dos Sprints. Os Desenvolvedores quebram os itens em um conjunto de tarefas correspondentes, cada uma com a duração máxima de um dia de trabalho.

Para criar as tarefas de cada item, os Desenvolvedores refletem sobre quais serão os passos de trabalho ou atividades necessárias para transformar o item em uma

parte funcionando do produto, ou seja, para que o item esteja pronto, de acordo com a Definição de Pronto.

Como referência para o trabalho de quebra em tarefas, os Desenvolvedores utilizarão os detalhes específicos para cada item escolhido, geralmente acordados com o Product Owner. Esses detalhes são frequentemente expressos como Testes de Aceitação (veja *Confirmação*, em *O que é a User Story?*, no capítulo *User Stories: representando o trabalho*), em especial no desenvolvimento de software. Ao quebrar cada item em tarefas, os Desenvolvedores provavelmente observarão atentamente tanto os detalhes de cada item quanto a Definição de Pronto.

As tarefas são geralmente pequenas, representando no máximo algumas horas de trabalho. Tarefas maiores que um dia de trabalho são de difícil acompanhamento e devem ser evitadas. Caso elas existam, a transparência diária que a reunião de Daily Scrum proporciona aos Desenvolvedores é prejudicada, já que essas tarefas podem estar em andamento por vários dias seguidos.

Durante essas atividades, o Product Owner pode ser requisitado com alguma frequência para fazer escolhas e sanar, o mais rapidamente possível, dúvidas sobre os itens e sobre o Objetivo do Sprint que invariavelmente surgirão.

O trabalho de definição das tarefas não é exaustivo nem completo. Os Desenvolvedores o fazem da melhor forma possível, com as informações e conhecimento que têm sobre os itens no momento da reunião. Inevitavelmente, à medida que os Desenvolvedores avançam no Sprint e entendem melhor o trabalho que estão realizando, novas tarefas surgirão para os itens do Sprint Backlog e outras não mais serão necessárias e desaparecerão.

Algumas opções para a reunião

No formato original da Sprint Planning, a reunião é dividida em duas. Os tópicos "Por que esse Sprint tem valor?" e "O que pode ser feito nesse Sprint?" são realizados na primeira parte e o tópico "Como o trabalho escolhido ficará pronto?", na segunda. Entretanto, existem variações importantes para esse formato.

Trago mais dois exemplos de formatos, o planejamento intercalado de Sprint e o planejamento just-in-time, além de tratar brevemente de estimativas de tarefas.

No planejamento intercalado de Sprint, os Desenvolvedores quebram em tarefas item a item, a partir do alto do Product Backlog, até que julguem que já selecionaram itens suficientes para o Sprint Backlog. Os Desenvolvedores, então, negociam o Objetivo do Sprint com o Product Owner.

No planejamento just-in-time, os Desenvolvedores apenas negociam e selecionam os itens para o Sprint Backlog, deixando para quebrá-los em tarefas apenas no momento da implementação de cada um deles durante o Sprint.

Curiosidade: Sprint Planning através dos tempos

O Guia do Scrum original de 2009 dividia a Sprint Planning em duas partes distintas e consecutivas: "Reunião de Sprint Planning Parte 1" e "Reunião de Sprint Planning Parte 2" (SCHWABER, 2009). O guia de 2010 passou a chamar as partes de "O quê?" e "Como?", ressaltando que alguns times combinam a execução delas. No guia de 2011, as duas partes passaram a ser tratadas como "O que será feito neste Sprint?" e "Como o trabalho escolhido ficará pronto?", a serem realizadas

uma após a outra. O guia de 2013 transformou as duas partes em tópicos, sem uma sequência obrigatória estabelecida para a sua execução. O primeiro tópico passou a se chamar "O que pode ser feito nesse Sprint?", explicitando uma maior flexibilidade no escopo do Sprint (SCHWABER; SUTHERLAND, 2013). Esse formato foi mantido no guia seguinte (SCHWABER; SUTHERLAND, 2017).

Já no guia de 2020, o tópico "O que pode ser feito nesse Sprint?" foi desmembrado, gerando o novo tópico "Por que esse Sprint tem valor?", que passou a carregar a criação do Objetivo do Sprint (SCHWABER; SUTHERLAND, 2020), como já mencionei anteriormente.

Sprint Planning 1 e Sprint Planning 2

O Sprint é dividido em duas partes, realizadas em sequência: Sprint Planning 1 e Sprint Planning 2. Na Sprint Planning 1, são realizadas as atividades relativas aos tópicos "Por que esse Sprint tem valor?" e "O que pode ser feito nesse Sprint?". Na Sprint Planning 2, são realizadas as atividades relativas ao tópico "Como o trabalho escolhido ficará pronto?". Esse é o formato mais tradicional e conhecido para a reunião.

Somente após a seleção dos itens para o Sprint Backlog e a definição do Objetivo do Sprint, os Desenvolvedores detalham o plano de trabalho para o Sprint, percorrendo e decompondo os itens escolhidos.

A participação do Product Owner na Sprint Planning 2 não é obrigatória. No entanto, é altamente recomendado que, no mínimo, ele fique acessível e disponível para fazer escolhas e tirar dúvidas dos Desenvolvedores.

Ao final da Sprint Planning 2, o Sprint Backlog inicial terá sido gerado, contendo o Objetivo do Sprint, os itens escolhidos na Sprint Planning 1 e um plano, ao menos parcial, de como esses itens serão implementados.

Planejamento intercalado de Sprint

Também chamado de "planejamento baseado em compromisso", o planejamento intercalado é uma alternativa a considerarmos ao formato mais conhecido da reunião de Sprint Planning e traz algumas vantagens concretas.

Em um planejamento intercalado, Product Owner e os Desenvolvedores planejam juntos durante todo o tempo. Observe que, ao contrário do que ocorre no formato tradicional, o Product Owner obrigatoriamente está presente durante toda a reunião. Métricas como a Velocidade e estimativas dos itens não são usadas, apenas a experiência, o conhecimento e um acordo entre os envolvidos.

Os tópicos "O que pode ser feito nesse Sprint?" e "Como o trabalho escolhido ficará pronto?" do Sprint Backlog são definidos de forma intercalada ao longo da reunião. Ou seja, o que será implementado e como será implementado não estão mais separados. E o tópico "Por que esse Sprint tem valor?" emerge dessa interação.

O item mais ao topo do Product Backlog é lido em voz alta. O Product Owner tira dúvidas sobre ele até que os Desenvolvedores estejam confortáveis com sua própria compreensão do item. Esse é geralmente um trabalho rápido se o Product Backlog estiver previamente refinado (veja o capítulo *Refinamento do Product Backlog (adicional)*).

Os Desenvolvedores, então, realizam as discussões técnicas necessárias e estabelecem um plano de como o item será implementado, o que geralmente é expresso por tarefas a serem realizadas durante o Sprint. O item e seu plano são adicionados ao Sprint Backlog.

Product Owner e Desenvolvedores seguem então para os itens seguintes, um a um, que também são lidos, discutidos e quebrados em tarefas (ou em alguma outra forma de plano). Ao fazê-lo, os Desenvolvedores têm uma maior compreensão sobre o trabalho necessário para transformar cada item em pronto, e podem julgar em que ponto já selecionaram itens suficientes para o Sprint Backlog.

Em seguida, o Product Owner e os Desenvolvedores, tendo em vista os itens selecionados, vão negociar e estabelecer o Objetivo do Sprint.

Planejamento intercalado de Sprint: passos

1. O próximo item de maior ordem (mais ao alto) do Product Backlog é lido, e dúvidas dos Desenvolvedores são tiradas pelo Product Owner;
2. os Desenvolvedores geram um plano de como o item será implementado, geralmente quebrando-o em tarefas;
3. a partir das tarefas, os Desenvolvedores decidem se acreditam que implementar o item durante o Sprint está dentro da sua capacidade de trabalho;
4. em caso positivo, o item é removido do Product Backlog e adicionado ao Sprint Backlog, que está sendo criado para o Sprint, e voltamos ao passo 1. Em caso negativo, o item não é selecionado e seguimos para o passo 5;
5. Product Owner e os Desenvolvedores estabelecem o Objetivo do Sprint.

Há diversas vantagens nessa abordagem. As principais são:

- a escolha dos itens, a princípio, será realizada sobre a melhor previsão possível que os Desenvolvedores podem dar naquele momento, já que se dará sobre um plano mais detalhado;

- a reunião é mais dinâmica, já que a participação de todos é ativa o tempo todo e as atividades são mais variadas, uma vez que se intercalam partes diferentes do planejamento. O dinamismo pode levar a um maior engajamento dos participantes;

- há um maior foco por parte dos Desenvolvedores. Ao iniciarem a discussão sobre o que é o item, os participantes naturalmente já começarão a pensar

em como implementá-lo. Caso sigam na discussão do próximo item sem criar o plano imediatamente, eles poderão ter maior dificuldade em manter seu foco;

- Desenvolvedores e Product Owner obtêm uma melhor compreensão conjunta sobre cada item, já que trabalham juntos do início ao fim da reunião;
- a colaboração mais próxima entre Product Owner e os Desenvolvedores aumenta o espírito de equipe e a responsabilidade conjunta sobre o Sprint.

O planejamento intercalado pode também ser útil quando usado temporariamente para times recém-formados, iniciando o trabalho de desenvolvimento de um produto. Nesse momento, os Desenvolvedores não têm noção da sua capacidade de trabalho e a escolha dos itens para o Sprint Backlog antes da sua quebra em tarefas poderá ser demasiado imprecisa. Eles podem, portanto, utilizar o planejamento intercalado no começo e, após alguns Sprints, terão uma melhor noção de sua capacidade de trabalho e podem passar a utilizar o formato de planejamento tradicional, em duas partes. A Velocidade é uma forma de definirmos sua capacidade e pode ser usada com esse fim (veja a seção *Velocidade* no capítulo *Story Points: estimando o trabalho*).

Estimativas de tarefas

A quebra em tarefas, como indiquei anteriormente, é a forma mais comum de planejar como os itens escolhidos para o Sprint serão implementados.

Caso optem, ainda na Sprint Planning, por quebrar todos os itens do Sprint Backlog em tarefas, podem ser adicionadas estimativas para o tempo de implementação de cada uma delas. A prática de estimar tarefas, que não é prescrita pelo Scrum, possui o único propósito de servir

aos Desenvolvedores para que consigam entender o progresso de seu trabalho em direção ao final do Sprint. Uma forma conhecida de realizar esse acompanhamento é por meio de um Gráfico de Sprint Burndown (veja *Gráfico de Sprint Burndown*).

Embora a maioria dos times use simplesmente "horas", diferentes unidades podem ser empregadas para essas estimativas. O principal problema com estimativas em "horas" é que a estimativa realizada por um Desenvolvedor provavelmente é diferente da estimativa realizada por outro para uma mesma tarefa. Por essa razão, os Desenvolvedores acabam sendo forçados a estabelecer prematuramente, ainda no planejamento, quem implementará cada tarefa. Essa definição, no entanto, deveria ser realizada dinamicamente ao longo do Sprint e conforme o necessário.

Já com o uso de uma escala mais simples como "tamanhos de camisa" (*T-Shirt Sizing*), ou seja, Pequeno, Médio e Grande (P, M, G), esse problema é minimizado devido à menor precisão. Podemos ver na figura a seguir um exemplo de um trecho do Quadro de Tarefas com as tarefas estimadas em Tamanhos de Camisas.

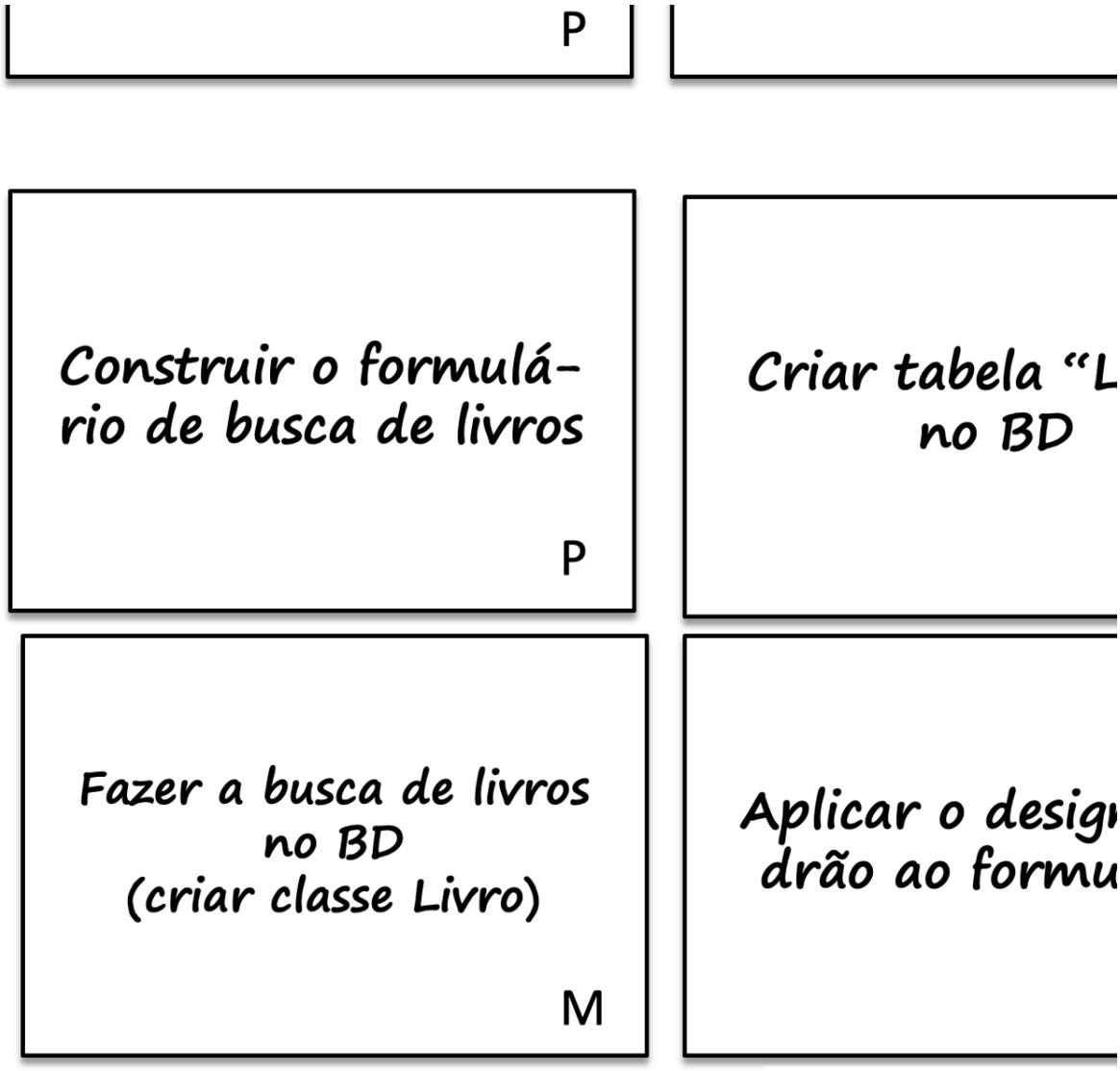


Figura 20.2: Trecho do Quadro de Tarefas com estimativas em Tamanhos de Camisas

Alternativamente, os Desenvolvedores podem quebrar cada item em tarefas pequenas e utilizar a contagem do número de tarefas restantes em cada dia para que possam realizar o acompanhamento (e, possivelmente, traçar o Gráfico de Sprint Burndown), sem a necessidade de estimativas. O resultado é muito parecido para um investimento e uma quantidade de problemas muito menores.

Uma outra alternativa é o uso das estimativas dos itens do Sprint Backlog para o acompanhamento do trabalho no Sprint, o que funciona inclusive se for criado na Sprint Planning apenas parte do plano de como esses itens serão implementados.

Planejamento just-in-time de Sprint

A máxima do planejamento *just-in-time* é a redução do desperdício. Esse formato considera que o melhor momento para realizarmos o planejamento de como um item será implementado é imediatamente antes do início da implementação do item. Ou seja, o último momento possível.

Se o planejamento dos itens for realizado na reunião de Sprint Planning, os Desenvolvedores podem não ter dados suficientes para criar um bom plano. Mudanças significativas no plano terão de ser feitas no decorrer do Sprint, o que caracterizará o desperdício.

Na reunião de Sprint Planning com o planejamento *just-in-time*, os Desenvolvedores e o Product Owner apenas selecionam os itens do Product Backlog para formar o Sprint Backlog daquele Sprint, e criam o Objetivo do Sprint (tópicos "O que poderá ser feito nesse Sprint?" e "Por que esse Sprint tem valor?", respectivamente). Sua execução é similar ao Sprint Planning 1, descrito anteriormente. O plano de como será realizada a implementação dos itens escolhidos para o Sprint Backlog (tópico "Como o trabalho escolhido ficará pronto?") não é definido nessa reunião.

Após a reunião de Sprint Planning, os Desenvolvedores planejam o trabalho a ser realizado para transformar o primeiro item do Sprint Backlog em pronto, em geral quebrando-o em tarefas a serem realizadas. O item é

implementado e, apenas quando estiver pronto, os Desenvolvedores seguirão para planejar a implementação do item seguinte. Eles então realizarão esse planejamento e seguirão dessa mesma forma pelos outros itens do Sprint Backlog, até o final do Sprint.

Capítulo 21
Daily Scrum

Conteúdo

1. O que é a Daily Scrum?
2. Como é a Daily Scrum?
3. O que NÃO deve acontecer na Daily Scrum?
    - Daily Scrums disfuncionais.
    - Disfunção: relatório de status ou prestação de contas a outros.
    - Disfunção: informe de impedimentos ao Scrum Master.
    - Disfunção: reunião de trabalho.
    - Disfunção: falta de interesse e atenção.

Resumo

- **Objetivo:** alinhar e planejar informalmente o próximo dia de trabalho.
- **Quando:** em cada dia de trabalho do Sprint, sempre na mesma hora.
- **Duração:** máxima de 15 minutos.
- **Participantes obrigatórios:** Desenvolvedores.
- **Saídas esperadas:** plano informal para o próximo dia de trabalho.

21.1 O que é a Daily Scrum?

A Daily Scrum é uma reunião curta e objetiva, realizada diariamente pelos Desenvolvedores do produto. Ela tem a duração máxima (ou *timebox*) de 15 minutos e acontece sempre no mesmo local e na mesma hora. Essa

regularidade visa a criar nos Desenvolvedores o hábito de realizarem a reunião por sua própria iniciativa.

A Daily Scrum é uma reunião dos Desenvolvedores para os Desenvolvedores. Ela é essencial para organizarem seu trabalho em busca do Objetivo do Sprint e, dessa forma, facilita o autogerenciamento do Time de Scrum como um todo.

O principal objetivo dos Desenvolvedores nessa reunião é, em cada dia de trabalho, coordenarem seu trabalho de implementação dos itens do Sprint Backlog, o que aumenta as chances de conseguirem realizar o Objetivo do Sprint. Como parte desse objetivo, a Daily Scrum proporciona entre os Desenvolvedores transparência sobre o trabalho executado e a executar, promove a comunicação sobre ele, traz visibilidade a quais obstáculos ou impedimentos estão atrapalhando e serve de oportunidade para decisões rápidas com relação ao progresso do trabalho no Sprint. Assim, essa reunião produz um plano informal para o próximo dia de trabalho dos Desenvolvedores, ou seja, para o tempo até a próxima reunião de Daily Scrum.

21.2 Como é a Daily Scrum?

A Daily Scrum dura, no máximo, 15 minutos, independente do número de Desenvolvedores.

Durante a reunião, os Desenvolvedores geralmente percorrem, uma a uma, as tarefas realizadas desde o dia anterior e projetam quais tarefas pretendem realizar até o dia seguinte, o que pode ser feito de diversas formas diferentes. No entanto, existe um padrão, que hoje já não faz mais parte do Scrum, utilizado por muitos times para

conduzi-la. Nesse formato, cada Desenvolvedor se dirige a seus colegas e responde a três perguntas:

- *O que eu fiz desde a última reunião de Daily Scrum (para ajudar a realizar o Objetivo do Sprint)?*
- *O que eu pretendo fazer até a próxima reunião de Daily Scrum (para ajudar a realizar o Objetivo do Sprint)?*
- *Vejo impedimentos que impedem a mim ou a meu time (de realizar o Objetivo do Sprint)?*

As perguntas têm o propósito de estimular a reflexão e criar alinhamento sobre o trabalho a ser realizado até a próxima reunião de Daily Scrum, mantendo o foco dos Desenvolvedores no Objetivo do Sprint.

Curiosidade: as três perguntas da Daily Scrum

O formato das três perguntas foi introduzido como obrigatório em um artigo sobre padrões do Scrum publicado em 1999 (BEEDLE ET AL., 1999), que tem Jeff Sutherland como coautor. Sutherland, cocriador do Scrum, afirma que era um padrão utilizado desde o primeiro uso de Scrum na Easel Corporation, em 1993 (SUTHERLAND, 2004). Esse formato seguiu como obrigatório, com pequenas variações, no primeiro livro sobre Scrum (SCHWABER; BEEDLE, 2002) e nos guias oficiais do Scrum desde a primeira edição, de 2009 (SCHWABER, 2009), até a edição de 2017 (SCHWABER; SUTHERLAND, 2017), quando foi retirado do guia.

Na edição de 2020, as três perguntas foram removidas do guia (SCHWABER; SUTHERLAND, 2020). Eu entendo que passaram a ser consideradas pelos autores como apenas uma das várias práticas possíveis para a reunião.

Os Desenvolvedores são sempre aqueles que conduzem a reunião, independente da presença de outras pessoas.

O Scrum Master participa, quando necessário, como um facilitador na reunião de Daily Scrum. Ele ajuda os Desenvolvedores a manterem o foco nos objetivos da reunião e a utilizarem apenas os 15 minutos previstos, o que pode ser particularmente importante para times com pouca maturidade no uso de Scrum. Sua presença na reunião, no entanto, não é obrigatória.

O Product Owner, em geral, não participa da Daily Scrum. Mas ele pode comparecer à reunião, por exemplo, para prestar esclarecimentos sobre questões relacionadas ao Objetivo do Sprint.

Discussão: Product Owner na Daily Scrum?

Dado o teor técnico e operacional da Daily Scrum, ela geralmente não é interessante nem útil para o Product Owner, e portanto pode não valer a pena ele investir o seu tempo em comparecer à reunião.

Um Product Owner inexperiente, no entanto, pode acreditar que, ao estar presente na Daily Scrum, será capaz de fiscalizar o trabalho dos Desenvolvedores e ter sobre eles um maior controle, a partir da visibilidade de detalhes do dia a dia no Sprint. Nesses casos, sua presença na reunião é nociva e seu comportamento, que é contrário aos princípios do Scrum, será tratado pelo Scrum Master.

Mesmo um Product Owner bem-intencionado pode levar a comportamentos disfuncionais ao comparecer à Daily Scrum. Ela ganha ares de reunião de status quando, talvez por desconhecimento ou por falta de maturidade, os Desenvolvedores dirigem-se principalmente a ele no seu decorrer.

Em um cenário diferente, no entanto, o Product Owner usa a Daily Scrum para se colocar presente no dia a dia dos Desenvolvedores. Ele comparece à Daily Scrum de forma benéfica e natural, mostrando-se disponível para tirar dúvidas e buscando incentivar um espírito de equipe com sua presença e colaboração.

A reunião de Daily Scrum é também informalmente conhecida como **Daily Meeting** (reunião diária) ou **Stand-up Meeting** (reunião em pé). Esse último nome vem do Extreme Programming, uma metodologia ágil de desenvolvimento de software, e indica uma prática potencialmente útil: a realização da reunião com todos os participantes de pé, com o objetivo de que eles não se permitam sentir confortáveis o suficiente para quebrar o *timebox* de 15 minutos. Porém, cabe destacar que não há obrigatoriedade no Scrum de realizarmos essa reunião em pé.

21.3 O que NÃO deve acontecer na Daily Scrum?

A reunião de Daily Scrum é importante para ajudar os Desenvolvedores a organizarem e a acompanharem seu trabalho durante o Sprint. Entretanto, há diversas disfunções comuns que podem atrapalhar a sua

execução e reduzir o seu valor. Listo algumas delas nesta seção.

Daily Scrums disfuncionais

O Scrum Master olha o relógio, é hora da Daily Scrum. Os Desenvolvedores não se mexem. O Scrum Master pensa: *ai meu deus, lá vou eu de babá de novo*. Ele chama, puxa um por um pelo braço, mas encontra resistência. *Vamos lá, pessoal, hora da Daily Scrum. Vamos!*

Todos finalmente entram na sala de reunião. Eles já conhecem bem a rotina, 15 minutos, todos de pé. Responder às três perguntas. O primeiro puxa a palavra e começa, monótono: *desde ontem fiz isso, isso e aquilo, até amanhã vou fazer aquilo, aquilo e aquilo e não há impedimentos*. O Desenvolvedor seguinte vai pela mesma linha, quase como um robô: *...e não há impedimentos*. E o seguinte. E o seguinte.

Enquanto isso, outros Desenvolvedores, desinteressados, consultam o celular ou simplesmente viajam em seus pensamentos, e quase dormem. Ninguém entende bem o que está fazendo ali. A reunião é muito chata e sem propósito. Afinal, cada um faz apenas a sua parte no dia a dia do trabalho e, ao não dividirem responsabilidades, não se interessam pelo que os outros estão fazendo.

Na sala ao lado, outro grupo de Desenvolvedores e seu Scrum Master também estão fazendo a Daily Scrum. Cada um deles se vira para o Scrum Master e responde às três perguntas. O Scrum Master faz caras e bocas, comenta, sugere e anota. A reunião termina e o Scrum Master está preocupado com o andamento da Sprint. Pensa em como ele vai cobrar mais empenho dos Desenvolvedores para que os itens planejados estejam prontos ao final da Sprint.

Por fim, mais para o final do dia, um terceiro grupo termina a Daily Scrum com quarenta e poucos minutos. Eles mais uma vez se perderam em discussões técnicas e esclarecimentos de questões de negócios. Parecia que eles queriam resolver todos os seus problemas na reunião, talvez devido à falta de colaboração ao longo do dia de trabalho. Alguns debates foram bem acalorados e completamente infrutíferos. E isso é uma constante para eles.

Mostrei aqui alguns cenários de Daily Scrums bastante disfuncionais. O fato é que a Daily Scrum não é uma reunião de status e não é uma reunião de prestação de contas, nem entre os Desenvolvedores, nem para o Scrum Master, nem para o Product Owner. Também não é uma reunião de resolução de problemas.

A Daily Scrum é uma reunião de **planejamento**, ainda que informal. Os Desenvolvedores, preferencialmente em frente ao seu quadro de tarefas, planejam o que vão fazer até o próximo dia de trabalho, ou seja, até a próxima Daily Scrum.

A interação pode se parecer com: *eu vou fazer essa tarefa aqui*. *E eu vou fazer aquela*. *Quem me ajuda nessa? Vamos puxar essa outra juntos? Não, essa aqui está impedida, mas o Scrum Master já está resolvendo*. Eles conversam e criam transparência entre eles sobre o seu trabalho, de forma a conseguirem se planejar. É claro que eles tiram muito mais benefício e têm muito mais interesse na reunião se de fato trabalharem juntos, sobre os mesmos itens, como um time deve fazer.

E as famosas três perguntas, hoje opcionais? Elas podem ser um bom guia para Desenvolvedores que estão começando com Scrum. Mas elas são sempre o meio, e

nunca o objetivo. Já vi muitos times maduros tirando muito mais benefício da reunião sem elas.

Disfunção: relatório de status ou prestação de contas a outros

A reunião de Daily Scrum não serve como instrumento de cobrança externa sobre os Desenvolvedores. Infelizmente, não é incomum ver o Scrum Master ou o Product Owner repassando a lista de tarefas e perguntando a cada Desenvolvedor o que cada um fez, o que pretende fazer e quais são os impedimentos.

Os Desenvolvedores, na verdade, não respondem por suas tarefas nem ao Scrum Master e nem ao Product Owner. Nesta reunião, portanto, cada Desenvolvedor não presta contas de seu trabalho a eles ou a quaisquer partes interessadas, mas sim apenas aos outros Desenvolvedores.

O Scrum Master pode estar presente na reunião de Daily Scrum apenas como facilitador e o Product Owner, como convidado.

Disfunção: informe de impedimentos ao Scrum Master

Informar impedimentos ao Scrum Master não é um dos objetivos da reunião de Daily Scrum. Qualquer membro do Time de Scrum informa o impedimento ao Scrum Master assim que ele é identificado, de forma que possa ser tratado o mais rapidamente possível. O objetivo de se informar, durante a reunião, quais impedimentos surgiram desde a última Daily Scrum é apenas o de trazer transparência para todos os Desenvolvedores sobre o que está atrapalhando o trabalho.

Essa disfunção pode levar Desenvolvedores, ao se depararem com impedimentos, a aguardarem até a reunião de Daily Scrum seguinte para solicitar a ajuda do Scrum Master. Esse comportamento acaba por atrasar a sua resolução, que pode atrapalhar por mais tempo do que o necessário o trabalho dos Desenvolvedores, e ameaçar a realização do Objetivo do Sprint.

Outra questão é que, ao informar um impedimento para o Scrum Master, o Desenvolvedor fornece a ele um nível de detalhes ao menos suficiente para ajudá-lo na remoção do impedimento. Essa é uma atividade que pode consumir tempo, e isso poderia atrapalhar o andamento da reunião ou até mesmo comprometer o seu *timebox* de 15 minutos.

Disfunção: reunião de trabalho

Ao trazer transparência sobre o trabalho realizado e a realizar, é comum emergirem assuntos relacionados, como discussões técnicas ou dúvidas de negócios. Com isso, é natural os Desenvolvedores quererem aproveitar o momento para discuti-los.

A reunião de Daily Scrum, no entanto, não deve ser transformada em uma reunião de trabalho. Os Desenvolvedores se limitam a realizar as conversas necessárias para definir o trabalho a realizar até o dia seguinte. Esse tipo de desvio deve ser evitado para que o foco da reunião seja mantido. No entanto, é normal nessas situações acontecer uma reunião de trabalho logo após a reunião diária, que pode contar inclusive com a presença do Product Owner, quando necessário.

Disfunção: falta de interesse e atenção

Durante a reunião de Daily Scrum, os Desenvolvedores mantêm o foco e prestam total atenção ao que seus colegas estão relatando. Nenhum outro trabalho ou conversa paralela acontecem durante a reunião.

O trabalho realizado no Sprint pertence aos Desenvolvedores como um grupo. Não há uma divisão de responsabilidades, seja por área de conhecimento ou por senioridade. Assim, todos são igualmente responsáveis pela execução e pelos resultados do trabalho. Quando essa divisão acontece, é comum o baixo interesse de Desenvolvedores no trabalho de seus colegas e, assim, a reunião de Daily Scrum torna-se naturalmente ineficiente. De todo modo, recomendo que conversas paralelas, telefones e computadores abertos sejam explicitamente proibidos durante a reunião.

Capítulo 22
Sprint Review

Conteúdo

Resumo

- **Objetivo:** obter feedback sobre o Incremento ou Incrementos do produto implementados no Sprint (inspeção e adaptação do produto).
- **Quando:** no último dia de cada Sprint, antes da reunião de Sprint Retrospective.
- **Duração:** máxima de quatro horas quando os Sprints são de um mês, em geral menos quando os Sprints são mais curtos.
- **Participantes obrigatórios:** Desenvolvedores, Product Owner, Scrum Master e quaisquer partes interessadas que possam oferecer feedback, como clientes, usuários e outros.
- **Saídas esperadas:** o Product Backlog ajustado para o próximo ou próximos Sprints, ou ao menos o feedback para tal, recebido sobre o Incremento ou Incrementos do produto implementados no Sprint; transparência sobre o progresso no desenvolvimento

do produto para clientes e demais partes interessadas.

22.1 O que é a Sprint Review?

Na reunião de Sprint Review (ou revisão do Sprint), os Desenvolvedores do produto e o Product Owner colaboram com clientes e demais partes interessadas para obter seu feedback sobre o trabalho pronto, de acordo com a Definição de Pronto, produzido durante o Sprint, ou seja, sobre o Incremento ou Incrementos do produto implementados pelos Desenvolvedores.

O feedback obtido nessa reunião é utilizado pelo Product Owner como matéria-prima para modificar o Product Backlog para Sprints futuros, o que pode ser realizado de forma colaborativa durante a própria reunião. A Sprint Review é, portanto, uma reunião de inspeção e adaptação do produto que está sendo desenvolvido.

O Product Backlog é formado por hipóteses daquilo que o Product Owner acredita que deva ser implementado para gerar o maior valor para clientes e usuários do produto em cada momento. Ele, portanto, não atua na Sprint Review como um tomador de pedidos. Pelo contrário, ainda que se trate de uma sessão colaborativa, o Product Owner detém a autoridade sobre as decisões acerca do que entra e o que não entra no Product Backlog, e de como ele é ordenado.

As presenças dos Desenvolvedores, do Product Owner e do Scrum Master são obrigatórias na Sprint Review. As demais pessoas convidadas podem ser clientes, usuários-chave, outros usuários, pessoas de negócio e outros de quem o feedback é considerado importante pelo Time de

Scrum e, possivelmente, para quem as funcionalidades implementadas no Sprint são relevantes.

Nessa reunião, o Time de Scrum busca obter diferentes visões relevantes sobre partes prontas do produto em desenvolvimento. Por meio desse feedback, podem entender como melhorar aquilo do produto que já foi desenvolvido, o que implementar em seguida e, quando necessário, realizar correções de rumo. O feedback obtido na Sprint Review, no entanto, não substitui aquele mais concreto, profundo e verdadeiro que buscamos obter de usuários ao utilizarem o produto, sempre que possível na forma de métricas de uso.

A reunião de Sprint Review acontece no último dia do Sprint, antes da reunião de Sprint Retrospective. Ela tem a duração máxima de quatro horas quando as Sprints são de um mês, e em geral são mais curtas quando as Sprints duram menos.

Discussão: Sprint Review, aprovação ou feedback?

A reunião de Sprint Review começa. Os Desenvolvedores mostram o que implementaram durante o Sprint, mas os clientes não interagem muito. Quando termina a demonstração, eles apenas fazem algumas perguntas básicas, aprovam o trabalho realizado, aplaudem e vão embora. A reunião terminou e foi considerada um sucesso!

Mas será que foi, de fato, bem-sucedida? Eu julgo que não. Acredito, sinceramente, que esse é um dos piores cenários possíveis para a reunião. Será que o que foi implementado era relevante para esses clientes naquele momento? Será que eles entenderam ou se importaram com o que lhes foi mostrado? Eles certamente vão se

preocupar quando o produto, uma vez entregue, não atender às necessidades de seus usuários!

O objetivo da Sprint Review não é obter uma aprovação formal do trabalho que foi implementado no Sprint. Trata-se de algo muito mais importante do que um polegar para cima ou um carimbo de "aprovado" no contrato. Na verdade, nem sequer é uma reunião de entrega. O principal objetivo da reunião de Sprint Review é obter feedback sobre o Incremento ou Incrementos implementados no Sprint, para alimentar o Product Backlog.

O time busca entender com os clientes e outras partes interessadas o que eles gostaram, o que não gostaram, do que sentiram falta e o que eles acharam que valeria a pena adicionar. Eles desejam esse feedback para ajustar o produto nos Sprints seguintes, adicionando, removendo ou modificando funcionalidades, e até mesmo mudando de rumo mais drasticamente, caso seja necessário. É do interesse dos Desenvolvedores e do Product Owner instigar os presentes na reunião a fornecerem seu feedback. Devem, portanto, convidá-los não apenas a seguir uma demonstração, mas também a usar o produto, enquanto os observam e aprendem sobre o seu uso real. É importante incentivá-los a fazer perguntas e despertar a sua criatividade perguntando por alternativas.

E se os clientes ou partes interessadas não gostarem do que viram? E se criticarem duramente algo que não ficou como eles acreditam que deveria ser? Ou até mesmo se entenderem que tudo o que foi feito no Sprint em nada lhes atende? O Product Owner e os Desenvolvedores não devem entrar em modo defensivo, nem se sentir mal. Ao contrário, eles devem sentir-se contentes por receber um feedback crítico, e cedo. Eles terão obtido informações

importantes sobre como potencialmente melhorar o produto. É por isso que, com Scrum, construímos e evoluímos produto pouco a pouco, de forma incremental, com revisões e entregas frequentes. Na realidade, essa é a melhor estratégia de mitigação de riscos possível.

Devemos ajudar os convidados da Sprint Review a entender os objetivos da reunião, passando da pergunta:

- *O que fizemos está aprovado?*

Para a pergunta:

- *Agora que você pôde ver o produto funcionando, o que você acredita que podemos modificar ou adicionar para melhor atender às suas necessidades de negócios e dos seus usuários?*

22.2 Como é a Sprint Review?

Preparação

A Sprint Review é uma reunião informal e não deve ser tratada nem como uma reunião de status, nem como uma reunião de entrega. Porém, pode ser interessante que o Time de Scrum realize uma breve sessão de preparação ao final do trabalho no Sprint. Nela, Desenvolvedores e Product Owner podem alinhar o entendimento sobre o que foi produzido durante o Sprint e definir como pretendem que a Sprint Review ocorra, antes de estarem face a face com os convidados para a reunião.

Não é necessário nem recomendado, no entanto, haver grandes preparações, nem a geração de relatórios ou de

apresentações bonitas. Toda a interação, durante a reunião, entre membros do Time de Scrum, clientes e demais partes interessadas está focada em obter feedback sobre o produto funcionando, ou seja, sobre o Incremento ou Incrementos implementados.

Gestão inicial de expectativas

É saudável que o Time de Scrum se preocupe com a gestão das expectativas dos convidados. Ao entenderem claramente o que se pretendeu gerar de valor como resultado do Sprint, eles vão compreender melhor o que os Desenvolvedores produziram e serão capazes de fornecer um feedback mais realista e alinhado.

Por essa razão, recomendo que Product Owner e Desenvolvedores esclareçam, logo na abertura da reunião, qual o Objetivo do Sprint que buscam realizar (veja o capítulo *Compromisso: Objetivo de Sprint*), e que o relembrem para os presentes sempre que necessário. Adicionalmente, os clientes e outras partes interessadas já podem ser informados sobre esse objetivo no momento do convite para a reunião, como parte da preparação para ela.

Essa prática tem o efeito adicional positivo de estimular que o foco dos convidados se desvie do plano e se mantenha no valor gerado.

Quem participa

Estão presentes na Sprint Review todos os Desenvolvedores, o Product Owner, o Scrum Master e convidados que possam trazer feedbacks relevantes sobre o Incremento ou Incrementos implementados no Sprint.

Esses convidados podem ser clientes, usuários-chave, pessoas de negócio, especialistas e quaisquer outras pessoas cujo feedback o Time de Scrum considera importante, mas que não estão disponíveis durante o Sprint para oferecê-lo. De forma geral, são pessoas que podem ser impactadas de alguma forma pelos resultados do Sprint ou que trazem conhecimento e experiência que possam contribuir para a evolução do produto. Por essa razão, pode ser ainda mais interessante haver usuários reais entre os presentes.

Ao verem, interagirem com e experimentarem Incrementos do produto prontos e funcionando, essas pessoas podem tornar-se capazes de trazer valiosos feedbacks sobre o produto, que ajudam a guiar a definição do que será feito em seguida. Esses feedbacks reduzem a geração de desperdício ao permitirem conformar os rumos dos Incrementos do produto seguintes de acordo com os objetivos e necessidades de quem realmente importa.

Em muitos casos, infelizmente, os únicos participantes da reunião de Sprint Review são os membros do Time de Scrum, e não há quaisquer pessoas externas presentes. Nesse cenário, os Desenvolvedores fazem a demonstração enquanto o Product Owner é o único que fornece feedback sobre o Incremento ou Incrementos prontos. Com isso, confiança em excesso está sendo depositada na representatividade do Product Owner e em seu conhecimento sobre as necessidades de clientes e sobre o mercado. As pessoas para as quais o produto desse Sprint é relevante somente poderão fornecer algum tipo de feedback após uma entrega para o seu uso. Esse feedback tardio aumenta consideravelmente os riscos de o time gerar um grande desperdício.

O Scrum Master possui o importante papel de facilitar a reunião de Sprint Review, mas ele não participa diretamente na apresentação do que foi feito, nem dá suas opiniões sobre o que os Desenvolvedores implementaram.

Colaboração

Em essência, o Time de Scrum e os convidados interagem, na Sprint Review, para revisar aquilo que foi implementado pelos Desenvolvedores durante o Sprint, o que serve para definir o que será feito em seguida.

Inicialmente, Desenvolvedores e o Product Owner se coordenam para apresentar ou demonstrar aos presentes os resultados do trabalho em busca de feedback. Eles podem fazê-lo integralmente em conjunto, mas não há regra de como isso deva ocorrer. O Product Owner também pode estar à frente, com Desenvolvedores oferecendo o apoio necessário. Ou, de outro modo, alguns ou todos os Desenvolvedores podem estar à frente com o Product Owner ao seu lado ou junto aos outros presentes.

O Incremento ou Incrementos implementados podem ser demonstrados como um todo, ou passando por cada item pronto do Sprint Backlog, daquele de maior ordem em direção ao de menor ordem. Mas a reunião não se limita a uma apresentação. Clientes e demais presentes trabalham colaborativamente com os Desenvolvedores e com o Product Owner, fazendo perguntas e obtendo respostas sobre o que lhes está sendo demonstrado, e apresentando suas ideias sobre o que esperam do produto nos próximos Sprints. Para trazer elementos importantes para essa discussão, o Product Owner pode oferecer detalhes do Product Backlog tal como está no momento e falar sobre perspectivas de mercado relativas

ao produto. Pode também ser interessante perguntar diretamente aos presentes o que esperam ver pronto nas próximas reuniões de Sprint Review. Estimulá-los a elaborar sobre o que acreditam que são as próximas necessidades de negócios mais importantes a serem atendidas.

Recomendo convidar os presentes a experimentarem diretamente o produto. Usar o produto os ajudará a entender melhor o que foi gerado, o que lhes serve e o que não lhes serve, além do que esperam para adiante. Ou seja, os estimulará a fornecerem feedback e permitirá que ele seja mais profundo e preciso.

A Sprint Review, no entanto, não é uma reunião para a realização de testes do produto e, assim, não deve ser utilizada para substituir práticas de testes que devam acontecer ao longo do Sprint.

Mudanças no Product Backlog podem ser realizadas pelos participantes, de forma colaborativa, durante a própria reunião. Prefiro, no entanto, deixar que o Product Owner decida qual a melhor forma de utilizar o feedback obtido para atualizar o Product Backlog. Ele, por exemplo, pode julgar que, dado o contexto e quem está presente na reunião, seja melhor atualizar o Product Backlog num momento posterior, utilizando o que viu e ouviu na reunião como matéria-prima para tal.

Foco

O foco dos presentes se mantém durante toda a Sprint Review no Objetivo do Sprint e no Incremento ou Incrementos que foram implementados para realizá-lo, isto é, no valor gerado. Por essa razão, em princípio, o Time de Scrum trata na reunião apenas dos itens do

Sprint Backlog que estejam prontos, de acordo com a Definição de Pronto.

O Time de Scrum também evita mostrar *slides* ou documentos que não fazem parte do produto, dar explicações excessivas sobre planos ou sobre o processo de trabalho, falar sobre intenções não realizadas e dar desculpas ou justificativas.

Pelas mesmas razões, questões técnicas são geralmente evitadas. Recomendo que somente sejam abordadas caso se mostrem do interesse das pessoas presentes, e sempre com o cuidado de não tornar a reunião tediosa.

Resultados

O feedback obtido a partir da interação entre Desenvolvedores, Product Owner e demais presentes é um dos principais resultados da reunião de Sprint Review. Ele é usado como matéria-prima para adicionar, remover ou modificar itens do Product Backlog e, portanto, é um ingrediente importante para a construção incremental do produto.

Outro resultado é a noção de progresso real oferecida aos clientes e demais partes interessadas ao verem e interagirem com partes prontas do produto e, principalmente, ao apreciarem o problema proposto para o Sprint resolvido, ou seja, o Objetivo do Sprint.

É importante observar que, para realizar esse objetivo, os Desenvolvedores não necessariamente terão completado todos os itens planejados. Os itens não prontos ao final do Sprint são geralmente os de mais baixa importância relativa à realização do Objetivo do Sprint, salvo quando algum impedimento tenha prevenido um item importante de ser implementado.

Na figura a seguir, por exemplo, podemos observar um quadro de tarefas representando o Sprint Backlog de um Sprint possivelmente bem-sucedido, apesar de um dos itens não ter sido completado.

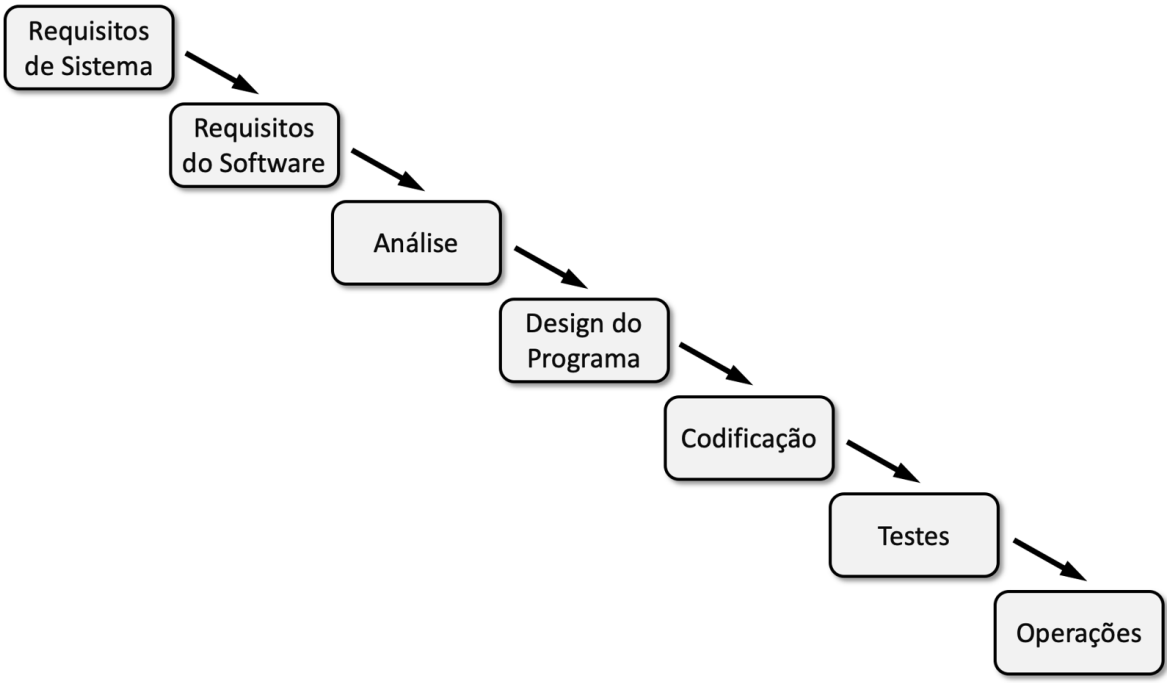


Figura 22.1: Quadro de tarefas (Sprint Backlog) ao final de um Sprint

Esses itens do Sprint Backlog que chegarem à reunião de Sprint Review não prontos, de acordo com a Definição de Pronto, poderão retornar ao Product Backlog e reaparecer em algum Sprint futuro, ou poderão ser descartados, se assim resolver o Product Owner. É papel dele, e apenas dele, decidir o destino desses itens, o que faz a partir do feedback recebido e de novos aprendizados sobre o produto.

Discussão: julgamento na Sprint Review

Em uma visão antiga do Scrum, o Product Owner deveria verificar, na reunião, se cada item planejado e apresentado está de fato pronto, de acordo com a

Definição de Pronto. Ainda de acordo com essa abordagem, o Product Owner estabeleceria, a partir do que foi e do que não foi implementado no Sprint, se os Desenvolvedores conseguiram realizar o Objetivo do Sprint.

Essa prática, para mim, não é compatível com o que esperamos de Times de Scrum. Nesses times, a relação de trabalho entre os seus membros deveria ser colaborativa e baseada na transparência. Lembro que a Definição de Pronto existe justamente para que, quando os Desenvolvedores digam que algo está pronto, todos entendam o que isso significa, sem a necessidade de maiores explicações. Além disso, o Product Owner colabora com os Desenvolvedores durante o Sprint, de acordo com o necessário, e não deveria ter surpresas sobre os seus resultados.

Os Desenvolvedores afirmarem que algo está pronto sem que, de fato, esteja, é geralmente um indicador de disfunções graves. Eles e o Product Owner podem simplesmente estar desalinhados quanto à Definição de Pronto. Mas pode também haver um problema grave no compromisso dos Desenvolvedores com a verdade. Ou, talvez, exista algum tipo de pressão para que eles aceitem mais trabalho do que são capazes de completar, enquanto são medidos por quanto entregam. Nesses casos, tratar o problema pode ser mais importante do que garantir que os itens estejam de fato prontos.

Qualquer que seja o caso, caracterizar a Sprint Review como um julgamento faz com que os membros do Time de Scrum se distanciem de seu objetivo primário, aquele de colaborar com os clientes e demais partes interessadas presentes na reunião para definir o que será feito em seguida.

Capítulo 23
Sprint Retrospective

Conteúdo

1. O que é a Sprint Retrospective?
    - Lições aprendidas em times tradicionais.
    - Melhoria incremental contínua em Times de Scrum.
    - A retrospectiva do Sprint.
2. Como é a Sprint Retrospective?
    - Regularidade, frequência e duração.
    - Participação do Scrum Master.
        - Facilitação.
        - Neutralidade.
        - Impedimentos.
    - Participação do Product Owner.
    - Dinâmica básica de uma retrospectiva.
    - Diferentes possibilidades na retrospectiva.
3. O que NÃO deve acontecer na Sprint Retrospective?
    - Busca de culpados.
    - Insegurança e medo de exposição.
    - Conflitos da diversidade.
    - Pensamento grupal.

Resumo

- **Objetivo:** definição de melhorias na forma como o Time de Scrum realiza o seu trabalho para aumentar a sua efetividade (inspeção e adaptação de como trabalha).
- **Quando:** no último dia de cada Sprint, após a reunião de Sprint Review.

- **Duração:** máxima de três horas quando os Sprints são de um mês, em geral menos quando os Sprints são mais curtos.
- **Participantes obrigatórios:** Desenvolvedores, Product Owner e Scrum Master.
- **Saídas esperadas:** planos de ação para melhorias a serem realizados já no Sprint seguinte.

23.1 O que é a Sprint Retrospective?

Em intervalos regulares, a equipe reflete sobre como se tornar mais efetiva e, então, refina e ajusta seu comportamento de acordo — Princípio Ágil.

Lições aprendidas em times tradicionais

Os acertos e erros em projetos que utilizam métodos tradicionais são geralmente levantados e discutidos apenas uma vez, ao final deles, na esperança de que essas lições aprendidas possam ser úteis em trabalhos futuros.

A reunião de Lições Aprendidas ou *postmortem*, como também é conhecida, busca cumprir esse papel, realizando uma verdadeira autópsia no projeto sem, no entanto, ter contribuído em nada para a sua saúde enquanto o projeto ainda estava vivo. Além de não ser útil para o próprio projeto, apenas uma pequena parte dessas lições pode ser aproveitada, se tanto, pois novos projetos envolvem novas condições, pessoas e desafios.

Melhoria incremental contínua em Times de Scrum

Com origem no Sistema Toyota de Produção, *kaizen* é um conceito japonês que literalmente significa "mudar para

melhor" e se traduz na prática da melhoria incremental contínua (veja a subseção *Buscar a perfeição* no capítulo *De onde veio o Scrum?*).

O objetivo da reunião de Sprint Retrospective (ou retrospectiva do Sprint) é estimular o Time de Scrum a realizar essa prática. Times que utilizam Scrum estão sempre buscando melhores formas de realizar seu trabalho, aprendendo com seus acertos e erros ao longo do tempo.

O Princípio Ágil citado no início deste capítulo reflete exatamente isso: o Time de Scrum, ou seja, os Desenvolvedores, Product Owner e Scrum Master se reúnem periodicamente com os objetivos de identificar pontos de melhoria em como fazem seu trabalho (inspeção) e de traçar planos de ação para executar essas melhorias (adaptação).

Por buscar a implementação de melhorias já no Sprint seguinte, a prática da retrospectiva é muito mais objetiva e eficiente do que a prática mais tradicional de avaliarmos as lições aprendidas somente ao final de um projeto.

Curiosidade: retrospectivas e o Scrum

Após a Sprint Review e antes da reunião de Sprint Planning seguinte, o ScrumMaster realiza uma reunião de Sprint Retrospective com o Time. - Ken Schwaber (2003).

A Sprint Retrospective foi citada como parte do Scrum pela primeira vez em um artigo escrito por Ken Schwaber em 2003, de onde foi retirado o trecho anterior. Apesar

de ser descrita nos Princípios Ágeis, criados em 2001, a reunião não fazia parte do framework no artigo seminal do Scrum, apresentado no congresso OOPSLA em 1995 (SCHWABER, 1997) e nem no livro *Agile Software Development with Scrum*, primeiro livro sobre o assunto, de 2002 (SCHWABER, 2002), um ano após a criação do Manifesto Ágil (BECK et al., 2001).

A retrospectiva do Sprint

Na reunião de Sprint Retrospective, o Time de Scrum inspeciona o Sprint que está se encerrando quanto a seus processos de trabalho, dinâmicas, pessoas, relacionamentos, comportamentos, práticas, ferramentas utilizadas e ambiente, e planeja as melhorias necessárias para o Sprint seguinte.

Os membros do Time de Scrum identificam o que foi bem, e que por essa razão querem manter no próximo Sprint, e o que podem melhorar. Eles buscam as causas raízes dos problemas enfrentados durante o Sprint, e traçam planos de ação objetivos para realizarem as melhorias.

A reunião é uma colaboração e nunca deve se configurar como um espaço para trocas de acusações ou discussões improdutivas. O Scrum Master atua como um facilitador, conforme necessário.

23.2 Como é a Sprint Retrospective?

Regularidade, frequência e duração

O Time de Scrum realiza a reunião de Sprint Retrospective no último dia de todo e cada Sprint, logo após a reunião de Sprint Review, com o objetivo de garantir a melhoria incremental contínua. A obrigatoriedade dessa prática visa a assegurar regularidade e frequência na realização dessas melhorias. Traz, dessa forma, oportunidades ao Time de Scrum de promover mudanças para corrigir problemas antes que estes afetem negativamente os resultados de seu trabalho em Sprints futuros.

Em cenários de prazos curtos e ritmo de trabalho consequentemente acelerado, é comum os Desenvolvedores sofrerem pressões internas e externas para suprimir quaisquer esforços que não façam parte do trabalho de desenvolvimento do produto propriamente dito. Nesse cenário, observo que a reunião de Sprint Retrospective é uma das primeiras a ser repetidamente cancelada, e até mesmo definitivamente descartada. Sem a reunião, o Time de Scrum dificilmente realiza a inspeção e a adaptação. Assim, sequer consegue avaliar o porquê de não ser capaz de realizar as tarefas em um ritmo sustentável, fechando-se em um círculo vicioso.

Como guardião das práticas do Scrum, é papel do Scrum Master estimular o Time de Scrum a realizar as retrospectivas com a regularidade exigida pelo framework.

A reunião de Sprint Retrospective deve durar no máximo três horas para quando os Sprints são de um mês, mas em geral duram menos quando os Sprints são mais curtos.

Participação do Scrum Master

Facilitação

O Scrum Master pode atuar como um facilitador na reunião de Sprint Retrospective, conforme necessário, trabalhando junto ao resto do Time de Scrum para minimizar as dificuldades de comunicação e de colaboração, ingredientes necessários para a geração de ideias. A partir das questões levantadas nessa reunião, o Scrum Master incentiva Product Owner e Desenvolvedores a encontrarem soluções para trabalharem com qualidade e tornarem-se cada vez mais efetivos, buscando sempre manter um ritmo sustentável de trabalho.

A variação na forma é, em geral, positiva para o resultado das reuniões de Sprint Retrospective. Assim, nesse trabalho de facilitação, o Scrum Master pode introduzir diferentes técnicas e práticas, propondo de tempos em tempos sua experimentação ao Time de Scrum, visando a tornar as retrospectivas mais efetivas.

O Scrum Master, enquanto facilitador, estimula todos os Desenvolvedores e o Product Owner a participarem igualmente das atividades. Aqueles pouco participativos, por exemplo, e aqueles que agem de forma oposta, deixando pouco espaço para os outros se manifestarem, representam um especial desafio à facilitação.

O Scrum Master também pode cuidar da agenda da reunião, assegurando que nela aconteçam todas as atividades necessárias para chegarem a resultados, e garantindo que o seu *timebox* seja cumprido.

Neutralidade

O Scrum Master, enquanto facilitador, deveria evitar de participar diretamente na realização das atividades, ou de interferir com suas opiniões nas ideias e no processo decisório do Time de Scrum. Para que as melhorias sejam

corretamente identificadas e colocadas em prática, é importante que os Desenvolvedores e o Product Owner sejam responsáveis pelo processo de geração de ideias, sentindo-se, assim, igualmente responsáveis por seus resultados.

Impedimentos

O Scrum Master também exerce na reunião o importante papel de não deixar passarem despercebidos pontos que exigirão sua atuação direta. Impedimentos dificultam consideravelmente ou obstruem o trabalho dos Desenvolvedores na realização do Objetivo do Sprint, e garantir sua remoção é responsabilidade do Scrum Master.

Os impedimentos são identificados normalmente pelos Desenvolvedores no seu trabalho diário e imediatamente comunicados ao Scrum Master. No entanto, não é incomum eles emergirem na reunião de Sprint Retrospective na forma de pontos a serem melhorados. A responsabilidade pela gestão da resolução desses impedimentos detectados na reunião também recai sobre o Scrum Master.

Participação do Product Owner

O Product Owner é membro do Time de Scrum e, como tal, participa integralmente das reuniões de Sprint Retrospective, colaborando com as melhorias incrementais contínuas. No entanto, alguns comportamentos disfuncionais podem prejudicar a sua participação nessas reuniões. Por essa razão, não é incomum encontrarmos recomendações equivocadas de que o Product Owner não deveria estar presente nelas.

Enquanto facilitador, todavia, é papel do Scrum Master trabalhar para garantir uma participação positiva e efetiva de todos os membros do Time de Scrum nas reuniões de Sprint Retrospective, o que certamente inclui o Product Owner.

Na presença de um Product Owner não colaborativo, que seja ou aja como um chefe ou como um cliente, os Desenvolvedores podem se sentir intimidados caso questões a serem levantadas toquem em pontos sensíveis ou estejam diretamente relacionadas com o trabalho do Product Owner. Dessa forma, essas questões podem acabar não se revelando durante a reunião, tornando-a ineficiente.

A intimidação pode também acontecer na direção contrária. Os Desenvolvedores podem entender que é apropriado aproveitar a presença do Product Owner para manter um foco excessivo em problemas relacionados ao trabalho dele. O Product Owner, nesse caso, passa a ser o centro das atenções durante a reunião ou, pior, um alvo de reclamações e acusações. Esse comportamento é comum quando o Product Owner é ausente durante o Sprint, o que, de fato, pode configurar-se como um problema a ser resolvido.

A ocorrência de algum desses casos depende de que tipo de relação o Product Owner e os Desenvolvedores cultivam. De forma geral, quanto mais próximos eles trabalharem, e quanto mais interagirem e colaborarem no seu dia a dia, mais a participação do Product Owner na reunião será vista como natural e positiva.

Dinâmica básica de uma retrospectiva

Descrevo em seguida um exemplo de como pode se parecer a dinâmica básica de uma reunião de Sprint

Retrospective. Destaco que existem diversas formas de conduzirmos essa reunião, mas com o objetivo final sempre de realizarmos melhorias na forma como o Time de Scrum realiza o seu trabalho a partir de planos de ação concretos.

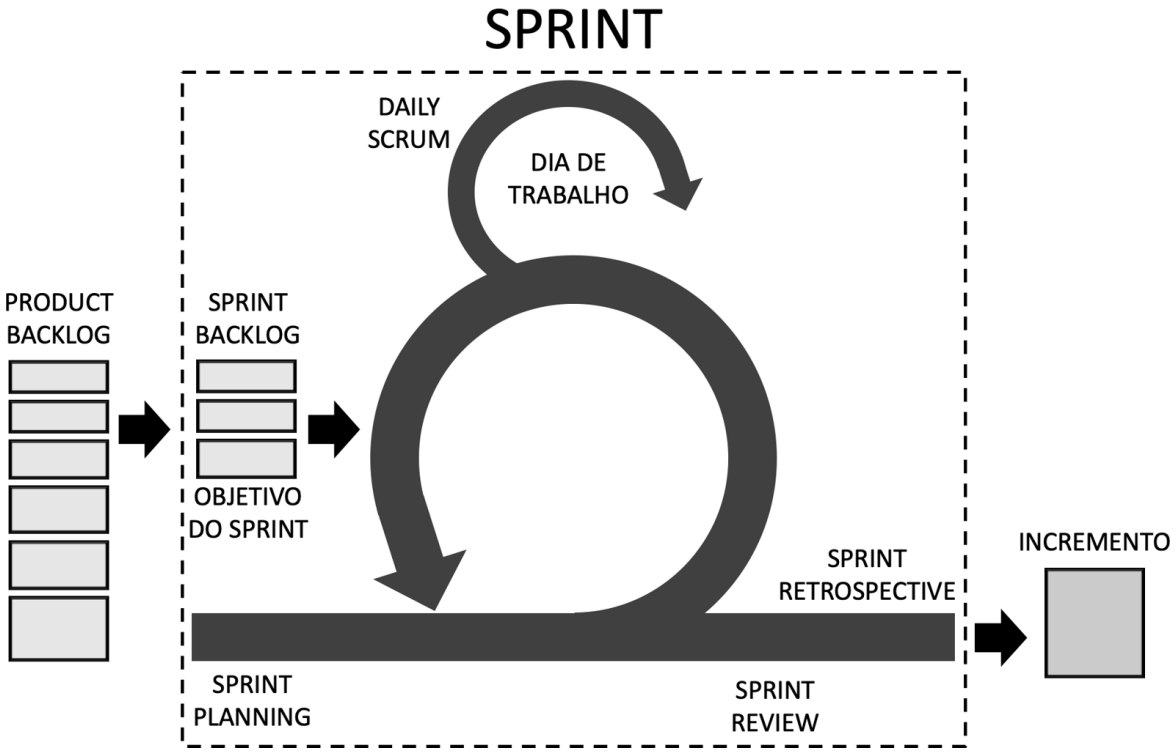


Figura 23.1: Quadro básico de retrospectiva

1. Todos os membros do Time de Scrum se reúnem em uma sala privada para realizar a reunião;
2. Desenvolvedores e Product Owner repassam rapidamente os pontos levantados na reunião de retrospectiva anterior. O objetivo é verificar se e como as ações definidas para realizar as melhorias foram executadas, e que pontos importantes ainda restaram para serem tratados no Sprint seguinte;
3. eles utilizam alguns minutos para relembrar os fatos mais marcantes do Sprint que está se encerrando. Para viabilizar essa atividade, o Scrum Master pode sugerir o uso de diferentes práticas;

4. mais alguns minutos são então reservados para que os Desenvolvedores e o Product Owner reflitam e escrevam (em geral, em notas adesivas) o que consideram que foi bem no Sprint que está se encerrando, e o que consideram que pode ser melhorado já no Sprint seguinte;
5. em seguida, eles colam suas notas adesivas em um quadro com as colunas "o que foi bem" e "o que pode melhorar" (um exemplo desse quadro pode ser visto na figura anterior);
6. eles agrupam itens parecidos ou relacionados, com a facilitação do Scrum Master, para em seguida ordenar esses grupos de itens de acordo com sua importância;
7. Desenvolvedores e Product Owner, em seguida, discutem em torno dos grupos de itens levantados. Se houver itens demais para serem discutidos dentro do *timebox* da reunião e para serem tratados no próximo Sprint, os participantes devem concentrar-se apenas nos itens mais importantes, em geral não mais que dois ou três. Para definir quais são os itens mais importantes, eles podem usar como critério o número de participantes que apontaram o item ou alguma forma de votação;
8. eles repassam rapidamente cada item da coluna "o que foi bem", comprometendo-se a manter ou repetir no Sprints seguintes esses pontos identificados como positivos, caso faça sentido, e refletem sobre como tornar isso possível;
9. com relação aos pontos a melhorar, os Desenvolvedores e o Product Owner buscam identificar as causas raízes dos problemas e, a partir daí, formulam as ações necessárias para colocar as melhorias correspondentes em prática já no Sprint seguinte. Dessa forma, buscam tornar seu trabalho mais efetivo o mais cedo possível. Planos de ação

são mais efetivos quando indicam "o quê" (qual é a questão a ser tratada), "quem" (quem ficará responsável por ela) e "quando" (em que prazo ela será tratada);
10. após repassarem todos os pontos, traçarem planos de ação adequados e registrá-los de alguma forma — em um papel à parte, por exemplo, ou até mesmo em notas adesivas coladas sobre os itens da coluna "o que pode melhorar" -, os presentes dão a reunião como encerrada.

Diferentes possibilidades na retrospectiva

O estabelecimento de uma rotina repetitiva pode ser uma receita de fracasso no médio prazo para retrospectivas de um Time de Scrum. A escolha de técnicas apropriadas para cada situação ou até mesmo a simples variação nas práticas utilizadas podem tornar a reunião mais dinâmica, o que mantém os membros do Time de Scrum envolvidos e interessados. O Scrum Master, enquanto facilitador, geralmente escolhe e sugere como se dará a reunião.

O livro *Agile retrospectives: making good teams great*, de Esther Derby e Diana Larsen (2006), oferece, além de um framework para a execução das retrospectivas, diversas práticas que podem ser usadas. O livro *Retrospectivas Divertidas: atividades e ideias para fazer retrospectivas ágeis mais envolvente*, de Paulo Caroli e Tainã Caetano, traz diversas práticas interessantes, que também podem ser encontradas em http://www.funretrospectives.com. Outras técnicas podem ser encontradas no livro eletrônico da K21 *Retrospectivas ágeis* (https://k21.global/lp/ebook-retrospectivas-ageis).

Mostro, a seguir, algumas das práticas mais populares entre times ágeis:

- **coluna "a fazer"**: adicionamos mais uma coluna ao quadro da retrospectiva, onde serão colocadas notas adesivas com as ações de melhorias planejadas. As melhorias podem estar diretamente alinhadas aos pontos da coluna "O que pode melhorar" correspondentes;
- **coluna "delta"**: a coluna "o que pode melhorar" é trocada pela coluna "delta", que simboliza mudança. Assim, o foco da discussão se desloca do que pode melhorar para a melhoria em si, ou seja, as notas adesivas coladas nessa coluna já descrevem as mudanças a serem realizadas;
- **começar-parar-tentar**: em vez do quadro tradicional, dividimos o quadro nas colunas "Começar a fazer", "Parar de fazer" e "A tentar". Os participantes colam na primeira coluna notas adesivas com ações que consideram que devem começar a fazer, na segunda coluna ações que realizaram no último Sprint, mas que consideram que devem parar de fazer e, na terceira, ações a tentarem no próximo Sprint;
- **classificação dos pontos**: existem diversas possíveis classificações para os pontos positivos e os pontos a melhorar que podem ser úteis para os membros do Time de Scrum, entre as quais destaco:
    - classificar os pontos a melhorar entre "Time de Scrum" e "Organização", o que significam, respectivamente, pontos sobre os quais os próprios membros do Time de Scrum devem agir e questões organizacionais a respeito das quais somente o Scrum Master pode fazer algo ou exercer influência;

    - classificar os pontos a melhorar entre "Ação" e "Gradual", ou seja, entre pontos para os quais é possível o time traçar planos de ação mais precisos, ou objetivos e pontos para os quais as mudanças são mais subjetivas e graduais, como mudanças comportamentais;
    - classificar tanto os pontos positivos quanto os pontos a melhorar entre "Causado pelo Scrum", quando a questão foi causada pelo uso do Scrum (repare que aqui geralmente vão pontos positivos), e "Visível pelo Scrum", nos casos em que foi o Scrum que tornou a questão visível;
- **linha do tempo**: os participantes traçam uma linha do tempo com os acontecimentos mais importantes do Sprint e, a partir deles, identificam os pontos positivos e os pontos a melhorar. Observe que o próprio Gráfico de Sprint Burndown pode ser usado como essa linha do tempo, com a vantagem que seus pontos de inflexão para cima e linhas retas podem indicar impedimentos ou problemas que ocorreram durante o Sprint (veja o capítulo *Burndown e Burnup: acompanhando o trabalho*). Nesse caso, pode ser interessante que os participantes colem notas adesivas sobre pontos relevantes no próprio Gráfico de Sprint Burndown, indicando esses acontecimentos (veja a figura a seguir);

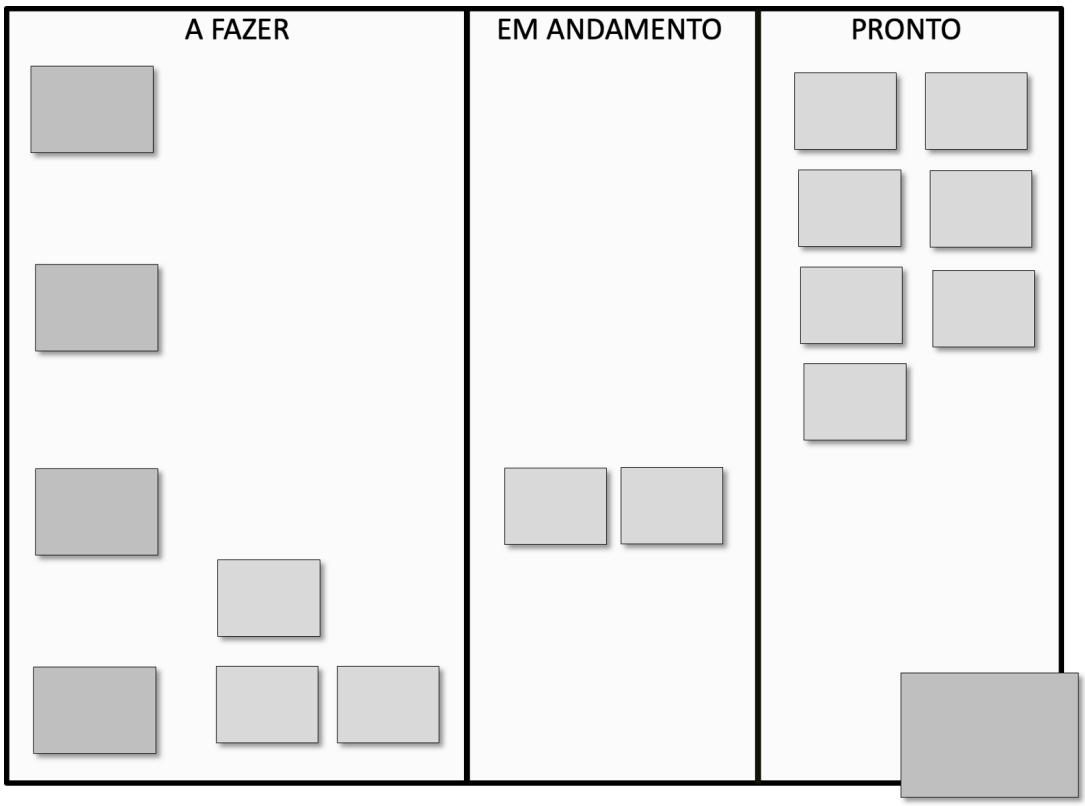


Figura 23.2: Sprint Burndown utilizado como linha do tempo para a retrospectiva

- **foco único**: caso tenha havido algum problema importante suficiente que tenha causado grande impacto no Sprint e que os membros do Time de Scrum acreditem que poderão se repetir, a busca de soluções para esse problema pode ser o tema único (ou, ao menos, central) da reunião.

23.3 O que NÃO deve acontecer na Sprint Retrospective?

A reunião de Sprint Retrospective não deve ser utilizada para identificarmos ações de melhoria no produto, trabalho esse que acontece na reunião de Sprint Review. O principal foco da reunião de Sprint Retrospective é a

identificação do que precisa ser melhorado na forma de trabalho do Time de Scrum. Assim, é necessário que questões ocorridas durante o Sprint que se encerra sejam identificadas e exploradas com franqueza.

Dessa forma, embora não devamos apontar erros individuais nessa reunião, o trabalho de inspeção envolve a capacidade dos participantes de expor, em algum grau, suas próprias limitações e pontos fracos e, principalmente, as do Time de Scrum como um todo. Diante dessa necessidade, o Time de Scrum enfrenta importantes desafios que interferem em sua capacidade de executar uma reunião de Sprint Retrospective eficaz. Descrevo a seguir algumas disfunções comuns.

Busca de culpados

Norm Kerth (2001), em seu livro *Project retrospectives*: a handbook for team reviews_, definiu o que chamou de Principal Diretiva das Retrospectivas.

> **Principal Diretiva das Retrospectivas**
>
> "*Independente do que descobrirmos, entendemos e verdadeiramente acreditamos que todos fizeram o melhor trabalho que puderam, dado o que sabiam no momento, suas competências e habilidades, os recursos disponíveis e a situação que se apresentou.*"

Os objetivos de reuniões de Sprint Retrospective são sempre construtivos. O foco dos participantes da reunião deve se manter no desejo objetivo de melhoria, aprendendo-se com os erros cometidos.

Buscar culpados para os problemas que aconteceram durante o Sprint ou "lavar roupa suja" tem efeitos

extremamente negativos sobre o andamento da reunião e sobre seus resultados. Em hipótese alguma devemos fazer acusações, apontar nomes ou citar erros individuais, relacionando-os com problemas que ocorreram no Sprint.

O Scrum Master, como facilitador, trabalha para que não aconteça uma busca de culpados, de forma que os Desenvolvedores e Product Owner se sintam mais seguros para falar livremente sobre os problemas.

Insegurança e medo de exposição

Algum nível de exposição dos participantes pode ser necessário para uma participação efetiva na reunião de Sprint Retrospective. No entanto, falar sobre o que não funcionou bem em seu próprio trabalho e no trabalho de outros que estão presentes não é uma tarefa fácil para a maioria das pessoas.

Muitos não estão acostumados com o nível de exposição necessário e se sentem desconfortáveis com a possibilidade de haver algum tipo de enfrentamento. Essa dificuldade é ainda mais evidente em indivíduos habituados até então com o trabalho no estilo comando e controle.

Os membros do Time de Scrum poderão se sentir ainda mais intimidados caso haja algum cliente ou parte interessada importante presente na reunião, ainda mais se estiver direta ou indiretamente relacionado a causas de problemas ocorridos durante o Sprint. Nesses casos, os membros do Time de Scrum não se sentem em um ambiente seguro para falar de problemas e, assim, não será surpreendente se até mesmo problemas bastante visíveis não forem mencionados na reunião. Por essa

razão, somente estão presentes na reunião os membros do próprio Time de Scrum.

Os Desenvolvedores, no entanto, devem se sentir confortáveis para discutir problemas com o Product Owner. Caso não haja um ambiente de confiança que permita essa transparência, uma disfunção está configurada e cabe ao Scrum Master, enquanto facilitador, trabalhar para ajudar os envolvidos a resolvê-la.

De forma geral, em uma retrospectiva saudável, os membros do Time de Scrum conseguem levantar os pontos a melhorar sem inibição e sem medo de sofrer qualquer retaliação, pois sabem que todos os presentes compreendem que identificar possíveis melhorias é positivo para toda a organização.

Conflitos da diversidade

A diversidade de personalidades presentes em um Time de Scrum e, portanto, em uma reunião de Sprint Retrospective, favorece a manifestação de uma variedade de conflitos durante a reunião. As pessoas naturalmente têm diferentes opiniões, pontos de vista e interesses e, quanto mais heterogêneo for o grupo, mais essas diferenças se acentuam.

Um ambiente hostil ou simplesmente não amigável pode levar os Desenvolvedores ou Product Owner a adotarem posturas defensivas e a se fecharem, deixando consciente ou inconscientemente de trazer à tona pontos que ameacem a sua imagem ou a de colegas mais próximos perante o resto do Time de Scrum.

O Scrum Master atua dentro e fora das retrospectivas, buscando dinâmicas que aumentem o entrosamento dos

membros do Time de Scrum e que os estimulem a construir objetivos comuns, levando-os a funcionar como um verdadeiro time.

Pensamento grupal

Uma armadilha relativamente comum em grupos muito coesos é a tendência de seus membros criarem pressões intensas sobre si mesmos para obter e manter o consenso ou a conformidade. Essas pressões internas anulam a motivação do grupo em buscar, por meio do pensamento crítico, caminhos alternativos de ação. Assim, o grupo acaba por tomar decisões disfuncionais.

Esse processo nocivo é chamado de pensamento grupal ou *groupthink* (JANIS, 1982). O Time de Scrum que incorre em pensamento grupal pode apresentar alguns dos seguintes sintomas:

- acredita ser invulnerável a falhas, apresentando um otimismo excessivo e assumindo riscos desnecessários;
- trata membros que questionam ou discordam da opinião da maioria como desleais e exerce pressões para que se adequem às opiniões do resto do time;
- acredita possuir uma moralidade inerente, independente das ações de seus membros, o que os leva a ignorar consequências morais e éticas de suas decisões;
- cria estereótipos de pessoas externas que podem representar uma ameaça a seu trabalho, caracterizando-as como más demais para que seja possível negociar com elas ou tão fracas e tolas que não representam uma ameaça a ser considerada;
- acredita existir uma unanimidade em torno da visão da maioria de seus membros, de forma que o silêncio

de uma minoria diante de discussões ou argumentações é tratado como concordância;
- induz seus membros a se autocensurarem por temerem a desaprovação dos outros membros ou por minimizarem a relevância de suas questões diante do aparente consenso do time;
- induz o surgimento de membros que protegem o time de informações externas que possam ameaçar os valores já estabelecidos;
- cria explicações coerentes e aceitáveis de forma a ser capaz de ignorar qualquer feedback negativo que, caso fosse levado em consideração, levaria seus membros a reconsiderarem suas concepções ou suposições.

Times autogerenciados possuem uma tendência natural a serem muito coesos e, assim, são particularmente vulneráveis ao pensamento grupal (MANZ; NECK, 1997). A reunião de Sprint Retrospective é, provavelmente, o momento em que esses fatores podem se tornar mais visíveis. Por essa razão, é quando o Scrum Master pode atuar com maior intensidade para estimular o time a detectar, criar transparência e resolver o problema.

Capítulo 24
Release Planning (adicional)

Conteúdo

1. O que é a Release Planning?
2. Como é a Release Planning?
    - Estratégia de entregas: quando.
        - Entrega por valor.
        - Entrega por Sprint.
        - Entrega por item ou contínua.
        - Entrega por plano.
    - Estratégia de entregas: quem a recebe.
        - Entrega para os usuários finais.
        - Entrega para usuários intermediários.
        - Entrega para usuários finais selecionados.
    - Conclusão.

Resumo

- **Objetivo:** definir ou revisitar a estratégia de entregas ou planejar a próxima entrega de um ou mais Incrementos do produto implementados para seu uso e para obtenção de feedback.
- **Saídas esperadas:** Estratégia de entregas ou plano da entrega, formal ou informal.

24.1 O que é a Release Planning?

Entregar software em funcionamento com frequência, desde a cada poucas semanas até a cada poucos meses, com uma preferência por prazos mais curtos — Princípio Ágil.

Podemos chamar de Release a entrega de um ou mais Incrementos do produto prontos, de acordo com a Definição de Pronto, implementados pelos Desenvolvedores do produto em um ou mais Sprints, para que sejam utilizados e para que obtenham feedback. Em conjunto e somados ao que já foi entregue anteriormente, esses Incrementos formam um produto que possui valor suficiente para ser utilizado por seus usuários.

A atividade de Release Planning é o ato, formal ou informal, de planejar como, quando e para quem a próxima entrega de uma fatia utilizável do produto será realizada.

Antigamente, o guia oficial do Scrum determinava a realização de uma reunião obrigatória de Release Planning. Ele definia que *o propósito de uma Release Planning é estabelecer um plano e objetivos que o Time de Scrum e o resto da organização possam entender e comunicar. (...) O plano da entrega estabelece o objetivo da entrega, o Product Backlog de maior prioridade, os maiores riscos e as funcionalidades que a entrega irá conter. A organização pode então inspecionar o progresso e realizar mudanças a esse plano de entrega em cada Sprint* (SCHWABER, 2009). A reunião, no entanto, foi retirada do guia já em 2011 (SCHWABER; SUTHERLAND, 2011) e não consta mais da definição oficial do Scrum.

Conforme o contexto, cada Time de Scrum tem sua estratégia para a realização de suas entregas. De acordo com essa estratégia, a atividade de Release Planning pode ser realizada em diferentes momentos, de diferentes formas.

A realização de entregas ao longo do trabalho de desenvolvimento do produto tem três objetivos principais:

- **gerar valor para clientes e usuários do produto** — cada vez que realizamos uma entrega para usuários finais do produto, visamos gerar valor para eles e, consequentemente, para o negócio. A estratégia adotada, no entanto, pode determinar a realização de entregas para usuários intermediários, o que em geral não constitui geração direta de valor de negócio;
- **obter feedback** — uma vez realizada uma entrega, o Product Owner e os Desenvolvedores buscam impressões, opiniões e métricas de uso sobre o que foi recebido e utilizado junto a clientes e usuários do produto. A partir desse feedback, são realizadas mudanças no Product Backlog e, dessa forma, o produto é construído incrementalmente;
- **demonstrar progresso concreto a clientes e a demais partes interessadas** — cada vez que os usuários recebem uma entrega, os clientes e outras partes interessadas que acompanham o desenvolvimento do produto têm visibilidade real do progresso atual na realização do Objetivo do Produto, isso é, a partir do que já está pronto, funcionando e gerando valor. O progresso decorrente de uma entrega é muito mais concreto e relevante do que o percebido a partir de reuniões de Sprint Review, já que se trata de uma entrega para uso e que gera valor.

As entregas, em si, são idealmente realizadas pelos próprios Desenvolvedores, que fazem em conjunto o trabalho de ponta a ponta.

24.2 Como é a Release Planning?

A estratégia de entregas é de responsabilidade do Product Owner, mas sempre deve estar alinhada ao contexto da organização. O Product Owner define essa estratégia e, sempre que necessário, a modifica e a comunica ao Time de Scrum e a outras partes interessadas.

Como regra geral, eu considero duas dimensões essenciais em uma estratégia de entregas: quando ou com que frequência serão realizadas, e quem vai recebê-las.

Estratégia de entregas: quando

Com relação à dimensão "quando ou com que frequência deverão ser realizadas", para facilidade na compreensão, vou classificar as entregas em: entrega por valor, entrega por Sprint, entrega por item ou contínua e entrega por plano.

Entrega por valor

Dependendo da natureza do negócio, a entrega pode ser realizada sempre que o Product Owner julgar que os Incrementos do produto já implementados pelos Desenvolvedores nos Sprints já representam valor de negócio ou são passíveis de feedback o suficiente para que valha a pena entregá-los para que usuários os utilizem. Dessa forma, os Desenvolvedores seguem implementando Incrementos do produto, Sprint a Sprint, até que o Product Owner decida que já tem o suficiente para realizar uma entrega.

Entrega por Sprint

Podemos realizar uma entrega ao final de cada Sprint, aproveitando a própria cadência do Scrum. Nesse cenário, a reunião de Sprint Planning já fornece o suficiente para que o Product Owner comunique a clientes e demais partes interessadas o que será recebido e quando, em geral por meio do Objetivo do Sprint (veja o capítulo *Compromisso: Objetivo do Sprint*).

Assim, os Desenvolvedores trabalham para que o resultado final do Sprint seja imediatamente entregue. Realizar a entrega em cada Sprint é extremamente ágil, pois antecipa a geração de valor e o feedback dos usuários sobre o que foi produzido, maximizando as oportunidades de melhorias no produto.

Esse cenário não invalida a necessidade da realização da Sprint Review, que visa garantir a obtenção de feedback de convidados-chave da reunião sobre o Incremento ou Incrementos produzidos no Sprint.

Entrega por item ou contínua

Em um cenário ainda mais ágil, cada item implementado pelos Desenvolvedores é entregue assim que todas as suas tarefas são terminadas, durante o próprio Sprint, no que podemos caracterizar como um processo de entrega contínua. Entregar o item, portanto, é parte da Definição de Pronto usada.

Entrega por plano

Para cada entrega, o Time de Scrum pode criar um plano de alto nível do que será implementado. Nesse caso, é comum também que realizem uma reunião de Release Planning para cada entrega, em geral durante o Sprint anterior ao início do seu trabalho correspondente. Nessa reunião, eles estabelecem o plano de entrega, que geralmente contém:

- o Objetivo da Entrega, que é um objetivo de negócios a ser realizado por meio da entrega, e representa um incremento à realização do Objetivo do Produto (veja *Objetivo de Entrega*, no capítulo *Compromisso: Objetivo do Produto*);
- a data em que a entrega será realizada, que pode ser exata, aproximada ou uma faixa de datas provável;
- um conjunto de itens selecionados do Product Backlog, que representam uma ideia inicial do que estará presente na entrega. Visando a realizar o Objetivo da Entrega, esses itens serão implementados, até a data da entrega, desde os mais importantes com relação a esse objetivo em direção aos menos importantes, que vão sendo refinados e modificados conforme necessário.

Para chegarmos a uma previsão do conjunto de itens a fazer parte dessa entrega, podemos calcular o número de Sprints até a data estabelecida para a entrega (também chamado de "tamanho da entrega") e utilizá-lo juntamente com a estimativa dos itens do Product Backlog e a média da soma das estimativas dos itens prontos nos últimos Sprints, também chamada de Velocidade (veja, no capítulo *Story Points: estimando o trabalho*, a seção *Velocidade*). Os itens são então escolhidos, a partir do alto do Product Backlog, de forma que a soma das suas estimativas seja aproximadamente igual à Velocidade multiplicada pelo tamanho da entrega em Sprints.

Repare que o escopo da entrega não é fixo, o que significa que novos itens vão surgir, outros vão desaparecer, ou serão reordenados, vão evoluir, vão ganhar mais detalhes e vão ser fatiados em itens menores durante os Sprints até a entrega. Repare

também que esses itens não são separados do Product Backlog, de forma que não existe um backlog de entrega. Podemos talvez fazer algum tipo de marcação, como uma etiqueta, nos itens que acreditamos que pertencerão à entrega planejada.

Não há duração oficial estabelecida para a reunião de Release Planning. Recomendo, no entanto, que o Time de Scrum estabeleça um tempo máximo para ela (*timebox*) e que não ultrapasse um dia de trabalho.

Destaco também que a realização da reunião de Release Planning de forma alguma substitui as reuniões de Sprint Planning, que serão realizadas em seus Sprints correspondentes. O conjunto de itens que entrará em cada Sprint Backlog será, portanto, definido apenas na sua Sprint respectiva, e não na reunião de Release Planning.

O progresso em direção à data da entrega e, portanto, à realização do Objetivo da Entrega, é inspecionado em cada Sprint. As ferramentas mais utilizadas com esse propósito são o Gráfico de Release Burndown e o Gráfico de Release Burnup (veja o capítulo *Burndown e Burnup: acompanhando o trabalho*).

Estratégia de entregas: quem a recebe

Idealmente, as entregas são realizadas para os usuários finais e com alta frequência, de forma a obtermos feedback sobre o uso dos Incrementos do produto e de forma a gerarmos valor de negócio para os clientes.

Quando esse cenário não é possível, a entrega pode ser feita, com frequências diferentes, para um ou mais grupos de usuários selecionados, como usuários intermediários ou subconjuntos dos usuários finais. Com

isso, visamos a garantir um nível de feedback suficiente para alimentar a definição do produto, antes que seja possível que ele chegue a seus usuários finais.

Entrega para os usuários finais

O Incremento ou Incrementos do produto são disponibilizados para os usuários finais do produto. Ao terem, em suas mãos, as funcionalidades que mais necessitam naquele momento, eles as utilizarão, gerando valor para clientes e possibilitando a obtenção de feedback.

Sempre que é possível de ser adotado, esse cenário representa o menor risco na definição do produto, pois permite obtermos feedback a partir do uso real do produto.

Entrega para usuários intermediários

Nem sempre é possível realizar, com uma alta frequência, entregas para os usuários finais do produto. Nesses casos, um grupo de pessoas pode ser escolhido para representarem seus usuários finais no uso do produto. Com frequência, esses usuários intermediários pertencem à própria organização que trabalha para desenvolver o produto.

Eles podem ser entendedores do negócio do produto, especialistas em usabilidade, bons conhecedores de seus usuários finais ou simplesmente pessoas capazes de utilizar o produto e opinar sobre o que creem que é necessário para atrair os usuários e satisfazer suas necessidades. Eles receberão essa entrega interna e serão responsáveis por prover feedback sobre o seu uso.

Esse cenário é comum nos produtos em que os usuários reais são anônimos e pode não haver sentido ou não ser

interessante disponibilizar o produto antes que ele atinja um certo número de funcionalidades. Produtos de prateleira são exemplos comuns.

Entrega para usuários finais selecionados

Em produtos com um grande número de usuários ou com esse potencial, um grupo menor e representativo desses usuários reais pode ser escolhido para receber entregas mais cedo que o resto dos usuários. Esse grupo utilizará cada conjunto de Incrementos do produto entregue, com a consciência de que se trata de uma entrega intermediária, e será responsável por prover feedback sobre ela.

Em muitos casos, o benefício e estímulo para que continuem usando o produto e provendo feedback é a possibilidade de experimentar versões do produto antes de todos. Em outros casos, é a possibilidade de utilizá-lo de graça ou a um preço especial, por exemplo. Outros usuários chegam até mesmo a pagar caro para fazer parte de programas desse tipo.

Os usuários selecionados com esse propósito são normalmente chamados de *beta-testers*, *beta-users*. Em outros casos, quando se trata de um uso controlado e monitorado, eles formam os chamados grupos de foco.

Esse também pode ser o caso em que parte de uma organização recebe Incrementos do produto antecipadamente e ajuda a construí-lo com seus feedbacks e dados de uso.

Conclusão

Em um cenário ideal, utópico para a maioria dos contextos, a melhor estratégia estabeleceria que as entregas fossem realizadas por item, continuamente, e

para o usuário final. Quando isso não é factível, o Product Owner busca definir uma estratégia de entregas de forma a se aproximar, o máximo possível, do uso frequente e real do produto. Ele faz isso conjugando diferentes "para quem" e "com que frequência" essas entregas serão realizadas, até que seja possível realizar uma entrega para o usuário final.

Durante os anos de 2012 e 2013, em meus treinamentos de Scrum no Canadá, eu recebi vários alunos de uma grande empresa que produzia jogos de computador de prateleira. Eles me contaram que a empresa utilizava, então, uma estratégia de entregas com Scrum que ilustra bem esse ponto:

- **entrega por Sprint para usuários intermediários** — a partir de um determinado Sprint, uma equipe interna da empresa recebia entregas internas ao final de cada Sprint e seu trabalho era jogar o jogo para prover feedback sobre a sua usabilidade, nível de realismo etc.;
- **entrega por valor para usuários finais selecionados** — com uma menor frequência e desde mais tarde no desenvolvimento do jogo, ele era entregue a usuários selecionados, os *beta-testers*, cujo papel era prover feedback sobre o jogo como um todo e permitir a coleta de métricas do uso real do produto;
- **entrega por valor para os usuários finais** — pouco após a distribuição de uma versão de demonstração, por fim, a entrega daquele ano era realizada para os usuários finais ao disponibilizarem o produto nas lojas.

Repare que o objetivo da estratégia de entregas dessa empresa de jogos era a de se aproximar ao máximo, dentro de suas limitações, do uso real do produto durante o seu desenvolvimento.

Capítulo 25
Refinamento do Product Backlog (adicional)

Conteúdo

1. O que é o Refinamento do Product Backlog?
2. Como é o Refinamento do Product Backlog?
    - Participantes.
    - Quando ocorre.
    - Definição de Preparado.

Resumo

- **Objetivo:** preparação de itens do alto do Product Backlog para a sua implementação.
- **Quando:** durante o Sprint, ou como um trabalho contínuo, ou como atividade eventual, ou realizado em sessões agendadas.
- **Duração:** não há duração estabelecida, mas em geral os Desenvolvedores não utilizam para atividades de refinamento mais do que 10% do seu esforço em cada Sprint.
- **Participantes obrigatórios:** Desenvolvedores, juntamente com o Product Owner e/ou com outras partes interessadas.
- **Saídas esperadas:** o Product Backlog ordenado, planejável, emergente e gradualmente detalhado. O Time de Scrum espera obter uma quantidade suficiente de itens no alto do Product Backlog preparados para serem implementados.

25.1 O que é o Refinamento do Product Backlog?

Devido à sua natureza emergente, o Product Backlog é frequentemente atualizado pelo Product Owner, que adiciona, remove e fatia seus itens. Mas são os Desenvolvedores do produto, os "consumidores" desses itens, que vão utilizá-los para realizar o seu trabalho nos Sprints, transformando-os em Incrementos do produto prontos.

Com esse fim, ao longo de todo o desenvolvimento do produto, os itens do alto do Product Backlog serão preparados para a sua implementação em um momento próximo a que isso aconteça. As atividades de preparação desses itens são chamadas de Refinamento do Product Backlog. Esse é um trabalho realizado pelos Desenvolvedores, geralmente em conjunto com o Product Owner. Com frequência, no entanto, essa interação inclui outras partes interessadas e pode até mesmo excluir o Product Owner por completo.

O Refinamento, também conhecido como *Backlog Grooming*, visa a garantir que o Product Backlog seja:

- **ordenado**, para maximizar o valor gerado para os clientes do produto;
- **planejável**, de forma que os Desenvolvedores e o Product Owner sejam capazes de planejar gradualmente o trabalho de desenvolvimento do produto;
- **emergente**, refletindo o dinamismo do ambiente de mudanças e de incerteza no qual o trabalho de desenvolvimento do produto está imerso;
- **gradualmente detalhado**, de forma que seus itens possuam um detalhamento adequado e que, assim,

um número suficiente de itens esteja preparado para o trabalho em seguida no Sprint.

As atividades de Refinamento do Product Backlog podem incluir o detalhamento de itens, seu reordenamento, o desmembramento de itens maiores em itens menores e, possivelmente a adição e a remoção de itens no Product Backlog. Ao final das sessões de Refinamento para um Sprint, os itens de maior ordem do Product Backlog possuem os detalhes necessários para serem implementados. O Time de Scrum prepara, assim, um número de itens que julgue o suficiente para preencher o Sprint Backlog a ser trabalhado. Mas é importante lembrar que a escolha dos itens, de fato, apenas ocorrerá na reunião de Sprint Planning.

Frequentemente, a geração de estimativas é parte das sessões de Refinamento do Product Backlog, quando a prática é adotada. Esse trabalho de estimar é realizado unicamente pelos Desenvolvedores, que contarão normalmente com a presença do Product Owner ou de outras partes interessadas para esclarecer dúvidas que invariavelmente vão surgir.

Não é incomum a realização das atividades de Refinamento do Product Backlog se restringir à própria reunião de Sprint Planning. A realidade nos mostra, no entanto, que essa pode ser uma prática arriscada.

Discussão: Sprint Planning dos infernos?

O Product Owner inicia o Sprint Planning, apresentando cada um dos itens do alto do Product Backlog que ele acredita que devam entrar no Sprint. Os Desenvolvedores ainda não tiveram contato com esses itens e fazem perguntas de todo o tipo. O Product Owner,

com muita dificuldade, se esforça para respondê-las. Ele, no entanto, sequer teve tempo de preparar qualquer coisa.

Product Owner e Desenvolvedores colaboram para definir os detalhes e os Testes de Aceitação de cada item, um a um. É um trabalho minucioso que gera muitas ideias e discussões e, por essa razão, demanda um tempo considerável para esclarecimentos. Uma vez detalhado o item, os Desenvolvedores realizam a atividade de Planning Poker para estimá-lo. Nela, surgem novas dúvidas que, novamente, consomem um esforço considerável de todos para serem tratadas, e nem sempre de forma satisfatória.

Os Desenvolvedores logo identificam que um dos itens é muito grande e, então, colaboram com o Product Owner para fatiá-lo e escolher apenas a sua parte mais importante. Eles decidem que o resto deve voltar para o Product Backlog. Não demoram a identificar outro item a ser fatiado e novamente realizam esse trabalho.

Para outro dos itens, os Desenvolvedores entendem que, naquele momento, não têm informações suficientes e, portanto, seria muito arriscado implementá-lo no Sprint. O Product Owner procura ainda negociar, mas afinal chegam a um acordo e o item retorna ao Product Backlog. Essa discussão eleva o nível de estresse da reunião e o Scrum Master, enquanto facilitador, faz o seu melhor para que o conflito seja resolvido.

Após várias horas de reunião, tantos detalhes foram discutidos e tantas negociações ocorreram que a exaustão toma conta de todos. Os itens restantes são discutidos mais rapidamente e, devido ao cansaço, todos estão mais propensos a aceitá-los com uma compreensão menor dos detalhes. Um Objetivo para o

Sprint é rapidamente negociado e, em seguida, os Desenvolvedores partem para a quebra de alguns dos itens em tarefas, o que ocorre sem o cuidado e a reflexão adequados. Como consequência, o Sprint Backlog resultante não representa um bom plano para o Sprint.

Essa reunião representa o que eu chamo de "Sprint Planning dos infernos". O Sprint consequente é bastante conturbado. Surpresas no trabalho surgem com frequência e muitos esclarecimentos são necessários. Ao final, o Objetivo do Sprint dificilmente será satisfatoriamente realizado. O que trouxe tantos problemas para o trabalho nesse Sprint? Será que uma preparação prévia não os teria ajudado?

25.2 Como é o Refinamento do Product Backlog?

Participantes

O trabalho de Refinamento do Product Backlog é realizado para resolver uma necessidade dos Desenvolvedores: devem existir itens suficientes preparados para que eles possam realizar o seu trabalho no Sprint. Sua participação, portanto, é imprescindível.

O guia oficial do Scrum define a atividade de Refinamento do Product Backlog como um processo contínuo de dividir e definir itens do Product Backlog em itens menores e mais precisos, adicionando detalhes (SCHWABER; SUTHERLAND, 2020). Embora edições anteriores do guia definissem que esse trabalho é realizado pelo Product Owner em conjunto com os Desenvolvedores (SCHWABER; SUTHERLAND, 2017), a

edição atual deixa a participação em aberto, abrindo espaço para que outras partes interessadas possam ser convidadas se o Time de Scrum acreditar que vale a pena. Esses terceiros são internos ou externos à organização, e podem aportar valor trazendo conhecimentos e opiniões relevantes para o detalhamento dos itens e para as decisões necessárias.

Times de Scrum mais maduros podem até mesmo realizar sessões de Refinamento do Product Backlog com pouca ou até mesmo nenhuma participação do Product Owner, com a interação direta entre os Desenvolvedores e partes interessadas. Essa prática somente é possível quando o Product Owner delega parte significativa de seu trabalho de definição do produto aos Desenvolvedores, trazendo a eles uma maior responsabilidade.

Alguns Times de Scrum preferem realizar as sessões de Refinamento do Product Backlog com a presença de apenas alguns de seus membros. Em seguida, eles transmitem os resultados para o restante dos Desenvolvedores que, durante a sessão, seguiram ocupados com o trabalho planejado para o Sprint corrente. Entretanto, eu particularmente recomendo a participação de todos, o que normalmente leva a um melhor compartilhamento das informações e a uma sensação maior de propriedade sobre os resultados do trabalho.

É recomendável o Scrum Master participar das sessões de Refinamento, atuando ali como um facilitador.

Quando ocorre

Cada Time de Scrum possui seu próprio processo para a colaboração necessária na realização das atividades de Refinamento do Product Backlog.

Para alguns, esse é um trabalho contínuo, que ocorre durante todo o Sprint. Para outros, é um trabalho eventual, apenas para quando o Product Owner traz novos itens a serem preparados para o Sprint seguinte.

Outros Times de Scrum preferem agendar as sessões de Refinamento em dias e horários específicos durante o Sprint, seja em sessões diárias, semanais ou agendadas apenas para os últimos dias do Sprint, dependendo da necessidade e da disponibilidade do Product Owner ou de outras partes interessadas. A realização de mais de uma sessão pode trazer tempo suficiente para obtenção de informações e para reflexões necessárias.

No caso em que itens são preparados cedo demais, há o risco de que, ao chegarem ao Sprint de seu desenvolvimento, seus detalhes estejam desatualizados ou que as prioridades tenham mudado. Por essa razão, eu normalmente recomendo que as sessões de Refinamento aconteçam mais próximas do final do Sprint anterior à sua implementação.

Há também times que realizam o Refinamento apenas durante a reunião de Sprint Planning do próprio Sprint em que os itens serão implementados, e por essa razão podem enfrentar alguns dos problemas descritos anteriormente neste capítulo, na sessão *Discussão: Sprint Planning dos infernos?*.

Seja qual for o formato escolhido, essa colaboração não deve ocupar muito tempo do trabalho do time. Usualmente, recomendamos não usar mais do que 10% do esforço dos Desenvolvedores no Sprint com essas atividades.

Definição de Preparado

A Definição de Preparado é um acordo entre Product Owner e Desenvolvedores que estabelece critérios para o estado de um item do Product Backlog para que seja considerado preparado para entrar no Sprint Backlog e, portanto, para poder ser implementado (veja o capítulo *Compromisso: Definição de Preparado (adicional)*).

Com a introdução dessa prática, as sessões de Refinamento do Product Backlog passam a utilizar a Definição de Preparado como referência e critério para esse trabalho de preparação.

Dessa forma, o principal resultado esperado das sessões de Refinamento do Product Backlog passa a ser um número suficiente de itens preparados para serem implementados, de acordo com a Definição de Preparado.

A Definição de Preparado não é parte oficial do Scrum.

Técnicas complementares

Nesta parte do livro, ofereço algumas técnicas e práticas muito utilizadas por Times de Scrum, detalhadas e enriquecidas com exemplos, que podem ajudá-los a obterem sucesso com o uso do framework. São elas:

- Contratos ágeis, com ideias e modelos que podem nos ajudar a oferecer o trabalho de desenvolvimento iterativo e incremental de produtos com Scrum;
- User Stories, provindas da metodologia Extreme Programming, que nos permitem representar o trabalho a ser realizado pela perspectiva do usuário;
- Story Points, que podem nos ajudar a estimar o trabalho com melhor precisão;
- Gráficos de Burndown e Burnup, que trazem transparência, permitindo o acompanhamento do cumprimento do trabalho planejado.

Capítulo 26
Contratos ágeis: pactuando sobre as entregas

Conteúdo

1. O que entendemos por contratos?
2. Contratos tradicionais.
    - Contrato tradicional: custo fixo com escopo fixo e detalhado.
    - Contrato tradicional: tempo e material com escopo fixo e detalhado.

3. Princípios básicos para contratos com Scrum.
    - Premissas básicas de contratos com Scrum.
    - O Triângulo Ágil.
    - Escopo não detalhado.
    - Ponto de parada.
4. Formatos de contratos com Scrum.
    - Contrato intermediário: custo fixo, prazo fixo, escopo detalhado mas flexível.
    - Contrato ágil: custo fixo, prazo fixo, escopo não detalhado.
    - Contrato ágil: custo fixo, escopo não detalhado, prazo fixo com opção de parada.
    - Contrato ágil: custo fixo, escopo não detalhado, prazo fixo com opção de parada taxada.
    - Contrato ágil: custo fixo, escopo não detalhado, prazo fixo com opção de troca de contexto.
    - Contrato ágil: incremental com pontos de verificação.

26.1 O que entendemos por contratos?

O objetivo deste capítulo é apresentar ideias e conceitos que podem nos ajudar a entender como oferecer o desenvolvimento de produtos com Scrum, e dessa forma embasar os contratos que se façam necessários.

Ao tratar de contratos, refiro-me a quaisquer acordos ou compromissos de entrega, sejam formais ou informais, entre fornecedor e cliente, seja interno ou externo. Para um cliente externo, ou seja, outra empresa que nos contrata como fornecedores do produto que será desenvolvido, geralmente criamos propostas e contratos formais, como aqueles que exigem assinaturas. Clientes internos podem ser, por exemplo, diferentes áreas de

nossa organização que nos solicitam, enquanto fornecedores internos, o desenvolvimento de algum produto para a própria organização ou para oferecerem a terceiros. Os contratos com clientes internos tendem a ser mais informais.

Não pretendo, assim, entrar nos detalhes do que deve estar presente em contratos formais em papel e assinados, nem descrever os passos para a sua elaboração ou entrar na linguagem jurídica que deve ser utilizada na sua elaboração.

26.2 Contratos tradicionais

O que vai ser entregue? Quanto vai custar? Quando será entregue? Essas são perguntas comuns que contratos tradicionais buscam responder, firmados entre fornecedores no desenvolvimento de um produto e clientes, sejam internos ou externos.

O "Triângulo de Ferro" (ou "Triângulo da Gestão de Projetos"), formado pela tríade de restrições "escopo", "custo" para o cliente (ou “orçamento”) e "prazo" (ou “cronograma”), é ainda hoje usado como referência para a formação de muitos desses contratos (veja a figura a seguir). Tendo esse modelo como referência, estabelecemos que cumpriremos o contrato e alcançaremos o sucesso se entregarmos o produto com o escopo conforme detalhado e acordado com o cliente, dentro do custo e prazo acertados.

Alguns autores ainda adicionam ao triângulo a "qualidade" como a quarta restrição. Isso significa para o contrato que o produto entregue deve estar dentro dos padrões de qualidade acordados.

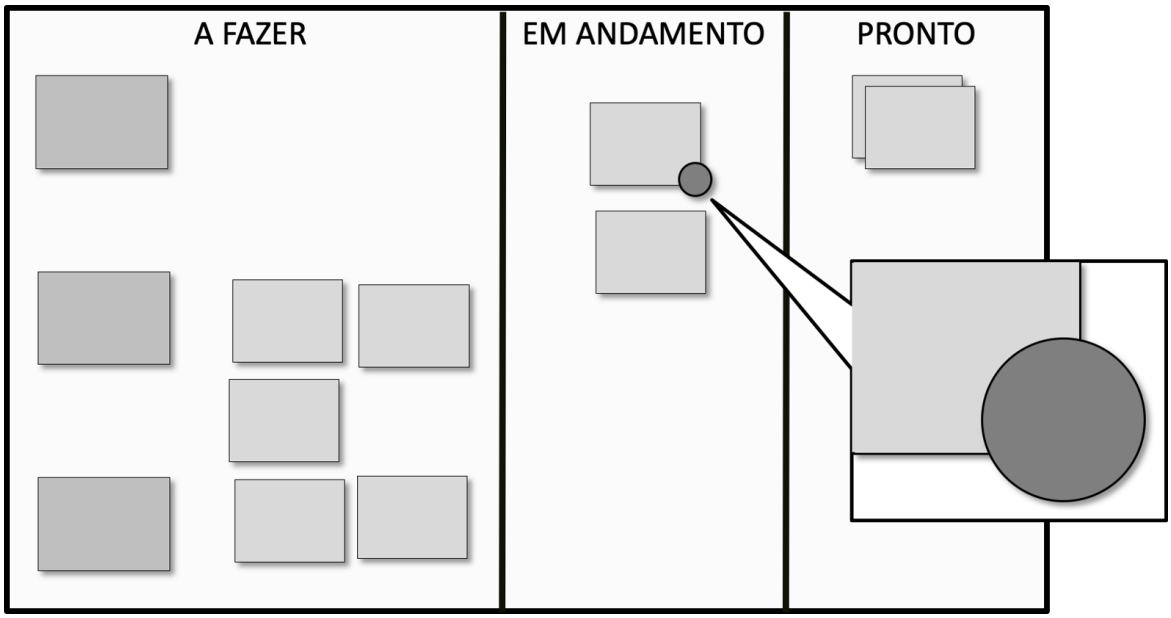


Figura 26.1: Triângulo da Gestão de Projetos

A necessidade de estabelecermos um contrato formal é comum quando há um cliente solicitando, a uma empresa fornecedora, o desenvolvimento de um novo produto ou de uma nova parte de um produto já existente.

No entanto, os conceitos que apresento neste capítulo também são úteis para quando tratamos de trabalho interno, nos quais o time (ou seu departamento) estabelece um contrato informal com clientes de dentro da própria organização. Esses conceitos também são úteis para quando tratamos do desenvolvimento de produtos para o mercado, quando há um contrato informal com pessoas ou departamentos internos que fazem o papel de clientes. Em qualquer caso, estamos tratando de uma relação entre cliente e fornecedor.

Contrato tradicional: custo fixo com escopo fixo e detalhado

A forma mais comum de contratação do desenvolvimento de um produto é por custo fixo. Na forma tradicional desse tipo de contrato, as três restrições são fixadas. O escopo do trabalho é esmiuçado em detalhes. Estimativas são realizadas sobre esses detalhes para, em seguida, estabelecermos o prazo do trabalho a ser realizado. A esse prazo, em geral, é ainda aplicado um fator de correção, também conhecido informalmente como "gordura".

A partir do escopo e prazo, o custo é fixado. Ou seja, partimos do escopo para determinarmos o prazo e o custo para o cliente. Mudanças no escopo são indesejadas, devendo ser gerenciadas de perto e cobradas em adição ao custo.

Alternativamente, podemos também partir de um custo ou prazo máximo (em geral, estabelecidos pelo cliente) e ajustar as duas outras restrições de acordo. Por exemplo, o cliente exige uma data máxima para a entrega final e, então, detalhamos o escopo para, a partir daí, negociarmos sua redução, juntamente com o custo correspondente.

Esse contrato, de forma geral, reflete uma relação de alta desconfiança entre cliente e fornecedor. Ao oferecer exatamente o que será entregue, quando será entregue e quanto lhe custará, essa forma de contrato traz uma sensação maior de segurança para o cliente. Não é à toa que é de longe o modelo preferido entre os clientes que não conhecem os benefícios do Scrum.

Mas essa segurança é inteiramente falsa: o modelo, para o trabalho de desenvolvimento de produtos, simplesmente não funciona. Como mencionei em outro capítulo, o Standish Group divulgou em 2015 um estudo mostrando que apenas 11% dos projetos (no

desenvolvimento de software) que fixam escopo, custo e prazo terminam com o custo e no prazo previstos e com o cliente satisfeito (The Standish Group, 2015).

Dessa forma, ao detalhar e fixar o escopo, o modelo de "custo fixo com escopo fixo e detalhado" para a contratação do desenvolvimento de um produto garante apenas que há uma enorme chance do contrato ser violado, o que observamos ser o caso mais comum nesse mercado.

Contrato tradicional: tempo e material com escopo fixo e detalhado

Tempo e Material é outra forma conhecida de contratar o desenvolvimento de produtos. A forma tradicional desse contrato em geral prevê o detalhamento do escopo no início sem, no entanto, que fixemos o custo e o prazo final.

Basicamente, o cliente é cobrado por hora ou dia de trabalho das pessoas contratadas para o trabalho até que o escopo estabelecido seja esgotado, sem qualquer garantia de quando isso vai ocorrer. O valor cobrado por dia ou por hora é normalmente estabelecido de acordo com o papel exercido por essas pessoas e por seu nível de senioridade.

Esse modelo é mais usado quando as pessoas contratadas para o trabalho estão alocadas no cliente, ou seja, trabalhando em suas dependências durante o trabalho de desenvolvimento do produto. Assim, em geral esse contrato reflete uma desconfiança menor na relação cliente-fornecedor, já que o cliente acredita exercer um maior controle sobre o trabalho que está sendo realizado.

Mas, na prática, tanto o custo total do trabalho quanto a data de seu término ficam imprevisíveis ao utilizarmos esse modelo. O cliente até pode tentar se proteger, ao estabelecer um limite para o custo e para o prazo final a partir de estimativas. Mesmo assim, ao definirmos os detalhes do produto de antemão, grande parte dos riscos recai sobre o cliente, pois ele, ainda assim, pode não receber um produto funcional na data limite estabelecida.

Caso optemos por permitir que o cliente modifique o que foi previamente acordado para o produto conforme desejar, esse escopo pode facilmente fugir do controle e não convergir para entregas. O fornecedor, em princípio, não verá problemas, pois continuará sendo pago, enquanto que novamente os riscos de um trabalho desgovernado recairão sobre o cliente.

Discussão: podemos utilizar Pontos de Função?

A Análise por Pontos de Função (APF) é ainda muito utilizada para a definição de estimativas em contratos de desenvolvimento de produtos de software.

Para que se realize a APF de um sistema a ser desenvolvido, a contagem dos pontos de função é feita a partir de sua especificação e utiliza um conjunto definido de regras e técnicas. Essa contagem determinará o tamanho funcional do sistema, ou seja, a quantidade de funcionalidades ou elementos do novo sistema tais como percebidas por seus usuários, para que então se realize a estimativa do esforço e custo necessários para seu desenvolvimento.

O principal argumento a favor do uso da APF é que, ao se pagar apenas pelas funcionalidades previstas, não entram nessa conta trabalhos que não são de implementação propriamente dita, como a análise, os testes, a correção de erros etc. O cliente, teoricamente, também não paga por uma possível ineficiência no trabalho de implementação. Talvez por essa razão a técnica tem uma grande aceitação em contextos muito regulados, como em contratações realizadas pelo governo.

O problema fundamental da APF é que, além de a contagem possuir um nível significativo de subjetividade, ela exige um escopo detalhado antes do início do trabalho de desenvolvimento, o que é incompatível com o trabalho ágil.

Uma solução paliativa muitas vezes adotada é, uma vez detalhado um escopo para o sistema e realizada a contagem sobre ele, a construção do produto é realizada sobre um novo escopo emergente, mas restrita à contagem inicial, enquanto se faz a gestão da mudança necessária para viabilizar essa adaptabilidade.

26.3 Princípios básicos para contratos com Scrum

Premissas básicas de contratos com Scrum

No capítulo *O que é Scrum?*, descrevi o trabalho de desenvolvimento de um produto como complexo e com um alto grau de incerteza (veja *Scrum é embasado no empirismo*, no capítulo *O que é Scrum*). Ou seja, não é viável definirmos as necessidades de negócios com

grande nível de detalhes no início do trabalho e, portanto, devemos utilizar uma abordagem empírica, na qual trabalhamos de forma incremental e guiados por feedbacks frequentes.

Também idealizei que o trabalho de desenvolvimento de um produto pode ser entendido como um Sistema Adaptativo Complexo, cujo comportamento é emergente e não previsível ao longo prazo (veja *Scrum se aplica a problemas complexos*, no capítulo *O que é Scrum*).

Como vimos no capítulo *Por que Scrum?*, resultados de estudos apontam que, ao fixarmos o escopo no princípio do trabalho, uma grande parte das funcionalidades entregues nunca é utilizada (veja, no capítulo citado, *Desenvolver o produto com base em seu próprio uso*, em *Redução do desperdício*).

Portanto, uma das premissas básicas no desenvolvimento de um produto com Scrum é a de que não é possível nem verdadeiro trazermos um alto nível de detalhes ao escopo a ser implementado no médio e no longo prazo (veja *Definição: analogia do horizonte*, em *Redução do desperdício*, no capítulo *Por que Scrum?*). Dessa forma, não há como estabelecermos um contrato de desenvolvimento de um produto com um escopo detalhado, a menos que esses detalhes se refiram, no máximo, ao curto prazo. Entendemos que, ao tentarmos fazê-lo, um enorme desperdício será gerado, e o prazo e o custo estabelecidos muito provavelmente não serão cumpridos.

Essa é uma realidade inerente ao desenvolvimento de produtos, independentemente da metodologia utilizada para o trabalho. Dessa forma, podemos concluir que ambos os modelos tradicionais de custo fixo e tempo e material não são modelos adequados para esse trabalho.

O Triângulo Ágil

Uma vez colocadas as premissas básicas de contratos com Scrum, podemos entender por que o Triângulo de Ferro, que mostrei anteriormente, é inadequado como base para o estabelecimento de contratos para o desenvolvimento de um produto. Entendemos que não é possível, a partir do detalhamento de um todo a ser implementado, estabelecermos o prazo e o custo para o desenvolvimento do novo produto (ou da nova parte do produto). Esse todo não existe de antemão e somente pode ser detalhado em retrospecto, após pronto.

Em outras palavras, não há como calcularmos o prazo e custo do trabalho a partir de um escopo fixo predefinido, como no Triângulo de Ferro. Podemos, no entanto, trabalhar com um triângulo invertido para o estabelecimento do contrato. Nele, são fixados o prazo e custo, e os detalhes do escopo são definidos apenas ao longo do trabalho de desenvolvimento do produto (veja a figura a seguir).

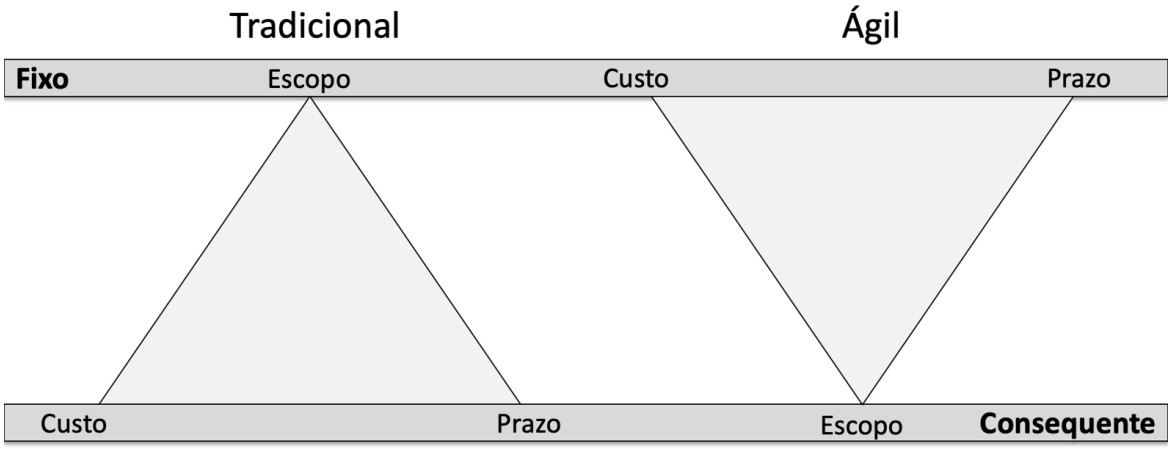


Figura 26.2: Triângulo de Ferro versus Triângulo Ágil

Repare que esse prazo e custo fixos podem ser curtos e renováveis. Nesse cenário, o produto é desenvolvido de

parte pronta em parte pronta, cada uma consumindo na totalidade o prazo e custo, que se renovam até que se decida parar.

Escopo não detalhado

Conforme afirmei anteriormente, contratos que fixam e detalham o escopo não são compatíveis com o trabalho com Scrum.

Escopo não detalhado, no entanto, não significa ausência de escopo. É importante esclarecermos no contrato, mesmo que em alto nível, qual o problema a ser resolvido (ou a necessidade a ser suprida, ou a oportunidade a ser aproveitada) com o uso do produto, que justifica a própria contratação do desenvolvimento do produto.

A definição clara desse problema é essencial para guiar o desenvolvimento do produto. No Scrum, ele é representado pelo Objetivo do Produto (veja o capítulo *Compromisso: Objetivo do Produto*). No contrato, esse Objetivo do Produto pode ser explicitamente desmembrado, caso se mostre necessário. Esse desmembramento é feito em objetivos um pouco mais granulares a serem distribuídos em um *roadmap* que mostra a expectativa sobre a evolução do desenvolvimento do produto, ainda em alto nível, no decorrer do trabalho.

Ponto de parada

Ao fixarmos o custo e o prazo e flexibilizarmos o escopo, o cliente não acostumado ao trabalho ágil pode se sentir inseguro sobre o que vai receber como resultado e, com isso, pode exigir o contrato com o escopo detalhado.

Já entendemos neste capítulo que esse tipo de contrato traz apenas uma ilusão de segurança e não deve ser uma opção na contratação do desenvolvimento de um produto. Isso porque aumenta em muito os riscos do cliente, embora ele não necessariamente perceba isso.

Na realidade, o contrato com Scrum deve trazer muito mais transparência e maior segurança para o cliente. Ele vê resultados concretos ao final de cada ciclo de desenvolvimento, na reunião de Sprint Review. A partir de seu feedback, o produto é ajustado e direções são definidas. Incrementos do produto funcionando são entregues com frequência e, na mão de seus usuários, geram valor para o cliente, permitindo um feedback ainda mais concreto.

Aliada ao feedback frequente, a ordenação do que será produzido orientada ao valor mitiga tanto o risco de necessidades importantes do cliente ficarem de fora do produto quanto o risco de trabalharmos apenas cumprindo tarefas, que representarão valor somente em uma data distante. A implementação ordenada garante que as necessidades do cliente sejam atendidas desde cedo, desde as mais importantes, com soluções de ponta a ponta.

Veja na figura a seguir o gráfico do incremento de valor em cada entrega versus o tempo. As marcações no eixo horizontal indicam as entregas dos Incrementos do produto para seus usuários, enquanto que o eixo vertical indica a quantidade de valor agregada ao produto com cada entrega. Repare que aqui não estou tratando do valor acumulado total do produto, mas sim do valor adicional de cada entrega.

Eu indico no gráfico, de forma qualitativa, o retorno esperado sobre o investimento. De forma simplificada,

podemos afirmar que o cálculo desse retorno opõe o valor de negócio obtido ao investimento realizado, sendo aqui esse último proporcional ao tempo de trabalho para uma entrega.

Podemos rodar uma ou algumas poucas iterações até chegarmos a uma primeira entrega, visando a realizar a parte mais importante do Objetivo do Produto. Essa entrega representa um alto valor de negócio e um consequente alto retorno sobre o investimento, o que pode ser observado na primeira marcação do gráfico.

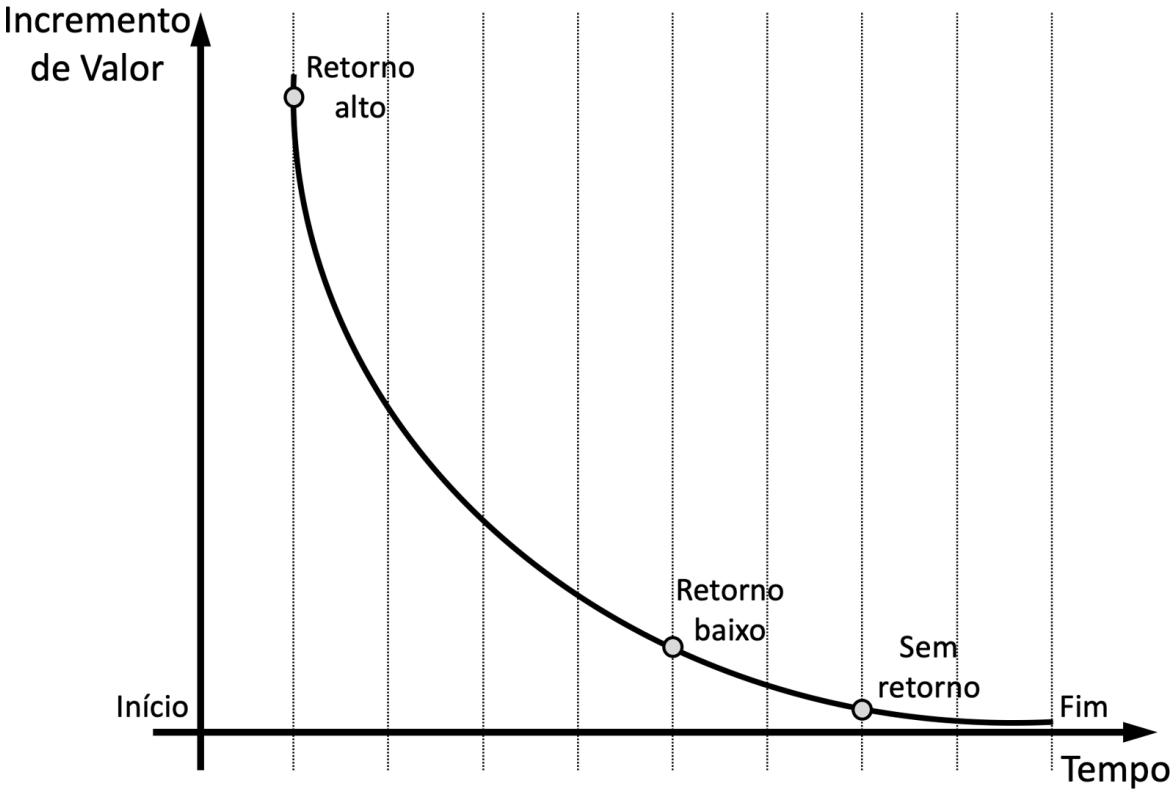


Figura 26.3: Incremento de valor versus Tempo

Em teoria, a entrega seguinte adiciona valor ao produto, mas um valor menor que o anterior, pois consiste de soluções para a próxima parte mais importante do Objetivo do Produto, que incluem tanto novas funcionalidades quanto melhorias para o que já foi

entregue. O trabalho segue de forma iterativa, com melhorias incrementais em como o Objetivo do Produto é realizado, mas adicionando cada vez menos valor ao que já foi entregue. Forma-se, idealmente, uma curva assintótica, cujos pontos se aproximam cada vez mais do eixo, mas sem nunca o tocar. Dependendo do contexto, essa curva pode ser mais ou menos acentuada que a da figura anterior.

Repare no gráfico como o retorno da entrega é altíssimo no princípio do trabalho, mas rapidamente decai entrega após entrega, até que, em algum ponto, podemos considerá-lo baixo o suficiente para que já não compense mais para o cliente seguir com o desenvolvimento do produto. A partir desse ponto, adicionar novas funcionalidades ao produto já não vale a pena, e o trabalho pode simplesmente ser interrompido.

Definição: analogia da iteração

Eu me lembro de quando aprendi na escola, nas aulas de matemática do professor Celso, a resolver equações do segundo grau. É simples. Aplicamos a fórmula de Bhaskara nos parâmetros da equação e pronto! Achamos as suas raízes (soluções). Mas o que acontece para a grande maioria de tipos de equações matemáticas é diferente. Para equações como, por exemplo, as do quarto grau e as logarítmicas, não existem fórmulas para encontrarmos suas soluções. Uma forma comum de resolvê-las é o método iterativo.

O processo para usar iterações para resolver uma equação matemática é simples, mas vou simplificá-lo ainda mais para o leitor. Começamos escolhendo um valor inicial. Ele é geralmente um palpite, que idealmente imaginamos ou sabemos ser próximo de uma

das soluções da equação, mas não o suficiente para ser considerado uma solução satisfatória.

Esse valor inicial é aplicado à equação, que o transforma em um valor um pouco mais próximo à solução da equação. Aplicamos então esse novo valor à própria equação, conseguindo assim um novo resultado ainda mais próximo à solução da equação. Repetiremos esse processo sucessivamente, utilizando a saída de uma iteração como a entrada da seguinte. Dessa forma, o resultado é progressivamente melhorado.

Minha pergunta é: quando paramos de iterar? Que critérios de parada podemos utilizar? Deixe-me por favor percorrer algumas opções.

Posso, talvez, estabelecer para a solução uma precisão desejada de cinco casas decimais. Ao verificar que uma nova iteração modificaria o resultado apenas a partir da sexta casa, considero que a solução é boa o suficiente. Alternativamente, posso rodar cinquenta iterações e me satisfazer com o resultado final, seja qual for. Ou talvez eu possa definir que irei iterar durante dez minutos e então ficar com o valor resultante. Ou talvez eu possa iterar, iterar, iterar... até eu achar que o resultado está bom o suficiente!

Todas essas soluções podem ser aceitáveis. O que estamos buscando aqui é iterar sobre uma solução até que ela possa, por algum critério, ser considerada boa o suficiente. Na prática, o ponto de parada é aquele em que entendemos que o retorno sobre o investimento de executarmos mais uma iteração não compensa.

Para o desenvolvimento de um produto, a ideia é a de que partimos, na primeira iteração, de uma solução simples (ou uma hipótese de solução) para a parte mais

importante do problema. Utilizamos o feedback obtido sobre essa solução para iterarmos sobre ela, melhorando, corrigindo e expandindo a solução. E seguimos iterando. A condição de parada poderia ser qualquer uma dessas que estabeleci para a equação nos parágrafos anteriores, como por exemplo um prazo fixo, um número predefinido de iterações ou até que o cliente se dê por satisfeito.

A premissa básica que essa analogia nos traz, importante para contratos de desenvolvimento de um produto, é que, ao criarmos e entregarmos o produto a partir dos problemas mais importantes, não tem mais sentido seguir esse trabalho a partir do momento em que o valor adicional que será obtido já não compensa o investimento de fazê-lo.

26.4 Formatos de contratos com Scrum

Nesta seção, busco listar os formatos de contrato mais comuns para o desenvolvimento de um produto com Scrum. Todos partem das mesmas premissas listadas em seções anteriores: custo e prazo fixos, escopo flexível.

Contrato intermediário: custo fixo, prazo fixo, escopo detalhado mas flexível

Para a elaboração desse contrato, fixamos um prazo e um custo para o trabalho e o escopo é detalhado. Desse escopo inicial, partes podem ser negociadas e trocadas ao longo do trabalho de desenvolvimento do produto.

Essa é uma solução paliativa e, embora possa se apresentar como o único acordo factível em um

ambiente em transição, ainda reflete um nível relevante de desconfiança entre cliente e fornecedor e é distante do que buscamos para o trabalho com Scrum. Esse formato de contrato gera um grande custo transacional, já que as mudanças serão frequentes e cada troca terá que ser negociada. Existe também o risco do apego ao escopo previamente detalhado, o que pode impor barreiras a que mudanças necessárias no escopo aconteçam.

O conhecido, embora antigo, modelo de contrato *Money for nothing and your change for free*, de Jeff Sutherland, um dos criadores do Scrum (SUTHERLAND, 2008), se enquadra nesse formato. O modelo do Jeff estabelece, em adição, uma opção de parada taxada, em que o cliente pode interromper o contrato de forma antecipada, em qualquer ponto ao longo de sua execução, mas ele pagará ao fornecedor 20% do orçamento que restou a ser cumprido.

Contrato ágil: custo fixo, prazo fixo, escopo não detalhado

Fixamos um prazo e um custo para o trabalho, mas o escopo não é detalhado em seu princípio e emerge em seu decorrer. Em um contrato desse tipo, o escopo pode ser definido em alto nível a partir do Objetivo do Produto e, talvez, de um *roadmap*.

O prazo e o custo podem ser fixados a partir de:

- o valor que o cliente pode ou está disposto a pagar (portanto, o custo ou orçamento), que compra um certo tempo de trabalho do fornecedor;
- uma data final imposta pelo cliente, definida a partir de uma necessidade específica do negócio;

- o tempo que o fornecedor, a partir de seu conhecimento e experiência, acredita ser suficiente para iterar sobre o problema até resolvê-lo satisfatoriamente.

O cliente, por exemplo, pode necessitar do produto terminado até certa data antes do início de um evento relacionado. Ou, em outro exemplo, o cliente possui reservado um determinado orçamento para o trabalho, e o fornecedor calcula que esse valor é suficiente para um certo número de meses de trabalho de uma de suas equipes. Ou o fornecedor simplesmente define, a partir de sua experiência, que em um certo número de meses será capaz de resolver o problema satisfatoriamente.

Nos dois últimos casos, o custo para o cliente pode ser calculado a partir do prazo que foi fixado e do custo de um time para o trabalho. Podemos também definir o custo a partir do valor que esperamos gerar para o cliente. Ou talvez como um percentual do valor futuro de fato gerado pelo produto, à medida em que isso acontece. Entretanto, esse é um cálculo difícil de ser realizado e de ser aceito pelos clientes.

Esse formato de contrato traz a segurança para o cliente de que não gastará mais (nem menos) do que o custo estabelecido. Como em todo e qualquer trabalho ágil, o cliente também tem uma maior segurança de que o produto desenvolvido irá trazer o valor esperado, já que receberá com frequência entregas para seus usuários e o trabalho de desenvolvimento do produto será orientado ao feedback provindo desse uso.

Contrato ágil: custo fixo, escopo não detalhado, prazo fixo com opção de parada

Esse formato de contrato carrega as mesmas características que o anterior, *custo fixo, prazo fixo, escopo não detalhado*. Nesse formato, no entanto, o cliente pode escolher interromper o trabalho em qualquer momento que desejar.

De forma geral, o cliente escolherá cessar o trabalho de desenvolvimento do produto se:

- ele estiver insatisfeito com o trabalho realizado até o momento e não quiser gastar mais com um trabalho no qual não mais acredita;
- ele estiver suficientemente satisfeito com o produto já entregue até o momento e acreditar que seguir investindo em entregas futuras não traria um retorno que valeria a pena seguir adiante.

Enquanto a primeira razão demonstra que esse tipo de contrato traz uma maior segurança para o cliente, a segunda é tratada como um sucesso alcançado. O gráfico de valor incremental *versus* tempo da figura anterior nos mostra que, caso a ordenação tenha sido bem-feita, com um envolvimento do cliente e das demais partes interessadas e com as entregas frequentes, existe uma grande possibilidade de que valha a pena parar o desenvolvimento do produto antes do prazo final estabelecido. Dessa forma, naturalmente, o cliente poderia escolher interrompê-lo, satisfeito com o que já está em suas mãos.

Esse formato traz uma maior segurança para o cliente ao impor um limite para o seu gasto. Da mesma forma que no modelo anterior, o cliente também tem uma maior segurança de que o produto desenvolvido irá trazer o valor esperado.

Contrato ágil: custo fixo, escopo não detalhado, prazo fixo com opção de parada taxada

Esse formato de contrato adiciona ao anterior, "custo fixo, escopo não detalhado, prazo fixo com opção de parada", uma taxa a ser paga pelo cliente caso deseje interromper o trabalho antes do prazo final acordado. Essa taxa pode corresponder a um valor fixo, mas é geralmente um percentual do valor restante do contrato. Pode também ser um valor calculado como o suficiente para cobrir o custo do time pelo tempo restante em que trabalharia no desenvolvimento do produto. Dessa forma, esse formato traz também uma maior segurança para o fornecedor.

Dessa forma, estabelecemos uma relação ganha-ganha. Por um lado, o cliente economiza com relação ao total que pretendia gastar no produto e ainda vê a necessidade de seu envolvimento cessando mais cedo. Por outro lado, a organização que desenvolve o produto recebe algo por um trabalho que não vai realizar e já pode alocar o time em outro trabalho.

Esse formato de contrato é uma evolução do modelo *Money for nothing and your change for free*, de Jeff Sutherland, que citei anteriormente (SUTHERLAND, 2008). A grande diferença é que o modelo do Jeff estabelece, em adição à opção de parada taxada, um escopo fixo inicial do qual partes podem ser negociadas e trocadas.

Contrato ágil: custo fixo, escopo não detalhado, prazo fixo com opção de troca de contexto

Adicionamos ao formato "custo fixo, escopo não detalhado, prazo fixo com opção de parada" a possibilidade de usar o tempo e orçamento restantes

após a decisão de interrupção do desenvolvimento do produto para outros fins, mas para o mesmo cliente. Podemos, por exemplo, utilizar o tempo restante para realizar uma pequena implementação ou iniciar o desenvolvimento de um outro produto a ser continuado em um novo contrato.

Contrato ágil: incremental com pontos de verificação

Desenvolvemos o produto buscando entregar partes prontas do produto o mais frequentemente possível, a partir das que acreditamos que gerarão maior valor. Para esse formato de contrato, estabelecemos pontos frequentes de verificação em que o cliente decide se prosseguimos com o trabalho ou não. Ele pode tomar essa decisão ao final de cada Sprint, por exemplo, ou após cada entrega, que pode ser realizada após alguns Sprints de trabalho.

A diferença fundamental do modelo "incremental com pontos de verificação" para o "custo fixo, escopo não detalhado, prazo fixo com opção de parada" é que, no primeiro, não estabelecemos um custo nem prazo final para o trabalho. No modelo incremental, o produto é desenvolvido com Scrum em incrementos e, após verificar ou receber um ou mais incrementos, o cliente decide se vale a pena continuar com o trabalho de desenvolvimento do produto.

Esse modelo mantém o prazo fixo, porém curto, estabelecido como a duração de um Sprint ou de uma entrega, e o custo é em geral fixado a partir desse prazo.

Repare que, com esse modelo, o trabalho de desenvolvimento pode seguir indefinidamente. Dessa forma, ele pode ser utilizado, por exemplo, para produtos internos à organização que devem ser mantidos e atualizados enquanto a organização existir.

Capítulo 27
User Stories: representando o trabalho

Conteúdo

1. O que é a User Story?
2. Como é a User Story?
    - Cartão.
    - Conversas.
    - Confirmação.

27.1 O que é a User Story?

User Story, ou “história de usuário”, é uma descrição concisa e simples de um comportamento ou característica do produto, sob o ponto de vista de um usuário desse produto para o qual o que está descrito tem valor. A partir da implementação da User Story, a característica ou comportamento será adicionada ou modificada no produto.

> *...a ideia de indicar trabalho a ser realizado em frases curtas (histórias de usuários) em um cartão (cartões de histórias). É importante notar que as frases curtas nos cartões não são “os requisitos”; elas são indicadores que prometem uma conversa sobre o que está associado com a frase no cartão. Neste sentido, elas são indicadores e lembretes eficientes... —* Alistair Cockburn, 2007.

Entre cada descrição e os resultados das conversas realizadas sobre elas, as User Stories visam a substituir o que chamamos, em projetos tradicionais, de “requisitos” (representados, por exemplo, por Casos de Uso), com o intuito de que o produto seja integralmente especificado por meio delas (JEFFRIES et al., 2000). Elas foram sugeridas pelos criadores do Extreme Programming, uma Metodologia Ágil para o desenvolvimento de software.

Repare que User Stories têm como referência um usuário do produto, ou seja, alguém que de fato o utiliza, e não de um cliente, do Product Owner, da empresa ou de alguma outra parte interessada.

Com o uso de Scrum, os comportamentos ou características do produto a serem criados são descritos por meio de itens do Product Backlog. Logo, ao optarmos pelo uso de User Stories, cada um dos itens do Product Backlog é representado por uma User Story, ou seja, a partir da perspectiva de um usuário do produto.

As User Stories são, a princípio, escritas pelo Product Owner e refinadas, para sua implementação, em conjunto com os Desenvolvedores do produto e, possivelmente, com outras partes interessadas. No entanto, as User Stories não fazem parte do framework Scrum e, assim, seu uso é opcional.

Veja um exemplo de User Story na figura a seguir.

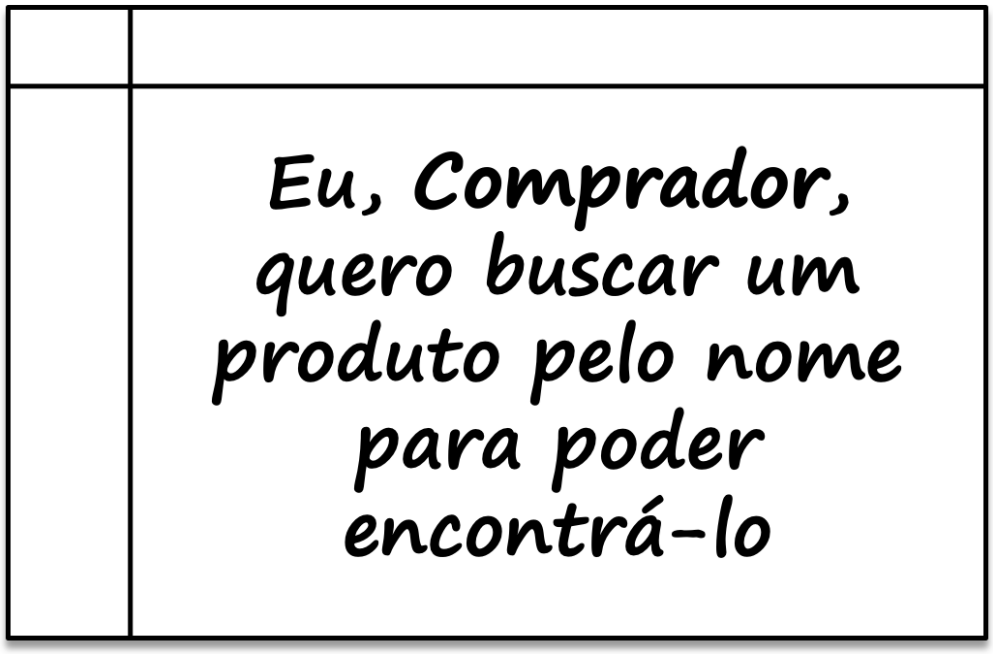


Figura 27.1: Exemplo de User Story, representando uma característica do produto a partir da perspectiva de seu usuário

A User Story é apenas uma promessa de uma conversa, um lembrete de que mais detalhes serão necessários para a sua implementação e que, antes disso, ela não deve ser considerada suficiente para tal (COCKBURN, 2007). Ela é o começo, mas deve ser seguida por uma série de conversas entre as pessoas de negócios (no Scrum, o Product Owner e, possivelmente, outras partes interessadas) e os Desenvolvedores, para apenas então esses últimos serem capazes de iniciar o seu trabalho com ela. Essas conversas visam a definir e a capturar os detalhes necessários para a implementação da característica do produto a que se refere a User Story.

Esses detalhes podem ser documentados de diferentes formas e anexados à sua User Story correspondente. Acredito, no entanto, que apenas Testes de Aceitação da User Story, descritos mais adiante, já representem documentação suficiente, uma vez que devem cobrir, no

mínimo, os aspectos essenciais da característica do produto.

Ao optarmos pelas User Stories, tarefas técnicas não serão, a princípio, representadas como itens de Product Backlog. Elas serão quase sempre parte de alguma User Story, discutidas e definidas apenas quando essa User Story for movida para o Sprint Backlog de algum Sprint. O mesmo podemos imaginar para a maioria das tarefas de pesquisa ou de investigação que se façam necessárias. Dessa forma, buscamos evitar (ou minimizar) o domínio da perspectiva mais técnica no Product Backlog e mantemos um maior foco na geração de valor.

No entanto, não faz sentido que certas tarefas sejam parte de uma User Story. Tarefas que perpassam várias User Stories ou tarefas de correção de problemas, por exemplo, dificilmente podem ser descritas sob a perspectiva do usuário. Essas tarefas serão itens do Product Backlog, mas não são representadas como User Stories.

Para essas, alguns times utilizam um formato parecido com o das User Stories e as chamam de “histórias técnicas”. Eu não recomendo essa prática, já que tende a criar confusão entre o que é trabalho técnico e o que gera valor direto para o usuário e deve ser o foco primário do trabalho de construção do produto.

27.2 Como é a User Story?

Ron Jeffries (2001) estabeleceu três aspectos críticos para a User Story, chamados de "os três C’s": o Cartão, as Conversas e a Confirmação.

Em resumo, os Desenvolvedores, o Product Owner e, possivelmente, outras partes interessadas preparam User Stories que acreditam que serão implementadas em seguida. Essa preparação pode ocorrer em sessões de Refinamento do Product Backlog, na reunião de Sprint Planning ou em alguma outra circunstância mais informal. Ao interagir, eles passam, User Story a User Story, pelos três C’s.

O **Cartão** de cada User Story serve como um lembrete e traz uma descrição breve do comportamento ou característica a ser criado ou modificado no produto. Eles leem o Cartão e realizam **Conversas** sobre essa User Story, para assim definir os detalhes da solução a ser implementada, com que o usuário vai interagir. Esses detalhes servirão posteriormente como **Confirmação** de que essa solução estará funcionando conforme acordado.

Veremos, em seguida, os detalhes do que cada um desses aspectos significa.

Cartão

O Cartão é a descrição do comportamento ou característica a ser criado ou modificado no produto, sob a perspectiva de um usuário desse produto. Ou seja, é a própria User Story. Essa descrição é concisa e propositalmente incompleta, suficiente apenas para identificar o que é e de que se trata essa característica, e em geral cabe em uma nota adesiva. Uma série de conversas ainda será necessária para chegarmos aos detalhes necessários para o seu trabalho de implementação. O nome "Cartão" é dado porque, inicialmente, as User Stories eram escritas em cartões ou fichas de índice.

Não existe um formato obrigatório para escrevermos as User Stories. No entanto, o pessoal da empresa Connextra, usuários precoces do Extreme Programming, criaram o que é hoje o padrão mais conhecido. Esse formato utiliza três parâmetros: “QUEM”, “O QUÊ” e “POR QUÊ” (COHN, 2004).

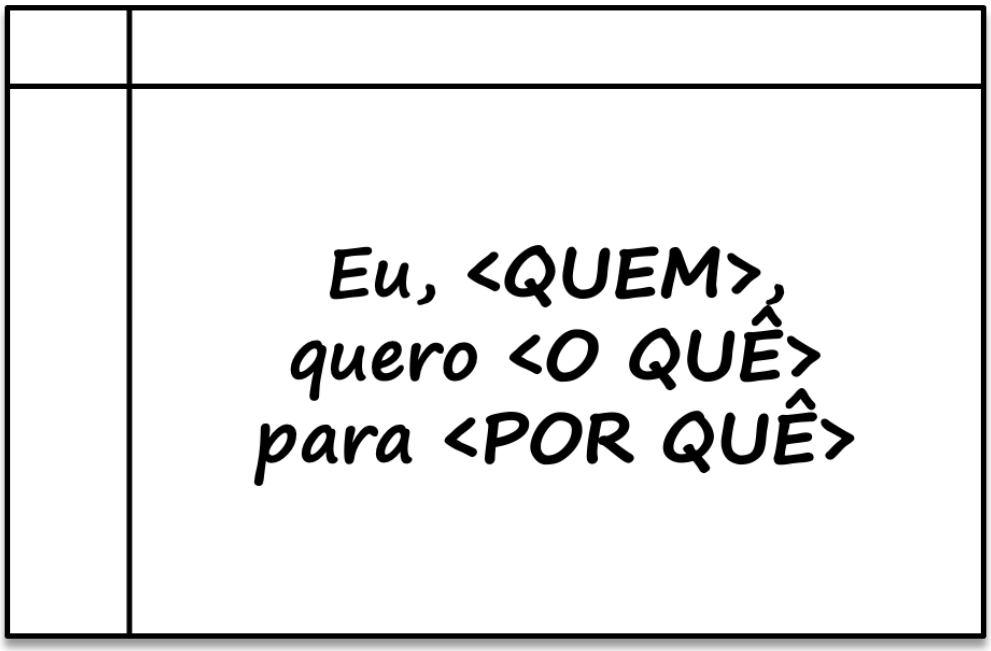


Figura 27.2: Formato baseado no padrão da Connextra de escrita de uma User Story

QUEM define quem é o usuário para que a User Story tem valor. Pode ser representado, por exemplo, por um tipo de usuário do produto (como "Comprador" ou "Administrador"), por uma *persona*, por uma *protopersona* ou até mesmo por um usuário específico, ou seja, uma pessoa que existe de fato.

"QUEM" ajuda a criar no leitor da User Story uma imagem mental desse usuário e a gerar uma empatia com ele, permitindo que se coloque em seu lugar. Em minha opinião, User Stories que começam "Eu, usuário..."

ou "Eu, cliente..." não atingem esse objetivo e devem ser evitadas.

A técnica de *personas* é frequentemente utilizada com o propósito de estimular o pensamento voltado ao usuário. Uma persona é um usuário fictício que representa um perfil distinto de usuários do produto, que terão objetivos em comum atendidos pelo produto. Com o uso dessa técnica, são criadas diferentes personas, cada uma representando um perfil relevante de usuários, geralmente a partir de suas necessidades, valores, objetivos, frustrações e desejos (BULEY, 2013). Personas são baseadas em um amplo trabalho de pesquisa e investigação dos usuários e geralmente possuem um bom nível de detalhes. São largamente usadas por profissionais de *marketing* e de *design* de experiência do usuário.

A técnica de *protopersonas* é uma versão simplificada de personas, mais leve, ad hoc e, portanto, menos rigorosa. A protopersona exige muito menos investimento, já que pode ser criada a partir de qualquer pedaço de informação e conhecimento disponíveis, inclusive a partir de interações entre pessoas que tenham conhecimentos, ideias e palpites sobre os usuários do produto (BULEY, 2013). Por essa razão, é a preferida para o trabalho com Scrum.

Eu geralmente represento a protopersona com um nome, uma imagem ou foto, um objetivo claro, características, comportamentos e motivações, além de um contexto que a humanize, traga empatia e, possivelmente, a coloque no contexto do uso do produto. Para manter o foco no usuário, prefiro não mencionar, a princípio, o produto nessa descrição.

O nome da protopersona pode ser totalmente inventado (como, por exemplo, “João”). Com o uso, todos se acostumarão com o nome e o vincularão à protopersona automaticamente. Mas pode ser interessante que esse nome traga alguma característica distinta da protopersona (no nosso exemplo abaixo, “João Baldeação” já nos diz algo sobre ela). Podemos também pegar emprestado o nome de alguma personalidade conhecida que compartilhe características com essa protopersona, ou misturarmos esse nome com algum outro elemento, de forma a não nos fecharmos demais em suas características particulares. Uma protopersona criada em aula, certa vez, se chamava “Maria Shakira”.

É importante criar um número pequeno de protopersonas. Se houver muitas, fica difícil delimitarmos onde termina uma e começa a outra e, assim, conseguirmos responder da melhor forma possível às necessidades dos usuários. Em geral, algo entre três e sete ou oito personas é o número recomendado.

Na figura a seguir, descrevemos "João Baldeação", uma protopersona de um aplicativo de transportes.

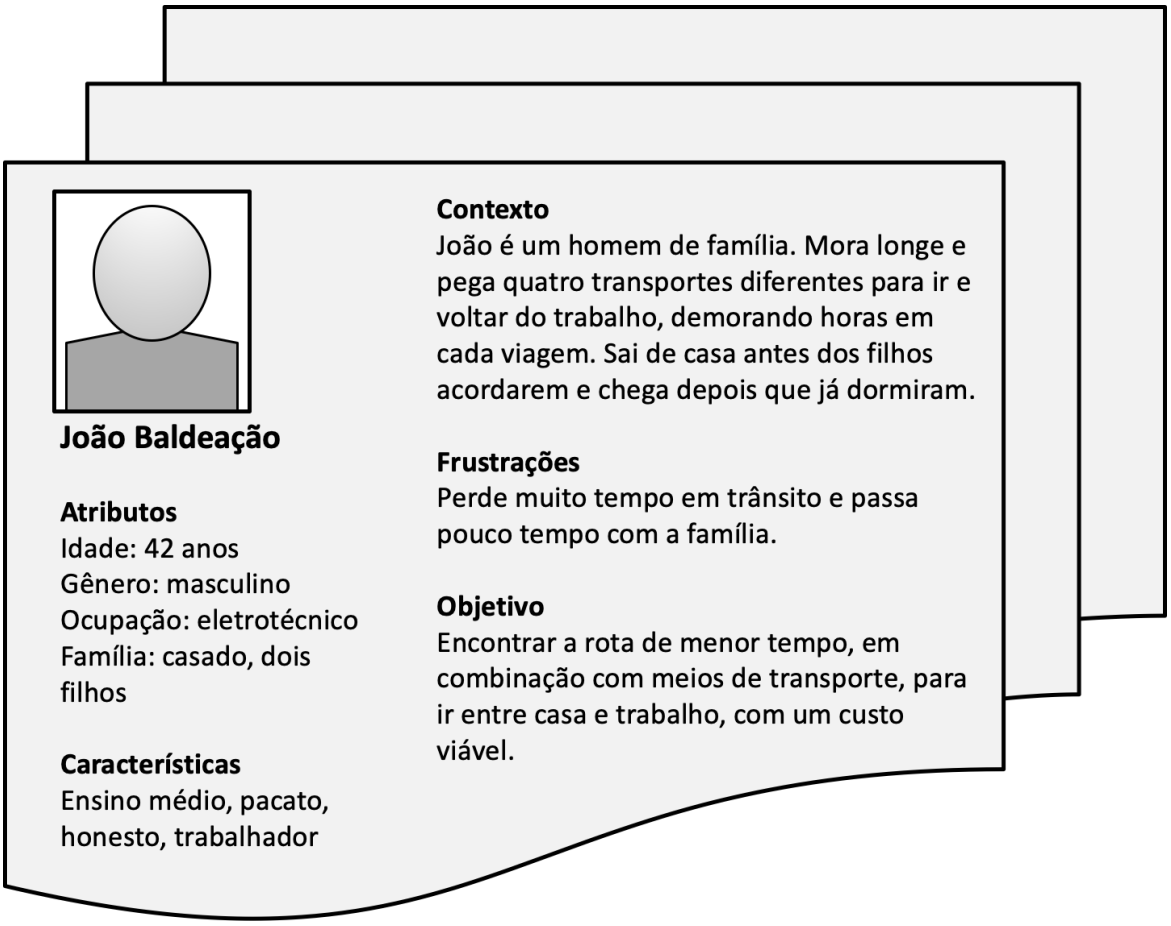


Figura 27.3: Exemplo de persona

Uma User Story para essa persona poderia ser algo assim: *para passar mais tempo com a minha família, eu, João Baldeação, quero encontrar a rota e meios de transporte de menor tempo entre casa e trabalho e com um custo viável*.

O QUÊ define qual é o comportamento ou característica do produto a ser criada, e que tem valor para o usuário representado em “QUEM”.

POR QUÊ define qual o benefício, problema resolvido ou valor imediato obtido pelo usuário ao poder usufruir do comportamento ou característica a ser criada. Aqui, devemos ter o cuidado para não nos remetermos a um

objetivo muito amplo, que será realizado a partir de múltiplas User Stories (como, por exemplo, o Objetivo do Sprint). O "POR QUÊ" é basicamente o que o usuário passa a ser capaz de realizar ao ter o seu "O QUÊ" atendido.

Podemos ver na figura a seguir uma outra forma de expressar a User Story, utilizando os mesmos elementos, mas com ênfase no benefício do usuário.

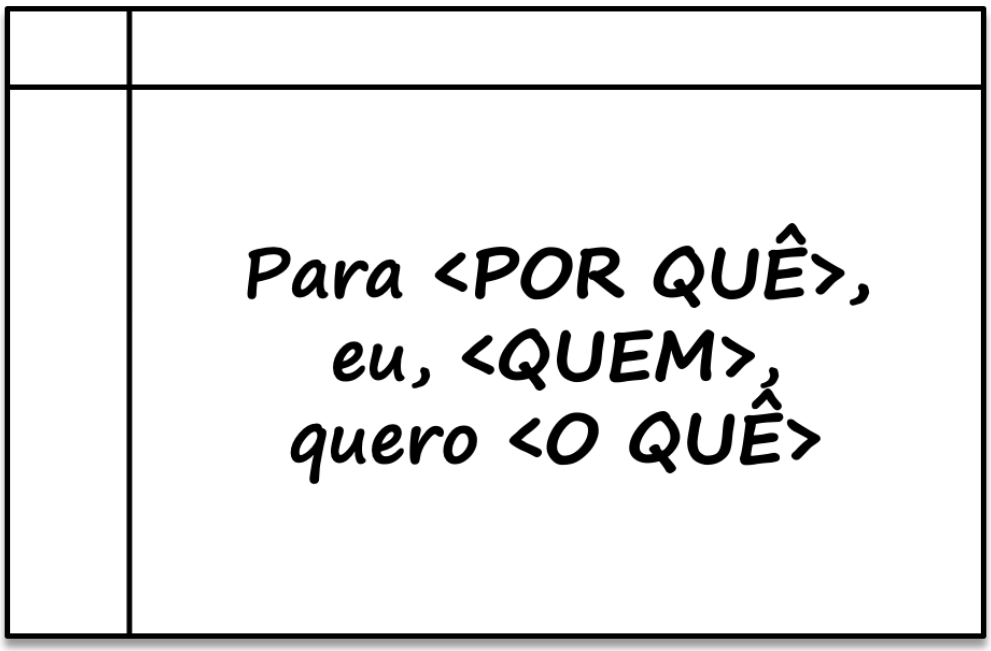


Figura 27.4: Mais um padrão de escrita de uma User Story

Outra forma simples de se escrever a User Story é: “Um [QUEM] pode [O QUÊ] para [POR QUÊ]”. Por exemplo, se imaginarmos como produto um aplicativo para viagens, uma de suas User Stories poderia ser: *Um Viajante de Turismo quer compartilhar sua viagem com seus amigos para lhes causar admiração*.

Discussão: será mesmo o ponto de vista do usuário?

O ponto de vista do usuário é o de quem tem um problema ou necessidade que o produto vai resolver ou atender, certo? Vejamos, a seguir, o exemplo de User Story que utilizamos anteriormente:

> *Eu, Comprador, quero buscar um produto pelo nome para poder encontrá-lo.*

Essa é a forma original e mais comum de escrevermos User Stories, e é correta. No entanto, “buscar um produto pelo nome” não estabelece um problema ou necessidade do usuário. Ao expressar um comportamento ou característica do produto, a User Story representa, na verdade, uma solução. Eu entendo que, assim, a User Story não expressa realmente a perspectiva do usuário. Qual seria o problema do usuário, então? Eu proponho então reescrevermos a User Story da seguinte forma:

> *Eu, Comprador, quero encontrar um produto de que sei o nome para então decidir comprá-lo.*

Ou algo parecido. Nesse caso, o problema do usuário é encontrar um produto cujo nome ele conhece. Uma entre as diferentes possíveis soluções é buscar produtos por nome, com uma caixa de texto e um botão, por exemplo. Outra seria mostrar simplesmente uma lista de todos os produtos ordenados por nome para que ele possa encontrar o que precisa. Essa alternativa poderia, por exemplo, ser mais adequada para uma versão inicial do produto devido a seu custo mais baixo.

Expressar a User Story a partir do problema e não de uma solução particular pode abrir espaço para discussões sobre as possíveis diferentes soluções para o

mesmo problema. Dessa forma, pode ajudar a que soluções mais adequadas sejam encontradas. Nesses casos, a solução posteriormente escolhida pode ser integralmente descrita nos Testes de Aceitação, que descreverei mais adiante.

A abordagem de escrevermos a User Story descrevendo um problema do usuário pode ser especialmente útil para User Stories maiores e mais vagas (os “épicos” descritos mais adiante neste capítulo), caso em que estamos ainda distantes da solução.

Conversas

São conversas sobre a User Story, nas quais os detalhes do comportamento ou característica a ser criado no produto são discutidos, negociados, definidos e, então, documentados na forma de Testes de Aceitação (a Confirmação, como veremos adiante). Elas geram uma compreensão compartilhada sobre o que é a solução a ser implementada para gerar valor para o usuário.

As conversas são imprescindíveis para o trabalho dos Desenvolvedores, uma vez que o Cartão não possui os detalhes necessários para permitir o trabalho de implementação do item. Em geral, participam dessas conversas o Product Owner, os Desenvolvedores e outras partes interessadas, como pessoas de negócios, que possam ajudar. Qualquer pessoa capaz de contribuir com detalhes pode ser convidada a participar.

Tomemos como exemplo a User Story de um exemplo anterior: *Um Viajante de Turismo quer compartilhar sua viagem com seus amigos para lhes causar admiração*. Os Desenvolvedores e o Product Owner, por meio de conversas, podem concluir que permitir o

compartilhamento de fotos e vídeos em uma rede social diretamente do aplicativo é a solução desejada. A partir daí, escrevem os Testes de Aceitação correspondentes.

As conversas podem ocorrer em qualquer momento, mas são geralmente parte de sessões de Refinamento do Product Backlog ou da reunião de Sprint Planning.

Confirmação

> *Itens de Product Backlog com frequência incluem descrições de testes que comprovarão que estão completos quando estiverem "Prontos"* - Guia do Scrum (SCHWABER; SUTHERLAND, 2017).

A Confirmação é o acordo resultante das Conversas. É um conjunto de afirmações que documentam os detalhes do que será implementado a partir da User Story, definem seus limites e regras e permitem a confirmação de que o comportamento ou característica nova, criada no produto, está conforme o que foi definido nas Conversas. Ao mesmo tempo, ajudam os Desenvolvedores a determinar, durante o Sprint, quando o que estão implementando está completo de acordo com o combinado. Quando escritas em forma de testes, essas afirmações são chamadas de Testes de Aceitação.

Para escrevermos os Testes de Aceitação, podemos utilizar, por exemplo, as costas do cartão ou da nota adesiva da User Story, ou um quadro branco, que devem ser preservados e permanecer visíveis durante todo o Sprint. Ou, quando possível, podemos utilizar uma ferramenta de automatização de testes.

Ao trabalharmos com Testes de Aceitação automatizados e executados frequentemente, reduzimos os riscos de

que as mudanças frequentes no produto gerem erros que passem desapercebidos e cheguem aos usuários. Existem inúmeras ferramentas para fazermos essa automatização para o desenvolvimento de software, e diversas inclusive permitem sua escrita em linguagem humana, servindo como uma forma de documentação executável.

Os Testes de Aceitação são, em geral, expressos por enunciados pequenos e de fácil entendimento. O uso de exemplos na definição desses testes é uma prática altamente recomendada, inclusive para situações complexas. Nesse caso, os testes descrevem, a partir dos exemplos, as respostas do produto que são esperadas diante de interações dos seus usuários com ele.

Os Testes de Aceitação, quando viável, são criados antes do início do Sprint, possivelmente durante sessões de Refinamento do Product Backlog no Sprint anterior. É um trabalho dos Desenvolvedores, em conjunto com o Product Owner, clientes e outras partes interessadas, de entendimento e definição do que é esperado para proporcionar valor. Esses testes também podem ser rascunhados em um momento inicial, para então serem formalizados - e talvez automatizados - durante o trabalho no Sprint pelos Desenvolvedores.

Enquanto todas as User Stories devem satisfazer à mesma Definição de Pronto ao serem implementadas, cada User Story possui seu próprio conjunto de Testes de Aceitação. Passar sem problemas pelos testes é uma das condições da Definição de Pronto, o que significará, ao final do Sprint, uma Confirmação de que o comportamento ou característica nova, criada a partir da User Story, está funcionando conforme o que foi definido e acordado.

Como exemplo, vamos imaginar que a seguinte User Story é forte candidata a entrar no próximo Sprint para implementação:

> *Eu, Comprador, quero utilizar meu cartão de crédito no pagamento dos produtos escolhidos, para ter praticidade e segurança no pagamento.*

Em uma das sessões de Refinamento do Product Backlog, poderemos rascunhar os seguintes Testes de Aceitação relativos a essa User Story, que serão automatizados em um script durante o Sprint, nesse exemplo:

Testes de aceitação:

- Comprador utiliza cartão de crédito Visa (passar).
- Comprador utiliza cartão de crédito MasterCard (passar).
- Comprador utiliza cartão de crédito Amex (falhar).
- Comprador utiliza cartão de crédito com expiração em 01/01/2030 (passar).
- Comprador utiliza cartão de crédito com expiração em 01/01/2000 (falhar).
- Comprador utiliza cartão de crédito com expiração hoje (passar).
- Comprador entra um número de cartão de crédito válido (passar).
- Comprador entra um número de cartão de crédito inválido (falhar).

Nesses exemplos, podemos ver claramente que as bandeiras de cartão aceitas são Visa e MasterCard, enquanto que Amex não. Podemos ver também que

cartões de crédito expirando no dia da compra serão aceitos.

Os Testes de Aceitação cobrem cada aspecto do comportamento ou característica referida pela User Story e, assim, são geralmente suficientes para documentá-la.

27.3 Criando boas User Stories

Bill Wake (2003), autor do livro *Extreme Programming explored*, descreveu em seu blog quais seriam as características de boas User Stories. Ele formou o acrônimo INVEST ("investir", em inglês) com a primeira letra de cada uma dessas características, afirmando que devemos "investir em boas User Stories".

De acordo com Wake, uma User Story deve ser:

- **independente** — User Stories devem ser tão independentes ou desacopladas umas das outras quanto possível, para facilitar sua ordenação e planejamento. User Stories com sobreposições de conceitos com outras devem, assim, ser evitadas. Buscamos ser viável alterar livremente a ordem em que serão implementadas e, ao fazê-lo, não ser necessário alterar suas estimativas. Em minha experiência, nem sempre isso é viável. Cabe adicionar que, enquanto independente, uma User Story se traduzirá sempre em um comportamento ou característica do produto que representa valor para o usuário. Além disso, deve ser possível entendermos uma User Story sem ser necessário lermos quaisquer outras;

- **negociável e negociada** — seus detalhes serão discutidos, negociados e definidos entre pessoas de negócios e os Desenvolvedores. São as Conversas dos três C’s;
- **valiosa** — devem representar valor visível e utilizável para os usuários do produto. Ao dividirmos uma User Story, o resultado dessa divisão deve ser User Stories menores que também representem comportamentos ou característica com valor para o usuário, e não partes ou camadas de trabalho técnico;
- **estimável** — estimativas fornecem o custo de transformar uma User Story em uma característica ou comportamento do produto. Os Desenvolvedores devem possuir compreensão e detalhes suficientes — tanto técnicos quanto de negócios — para poderem estimar o trabalho, e assim o Product Owner possa ordenar as User Stories apropriadamente.

Dessa forma, conversas suficientes devem ser realizadas entre os Desenvolvedores, o Product Owner e quem mais for necessário para que os Desenvolvedores obtenham os detalhes necessários para que possam realizar as estimativas. Eles devem discutir possíveis soluções e, caso necessário, executar pequenos experimentos para reduzir os riscos técnicos dessa User Story. É claro que o nível de detalhes necessário e a precisão da estimativa dependem de quão próxima a User Story está de entrar em um Sprint para ser implementada.

Mesmo no caso em que não utilizemos estimativas, conversas que levem a uma boa compreensão das User Stories são importantes para entendermos quando são muito grandes e devem ser fatiadas, de forma que

fiquem pequenas o suficiente para poderem ser implementadas;

- **pequena (*small*, em inglês) ou dimensionada apropriadamente** — apenas User Stories pequenas e com um bom nível de detalhes podem ser implementadas em um Sprint. Quanto mais no alto do Product Backlog a User Story estiver, de menor granularidade ela será.

User Stories pequenas têm maior precisão em suas estimativas, trazendo uma melhor previsibilidade, permitem um melhor ordenamento e permitem que mais User Stories sejam prontas em um Sprint, aumentando as chances de o Objetivo do Sprint ser realizado.

A descrição da User Story também deve ser pequena.

- **testável** — deve ser possível verificar e confirmar que a User Story está completa, ou seja, que foi transformada em parte do produto e está funcionando conforme esperado. A verificação é realizada por meio dos Testes de Aceitação. É a Confirmação dos três C’s.

27.4 Épicos e Temas

O Product Backlog é ordenado, de forma que seus itens são tão menores e mais detalhados quanto mais acima estiverem nessa lista. Os itens da parte superior do Product Backlog são pequenos e possuem detalhes suficientes que os permitem serem aceitos para o trabalho do próximo Sprint. Os itens mais abaixo são cada vez maiores e carregam menos detalhes.

Épicos são User Stories que representam itens grandes demais ou sem detalhes suficientes para serem implementados em um Sprint. Enquanto épicos, essas User Stories somente obterão mais detalhes e evoluirão em User Stories menores quando partes do que dizem respeito ganharem importância.

Épicos podem ser maiores ou menores, mas jamais possuem granularidade e detalhes suficientes para fazerem parte de um Sprint Backlog e, assim, não podem ser implementados em um Sprint.

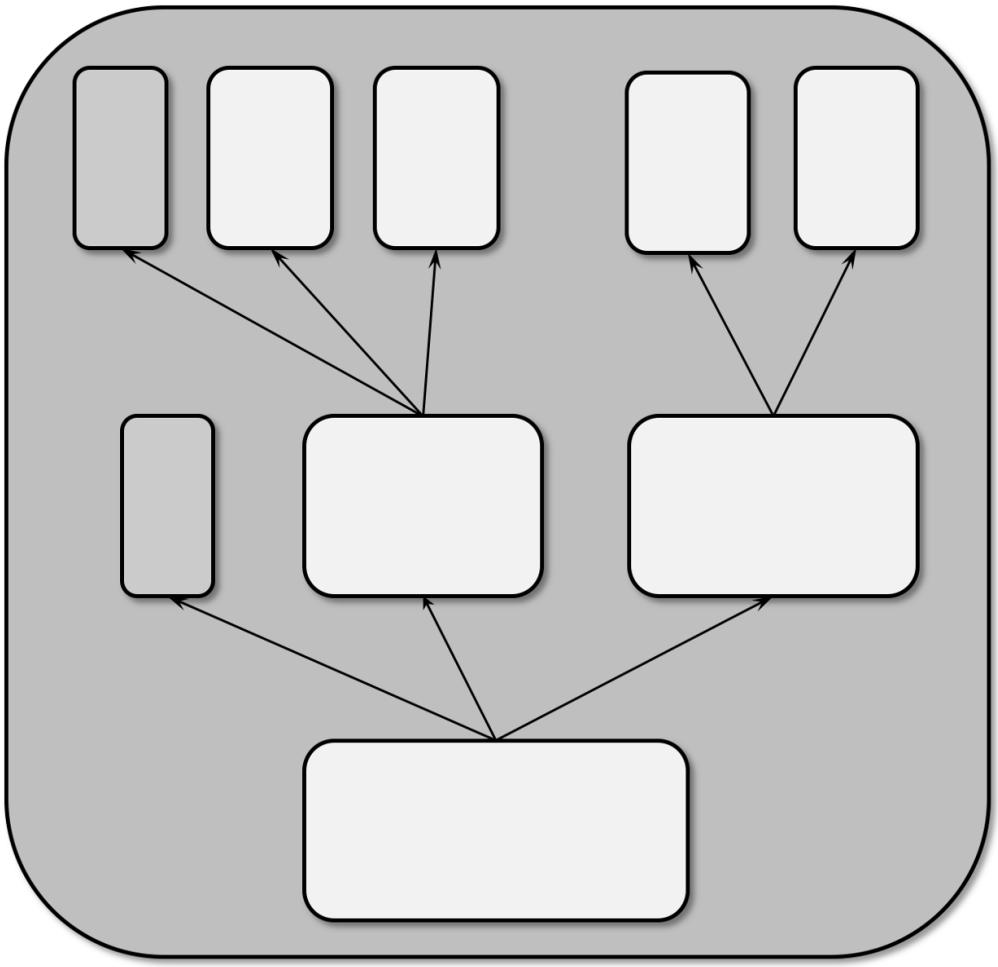


Figura 27.5: Épicos evoluem para User Stories menores

À medida que o problema endereçado pelo épico ganha importância, emergem dele uma ou algumas poucas User Stories com granularidade mais fina, que representem apenas suas próximas partes mais importantes. Por ter menor importância naquele momento, o resto do épico segue no Product Backlog ainda grande e vago, com uma ordem mais baixa, e lá se mantém enquanto a questão tratada por ele seguir existindo.

Na figura anterior, as User Stories em cor mais escura possuem granularidade fina o suficiente para o trabalho, enquanto que as outras são épicos.

“Enquanto comprador, quero pagar para confirmar a compra” é um épico porque, além das diversas formas de pagamento que poderão vir a ser disponibilizadas (mas que não são importantes no momento), outras questões relativas à compra poderão futuramente ser tratadas, como cupons de desconto, escolha de parcelamento, definição de endereço de entrega etc. Em um primeiro momento, “Enquanto comprador, quero pagar via transferência bancária para fechar a compra” e “Enquanto comprador, quero entrar o endereço de entrega para poder receber o produto” podem ser as duas primeiras fatias finas, e portanto não mais épicos, a serem implementadas.

Uma definição alternativa para épicos, adotada por diversas ferramentas de software, os caracteriza como contêineres de User Stories. Suas User Stories se mantêm estruturalmente vinculadas ao épico enquanto existirem. Nessa definição, o épico e suas histórias são geralmente relacionados a um objetivo de negócios ou do usuário específico, o que se assemelha ao que chamamos de tema.

Temas são coleções ou conjuntos de User Stories que estão de algum modo relacionadas e, por essa razão, podem ser agrupadas. Motivos para agrupar User Stories em temas incluem, entre outros:

- juntas visam a realizar um mesmo objetivo de negócios ou do usuário;
- são provenientes de um mesmo épico;
- pertencem a um time em um ambiente com múltiplos times trabalhando a partir de um mesmo Product Backlog.

O agrupamento de User Stories em temas relacionados a objetivos de negócios pode ajudar a manter o foco do Time de Scrum nesses objetivos, ao conectar com eles cada item individual.

Um tema geralmente possui um curto título que o identifica. As User Stories relativas ao pagamento, por exemplo, podem pertencer ao mesmo tema “Pagamento”.

Capítulo 28
Story Points: estimando o trabalho

Conteúdo

1. Para que serve estimar?
2. Precisão e acurácia de estimativas baseadas em juízo.
    - Conceitos básicos.
    - Precisão e acurácia suspeitas: Lei de Parkinson.
    - Precisão e acurácia nas estimativas ágeis.
3. Unidades para as estimativas.
    - Estimativas absolutas: tempo real e tempo ideal.
    - Estimativas relativas: Story Points.
        - Por que estimativas relativas?
        - O que são Story Points?
        - Escalas para Story Points.
        - Criação da escala a partir de referências.
        - Por que Fibonacci?
        - Story Points: complexidade, risco, esforço ou tamanho?
        - Story Points: tempo ou esforço?
4. Como realizar a estimativa?
    - Planning Poker.
5. Velocidade.
6. No estimates: estimativas são desperdício.

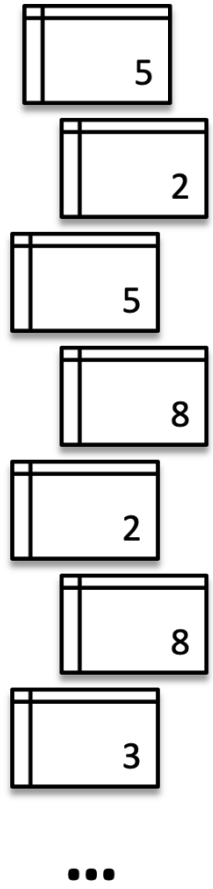
28.1 Para que serve estimar?

Entendemos por estimar o ato de julgar e formar uma opinião sobre o valor de algo.

O ato de estimar o tempo que será necessário para realizar um determinado trabalho pode ser útil para nos ajudar a planejarmos melhor. No trabalho com Scrum, especificamente, estimar o tempo necessário para implementar os itens do alto do Product Backlog facilita ao Time de Scrum planejar o que caberá no próximo Sprint.

Ao entendermos as estimativas de tempo desses itens como o seu custo e ao estimarmos o seu valor de negócio, podemos ainda definir a ordem em que eles serão implementados. Seguindo essa linha, podemos estabelecer que os itens de maior valor e de menor custo pertencerão ao alto do Product Backlog e serão implementados primeiro, pois trazem um maior retorno sobre o investimento. Quanto menor a relação entre seu valor e seu custo, menor o seu retorno estimado e mais baixa será a ordem do item.

No exemplo da figura a seguir, determinamos a ordenação, de forma simplificada, a partir do retorno individual de cada item, utilizando o valor de negócio relativo, estimado dentro de uma escala de 1 a 10, e o custo relativo de cada item, dado pela estimativa de seu tempo de implementação em Story Points, unidade que definiremos mais adiante.



Valor Relativo (1-10)	Custo Relativo (Story Points)	Retorno Relativo (Valor/Custo)
10	5	2
2	2	1
8	8	1
6	8	0,75
1	2	0,5
3	8	0,375
1	3	0,333
...	...	...

Figura 28.1: Utilizando o retorno para priorizar os itens

As estimativas dos tempos são feitas por todos os Desenvolvedores, em conjunto, que compartilham essa responsabilidade. Por essa razão, o simples exercício de se reunir e discutir para realizar essas estimativas os leva a uma compreensão compartilhada e mais profunda do que é cada item e de como ele será implementado. Dessa forma, a importância não está somente nos números, mas também no caminho para chegarmos a eles.

28.2 Precisão e acurácia de estimativas baseadas em juízo

Conceitos básicos

Ao estimarmos o tempo que levaremos para realizar um determinado trabalho, não estamos usando cálculos avançados nem aplicando conceitos de probabilidade e estatística. Na realidade, construímos essa estimativa simplesmente utilizando-nos de nossos conhecimentos, bom-senso, experiências passadas, contribuições de terceiros e de demais informações disponíveis.

De forma simplificada, podemos definir uma estimativa a partir de um **valor de referência** e, em torno dele, uma faixa de valores com tamanho maior ou igual a zero, ambos dados em alguma unidade. Por exemplo, podemos estimar que uma determinada atividade durará entre 3 e 5 horas ou 4 ± 1 horas.

Essa faixa de valores é chamada de **intervalo de confiança** da estimativa. Esse intervalo é tal que existe uma determinada probabilidade de que o valor real estará dentro dele. De forma prática, escolhemos para nossas estimativas um intervalo de confiança maior, quanto maior for a incerteza relacionada a essa estimativa.

Definimos a **precisão** de uma estimativa como a sua granularidade, que é inversamente relacionada ao tamanho do intervalo de confiança em torno do valor de referência. Com isso, quanto menor é o intervalo de confiança usado para a estimativa, mais precisa ela é (veja a figura a seguir).

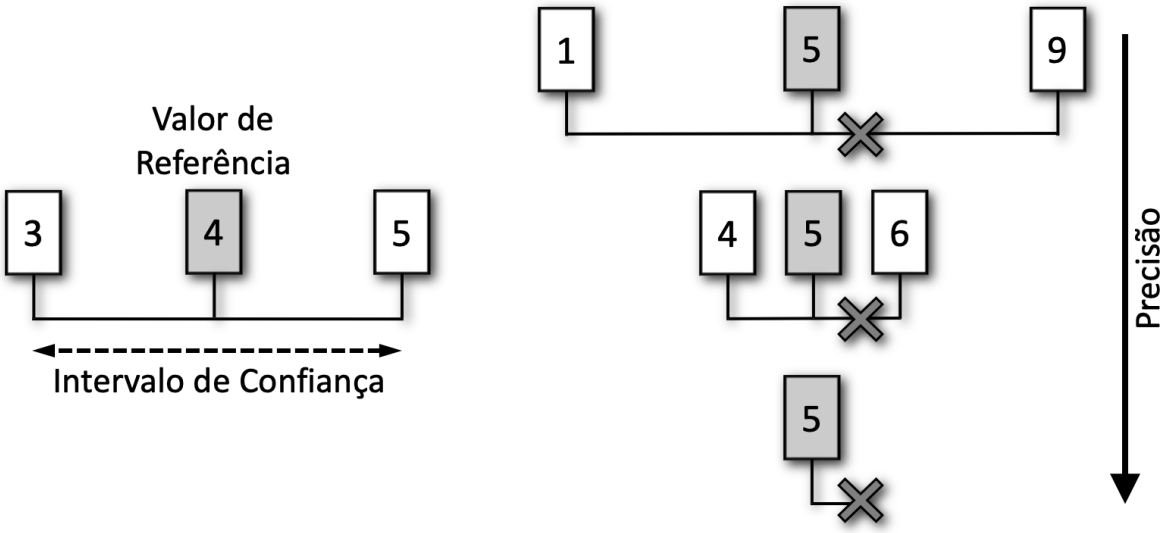


Figura 28.2: A estimativa e a precisão de uma estimativa

Por exemplo, a estimativa de que uma atividade durará exatamente 5 horas é mais precisa do que uma estimativa de que essa atividade vai durar 5 ± 1 horas (isso é, entre 4 e 6 horas). Ela é mais precisa ainda do que uma estimativa de que essa mesma atividade durará 5 ± 4 horas (entre 1 e 9 horas).

É importante observar que também podemos aumentar a precisão da estimativa utilizando unidades com maior granularidade. Por exemplo, a estimativa de 5 horas, 3 minutos e 10 segundos pode ser interpretada como mais precisa do que a estimativa de que a atividade vai durar 5 horas.

Repare que a precisão independe totalmente do valor real verificado posteriormente. Ou seja, a precisão não é diferente para qualquer dessas estimativas se a atividade, na realidade, durar 4 ou durar 20 horas, por exemplo.

O grau de **acurácia** ou de exatidão de uma estimativa é definido por quão perto do valor real ela está. Dessa forma, quanto menor a diferença entre a estimativa e o

valor real verificado posteriormente, maior terá sido o seu grau de acurácia.

O cálculo de um valor numérico para o grau de acurácia pode ser complicado, já que a estimativa é, na realidade, um intervalo. Aqui, porém, usarei uma definição propositalmente simplificada: consideramos a estimativa com acurácia se ela contiver o valor real verificado posteriormente (quer dizer, a estimativa se mostrou certa), e sem acurácia caso não o contenha (ou seja, a estimativa se mostrou errada).

Imagine, por exemplo, uma atividade para cuja duração diferentes estimativas foram realizadas. Digamos que a sua execução tenha durado 5,5 horas. Podemos afirmar que a estimativa de que ela duraria exatamente 5 horas não teve acurácia, de acordo com as nossas definições, pois não contém o valor real. A estimativa de que ela duraria 5 ± 1 horas tem menor precisão, mas teve acurácia. A estimativa de que ela duraria 5 ± 3 horas também teve acurácia, mas tem uma precisão bem menor.

Em outro exemplo, ao estimarmos tendo "dias" como unidade, se afirmarmos em nosso planejamento que uma determinada funcionalidade ficará pronta no dia 18 de novembro, podemos considerar essa estimativa como de alta precisão, independente da data em que a funcionalidade de fato fique pronta. Caso essa funcionalidade fique pronta apenas no dia 25 de novembro, essa terá sido uma estimativa sem acurácia, de acordo com nossa definição de acurácia.

Se tivéssemos afirmado, no entanto, que a funcionalidade estaria pronta no mês de novembro (o que na realidade significa entre os dias 1 e 30 de

novembro), nossa estimativa teria sido de mais baixa precisão, porém com acurácia.

Repare pelos dois exemplos anteriores que, ao reduzirmos a precisão de uma estimativa, aumentamos as chances de ela ter acurácia. Uma precisão cada vez menor (ou intervalo de confiança cada vez maior) é como um alvo que fica cada vez maior e que, dessa forma, aumenta as nossas chances de acertá-lo, ou seja, de termos acurácia. Portanto, precisão e chance de acurácia têm uma relação inversa.

Precisão e acurácia suspeitas: Lei de Parkinson

A Lei de Parkinson ilustra que, a partir de uma estimativa de alta precisão para o tempo de realização de alguma atividade, obter um alto grau de acurácia (e, portanto, de acerto) é geralmente um resultado artificial. Em outras palavras, se ao estimarmos o tempo necessário para realizarmos uma determinada atividade em um número de horas e, de fato, a atividade estiver completa após esse exato número de horas de trabalho, há possivelmente algo de errado (COHN, 2005).

A **Lei de Parkinson** foi criada por Cyril Northcote Parkinson nos anos 1950 e publicada em um artigo bem-humorado na revista britânica *The Economist* e depois em um livro. Essa "lei" afirma que *a utilização de um recurso se expande até a sua disponibilidade*. Ou, para o nosso caso específico, *o trabalho se expande de modo a preencher o tempo disponível para sua realização*. Dessa forma, ao predeterminarmos o tempo disponível (ou a data de entrega) para a realização de uma atividade, o tempo usado para a sua realização naturalmente se expandirá até a duração preestabelecida, mesmo que menos tempo pudesse ter sido suficiente.

Uma manifestação comum da Lei de Parkinson ocorre quando poderíamos terminar a atividade mais cedo, mas estendemos a sua execução ao gastarmos tempo com detalhes sem grande utilidade ou atividades paralelas que consideramos mais estimulantes.

Por exemplo, digamos que um gerente de projetos, em um projeto tradicional, tenha estimado que um membro de sua equipe demorará um número de dias para realizar uma determinada atividade. Ao longo do trabalho, esse membro naturalmente preencherá todo esse tempo disponível. Se há a percepção de que lhe sobrará tempo, ele deverá realizar o trabalho com uma maior minúcia e adicionando atributos não essenciais. Além disso, ele poderá utilizar uma parte relevante do tempo para realizar pesquisas, leituras etc. e outras atividades que considere mais interessantes.

O conceito de **Síndrome do Estudante** (COHN, 2005) está relacionado a como o estudante realiza seus trabalhos de casa, ou seja, deixando-os para a última hora. É outra forma de manifestação da Lei de Parkinson. Diante de um prazo para realizar uma determinada atividade, é um comportamento comum deixarmos sua realização para a última hora, ocupando todo esse tempo disponível. Muitas vezes, para conseguirmos terminar a tempo, deixaremos de lado o cuidado e a qualidade necessários. Algumas vezes, perderemos o prazo.

Em consequência, de acordo com a Lei de Parkinson, em qualquer das suas manifestações, um gerente de projetos tradicional que frequentemente acerta estimativas impostas para o trabalho de seus funcionários não tem razão para comemorar. O fato de existir essa estimativa, na realidade, acaba por produzir artificialmente o "acerto".

A Lei de Parkinson não possui caráter científico, mas ilustra bem o comportamento humano.

Precisão e acurácia nas estimativas ágeis

Abordagens tradicionais para projetos usam um alto grau de precisão em seu planejamento, o que, a uma primeira vista, pode parecer benéfico. Datas e durações exatas para o cumprimento de atividades são marcadas em longos cronogramas, que vemos muitas vezes representados por meio de gráficos de Gantt em sistemas de gerenciamento de projeto.

Sabemos, no entanto, que essas datas e durações estimadas muito provavelmente se mostrarão erradas quando tratamos do trabalho de desenvolvimento de produtos, pois a precisão utilizada ignora a realidade de incertezas que permeia atividades dessa natureza. Portanto, nesse contexto, podemos afirmar que abordagens tradicionais usam estimativas de alta precisão, mas de baixa acurácia.

Ao utilizarmos estimativas para o planejamento do trabalho com Scrum, uma maior acurácia e, portanto, um maior acerto, é mais importante do que uma precisão máxima. Em outras palavras, **a acurácia é mais importante que a precisão**. O que não significa que a precisão seja desimportante.

Em oposição aos planos com altíssima precisão usados em projetos tradicionais, devemos reconhecer que, para obtermos uma acurácia razoável, quanto menos detalhes possuirmos acerca de um trabalho a ser realizado, menor deve ser a precisão utilizada.

No ambiente de incertezas e mudanças em que está imerso o trabalho a que Scrum se presta, quanto mais

distantes no tempo estivermos da realização de um trabalho, menos detalhes teremos acerca dele. Logo, menor deverá ser a precisão utilizada para se estimá-lo, de modo a obtermos estimativas com maior acurácia.

28.3 Unidades para as estimativas

Estimativas absolutas: tempo real e tempo ideal

Para estimarmos o tempo necessário para a realização de uma atividade, é natural e intuitivo que utilizemos dias ou horas reais de trabalho, que chamo aqui de **tempo real**. Ou seja, fazemos uma estimativa de quanto tempo em dias ou horas acreditamos que a realização da atividade durará.

O tempo real é, de fato, a unidade mais utilizada para estimativas no gerenciamento tradicional de projetos. No entanto, como expliquei anteriormente, ao oferecermos uma estimativa absoluta para a realização da atividade, expressa em um número de dias ou horas ou em uma data final, induzimos as disfunções descritas pela Lei de Parkinson.

Ainda assim, a estimativa em tempo real pode ser útil para atividades lineares ou repetitivas, como as do trabalho fabril e do trabalho braçal. Se, por exemplo, uma fábrica produziu cinquenta carros por dia nos últimos três meses, em média, podemos estimar que ela vai demorar aproximadamente um ano para produzir dezoito mil carros. Se um pintor pintou metade de uma parede em uma hora, podemos esperar que ele conclua o seu trabalho em cerca de mais uma hora. E se um pedreiro levantou um muro de um metro de extensão em quatro horas, podemos esperar que o muro tenha cerca

de dois metros de extensão após oito horas (DE TOLEDO, 2019).

Scrum, no entanto, é próprio para o trabalho criativo, como o de desenvolvimento de produtos, imerso em um ambiente de incertezas e mudanças, emergente, não linear e não repetitivo. Não podemos, por exemplo, esperar de uma equipe, que criou uma brilhante campanha de marketing em uma manhã, repetir o feito à tarde ou no dia seguinte. Um par de horas muito produtivas para um desenvolvedor de software não garante sequer que ele terá alguma outra hora produtiva naquele mesmo dia.

Além disso, o trabalho criativo não se limita às horas normais da jornada de trabalho. O cérebro continua atuando sobre os problemas a caminho de casa, durante o banho e até no decorrer do sono. E como considerar nas estimativas aquele tempo necessário de descanso mental, que utilizamos para nos distanciar do problema e só assim conseguimos chegar a uma solução?

E como considerar a colaboração? Não podemos nos esquecer que cada item é implementado pelos Desenvolvedores em conjunto. Desse modo, sua duração também dependerá, entre outros fatores, de como ocorrerão as interações entre eles.

As estimativas em dias ou horas reais tendem ainda a carregar em si uma promessa ou obrigação irreais. Em vez de serem entendidas como meras previsões para facilitar o planejamento, elas são tratadas como compromissos assumidos, o que leva a uma cobrança impossível. Quando alguém, por exemplo, diz que vai demorar cinco dias para executar uma determinada atividade, a expectativa criada é de que, após cinco dias,

a atividade esteja pronta e essa pessoa será questionada caso isso não ocorra.

Tempo ideal era uma unidade utilizada inicialmente com Extreme Programming, uma metodologia ágil que muitas vezes anda de mãos dadas com Scrum. São dias ou horas ideais de trabalho, que significam quanto tempo levaríamos para realizar a atividade caso todo o foco de trabalho estivesse nela e não existissem quaisquer interrupções, como conversas, leitura de mensagens e e-mails, idas ao banheiro, reuniões, cafezinho etc.

O tempo ideal tende a acentuar ainda mais o problema da cobrança sobre um “compromisso assumido”, pois a execução certamente será mais longa que a estimativa realizada. Ao dizermos que uma atividade levará cinco dias ideais, é difícil justificar o porquê de essa atividade não estar pronta após os cinco dias, pois será natural a impressão de que o tempo não foi bem utilizado. Ou seja, ao se basear em uma situação muito idealizada e longe da realidade, tempo ideal é de difícil compreensão.

Mike Cohn (2005) afirma que, para estimarmos em tempo ideal, devemos assumir que:

- a User Story estimada é a única coisa em que vamos trabalhar;
- tudo o que for necessário para a realização da atividade deve estar disponível, e desde o começo;
- não haverá interrupções ao trabalho.

Sinceramente, não vejo no uso de tempo ideal nenhum benefício para o planejamento.

Levando em consideração as questões aqui levantadas, tanto as estimativas em tempo real quanto em tempo ideal se mostram inapropriadas para o trabalho com Scrum.

Estimativas relativas: Story Points

Por que estimativas relativas?

Estimativas relativas são aquelas que realizamos de forma comparativa entre os elementos sendo estimados. A unidade utilizada é considerada relativa pois é definida a partir de comparações entre esses elementos.

Por exemplo, em vez de estimarmos que uma determinada atividade exigirá um número de dias para ser realizada, nós a compararemos com uma outra atividade já estimada em nossa unidade relativa que, assim, servirá como referência. Dessa forma, se julgarmos que a nossa atividade vai demorar cerca de duas vezes mais tempo que a atividade de referência, e essa referência está estimada em dois pontos na nossa unidade relativa, então estimaremos nossa atividade em quatro pontos. A próxima atividade pode ser estimada utilizando essas duas últimas como referência para a comparação. Seguiremos comparando a atividade seguinte com as últimas estimadas. Contanto que as referências sejam propagadas de estimativa em estimativa, as estimativas relativas se manterão consistentes.

Observamos que estimar de forma comparativa é muito mais fácil e intuitivo do que de forma absoluta. Por exemplo, estimar o peso de um cavalo e de um elefante não é simples, mas estabelecer qual dos dois pesa mais é trivial (DE TOLEDO, 2009). É também mais fácil estimar quantas vezes esse elefante é mais pesado que o cavalo.

Realizei várias vezes, em minhas aulas, um experimento que aprendi há muitos anos com um amigo na Índia. Em um primeiro exercício, os alunos estimam em metros a altura de um edifício visível pela janela. Os resultados

são sempre muito diferentes entre eles, distribuídos em uma faixa muito larga. Seguindo o experimento, defino um edifício como referência, com "1 ponto de altura", e peço que os alunos estimem um outro próximo, mais alto, em "pontos de altura". Na prática, eles estimarão quantas vezes mais alto o segundo prédio é em comparação com o primeiro. Com esse segundo exercício, obtemos estimativas mais próximas, em uma faixa muito mais curta do que a obtida com a estimativa absoluta. Portanto, com uma precisão muito maior.

Há vantagens adicionais ao estimarmos de forma relativa. Ao deixarmos de expressar as estimativas em dias ou horas absolutos, desacoplamos a execução de uma atividade de uma data ou hora exata de entrega. Dessa forma, a abordagem de estimativas relativas evita, ao mesmo tempo, os efeitos da Lei de Parkinson e do tratamento incorreto da estimativa como compromisso ou obrigação (DE TOLEDO, 2009).

O que são Story Points?

Story Point é a unidade mais usada por equipes que utilizam o Scrum. É uma unidade relativa de tempo criada pelos Desenvolvedores e por eles utilizada para estimar o tempo necessário para implementarem, em conjunto, um item de trabalho. No caso de Scrum, essa implementação se refere a transformar um item do Product Backlog em pronto, de acordo com a Definição de Pronto.

Em vez de utilizar horas ou dias, os Desenvolvedores criam sua própria escala relativa a partir de um ou mais itens de referência selecionados. Com isso, passam a estimar os itens de trabalho dentro dessa escala, comparando-os primeiramente com esses itens de referência e, depois, com os últimos itens estimados.

Escalas para Story Points

As escalas mais utilizadas por times ágeis para representar os Story Points são a sequência de Fibonacci e a chamada "tamanhos de camisa" (*T-Shirt Sizing*), embora escalas simples de um a dez ou potências de dois podem funcionar bem.

Na sequência de Fibonacci, cada número é igual à soma dos dois números anteriores, começando com zero e um. Temos, portanto: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 e assim por diante. Para efeitos práticos, usamos a escala da seguinte forma:

1 -- 2 -- 3 -- 5 -- 8 -- 13 -- 21 -- 34 -- 55 -- 89...

Podemos utilizar uma versão adaptada, criada por Mike Cohn (2005) para maior simplicidade. Essa escala, ao arredondar os números mais altos, diminuímos neles a sensação de precisão:

1 -- 2 -- 3 -- 5 -- 8 -- 13 -- 20 -- 40 -- 100

Nessa versão, 40 e 100 simplesmente significam que um item de trabalho é muito grande e grande demais.

Para os tamanhos de camisa (*T-Shirt Sizing*), a escala é a seguinte:

PP -- P -- M -- G -- GG -- XG

Podemos estipular que **P** significa mais ou menos o dobro do tempo que **PP**, **M** significa mais ou menos o dobro do tempo que **P** e mais ou menos quatro vezes o tempo que

PP, e assim por diante. Na maioria dos casos, uma escala menor e mais simples é suficiente:

P -- M -- G

Outras escalas que passem a ideia de tamanho relativo podem funcionar bem. Eu já vi times utilizando raças de cachorro (do chihuahua ao dogue alemão, por exemplo), tamanhos de pedras (do pedregulho à montanha) e muitas outras alternativas criativas.

Criação da escala a partir de referências

Uma vez selecionada a escala, escolhemos um ou mais itens como referências para criarmos os outros pontos dessa escala.

Os Desenvolvedores podem escolher como referência de unidade um dos itens do Product Backlog que eles concordem que é o menor, mais simples, mais conhecido e com menos risco envolvido em sua implementação naquele momento. Esse item de referência provavelmente está próximo ao topo do Product Backlog, onde estão os itens menores e mais detalhados.

Dessa forma, o item de referência passa a ter a estimativa de um Story Point ou "P" ou "chihuahua", por exemplo, dependendo da escala escolhida. Assim, os próximos itens a serem estimados serão comparados com essa referência. Por exemplo, um item que, na estimativa dos Desenvolvedores, demande cerca de duas vezes o tempo para realizar o item de referência, receberá dois Story Points. Em vez de um ponto, muitos times preferem atribuir a esse item de referência o segundo ponto da escala (dois Story Points, por exemplo) para que no futuro haja espaço para itens menores.

Alternativamente, podemos conseguir melhores resultados ao escolhermos um ou dois itens de referência de tamanho médio em vez de apenas um pequeno. Atribuímos a esse ou esses itens 5 ou 8 Story Points (ou "M", por exemplo). Ou podemos escolher um item pequeno e um item grande, atribuindo a eles 2 e 13 Story Points (ou "P" e "G", respectivamente). Os próximos itens a serem estimados são comparados com esses dois itens de referência, em uma triangulação.

Uma vez que criamos a referência ou as referências, partimos do alto do Product Backlog estimando o primeiro item, comparando-o com o item ou itens de referência. Digamos que os Desenvolvedores escolheram um item como referência e atribuíram dois Story Points a ele. Assim, se eles julgam, por exemplo, que o primeiro item do Product Backlog levará mais que o dobro do tempo que o item de referência, esse item é então estimado em cinco Story Points (ou "M", por exemplo). Para o item seguinte, fazemos uma triangulação com os dois itens já pontuados, ou seja, comparando com eles esse item seguinte.

Por consequência, se os Desenvolvedores julgam que a implementação desse item seguinte vai demorar bem mais que a do item de referência e um pouco mais que a do outro item já estimado, podem estimar esse item seguinte em oito pontos (ou "G", por exemplo).

A partir daí, eles seguem estimando os itens a partir do alto do Product Backlog, Sprint após Sprint, sempre os comparando com os últimos itens estimados. É importante ressaltar que o item ou itens de referência são escolhidos apenas uma vez, geralmente antes do princípio de todo o trabalho de implementação, e não Sprint após Sprint.

Por que Fibonacci?

O uso da sequência de Fibonacci como escala para as estimativas traz uma vantagem interessante. Cada número da escala carrega em si um intervalo de confiança, que é o intervalo entre o número anterior e o posterior da escala. Com isso, o tratamento da precisão e acurácia da estimativa já está embutido nessa escala.

Por exemplo, ao estimarmos um item do Product Backlog em 3 Story Points, implicitamente estamos afirmando que a duração desse item deverá ser algo entre 2 e 5 Story Points - um intervalo de confiança de tamanho (5 - 2) = 3. No entanto, ao estimarmos outro item como 8 Story Points, implicitamente estamos afirmando que a duração desse item deverá ser algo entre 5 e 13 Story Points - um intervalo de confiança de tamanho (13 - 5) = 8 (veja a figura a seguir).

Figura 28.3: Fibonacci, precisão e acurácia

Ao usarmos a sequência de Fibonacci, podemos verificar que, quanto maior for a estimativa de um item, maior será seu intervalo de confiança e, assim, maior será a incerteza associada a ela e menor será sua precisão. De fato, sabemos intuitivamente que, para termos uma maior possibilidade de acerto, a precisão que podemos utilizar para estimar itens menores é maior (menor intervalo) do que para itens maiores (maior intervalo).

Como vimos anteriormente, quanto menor for a precisão de uma estimativa (maior intervalo), mais chance a estimativa tem de estar correta. Portanto, a sequência de

Fibonacci automaticamente reduz a precisão de uma estimativa quanto maior ela for (maior intervalo), aumentando assim sua possibilidade de acerto, ou seja, de acurácia.

Story Points: complexidade, risco, esforço ou tamanho?

Existe muita confusão acerca do significado dos Story Points. Eles representam a complexidade de uma funcionalidade? Talvez com um componente do risco envolvido em implementá-la, derivado da incerteza? Ou será que representam o esforço necessário para implementar a funcionalidade? Ou talvez o tamanho? Mas então o que exatamente significa tamanho de uma funcionalidade?

Story Points foram criados por alguns dos inventores do Extreme Programming (JEFFRIES et al., 2000). Seu propósito era o de burlar estimativas em unidades tradicionais de tempo, como horas ou dias, pois a gerência equivocadamente entendia essas estimativas como promessas, e não como estimativas. Era comum eles ouvirem: *se você disse que ficaria pronto em três dias, por que não está pronto três dias depois?*.

Assim, Story Points foram inicialmente criados para que pudessem continuar realizando estimativas necessárias para o planejamento, mas ofuscando o tempo para evitar o compromisso com uma alta precisão e, dessa forma, evitar a pressão da gerência sobre essas estimativas.

Story Points representam, nada mais, nada menos, que tempo.

Utilizamos Story Points justamente para estimarmos quanto trabalho é possível ser feito em um determinado período de tempo (um Sprint, por exemplo) ou quanto tempo os Desenvolvedores vão demorar para realizar um determinado objetivo.

É claro que, para isso, levamos em consideração parâmetros como complexidade, incerteza e risco, e sabemos que custo e esforço são derivados do tempo. Mas Story Points representam, simplesmente, tempo. Esse é, porém, um tempo que não é medido em horas ou dias, mas sim em uma escala relativa criada pelos próprios Desenvolvedores. Assim, *n* Story Points nada mais significam que *n* dias multiplicados por algum fator.

É importante ressaltar que, ao utilizar Story Points, buscamos justamente mascarar e evitar expressar a estimativa em dias ou horas. Logo, não faz sentido realizar sistematicamente qualquer tradução dos pontos em dias ou horas (por exemplo, *para nosso time, 3 Story Points significam dois dias de trabalho*) ou explicitar o fator de conversão de um para outro.

Story Points: tempo ou esforço?

De forma geral, em vez de estimarmos o tempo, real ou ideal, podemos estimar o esforço necessário para a realização de uma atividade. Assim, em vez de tempo, passamos a tratar de pessoas X tempo, substituindo "horas" ou "dias" por "pessoas-horas" ou "pessoas-dias". Em outras palavras, esforço significa o número de dias ou horas, necessários para completar a atividade, que devem ser distribuídos entre as pessoas executando o trabalho.

Assim, 10 pessoas-horas podem significar tanto uma pessoa realizando o item em dez horas, como duas

pessoas realizando o item em cinco horas ou quatro pessoas o realizando em duas horas e meia.

Muitos na comunidade ágil defendem que Story Points devem ser definidos como esforço, ao invés de tempo. No entanto, posso argumentar que o uso de Story Points como tempo ou como esforço trazem o mesmo efeito. Ou posso ir além e afirmar que não faz sentido defini-los como esforço. Isso se dá pelas seguintes razões:

1. uma estimativa com Story Points trata do tempo de trabalho dos Desenvolvedores em conjunto, e não de algum deles em específico. Poderíamos, portanto, apenas falar do esforço coletivo, e não individual;

2. a estimativa com Story Points somente faz sentido se as pessoas que realizarão o trabalho são sempre as mesmas. Se houver qualquer mudança no conjunto dos Desenvolvedores do Time de Scrum, a sua definição de Story Points perde o sentido e deve ser refeita. Por essa razão, o componente "pessoas" do esforço permanece constante e pode ser desconsiderado, restando apenas o "tempo";

3. esforço, ao tratarmos de trabalho de criação de um produto, não pode ser pensado como apenas braçal. Há também o esforço mental e o esforço de interação e colaboração. Ambos são não lineares, como já falamos anteriormente. Por essa razão, o esforço coletivo não pode ser pensado como a soma dos esforços individuais. Com isso, a própria definição de esforço em termos de "tempo" e "pessoas" perde sentido nesse contexto.

Seguindo essas justificativas e para efeitos de simplicidade, refiro-me a Story Points neste capítulo apenas como tempo, e não como esforço.

28.4 Como realizar a estimativa?

Qualquer que seja o método usado para realizar as estimativas, é importante que os princípios básicos a seguir sejam respeitados:

- as estimativas dos itens do Product Backlog somente podem ser realizadas pelos Desenvolvedores. Elas não devem ser realizadas pelo Product Owner, pelo Scrum Master ou por quaisquer outras partes interessadas. Acreditamos que as melhores estimativas são aquelas feitas pelas próprias pessoas que realizam o trabalho a ser estimado. Essa premissa, apesar de representar o senso comum, é muitas vezes ignorada em um grande número de contextos, nos quais frequentemente os gerentes criam as estimativas para suas equipes;
- as estimativas são definidas por um consenso de todos os Desenvolvedores. Como cada item do Product Backlog deve ser implementado pelos Desenvolvedores em conjunto, e não individualmente, a estimativa de um item do Product Backlog não pode pertencer a apenas um ou a alguns poucos indivíduos. Ela deve ser fruto da colaboração entre todos;
- o processo de definição das estimativas não pode ser custoso nem demorado. Se esse processo não for rápido e objetivo, o tempo despendido nele será relevante diante do tempo necessário para a realização do trabalho, de forma que a própria atividade de estimar necessitará de estimativas. Nada disso faz o menor sentido.

Os Desenvolvedores se reúnem para a realização de estimativas de itens do Product Backlog. Esse encontro,

em geral, se dá em sessões de Refinamento do Product Backlog ou, no mais tardar, na reunião de Sprint Planning (veja os capítulos *Refinamento do Product Backlog (adicional)* e *Sprint Planning*). O Product Owner, presente durante a atividade, pode esclarecer dúvidas que inevitavelmente surgirão quanto aos itens, enquanto que o Scrum Master atua como facilitador. Nem Product Owner, nem Scrum Master participarão da escolha dos valores para as estimativas.

Não recomendamos gastar um tempo demasiadamente longo com a estimativa de cada item. O processo de estimar é idealmente rápido e objetivo, de forma que o tempo total gasto não seja relevante com relação ao tempo de trabalho, como já mencionei anteriormente. Além disso, a partir de algum momento, mais conversas sobre um item já não trarão melhores estimativas. Consideramos desperdício, por exemplo, discutir profundamente e em detalhes como o item será implementado.

O uso de Story Points com a sequência de Fibonacci como escala de estimativas facilita significativamente o processo dos Desenvolvedores chegarem a um consenso. Já expliquei neste capítulo que, como a escala não é linear, quanto maior é o item, menor é o número de escolhas possíveis para sua estimativa. Assim, não desperdiçam tempo com estimativas de mais baixa precisão. Por exemplo, faz mais sentido gastarmos mais tempo discutindo se um item do Product Backlog deve ter a estimativa de 2 ou 3 Story Points (mais alta precisão), enquanto que, com a escala de Fibonacci, não é possível e nem faria sentido discutir se outro item deve ter 15 ou 16 Story Points. Ou o item será estimado em 13, ou em 21 pontos (mais baixa precisão).

Planning Poker

O Planning Poker é a técnica mais conhecida e utilizada por times ágeis para chegarem a um consenso quanto às estimativas de itens a serem implementados. Foi criado por James Greening (2002), um dos signatários do Manifesto Ágil, e satisfaz os princípios descritos anteriormente se usado corretamente. Apesar de sua popularidade, o Planning Poker não é parte integrante do Scrum e seu uso é opcional.

Para utilizar o Planning Poker, cada Desenvolvedor deve possuir um conjunto de cartas com as alternativas possíveis da escala utilizada para as estimativas. Dessa forma, se a sequência de Fibonacci adaptada for a escala escolhida, cada um possuirá as cartas 1, 2, 3, 5, 8, 13, 20, 40, 100. Existem também diversos aplicativos disponíveis para celulares com essa finalidade.

Algum membro do Time de Scrum lê o próximo item do Product Backlog a ser estimado. Os Desenvolvedores refletem brevemente sobre esse item e, caso seja necessário, esclarecem dúvidas com o Product Owner.

Cada Desenvolvedor então escolhe a carta com sua estimativa para todo o trabalho necessário para implementar o item, comparando-o mentalmente com estimativas realizadas anteriormente. Cada um pensa no tempo de trabalho dos Desenvolvedores em conjunto, e não apenas no tempo que acredita que tomará seu próprio trabalho.

É importante que o valor escolhido não seja imediatamente discutido nem mostrado aos outros para não influenciar seu julgamento inicial, de forma que não ancorem sua escolha ao que já foi mostrado.

Em seguida, cada um coloca sua escolha sobre a mesa, de forma que todos possam vê-la (veja a figura a seguir). Dificilmente um consenso será atingido nessa primeira rodada de votação. As diferenças, na realidade, vão gerar conversas que levarão a uma melhor compreensão conjunta do item, o que também contribui para a identificação de problemas mais cedo. Uma ou poucas mais rodadas de votação são então realizadas, sempre facilitadas pelo Scrum Master, até que um consenso seja atingido. A necessidade de um consenso faz com que os Desenvolvedores compartilhem a responsabilidade sobre a estimativa.

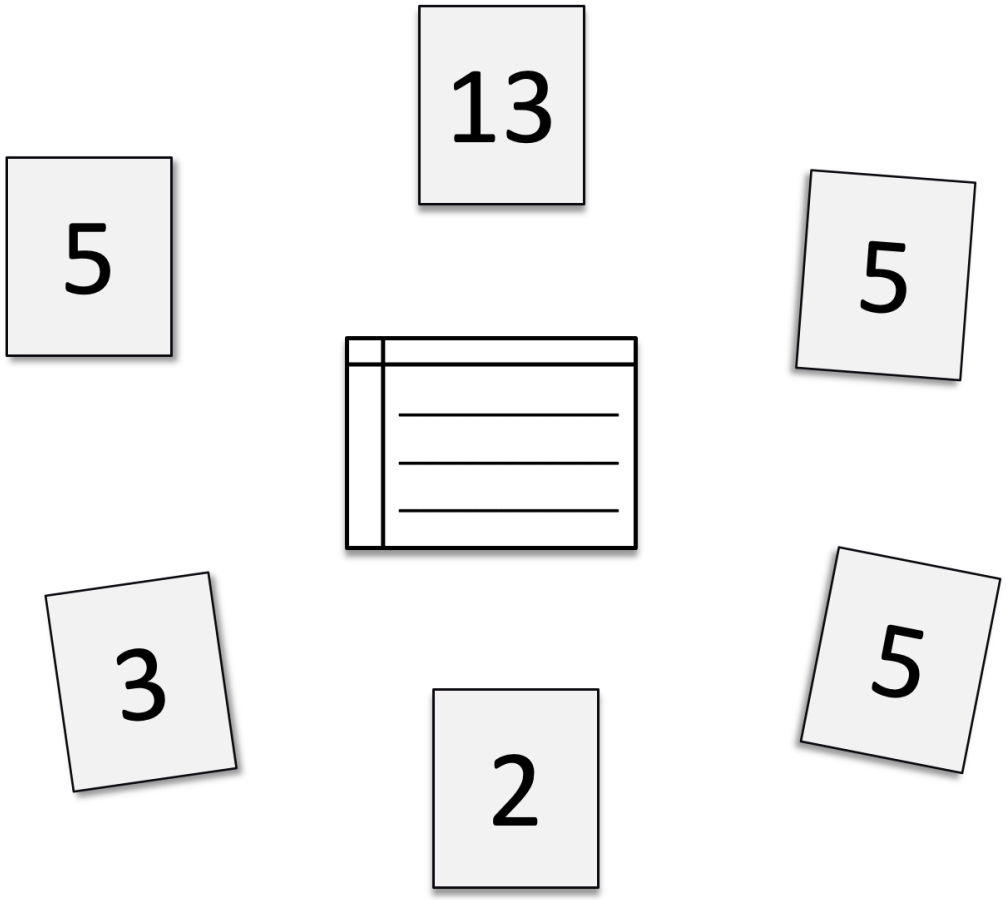


Figura 28.4: Planning Poker

Após cada rodada de votação, as pessoas que deram a estimativa mais alta e a mais baixa começam a conversa, justificando o porquê de suas escolhas. O fato de terem que justificar suas estimativas leva os Desenvolvedores a realmente se engajarem em buscar estimativas que lhes façam sentido.

Um Desenvolvedor que estimou o item em dois Story Points pode dizer, por exemplo: *esse item é fácil, já fiz algo muito parecido antes*. O outro Desenvolvedor que escolheu oito Story Points pode, então, dizer: *sim, mas você está se esquecendo que testar isso é bem complicado e que é necessário construir um ambiente para esse tipo de teste. Mas, como você já fez algo parecido, talvez demore menos do que pensei*. Em uma segunda rodada, eles terão estimativas mais próximas (três e cinco Story Points, provavelmente), assim como o resto dos Desenvolvedores, e o consenso poderá até mesmo ser atingido sem uma nova votação.

Pode ser uma boa ideia determinarmos o tempo máximo para cada discussão, estabelecendo um *timebox*. Além disso, esperamos que o consenso seja atingido com não mais que duas ou três rodadas de votação. Caso ocorram mais rodadas que isso, esse pode ser um indicador da necessidade de uma atuação mais efetiva do facilitador. No caso de impasses, os Desenvolvedores podem ter seus próprios acordos de como proceder. Eles podem, por exemplo, determinar nesses casos a escolha da estimativa mais alta ou, talvez, de uma intermediária.

Planning Poker: a atividade de Planning Poker se resume aos seguintes passos:

1. alguém — Scrum Master, Product Owner ou Desenvolvedor — lê o próximo item a ser estimado. O Scrum Master facilita a sessão;
2. os Desenvolvedores rapidamente refletem sobre esse item e esclarecem dúvidas com o Product Owner;
3. cada Desenvolvedor então escolhe uma carta com a sua estimativa para o trabalho necessário para que implementem o item em conjunto, sem mostrar a carta imediatamente;
4. todos mostram as suas cartas, colocando-as sobre a mesa ao mesmo tempo. Dessa forma, evitamos a ancoragem, isso é, que aceitem sem reflexão a influência dos outros;
5. os Desenvolvedores que deram a estimativa mais alta e a mais baixa conversam diante dos outros e defendem suas estimativas, justificando o porquê dos valores escolhidos;
6. uma ou mais rodadas de escolha de estimativas para o item são realizadas, até os Desenvolvedores chegarem a um consenso.

28.5 Velocidade

Conhecer a Velocidade pode ajudar aos Desenvolvedores e ao Product Owner na tarefa de planejar.

Digamos, por exemplo, que os Desenvolvedores tenham implementado, nas últimas três Sprints, itens cujas estimativas somadas correspondem a 30, 32 e 29 pontos, respectivamente. Esses pontos podem ser Story Points ou qualquer outra unidade utilizada pelos Desenvolvedores para estimarem o seu trabalho. Podemos tirar a média desses valores e dizer que sua Velocidade, neste momento, é de aproximadamente 30 pontos por Sprint (veja a figura mais adiante).

A Velocidade é, portanto, a taxa média de pontos dos itens implementados por Sprint. Podemos esperar que os Desenvolvedores implementem itens com estimativas correspondentes a algo próximo desse valor no próximo ou nos próximos Sprints ou, ao menos, sabemos que essa é a nossa melhor previsão possível.

O primeiro Sprint se inicia, é claro, sem nenhuma referência de Velocidade. A partir do segundo ou terceiro Sprints, no entanto, já há sentido em utilizarmos para seu cálculo os resultados de Sprints anteriores, ainda que a média não tenha se estabilizado. A partir de um certo ponto, esperamos uma Velocidade relativamente estável dali em diante, dado que os Desenvolvedores se mantenham os mesmos e, naturalmente, que isso também ocorra com a duração dos Sprints.

No entanto, é considerada uma boa prática usarmos apenas os últimos três Sprints para calcular a Velocidade, em vez de todos os valores históricos. Recomendo também, para cada Sprint, computar para a Velocidade apenas as estimativas dos itens que chegarem prontos ao final do Sprint, sem considerar trabalhos parciais. Com

esses cuidados, produzimos um valor mais realista e atualizado para a Velocidade.

O uso mais comum para a Velocidade é o de servir como referência no planejamento do Sprint. No nosso exemplo anterior, durante a reunião de Sprint Planning, os Desenvolvedores limitarão seu Sprint Backlog a itens cujas estimativas não superem em torno dos 30 pontos, Velocidade que calculamos. O Product Owner, por sua vez, esperará algo próximo dos 30 pontos nesse Sprint Backlog.

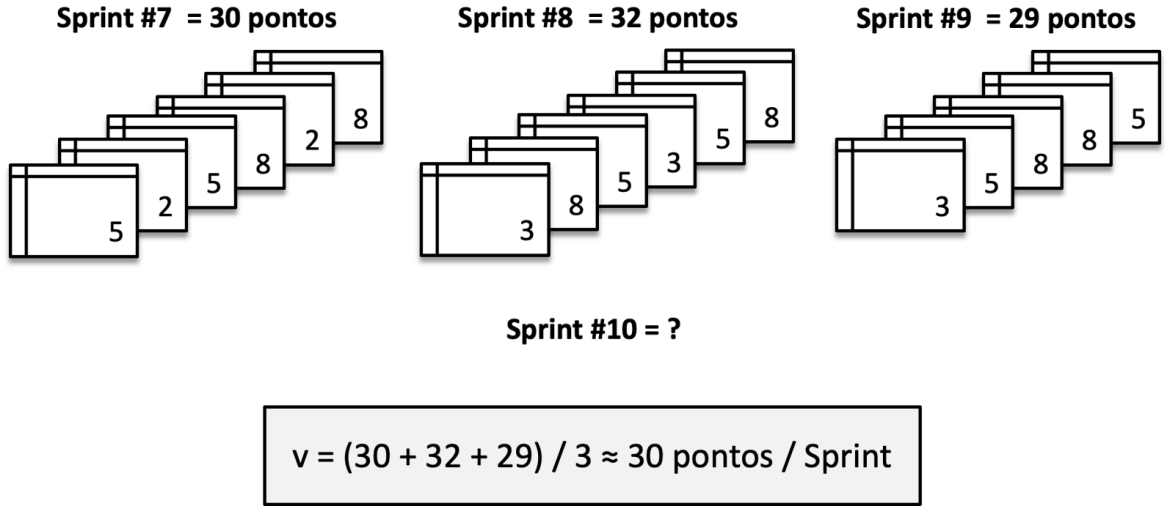


Figura 28.5: Medir a Velocidade pode aumentar a previsibilidade

A Velocidade também pode ser útil para o planejamento do trabalho para uma entrega a ser realizada após mais de um Sprint, mesmo que seja necessária uma extrapolação de baixa precisão. Tendo estimativas para os itens do Product Backlog e conhecendo a Velocidade, podemos planejar qual objetivo de negócios poderá ser realizado dado um prazo ou um número de Sprints. Ou, dado um objetivo de negócios, podemos projetar em quantos Sprints ele poderá ser realizado. Essa técnica é geralmente usada na reunião de Release Planning (veja, no capítulo *Release Planning*, a seção *Entrega por plano*).

28.6 No estimates: estimativas são desperdício

Não tenho certeza se eu inventei os Story Points, mas se eu o fiz, sinto muito agora. -- Ron Jeffries, 2017, em resposta a um tweet.

Curiosamente, os próprios criadores dos Story Points não mais os recomendam. Devido a seu uso exageradamente incorreto, eles preferem práticas alternativas. O fato é que existem opções importantes para facilitar o planejamento, que podem substituir completamente a abordagem de estimarmos os itens do Product Backlog individualmente.

O movimento *no estimates* (ou "sem estimativas"), por exemplo, ganhou muito espaço nos últimos anos. O princípio básico de trabalharmos sem estimativas reside na realidade de que, apesar de que as estimativas nos trazem uma sensação de segurança, a sua acurácia é muito limitada no trabalho de criação de produtos e, com isso, elas têm uma distância muito grande da realidade. Há uma grande desconexão entre o que cremos que vai acontecer e o que realmente acontece, o que nos mostra que essa segurança é falsa. Além disso, como vimos anteriormente, o comportamento das pessoas se modifica de forma disfuncional devido à simples existência de uma estimativa (lembre-se, por exemplo, da Lei de Parkinson). A conclusão lógica é que estimativas constituem desperdício.

A partir desses conceitos, existem diferentes abordagens para o trabalho sem estimativas. A mais comum estabelece que, para viabilizar o planejamento sem estimativas, devemos utilizar a boa prática de manter os itens do alto do Product Backlog consistentemente

pequenos, ou seja, requerendo cada um apenas alguns poucos dias de trabalho. O tamanho pequeno e a regularidade consequente trazem uma maior previsibilidade: podemos determinar a Velocidade simplesmente calculando a média da quantidade de itens prontos nos últimos Sprints, e usar esse valor para determinar a quantidade de itens que entrará no Sprint seguinte (veja a figura a seguir).

Por exemplo, os Desenvolvedores de um Time de Scrum que tenham implementado 6, 7 e 5 itens nos últimos três Sprints têm a Velocidade de 6 itens por Sprint (a média), e escolherão essa quantidade de itens na reunião de Sprint Planning seguinte.

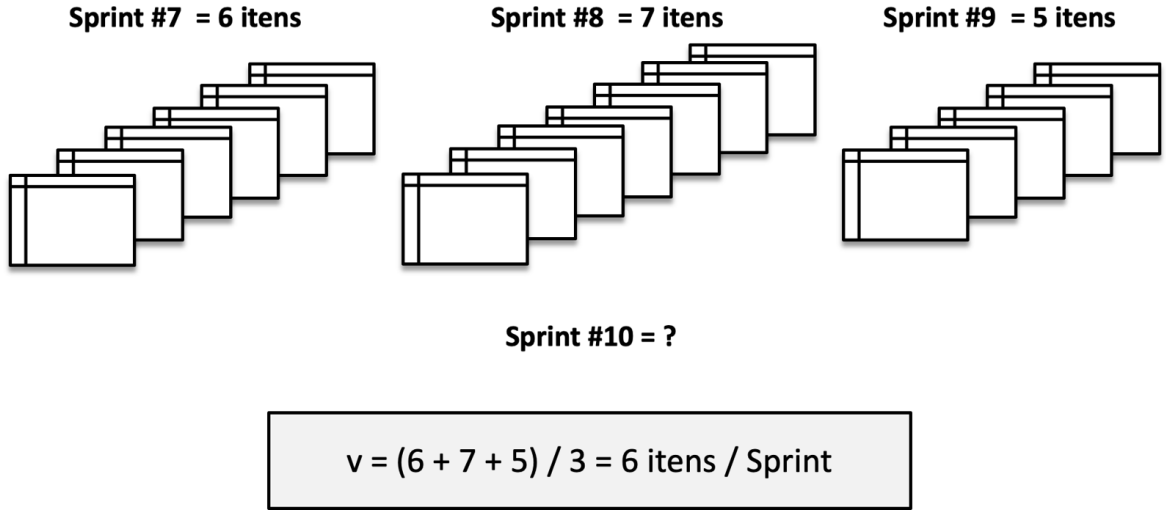


Figura 28.6: Medindo a Velocidade sem estimativas

Podemos considerar também que o planejamento para um horizonte muito maior que um ou alguns poucos Sprints possui uma precisão tão pequena que não vale a pena ser realizado. Logo, evitamos planejamentos detalhados para longas entregas.

Há equipes que sequer calculam sua Velocidade. Eles consideram que trabalhar com qualquer conceito

relacionado a estimativas é desperdício. Apenas com seu conhecimento e experiência são capazes de estabelecer que objetivo de valor acreditam que alcançarão naquele Sprint, trabalhando sempre com itens pequenos, do mais importante para o menos importante. Ao trabalharem para realizar objetivos de valor, seu foco se desvia do plano estabelecido para o valor a ser gerado, eliminando a importância da pontuação estimada.

Confesso que me incluo no grupo dos que não mais utilizam Story Points ou qualquer tipo de estimativa formal no meu trabalho.

Capítulo 29

Burndown e Burnup: acompanhando o trabalho

Conteúdo

1. O que são os gráficos de acompanhamento do trabalho?
2. Gráfico de Release Burndown.
    - O que é o Gráfico de Release Burndown?
    - Como é o Gráfico de Release Burndown?
3. Gráfico de Release Burnup.
    - O que é o Gráfico de Release Burnup?
    - Como é o Gráfico de Release Burnup?
4. Gráfico de Sprint Burndown.
    - O que é o Gráfico de Sprint Burndown?
    - Como é o Gráfico de Sprint Burndown?
    - Linha ideal.
        - O que é linha ideal?
        - Comportamento da linha ideal.
        - Críticas à linha ideal.

29.1 O que são os gráficos de acompanhamento do trabalho?

Gráficos de acompanhamento do trabalho são ferramentas para a visualização do progresso do cumprimento de uma quantidade mensurável ou estimada de trabalho em um determinado tempo. Eles servem para criar transparência de uma forma simples,

ajudando a rapidamente identificar problemas nesse progresso e, assim, reduzir riscos.

Os Gráficos de Burndown são os mais utilizados por Times de Scrum com esse propósito. A ideia de *burndown* (queima) é que o trabalho previsto é "queimado" em direção a um momento futuro fixo, formando um gráfico de inclinação negativa, ou seja, para baixo.

Os Gráficos de Burndown envolvem, portanto, alguma unidade de trabalho no eixo vertical, que representa o trabalho restante, e alguma unidade de tempo no eixo horizontal, que representa o tempo cumprido (veja a figura a seguir). Unidades comumente usadas para trabalho restante incluem número de tarefas restantes, quantidade de horas estimadas para as tarefas restantes, número de itens restantes e quantidade de Story Points estimados para os itens restantes. Unidades comuns para tempo incluem dias, semanas e Sprints, entre outros.

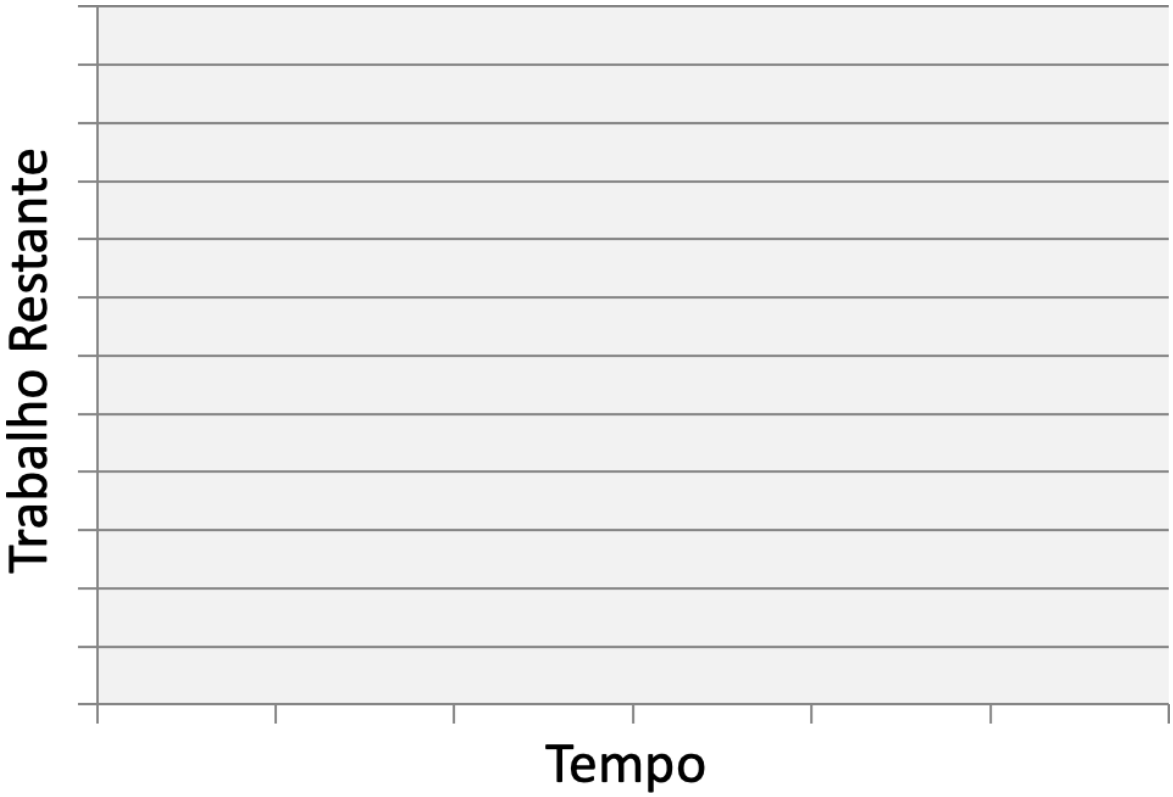


Figura 29.1: Eixos do Gráfico de Burndown: trabalho restante x tempo

Os Gráficos de Release Burndown e de Sprint Burndown são os dois tipos de Gráficos de Burndown mais utilizados por Times de Scrum.

Curiosidade: Gráficos de Burndown são artefatos do Scrum?

Os Gráficos de Release Burndown e de Sprint Burndown já fizeram parte do framework Scrum. Mas, ao contrário do que muitos pensam, já há muito tempo foram retirados da definição oficial.

De acordo com a Agile Alliance, o gráfico de Burndown foi inventado por Ken Schwaber no ano 2000 (SCHWABER, 2000, apud BURNDOWN CHART, 2017), um dos criadores do Scrum. Não encontrei qualquer menção ao gráfico anterior a essa data.

O uso do Gráfico de Sprint Burndown é explicado em um artigo de Jeff Sutherland (SUTHERLAND, 2001), onde é chamado de Gráfico de Burndown, e no primeiro livro de Scrum (SCHWABER; BEEDLE, 2002), onde é chamado de Gráfico de Backlog. O Gráfico de Release Burndown é descrito, ainda de forma primitiva, nesse mesmo livro, onde se refere ao projeto como um todo e é chamado de Plano do Projeto.

Ambos apareceram como parte do framework nas duas primeiras edições do guia oficial (SCHWABER, 2009 e SCHWABER; SUTHERLAND, 2010). No entanto, foram retirados do Scrum já na edição seguinte (SCHWABER; SUTHERLAND, 2011). Em seu lugar ficaram as recomendações de monitorarmos o trabalho no Sprint e de monitorarmos o progresso do trabalho em direção a um objetivo, ambas substituídas na edição de 2020 pela simples indicação da existência dos Gráficos de Burndown, juntamente a Gráficos de Burnup e Diagramas de Fluxo Cumulativo, como úteis para prevermos o progresso (SCHWABER; SUTHERLAND, 2020).

Os Gráficos de Burnup, diferentemente dos de Burndown, mostram duas curvas: a quantidade de trabalho já realizado no tempo e o trabalho total previsto a ser realizado. A quantidade de trabalho já realizado é uma curva ascendente no tempo, enquanto que o trabalho total previsto se modifica conforme o plano é atualizado.

O Gráfico de Burnup mais usado por Times de Scrum é o Gráfico de Release Burnup.

Nenhum dos gráficos de acompanhamento do trabalho é parte do framework Scrum, e assim seu uso é opcional.

29.2 Gráfico de Release Burndown

O que é o Gráfico de Release Burndown?

O Release Burndown é um gráfico mantido pelo Product Owner e utilizado tanto pelo Product Owner quanto pelos Desenvolvedores do produto para monitorar o progresso do trabalho em direção a uma entrega planejada. O Gráfico de Release Burndown não é parte do framework Scrum, mas pode ser útil quando usamos um plano de entrega, geralmente estabelecido em uma reunião de Release Planning (veja no capítulo *Release Planning (adicional)*).

Ao avaliar o gráfico, o Product Owner pode decidir por mudanças de rumo, como alterações no escopo ou no objetivo da entrega, adiamento da entrega e até mesmo seu cancelamento.

O Gráfico de Release Burndown mostra a quantidade de trabalho restante previsto para uma entrega (eixo vertical), correspondente aos itens do Product Backlog selecionados para essa entrega (a soma de suas estimativas, por exemplo), em determinados momentos no tempo, em geral ao final de cada Sprint que faz parte dessa entrega (eixo horizontal). O Gráfico de Release Burndown é, assim, uma maneira rápida e prática de visualizarmos o andamento da entrega.

Nos exemplos desta seção, usarei Story Points como unidade de "trabalho restante previsto". Os Story Points são atribuídos pelos Desenvolvedores aos itens do Product Backlog selecionados para a entrega. Entretanto, existem diversas outras opções (veja, no capítulo *Story Points: estimando o trabalho*, a seção *Unidades para as estimativas*).

Para unidade de "tempo", podemos utilizar como marcos o final de cada Sprint previsto para a entrega. Nesse caso, marcamos cada ponto do gráfico com a soma das estimativas dos itens do Product Backlog restantes naquele momento, entre os selecionados para a entrega. Alternativamente, podemos utilizar como marcos de tempo o final de cada uma das semanas previstas até a entrega.

A figura a seguir mostra um exemplo de Release Burndown antes do início do último Sprint de uma entrega. Nesse exemplo, a quantidade inicial de trabalho é de 235 Story Points, a serem queimados em seis Sprints.

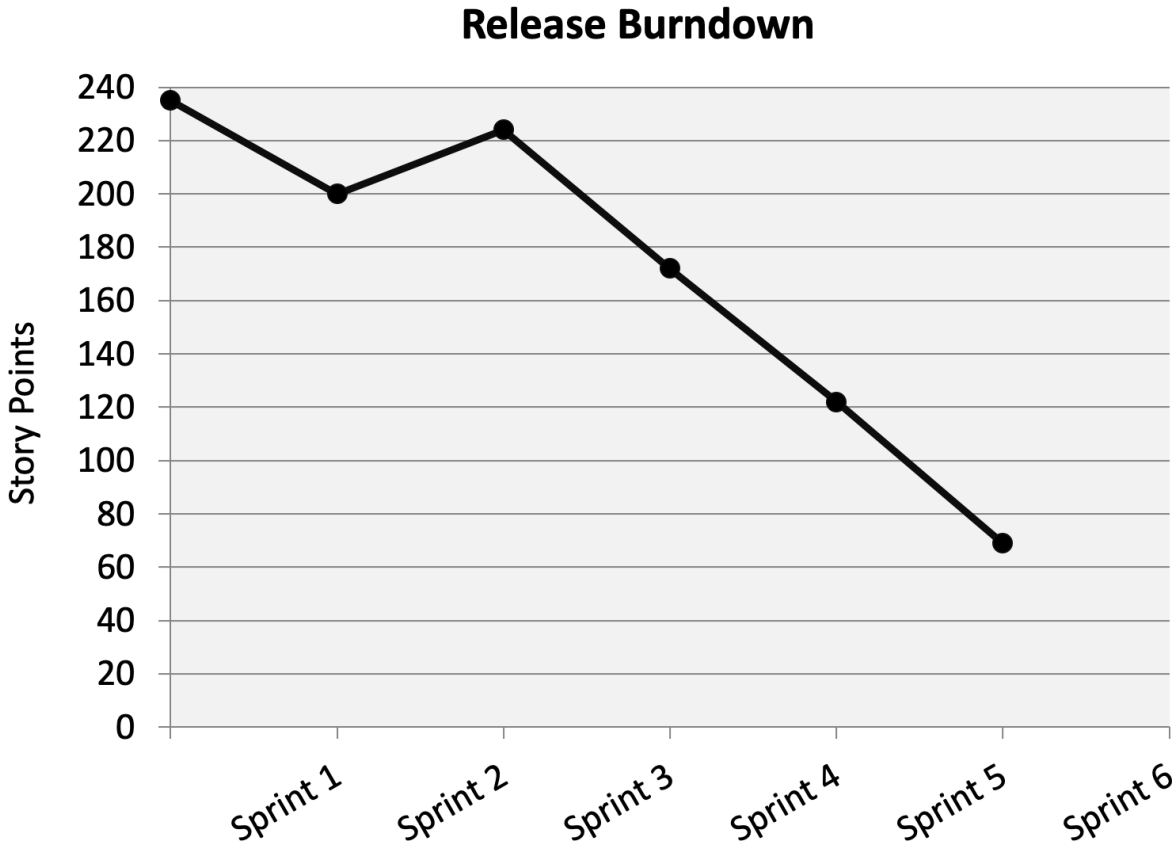


Figura 29.2: Um exemplo de Release Burndown ao final do penúltimo Sprint correspondente àquela entrega

No exemplo da figura anterior, podemos verificar pelo gráfico que, nos Sprints 3, 4 e 5, os Desenvolvedores "queimaram" 52, 50 e 53 Story Points respectivamente, atingindo quase uma constância.

O Gráfico de Release Burndown é geralmente criado em uma reunião de Release Planning. Ao final de cada Sprint ou de cada semana, ele é atualizado com o valor da quantidade restante de trabalho, naquele momento, previsto para a entrega programada.

Como é o Gráfico de Release Burndown?

O exemplo da figura a seguir mostra um Gráfico de Release Burndown que acaba de ser atualizado logo após o final do terceiro Sprint correspondente à entrega, de um total de seis previstos.

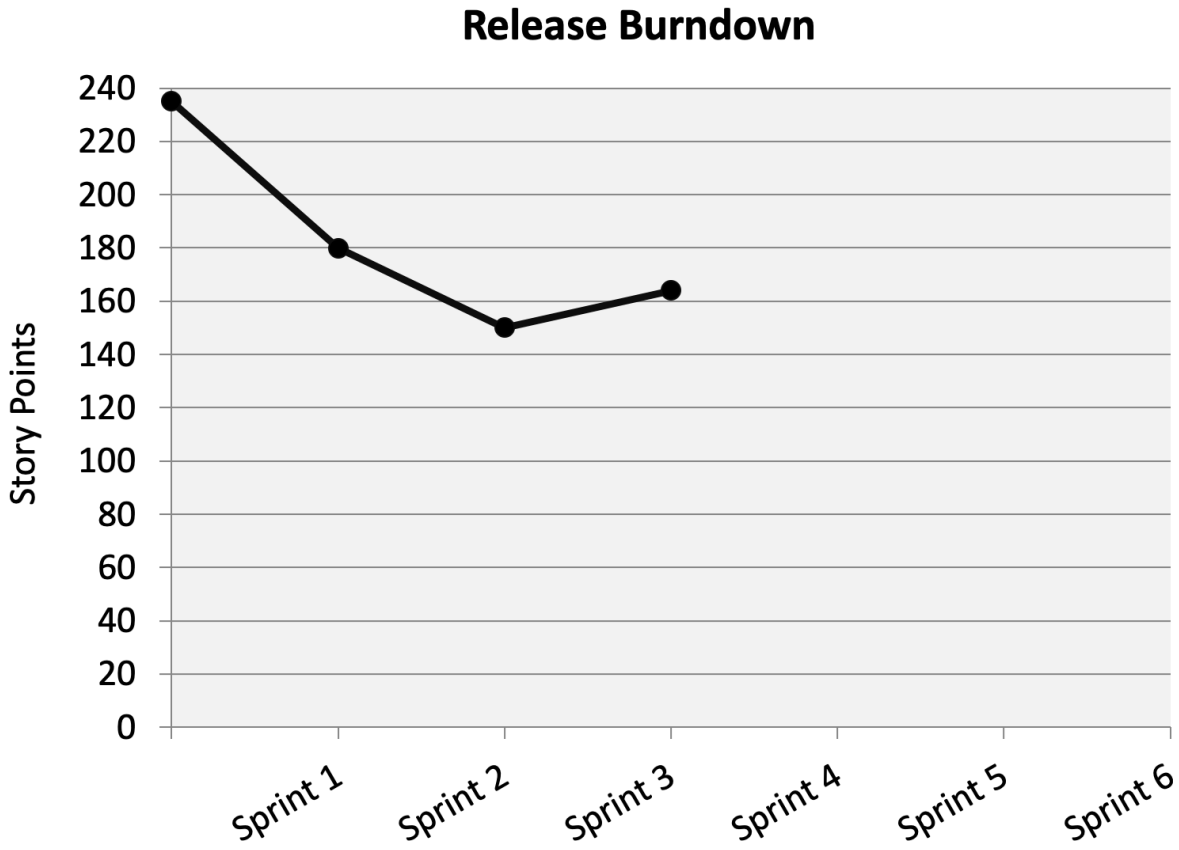


Figura 29.3: Release Burndown com inclinação para cima no final do terceiro Sprint

Observe que a inclinação súbita para cima, de 150 para 164 Story Points, serve como sinal de alerta, indicando que entrou no plano mais trabalho do que o que foi "queimado" durante o Sprint 3. Essa subida pode acontecer, por exemplo, quando há mudanças no escopo planejado para a entrega, ou com a inclusão de novos itens no Product Backlog, ou quando há mudanças nas estimativas dos itens restantes do Product Backlog previstos para a entrega. A subida pode ainda se acentuar quando parte significativa do que estava previsto para o Sprint não for implementada.

Uma forma prática de verificarmos o andamento do trabalho para a entrega é traçar uma reta partindo do ponto de algum dos marcos de tempo para trás, conectá-

la ao ponto correspondente ao momento atual e prolongá-la até cruzar com o eixo horizontal. No exemplo, partimos de três Sprints para trás, que no exemplo é o ponto inicial do gráfico em (**[Início]**, **235**), cruzamos o ponto atual, que é (**[final do Sprint 3]**, **164**), e prolongamos até cruzar o que parece ser uns quatro Sprints após o último Sprint previsto.

Esse ponto no eixo horizontal é o momento em que projetamos que os Desenvolvedores terão "queimado" todos os Story Points planejados, mantendo a Velocidade atual. Assim, podemos ver no exemplo que, no momento atual, as maiores chances são que o trabalho estará longe de ser terminado ao final do Sprint 6. Esse cenário de possível atraso provavelmente levará o Time de Scrum a negociar sobre o futuro da entrega.

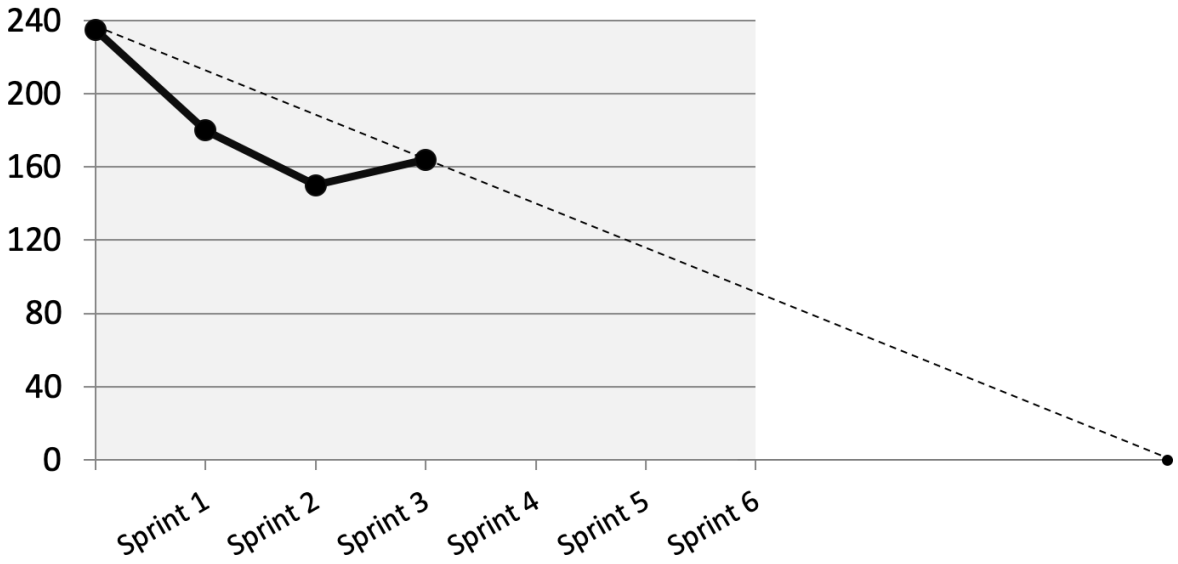


Figura 29.4: Verificando o andamento da entrega

É importante destacar que a projeção mostra apenas uma tendência e que não esperamos que os Desenvolvedores completem todos os itens definidos inicialmente para a entrega. Eles trabalham a partir dos

itens de ordem mais alta em direção aos de ordem mais baixa previstos para a entrega.

Esses itens vão evoluindo para itens menores e mais detalhados à medida que o trabalho segue e eles ganham importância com relação ao objetivo de negócios da entrega (veja a seção *Objetivo da Entrega* no capítulo *Compromisso: Objetivo do Produto*). Partes de menor importância com relação ao objetivo são movidas para baixo no Product Backlog. Novos itens também surgirão. Dessa forma, esperamos que os que sobrarem ao final do trabalho para a entrega sejam os menos importantes, assim aumentando as chances de que o objetivo definido para a entrega tenha sido realizado.

29.3 Gráfico de Release Burnup

O que é o Gráfico de Release Burnup?

O Release Burnup, assim como o Release Burndown, é um gráfico mantido pelo Product Owner e usado tanto pelo Product Owner quanto pelos Desenvolvedores para monitorar o progresso no trabalho em direção a uma entrega. O Gráfico de Release Burnup também não é parte do framework Scrum, mas pode ser útil quando usamos um plano de entrega, geralmente estabelecido em uma reunião de Release Planning (veja no capítulo *Release Planning (adicional)*).

O Gráfico de Release Burnup mostra duas linhas. Uma das linhas representa a quantidade de trabalho já realizado para essa entrega (apenas itens prontos de acordo com a Definição de Pronto). A outra linha representa a quantidade de trabalho total previsto para a

entrega, correspondente aos itens escolhidos para o plano de entrega.

Enquanto os Desenvolvedores trabalham nos Sprints correspondentes à entrega, a quantidade de trabalho realizado sempre aumenta e, assim, a tendência é de que essa linha se aproxime da outra. A quantidade de trabalho total prevista, no entanto, pode variar, de forma que sua linha pode subir ou descer à medida que novos itens são introduzidos, itens previstos são removidos, outros itens são divididos em itens menores e estimativas são refeitas.

O Gráfico de Release Burnup, portanto, oferece mais informações que o de Burndown: além de mostrar a variação no total de quantidade de trabalho previsto para a entrega, podemos ver, em cada momento, a distância entre o trabalho realizado e o previsto.

A figura a seguir mostra um exemplo de Release Burnup ao final do quinto Sprint correspondente à entrega. Repare que, nesse exemplo, a quantidade total de trabalho, representada pela linha pontilhada, sobe ao longo do tempo.

Nos exemplos desta seção, utilizo Story Points como unidade de "trabalho realizado" e "trabalho total", que são atribuídos pelos Desenvolvedores aos itens do Product Backlog selecionados para a entrega. No entanto, outras unidades podem ser usadas (veja, no capítulo *Story Points: estimando o trabalho*, a seção *Unidades para as estimativas*).

Para unidade de "tempo", é comum usarmos como marco o final de cada Sprint previsto até a entrega. Nesse caso, cada ponto do gráfico mostrará a soma das estimativas dos itens do Product Backlog já implementados, ao final

de cada um dos Sprints previstos para essa entrega. Alternativamente, podemos utilizar como marcos de tempo o final de cada uma das semanas previstas até a entrega ou alguma outra unidade de tempo.

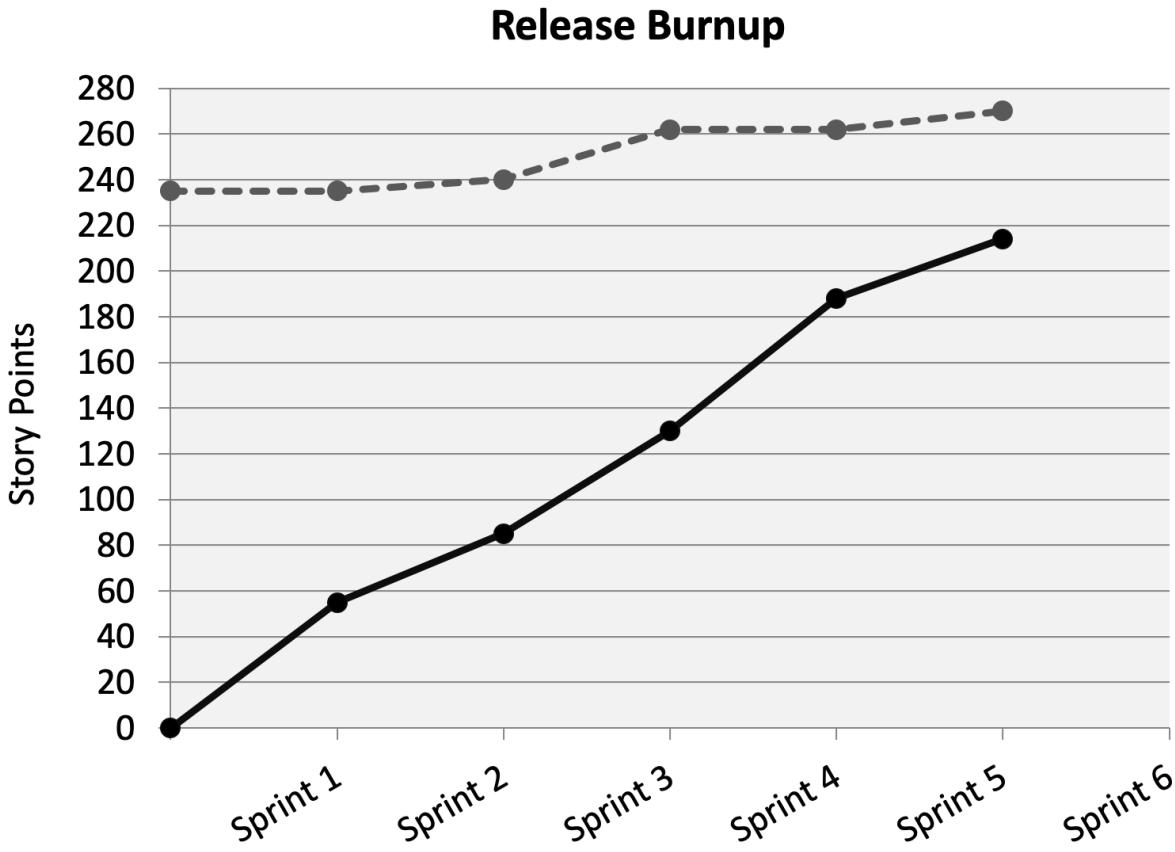


Figura 29.5: Um exemplo de Release Burnup no final do penúltimo Sprint correspondente à entrega

Outra forma comum de definirmos o trabalho já realizado e o trabalho total é utilizando o valor de negócio estipulado para cada item. Esse valor de negócio é estabelecido pelo Product Owner ou diretamente por clientes do produto. É, em geral, uma unidade relativa, ou seja, obtemos o valor de negócio de um item comparando-o com o valor de negócio de outros.

Nesse caso, a linha do trabalho total no Release Burnup representa o valor de negócios esperado do total de itens

planejados em cada momento até a entrega. E a linha do trabalho já realizado representa o valor de negócios gerado em cada momento com os itens já implementados.

O Gráfico de Release Burnup é geralmente criado em uma reunião de Release Planning, e atualizado ao final de cada Sprint ou de cada semana de trabalho.

Como é o Gráfico de Release Burnup?

Para efeitos didáticos, o exemplo da figura a seguir mostra um Gráfico de Release Burnup e um Gráfico de Release Burndown juntos. Os gráficos mostram o trabalho em Story Points e foram atualizados logo após o final do quarto Sprint (de um total de cinco até a entrega). O gráfico pontilhado é o total de trabalho previsto, o gráfico na descendente é o Gráfico de Burndown, e o gráfico na ascendente é o Gráfico de Burnup.

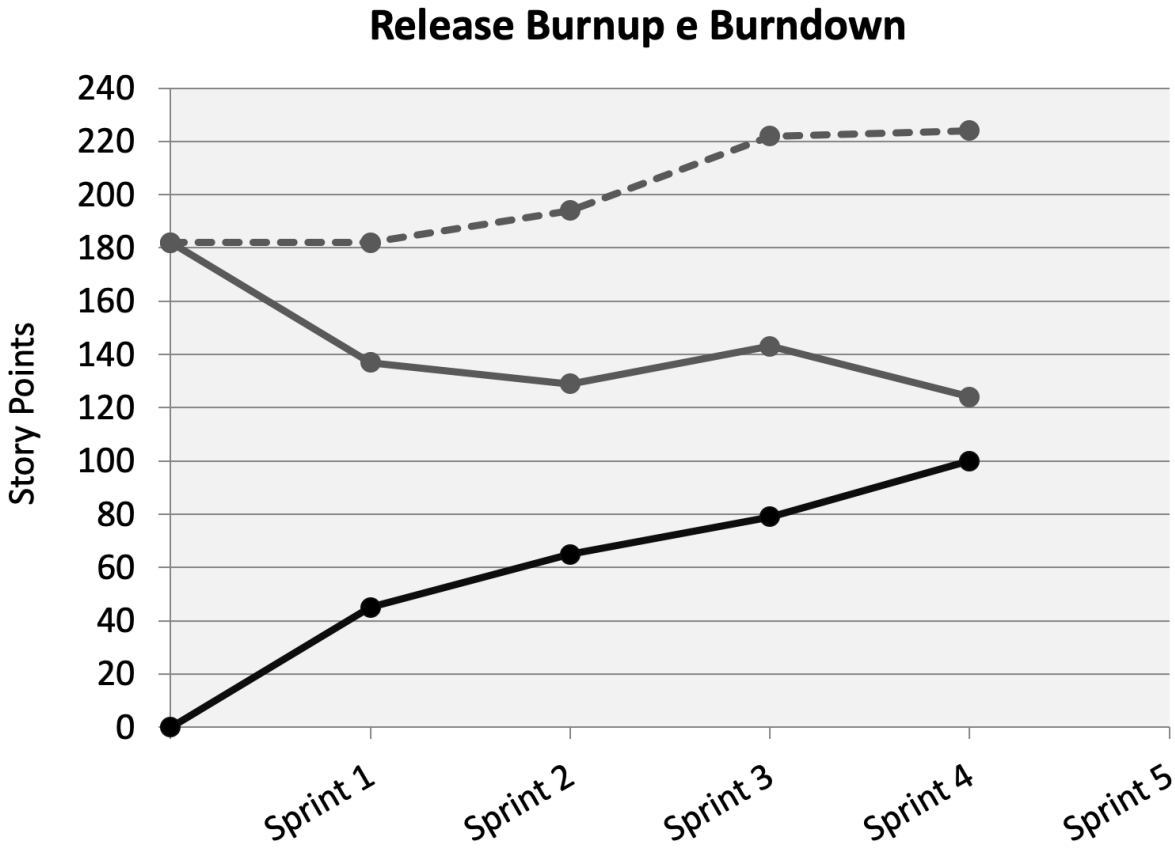


Figura 29.6: Release Burnup e Burndown juntos

À medida que o trabalho é realizado Sprint a Sprint, o Gráfico de Burnup acumula mais pontos e sobe, aproximando-se da linha que representa o trabalho total. Quanto mais próximo dessa linha o Gráfico de Burnup estiver ao final dos Sprints correspondentes à entrega, maiores são as chances de completarmos o trabalho previsto ou, ao menos, de realizarmos os objetivos da entrega.

Observe que a inclinação súbita para cima do Gráfico de Burndown durante o Sprint 3 é melhor explicada pelo Gráfico de Burnup, em que podemos ver que a quantidade de trabalho total cresceu mais do que o trabalho realizado naquele Sprint. Esse é um sinal de atenção para o Product Owner, especialmente se acontecer com frequência. Nesse caso, o Product Owner

analisará o que ocorreu e poderá tomar alguma providência.

Novamente, é importante destacar que não esperamos que os Desenvolvedores completem todos os itens definidos inicialmente para a entrega, mas sim que realize os objetivos de negócios, trabalhando a partir dos itens de maior ordem.

29.4 Gráfico de Sprint Burndown

O que é o Gráfico de Sprint Burndown?

O Sprint Burndown é um gráfico mantido e usado pelos Desenvolvedores para monitorar seu progresso no trabalho em direção ao final de um Sprint. Ele mostra a quantidade de trabalho restante estimado para o Sprint (eixo vertical), em cada dia de trabalho do Sprint (eixo horizontal).

Embora não seja parte integrante do framework Scrum, o Gráfico de Sprint Burndown é uma maneira rápida e prática para os Desenvolvedores visualizarem o andamento do Sprint. A figura a seguir mostra um exemplo de Sprint Burndown ao final de um Sprint bem-sucedido.

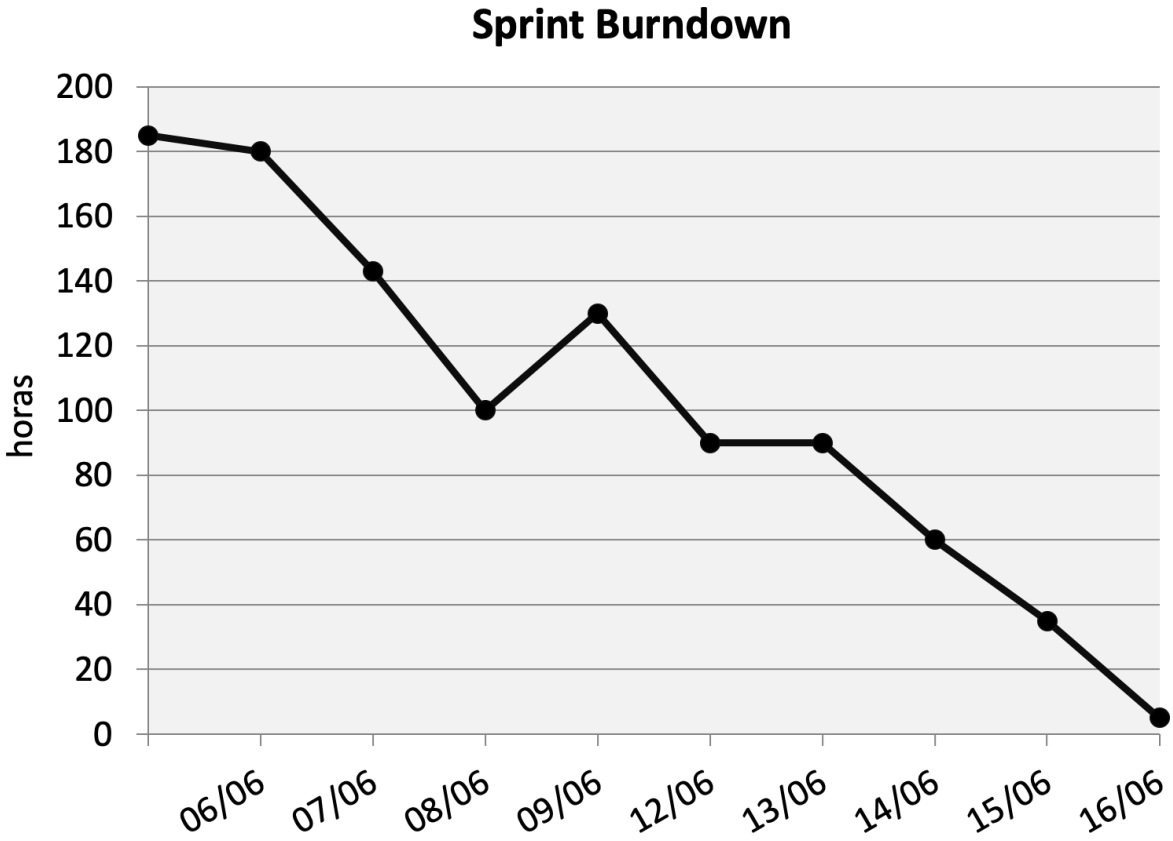


Figura 29.7: Gráfico de Sprint Burndown ao final de um Sprint bem-sucedido

O trabalho restante em cada dia pode ser contabilizado a partir dos itens do Sprint Backlog e de suas estimativas (em Story Points, por exemplo). Nesse caso, geralmente consideramos os pontos de um item como "queimados" apenas quando o item está pronto, de acordo com a Definição de Pronto, o que certamente gerará saltos (ou degraus) no gráfico. Alternativamente, podemos considerar a "queima" de pontos parciais, de acordo com a proporção entre a quantidade de trabalho daquele item já realizada e a quantidade total de trabalho planejada para o item.

Na prática, para o Gráfico de Sprint Burndown, é mais habitual o uso de tarefas como a unidade de quantidade de trabalho restante. A quebra em tarefas é a forma mais comum que os Desenvolvedores utilizam para decompor

os itens do Sprint Backlog para sua implementação, o que geralmente se inicia na reunião de Sprint Planning e segue ocorrendo durante o Sprint. No entanto, para que as tarefas possam ser utilizadas para o gráfico, a quebra de todos os itens deve ser realizada inteiramente na reunião de Sprint Planning.

Nesses casos, cada tarefa pode ser estimada individualmente. Se os Desenvolvedores estimam as tarefas em horas de trabalho, então cada ponto do gráfico mostrará a soma das horas estimadas de todas as tarefas restantes do Sprint Backlog (tanto aquelas não iniciadas quanto as que já estão sendo implementadas), versus o tempo, representado por cada dia útil de trabalho do Sprint.

Alternativamente, os Desenvolvedores podem usar uma escala não numérica para as estimativas das tarefas. Por exemplo, a escala de "tamanhos de camisas" (ou *T-Shirt Sizing*) utiliza como pontos (P)equeno, (M)édio e (G)rande. Podemos estabelecer que M é duas vezes P e G é duas vezes M (ou quatro vezes P), e somar as estimativas das tarefas restantes em função de P. Dessa forma, se no dia de hoje restam quatro tarefas P, duas M e uma G, o ponto correspondente a hoje terá o valor 4P + (2 x 2P) + (1 x 4P) = 12P no eixo vertical. Outra forma consiste em simplesmente designar um valor para cada unidade da escala (como P = 1, M = 2 e G = 4). Caso os Desenvolvedores utilizem uma escala não numérica diferente de "tamanhos de camisas", basta igualmente estabelecerem uma relação de proporção entre os pontos dessa escala.

Se os Desenvolvedores não estimam as suas tarefas, eles podem somar, em cada dia, o número de tarefas restantes e marcar no gráfico.

O Gráfico de Sprint Burndown é geralmente criado ao final da reunião de Sprint Planning, ou antes da primeira reunião de Daily Scrum, e é atualizado diariamente com o valor do trabalho restante naquele momento. Em alguns times, os Desenvolvedores preferem atualizá-lo logo antes de cada reunião de Daily Scrum, enquanto que, em outros, eles preferem fazê-lo no início ou no final do dia. O último ponto do gráfico é marcado quando o trabalho se encerra no Sprint.

Como é o Gráfico de Sprint Burndown?

No exemplo da figura a seguir, os Desenvolvedores realizam a quebra dos itens do Sprint Backlog inteiramente na Sprint Planning e usam horas de trabalho para estimar suas tarefas. Vamos imaginar que, nesse exemplo, o Gráfico de Sprint Burndown é atualizado logo antes do início da reunião de Daily Scrum e que essa reunião ocorre no meio da manhã de cada dia de trabalho (de um total de nove).

A reunião de Sprint Planning foi realizada na primeira metade do primeiro dia, e o primeiro ponto (sobre o eixo vertical) reflete a soma do total de horas de todas as tarefas definidas na reunião. O segundo ponto do gráfico (dia 06/06) reflete a quantidade de horas estimadas para as tarefas restantes (ainda não implementadas ou em andamento) logo antes da primeira reunião de Daily Scrum, no segundo dia do Sprint, momento em que algumas tarefas já foram realizadas. O gráfico acaba de ser atualizado no quinto dia do Sprint e a reunião de Daily Scrum está para começar. Observe que o último ponto do gráfico (16/06) será atualizado no último dia do Sprint, logo antes da reunião de Sprint Review.

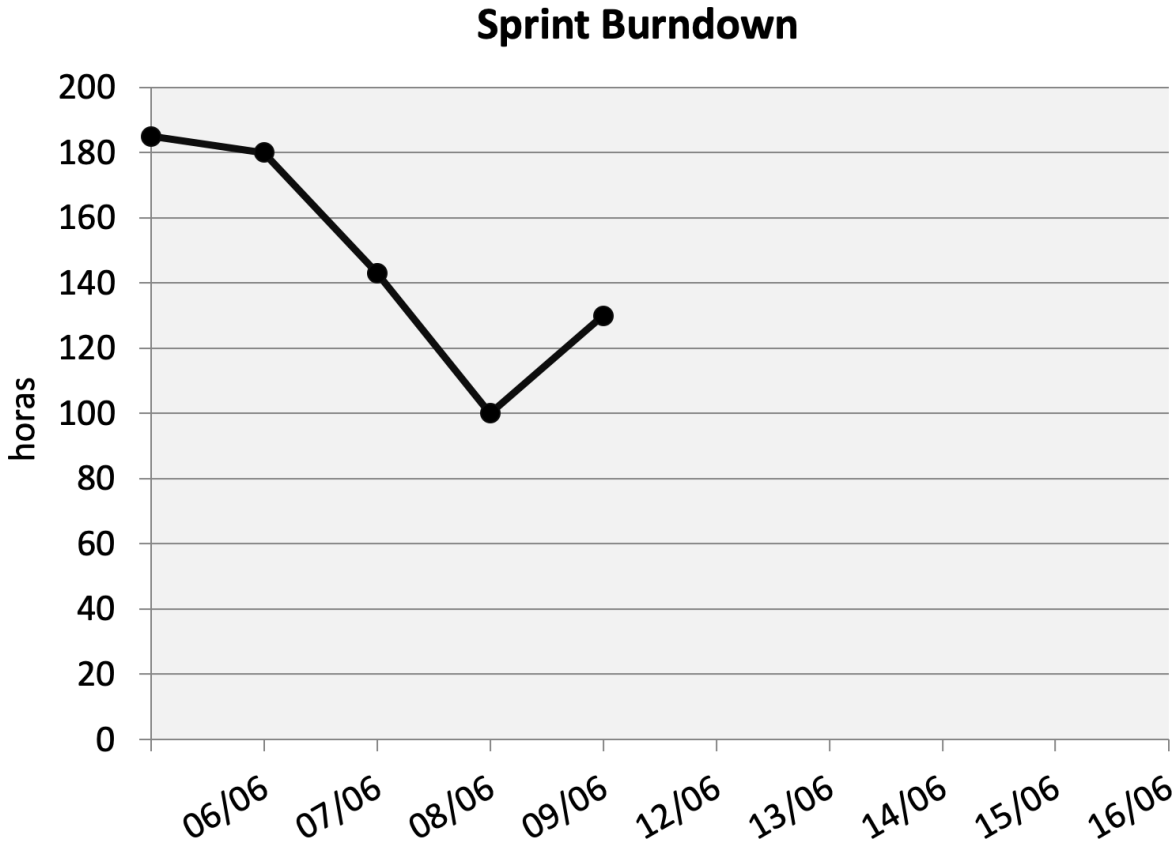


Figura 29.8: Gráfico de Sprint Burndown com inclinação para cima

A inclinação súbita para cima no dia 09/06, de 100 para 135 horas de trabalho, serve como sinal de alerta, podendo indicar que ocorreu algum problema entre a reunião de Daily Scrum dos dias 08/06 e 09/06. A reestimativa de tarefas já existentes ou a introdução de novas tarefas são perfeitamente normais e esperadas, mas aqui alguma dessas questões pode ter ocorrido em excesso, pode ter ocorrido algum impedimento ao trabalho dos Desenvolvedores, ou pode ter ocorrido mais de um desses problemas ao mesmo tempo.

Nesse outro exemplo da figura a seguir, vemos pelo Gráfico de Sprint Burndown que os Desenvolvedores trabalham em Sprints de quatro semanas, e utilizam o número de tarefas como unidade de trabalho restante. Vemos, no exemplo, que a quantidade de tarefas

restantes se manteve a mesma por três dias seguidos, formando uma linha reta.

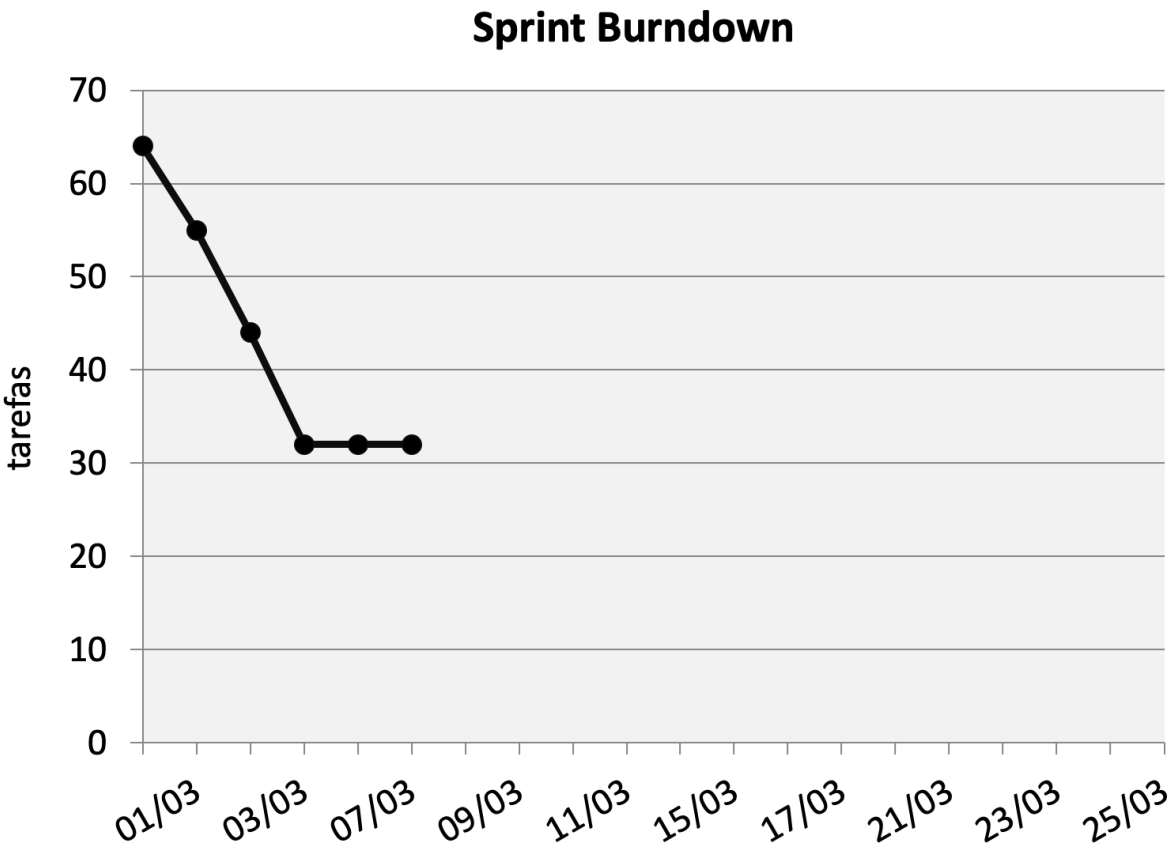


Figura 29.9: Gráfico de Sprint Burndown com linha reta

Da mesma forma que no caso anterior, esse comportamento do gráfico também serve como sinal de alerta, já que os Desenvolvedores estão cada vez mais distantes de "queimar" o trabalho estimado restante.

É importante destacar que, mesmo que os Desenvolvedores cheguem ao final de um Sprint com tarefas sobrando, ainda assim poderão ter sido bem-sucedidos. Embora deem o seu melhor para completar o trabalho, o compromisso dos Desenvolvedores é o de perseguir o Objetivo do Sprint. Mesmo que nem todos os itens selecionados tenham sido completados, ainda assim esse objetivo poderá ter sido realizado (veja o

capítulo *Compromisso: Objetivo de Sprint*). Não é incomum chegarmos ao final de um Sprint sem que o Sprint Burndown alcance o zero do eixo vertical, e mesmo assim podemos considerar o Sprint um sucesso.

Nos dois exemplos da figura a seguir, vemos que, ao final do Sprint, não foi "queimado" todo o trabalho estimado.

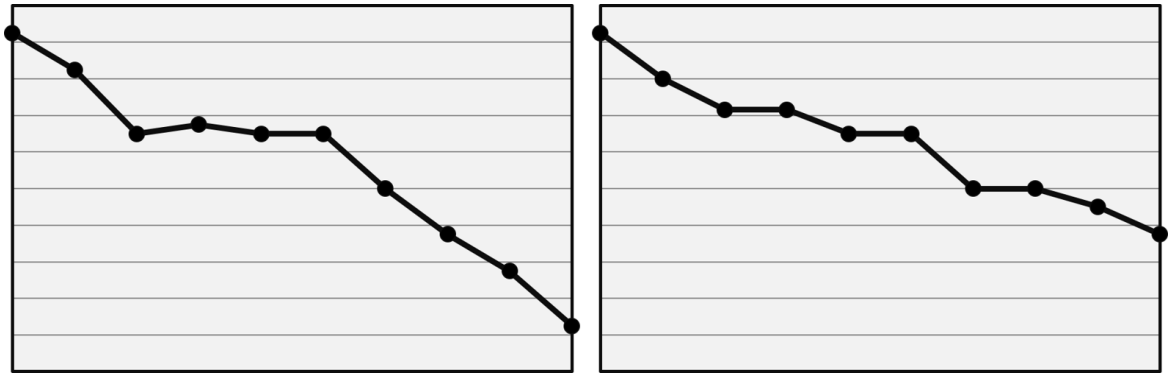


Figura 29.10: Dois exemplos de gráficos que não chegam a zero trabalho restante

No entanto, são dois cenários com resultados bastante distintos. No primeiro caso, os Desenvolvedores provavelmente geraram valor suficiente para realizar o Objetivo do Sprint. Mas, no segundo caso, há uma grande chance de o Objetivo do Sprint não ter sido realizado, já que muito deixou de ser feito.

O Gráfico de Sprint Burndown é criado pelos Desenvolvedores, para os Desenvolvedores. E para mais ninguém. Ele serve para sinalizar, para eles mesmos, o quão perto ou o quão longe estão de cumprir o trabalho planejado para o Sprint e, assim, terem uma ideia de seu andamento. O Gráfico de Sprint Burndown não deve, de forma alguma, servir como instrumento de cobrança para Product Owner, Scrum Master ou qualquer parte interessada exercer pressão sobre os Desenvolvedores.

Linha ideal

O que é linha ideal?

A linha ideal serve como guia visual para o comportamento do Gráfico de Sprint Burndown ao longo do Sprint. A linha ideal é opcional e, caso utilizada, é traçada no momento que o gráfico é criado.

Ela é uma linha diagonal que liga a quantidade de trabalho total sobre o eixo vertical, ou seja, no momento inicial, ao último dia do Sprint sobre o eixo horizontal, ou seja, com zero trabalho restante. Dessa forma, ela liga (*[primeiro dia do Sprint]*, *[trabalho total]*) a (*[último dia do Sprint]*, *[zero trabalho restante]*). A figura a seguir mostra a linha ideal.

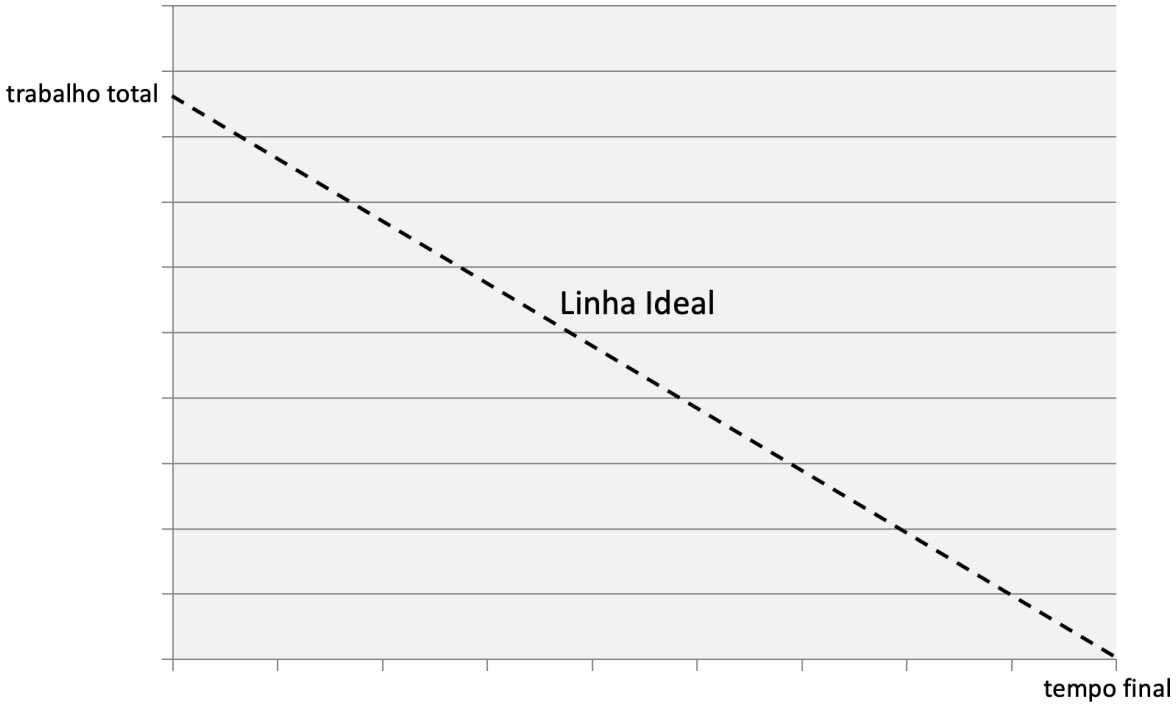


Figura 29.11: Linha ideal

Comportamento da linha ideal

É natural, ao longo de um Sprint que está correndo bem, o gráfico oscilar em torno da linha ideal. Um exemplo de oscilação saudável pode ser visto na figura a seguir.

Repare que o gráfico flutua em torno da linha ideal, mas se distanciar muito dela.

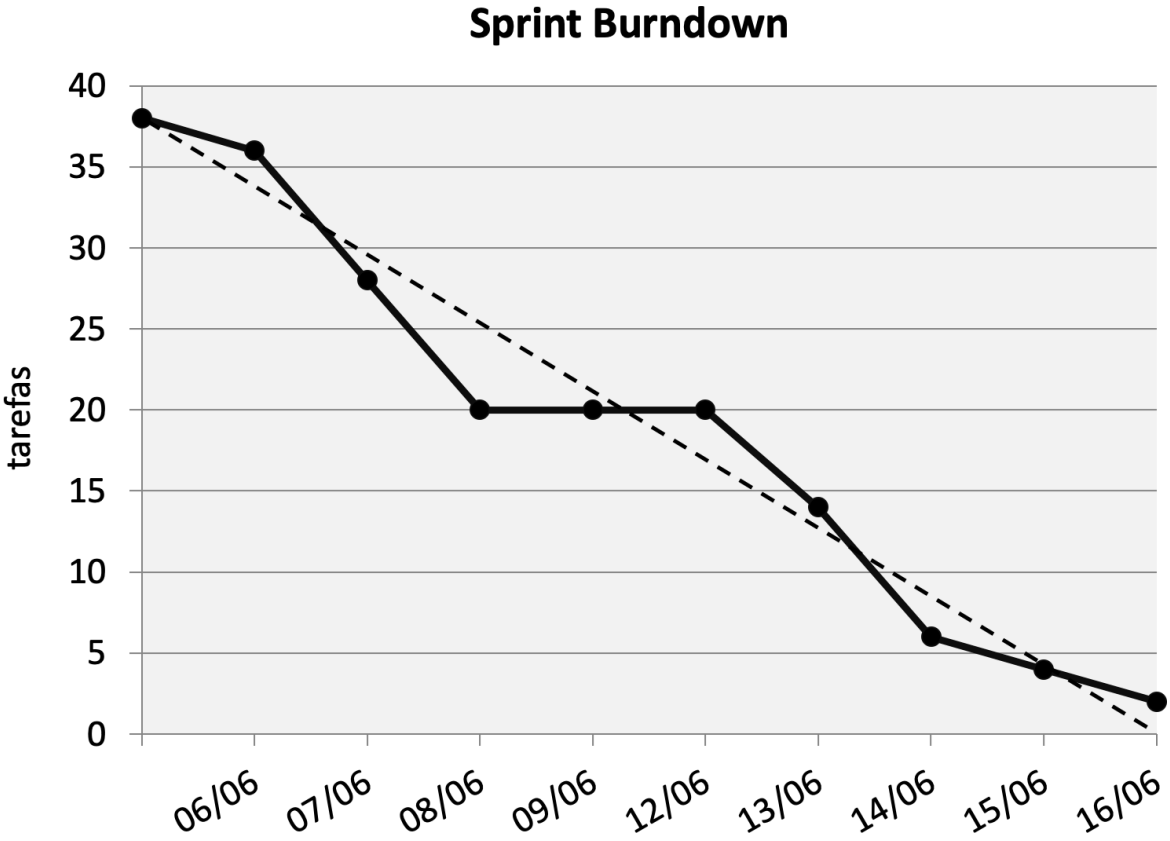


Figura 29.12: Oscilação natural do gráfico em torno da linha ideal

No entanto, se o ponto do gráfico referente ao momento atual estiver muito acima da linha, os Desenvolvedores ficarão em alerta. Quanto mais próximo do final do Sprint isso acontecer, em mais risco o Objetivo do Sprint poderá estar. No exemplo da figura a seguir, a situação parece crítica a três dias do final do Sprint.

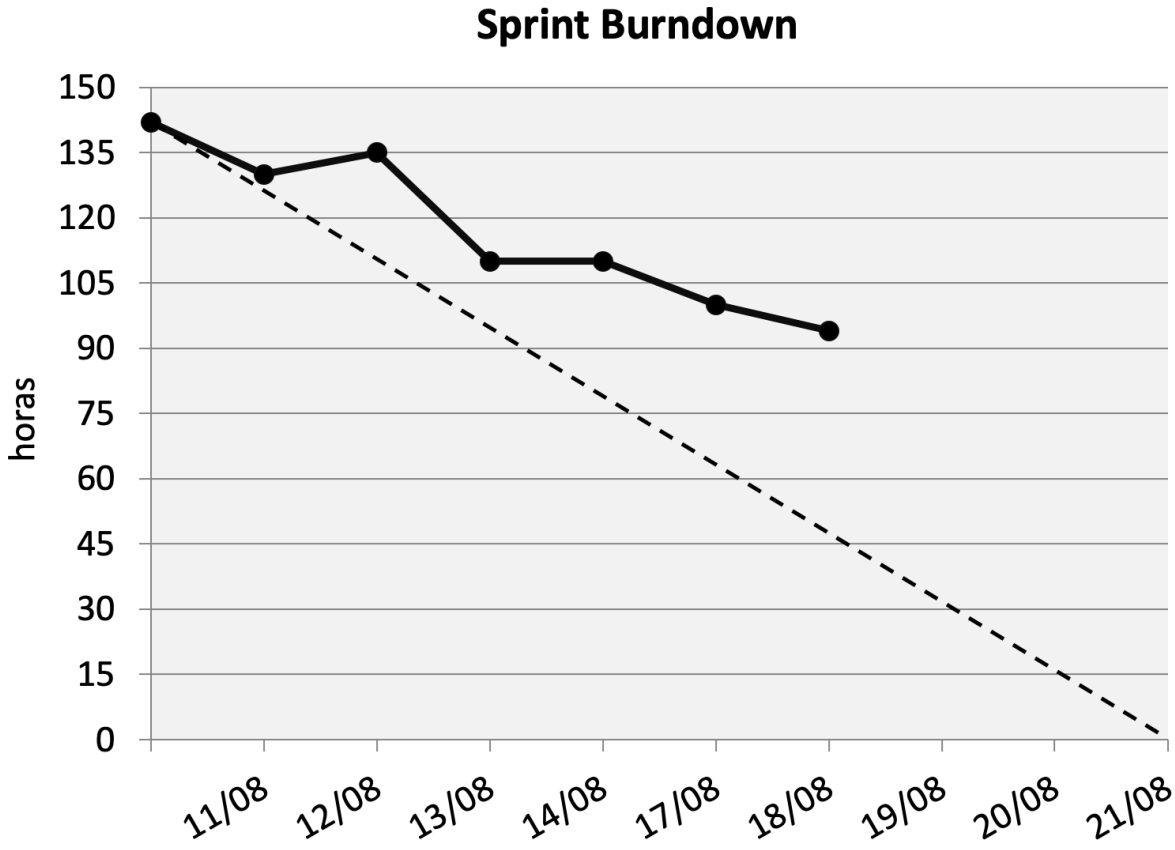


Figura 29.13: Nesse momento, a distância do ponto atual à linha ideal pode indicar uma situação crítica

A linha ideal pode cumprir bem seu propósito se compreendida apenas como um guia visual para o comportamento do Gráfico de Sprint Burndown. Ela também pode ser usada para o Gráfico de Release Burndown, ligando a quantidade inicial de trabalho previsto sobre o eixo vertical ao final do último Sprint correspondente à entrega sobre o eixo horizontal. Ou seja, ela ligará (*[início do primeiro Sprint]*, *[trabalho total]*) a (*[final do último Sprint]*, *[zero trabalho restante]*).

Críticas à linha ideal

Por diversas razões, a linha ideal é muito criticada por vários autores e praticantes de Scrum, que preferem não

a utilizar. Alguns acreditam que essa linha somente serve de instrumento de pressão sobre os Desenvolvedores. Eu inclusive já a vi ser chamada de "linha da opressão". Outros acreditam que, ao buscarem se adequar ao comportamento da linha ideal, os Desenvolvedores acabam por não serem honestos em suas estimativas.

Outra questão é que, ao traçarmos a linha ideal, estamos considerando uma precisão irreal na definição e nas estimativas iniciais das tarefas. Uma parte significativa das tarefas surge no decorrer do Sprint e diversas delas são reestimadas, fazendo com que o ponto inicial da linha ideal (e, portanto, a própria linha) não tenha significado real.

Final

Capítulo 30
Apêndice - Glossário

Em ordem alfabética.

Backlog Grooming: veja *Refinamento do Product Backlog*.

BDUF: *Big Design Up Front* é a prática de se especificar detalhadamente um trabalho a ser realizado (originalmente, de desenvolvimento de software), antes de começar esse trabalho, e a crença no sucesso associada a essa prática.

Clientes: pessoas, grupos ou organizações que solicitam o desenvolvimento do produto ou apenas o patrocinam, e que têm o objetivo de obter valor de negócio sempre que aquilo que é produzido é entregue aos usuários do produto.

Critérios de Aceitação: ver *Testes de Aceitação*.

Cynefin: modelo que se propõe a ajudar na tomada de decisões ao se entender a relação entre causa e efeito que ocorre em um determinado sistema ou contexto conforme interagem seus agentes. Criado em 1999 por Dave Snowden, o modelo oferece cinco domínios distintos para que se possa escolher o mais adequado ao sistema ou contexto em questão e, assim, tomar decisões apropriadas.

Daily Scrum: reunião curta e objetiva, realizada diariamente pelos Desenvolvedores, com o propósito de planejar informalmente o próximo dia do trabalho de

implementação dos itens do Sprint Backlog em busca do Objetivo do Sprint. A Daily Scrum proporciona transparência sobre o trabalho executado e a executar, promove a comunicação sobre ele, traz visibilidade a possíveis impedimentos e serve de oportunidade para decisões rápidas com relação ao progresso do trabalho no Sprint. É um evento do Scrum.

Definição de Preparado: entendimento formal compartilhado entre o Product Owner e os Desenvolvedores sobre o estado em que um item do Product Backlog deve estar para ser considerado qualificado para ser implementado, ou seja, para poder ser escolhido para compor o Sprint Backlog durante a reunião de Sprint Planning. A Definição de Preparado visa a garantir que o item esteja preparado segundo um critério bem definido.

Definição de Pronto: entendimento formal compartilhado entre o Product Owner e os Desenvolvedores sobre o que é necessário para considerarem que qualquer item ou Incremento do produto, implementado pelos Desenvolvedores, está pronto dentro de um Sprint. A Definição de Pronto é um compromisso de todo e cada Incremento, que deve cumpri-la integralmente, da mesma forma que os itens implementados que o compõem.

Desenvolvedor: responsabilidade atribuída a um conjunto de pessoas que trabalham em equipe na realização do trabalho de desenvolvimento do produto propriamente dito, criando Incrementos Sprint após Sprint. Os Desenvolvedores são parte do Time de Scrum.

Desenvolvimento de produto: trabalho de criação e evolução do produto, com a adição e modificação de funcionalidades, características e comportamentos do

produto, além da correção de problemas e erros. Com Scrum, o trabalho de desenvolvimento de um produto é feito continuamente, de forma iterativa e incremental.

Entrega: entrega de um ou mais Incrementos do produto prontos, implementados pelos Desenvolvedores em um ou mais Sprints sucessivos, que em conjunto possuem valor suficiente para serem utilizados. A entrega é realizada para quem, de alguma forma, vai utilizar o produto. Chamada de *release* em inglês.

Equipe: veja *Time*.

Gráfico de Release Burndown: gráfico mantido pelo Product Owner, usado tanto por ele quanto pelos Desenvolvedores para monitorar o progresso no trabalho em direção a uma entrega, a partir da "queima" do trabalho previsto para a entrega.

Gráfico de Release Burnup: gráfico mantido pelo Product Owner, utilizado tanto por ele quanto pelos Desenvolvedores para monitorar o progresso no trabalho em direção a uma entrega, a partir do acúmulo de trabalho realizado em direção ao trabalho previsto para a entrega.

Gráfico de Sprint Burndown: gráfico mantido e usado pelos Desenvolvedores para monitorarem seu progresso no trabalho em direção ao final de um Sprint, a partir da "queima" do trabalho previsto para o Sprint.

***Grooming*:** veja *Refinamento do Product Backlog*.

Impedimento: obstáculo ou barreira que dificulta significativamente ou impede que o trabalho dos Desenvolvedores seja realizado, bloqueando um ou mais itens do Sprint Backlog de forma a ameaçar o Objetivo do Sprint.

Incremento: um incremento é o resultado da implementação de um ou mais itens do Sprint Backlog, que se traduz em valor visível e utilizável por clientes e usuários do produto. Cada Incremento se soma aos Incrementos anteriores, visando a realizar o Objetivo do Produto. Ao menos um Incremento é produzido em cada Sprint.

Objetivo de Entrega: objetivo ou necessidade de negócios de alto nível a ser realizado por meio do trabalho dos Desenvolvedores para uma entrega. O Objetivo da Entrega não é parte do framework Scrum.

Objetivo de Sprint: objetivo ou propósito bem definido, alinhado ao Objetivo do Produto, estabelecido e acordado entre Product Owner e os Desenvolvedores durante a reunião de Sprint Planning. O Objetivo do Sprint representa o valor que o Time de Scrum espera que seja produzido no Sprint ao implementar itens do Sprint Backlog, a partir daquele de maior ordem. Ele é uma coerência que conecta esses itens e visa levar os Desenvolvedores do produto a trabalharem juntos, e não em diferentes iniciativas. O Objetivo do Sprint é o compromisso do Sprint Backlog, e forma parte dele.

Objetivo do Produto: objetivo ou necessidade de negócios ou do usuário de alto nível que expressa o propósito ou motivação central para o Time de Scrum no desenvolvimento do produto, a ser satisfeito a partir da implementação de itens do Product Backlog. O Objetivo do Produto fornece contexto, orientação, motivação e inspiração para esse trabalho, e alinha o entendimento de todas as partes interessadas sobre o que será realizado. Similar à definição de mercado para "visão de produto". O Objetivo do Produto é o compromisso do Product Backlog, e forma parte dele.

Partes interessadas: pessoas, grupos ou entidades que podem afetar, ser afetadas por ou se entenderem afetadas por decisões, atividades ou resultados do desenvolvimento do produto e que, assim, possuem algum interesse em jogo. Clientes, usuários, o próprio time, departamentos determinados da organização e gestores são exemplos de possíveis partes interessadas. Termo traduzido de *stakeholders*, em inglês.

Plano da entrega: plano que indica qual o objetivo a ser realizado por meio de uma entrega do produto para seus usuários e quando será realizado. O Plano da entrega geralmente contém o Objetivo da Entrega, a data exata ou aproximada em que a entrega será realizada e um conjunto de itens selecionados do Product Backlog para a entrega.

Product Backlog: lista ordenada, planejável, emergente e gradualmente detalhada, gerenciada pelo Product Owner, que evolui ao longo de todo o desenvolvimento do produto. O Product Backlog é a única fonte de trabalho para os Desenvolvedores e contém, em cada momento, o que se acredita que será implementado por eles para realizar o Objetivo do Produto, tais como características, comportamentos e funcionalidades do produto, experimentos, melhorias, correções de problemas, pesquisas que se façam necessárias, objetivos de negócios dos clientes e demais partes interessadas e necessidades dos usuários. É um artefato do Scrum.

Product Owner: responsabilidade atribuída a uma pessoa de gerenciar, a partir do Product Backlog, a definição do produto a ser desenvolvido, de forma a garantir e maximizar, por meio do trabalho dos Desenvolvedores, o valor de negócio do produto. O Product Owner é parte do Time de Scrum.

Produto: meio pelo qual o Time de Scrum, motivado e direcionado pelo Objetivo do Produto, cria valor para seus clientes, a partir da implementação de itens do Product Backlog. O produto possui limites bem definidos, partes interessadas conhecidas e usuários ou clientes bem definidos. Um produto pode ser um serviço, um produto físico ou algo mais abstrato.

Quadro de tarefas: quadro frequentemente utilizado por Desenvolvedores para representar o Sprint Backlog, dividido em colunas que contêm os itens selecionados para o Sprint e as suas tarefas correspondentes. As tarefas são distribuídas entre as colunas "A Fazer", "Fazendo" e "Feito" (ou termos similares), de acordo com seu andamento no Sprint. Os itens e as tarefas podem ser escritos em notas adesivas. Diversos softwares simulam o Quadro de Tarefas, representando-o virtualmente.

Refinamento do Product Backlog: trabalho de preparação de itens do Product Backlog para a sua implementação em um momento próximo a que isso aconteça, que pode envolver o detalhamento de itens, seu reordenamento, o desmembramento de itens maiores em itens menores e, possivelmente a adição e a remoção de itens. Esse é um trabalho realizado pelos Desenvolvedores, geralmente em conjunto com o Product Owner, mas que com frequência inclui outras partes interessadas e pode até mesmo excluir o Product Owner por completo.

Release: veja *Entrega*.

Release Planning: reunião na qual o Time de Scrum planeja a próxima entrega, a partir da criação de um plano da entrega (veja *Plano de entrega*).

Roadmap do produto: plano em alto nível de como o produto evoluirá ao longo do tempo até um momento futuro determinado, expresso por uma linha do tempo com marcos, cada um contendo uma data exata ou aproximada no futuro e um objetivo do produto a ser realizado com o desenvolvimento e/ou entregas do produto até a data.

Scrum Master: responsabilidade atribuída a uma pessoa de facilitar e potencializar o trabalho do Time de Scrum, ajudando seus membros a buscarem continuamente melhorar seus processos e práticas, de forma a se tornarem mais eficientes e eficazes na realização do seu trabalho e a tornarem o seu trabalho mais prazeroso. O Scrum Master é responsável por promover e apoiar o entendimento e o uso do Scrum e por velar pela qualidade das interações entre os membros do Time de Scrum, e entre eles e pessoas externas. O Scrum Master é parte do Time de Scrum.

Sprint: ciclo de desenvolvimento de duração fixa, em que um ou mais Incrementos do produto pronto são implementados pelos Desenvolvedores a partir dos itens de maior ordem do Product Backlog, e que vai desde a reunião de planejamento até as reuniões de encerramento. É um evento do Scrum.

Sprint Backlog: lista de itens selecionados na reunião de Sprint Planning do alto do Product Backlog para a implementação de um ou mais Incrementos do produto no Sprint (o quê), adicionada de um plano evolutivo de como esse trabalho será realizado (como), geralmente expresso por um conjunto de tarefas correspondente a cada item, e de um objetivo a ser cumprido (por quê), chamado de Objetivo do Sprint. É um artefato do Scrum.

Sprint Planning: reunião realizada no primeiro momento do primeiro dia do Sprint, na qual o Time de Scrum planeja e define por que o Sprint será realizado, o que será implementado no Sprint corrente e como o que foi selecionado será implementado. É um evento do Scrum.

Sprint Retrospective: última reunião realizada no Sprint, em que o Time de Scrum inspeciona o Sprint que está se encerrando quanto a seus processos de trabalho, dinâmicas, comportamentos e ambiente, e planeja as melhorias necessárias a serem executadas no próximo Sprint. É um evento do Scrum.

Sprint Review: reunião realizada no último dia do Sprint em que o Time de Scrum colabora com clientes e demais partes interessadas para obter seu feedback sobre o Incremento ou Incrementos do produto implementados no Sprint, para então utilizá-lo como matéria-prima para modificar o Product Backlog para Sprints futuros, o que pode ser realizado de forma colaborativa durante a própria reunião. É um evento do Scrum.

Story Point: unidade relativa criada pelos Desenvolvedores para ser utilizada em estimativas do tempo necessário para implementar um item de trabalho.

Testes de Aceitação: critérios que documentam os detalhes da User Story a partir de exemplos, expressos por enunciados pequenos e de fácil entendimento, que são utilizados para determinar quando o comportamento ou característica nova, criada no produto pelos Desenvolvedores, está conforme o que foi definido. Os Testes de Aceitação são preferivelmente automatizados.

Time: grupo de pessoas que trabalham juntas e colaboram em busca de um objetivo comum. O mesmo que *equipe*.

Time de Scrum: time formado pelo Product Owner, Scrum Master e Desenvolvedores.

Timebox: alocação máxima de tempo dentro da qual uma atividade deverá ocorrer.

User Story: descrição concisa e simples de um comportamento ou característica do produto, sob o ponto de vista de um usuário desse produto para o qual o que está descrito tem valor, que funciona como um convite para uma conversa entre os Desenvolvedores do produto e pessoas de negócios para definirem os detalhes do que será implementado.

Usuários do produto: pessoas que recebem e efetivamente utilizam o produto.

Velocidade: é a média da quantidade de trabalho implementado pelos Desenvolvedores, em conjunto, nos últimos Sprints, geralmente medida pela taxa média de queima de Story Points por Sprint ou de itens por Sprint.

Visão do produto: veja *Objetivo do Produto*.

Capítulo 31
Apêndice - Bibliografia

ÁGIL. *Michaelis Dicionário Brasileiro da Língua Portuguesa*, 2015. Disponível em http://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=ágil. Acesso em 06 out. 2020.

ARIELY, D. *What's the value of a big bonus?* New York Times, nov. 2008. Disponível em https://www.nytimes.com/2008/11/20/opinion/20ariely.html. Acesso em 10 ago. 2018.

ARMONY, R. S. 196 f. *Fatores críticos para a prática de Valores Ágeis em equipes de tecnologia da informação.* Dissertação (Mestrado em Administração de Empresas) — Instituto de Administração e Gerência, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, 2010.

BECK, K. et al. *Manifesto for agile software development*, 2001. Disponível em http://agilemanifesto.org/. Acesso em: 14 dez. 2012.

BEEDLE, M. et al., *SCRUM*: An Extension Pattern Language for Hyperproductive Software Development. In: Pattern Languages of Program Design, N. Harrison, B. Foote, and H. Rohnert (eds.). Addison-Wesley, Reading, Massachussets, 1999, v. 4, p. 637–651.

BIG DESIGN UP FRONT. *WikiWikiWeb*, 2013. Disponível em http://wiki.c2.com/?BigDesignUpFront. Acesso em 30 jul. 2018.

BULEY, L. *The user experience team of one*: a research and design survival guide. Rosenfeld Media, 2013.

BURNDOWN CHART. *Agile Alliance: Agile Glossary*, 2017. Disponível em https://www.agilealliance.org/glossary/burndown-chart/. Acesso em 19 jul. 2020.

COCKBURN, A. *Agile software development*: the cooperative game. 2. ed. Massachusetts: Addison-Wesley, 2007.

COHEN, D.; LARSON, G.; WARE, B. *Improving software investments through requirements validation.* In: Proceedings of 26th Annual NASA Goddard Software Engineering Workshop. Greenbelt, MD, Estados Unidos: IEEE, nov. 2001, p. 106-114. Disponível em http://www.sente.com/Presentations/software%20investments%20sew26.pdf. Acesso em 30 mar. 2016.

COHN, M. *User Stories applied*: for Agile software development. Massachusetts: Addison-Wesley, 2004.

COHN, M. *Agile estimating and planning.* Englewood Cliffs: Prentice Hall PTR, 2005.

COMBLEY, R. *Cambridge Business English Dictionary*. Cambridge University Press, 2011.

COPLIEN, J. *Ordered Not Prioritized.*, 2011. Disponível em: https://www.scrum.org/resources/ordered-not-prioritized. Acesso em: 5 mar. 2019.

CRISPIN, L.; GREGORY, J. *Agile testing*: a practical guide for testers and Agile teams. Boston: Addison-Wesley Professional, 2009.

CUMMINGS, T. G. *Self-regulating work groups*: A socio-technical synthesis. The Academy of Management Review, v. 3, n. 3, p. 625-634, jul. 1978.

DEGRACE, P.; STAHL L. H. *Wicked problems, righteous solutions*: a catalogue of modern software engineering paradigms. Nova Jersey: Yourdon Press, 1990.

DERBY, E.; LARSEN, D. *Agile retrospectives*: making good teams great. Texas: Pragmatic Bookshelf, 2006.

DRUCKER, P. F. *Landmarks of tomorrow*. New York: Harper & Brothers, 1959.

DRUCKER, P. F. *Management challenges for the 21st Century*. HarperCollins e-books. Kindle Edition, 1999.

DRUSKAT, V. U.; WHEELER, J. V. *Managing from the boundary*: the effective leadership of self-managing work teams. Academy of Management Journal, Mississippi State, MS, Estados Unidos, v. 46, n. 4, p. 435-457, 2003.

GREENING, J. *Planning Poker or how to avoid analysis paralysis while release planning.* 2002. Disponível em: http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf. Acesso em: 15 dez. 2012.

GREENLEAF, R. K.; SPEARS, C. *Servant leadership*: a journey into the nature of legitimate power and greatness. 25th anniversary edition. New York: Paulist Press, 2002.

GROSS, J. M.; MCINNIS, K. R. *Kanban made simple*: demystifying and applying Toyota's legendary manufacturing process. Nova Iorque: AMACOM, 2003.

HACKMAN, J. R.; OLDHAM, G.; JANSON, R.; PURDY, K. *A new strategy for job enrichment.* California Management Review, Berkeley, CA, Estados Unidos, v. 17, n. 4, p. 55-71, 1975.

HAMPSON, I. *Lean Production and the Toyota Production System or, the case of the forgotten production concepts.*

Economic and Industrial Democracy, v. 20, p. 369-391, 1999.

HIGHSMITH, J. *Agile software development ecosystems.* Boston: Addison-Wesley, 2002.

HIGHSMITH, J. *Agile project management*: creating innovative products. Massachusetts: Addison-Wesley, 2004.

HOLLAND, J. H. *Studying Complex Adaptive Systems.* Journal of Systems Science and Complexity, v. 19, p. 1-8, 2006.

JAIN, R.; MESO, P. *Theory of Complex Adaptive Systems and Agile software development.* In: Proceedings of the Tenth Americas Conference on Information Systems, New York, NY, Estados Unidos: AMCIS, ago. 2004. Disponível em: http://aisel.aisnet.org/amcis2004/197. Acesso em: 3 mai. 2016.

JANIS, I. L. *Groupthink*: psychological studies of policy decisions and fiascoes. 2. ed. Boston: Houghton Mifflin, 1982.

JEFFRIES, R.; ANDERSON, A.; HENDRICKSON, C. *Extreme Programming installed.* Massachusetts: Addison-Wesley, 2000.

JEFFRIES, R. [@RonJeffries]. "I am not sure I invented story points but if I did I’m sorry now.". Dec 21 dez. 2017, 11:29 am. Tweet. Disponível em: https://twitter.com/ronjeffries/status/943790497981706242?lang=en. Acesso em: 24 mai. 2020.

JEFFRIES, R. *Essential XP: Card, Conversation, and Confirmation.* XP Magazine, Ago. 2001. Disponível em:

https://ronjeffries.com/xprog/articles/expcardconversationconfirmation/. Acesso em: 19 set. 2019.

KERTH, N. *Project retrospectives*: a handbook for team reviews. Nova Iorque: Dorset House, 2001.

LARMAN, C. *Agile and iterative development*: a manager's guide. Massachusetts: Addison-Wesley, 2003.

LIKER, J. K. *The Toyota way*: 14 management principles from the world's greatest manufacturer. Blacklist: McGraw-Hill Professional Publishing, 2003.

LOCKE, E. A. *Motivation through conscious goal setting.* Applied & Preventive Psychology, Amsterdã, Holanda, v. 5, p. 117-124, 1996.

LOCKE, E. A.; LATHAM, G. P. *New directions in goal-setting theory.* Current Directions in Psychological Science, Nova Iorque, NY, Estados Unidos, v. 15, n. 5, p. 265-268, out. 2006.

MANZ, C. C.; NECK, C. P. *Teamthink*: beyond the groupthink syndrome in self-managing work teams. Team Performance Management, v. 3, n. 1, p. 18-31, 1997.

MANZ, C. C.; SIMS, H. P. Jr. *Leading workers to lead themselves*: the external leadership of self-managing work teams. Administrative Science Quarterly, v. 32, n. 1, p. 106-129, mar. 1987.

MARINHO, T. *How to use Merit Money to change company practices.* Management 3.0, 2018. Disponível em: https://management30.com/experience/case-studies/knowledge21/. Acesso em 26 jun. 2020.

MILLER, G. *The maginal number seven, plus or minus two*: Some limits on our capacity for processing information. Psychological Review, v. 63, p. 81-97, 1956.

Disponível em: http://psychclassics.yorku.ca/Miller/. Acesso em 9 jul. 2020.

MOORE, G. A. *Crossing the chasm*: marketing and selling high-tech products to mainstream customers. Nova Iorque: PerfectBound, 2001.

MORIEUX, Y. *Cooperation: when the whole is greater than the sum of its parts.* Set. 2015. Disponível em: https://www.linkedin.com/pulse/cooperation-when-whole-greater-than-sum-its-parts-yves-morieux/. Acesso em 11 jul. 2020.

MORIEUX, Y. *To boost productivity, try smart simplicity.* An Interview with BCG's Yves Morieux. Transcrição do BCG's business podcast, 21 set. 2017. Disponível em: https://www.bcg.com/publications/2011/strategic-planning-people-management-human-resources-morieux-yves-key-to-boosting-productivity.aspx. Acesso em 12 jul. 2020.

MOTTA, P. R. *Gestão contemporânea*: a ciência e a arte de ser dirigente. Rio de Janeiro: Record, 1991.

OGUNNAIKE, B. A.; RAY. W. H. *Process dynamics, modeling and control.* Nova Iorque: Oxford University Press, 1994.

OSONO, E.; SHIMIZU, N.; TAKEUCHI, H. *Relatório Toyota*: contradições responsáveis pelo sucesso da maior montadora do mundo. Tradução de Carlos Szlak. São Paulo: Ediouro, 2008.

PICHLER, R. *Agile product management with Scrum*: creating products that customers love. Boston: Addison-Wesley, 2010.

PIMENTEL, M. *Tenha nojo dos impedimentos.* InfoQ Brasil, dez. 2009. Disponível em: http://www.infoq.com/br/articles/tenha-nojo-impedimentos. Acesso em: 17 set. 2011.

PINK, D. H. *Drive*: the surprising truth about what motivates us. New York: Riverhead Books, 2009.

ROYCE, W. *Managing the development of large software systems*: concepts and techniques. In: Proceedings of IEEE WESCON. Piscataway, NJ, Estados Unidos: IEEE Press, ago. 1970, p. 1-9.

SCHWABER, K. *SCRUM Development Process*. In: Sutherland J., Casanave C., Miller J., Patel P., Hollowell G. (eds) Business Object Design and Implementation. Springer, London, 1997.

SCHWABER, K. *What is Scrum?* 2003. Disponível em: http://pages.cpsc.ucalgary.ca/~sillito/seng-301/lecture-notes/what-is-scrum.pdf. Acesso em: 9 jul. 2020.

SCHWABER, K. *Agile project management with Scrum.* Redmond: Microsoft Press, 2004.

SCHWABER, K. *The Enterprise and Scrum.* Redmond: Microsoft Press, 2007.

SCHWABER, K. *Scrum guide*. Scrum Alliance, Mai. 2009. PDF do acervo do autor.

SCHWABER, K. *Product Owners not proxies.* Ken Schwaber's Blog: Telling It Like It Is, jan. 2011. Disponível em: https://kenschwaber.wordpress.com/2011/01/31/product-owners-not-proxies/. Acesso em: 2 jul. 2020.

SCHWABER, K.; BEEDLE, M. *Agile software development with Scrum.* Upper Saddle River: Prentice Hall, 2002.

SCHWABER, K.; SUTHERLAND, J. *Scrum*. Scrum.org, Fev. 2010. PDF do acervo do autor.

SCHWABER, K.; SUTHERLAND, J. *The Scrum guide* -- the definitive guide to Scrum: the rules of the game. Jul. 2011. PDF do acervo do autor.

SCHWABER, K.; SUTHERLAND, J. *The Scrum guide* -- the definitive guide to Scrum: the rules of the game. Jul. 2013. PDF do acervo do autor.

SCHWABER, K.; SUTHERLAND, J. *The Scrum guide* -- the definitive guide to Scrum: the rules of the game. Jul. 2016. Disponível em: https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf. Acesso em: 12 jun. 2020.

SCHWABER, K.; SUTHERLAND, J. *The Scrum guide* —- the definitive guide to Scrum: the rules of the game. Nov. 2017. Disponível em: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf. Acesso em: 9 mai. 2018.

SCHWABER, K.; SUTHERLAND, J. *The Scrum guide* —- the definitive guide to Scrum: the rules of the game. Nov. 2020. PDF do acervo do autor.

SCHWARZ, R. *The skilled facilitator*: a comprehensive resource for consultants, facilitators, managers, trainers and coaches. 2. ed. San Francisco: Jossey-Bass, 2002.

SNOWDEN, D. J.; BOONE, M. E. *A leader's framework for decision making.* Harvard Business Review, Boston, MA, Estados Unidos, v. 85, n. 11, p. 68-76, nov. 2007.

SPEARS, L. C.; *Character and servant leadership*: ten characteristics of effective, caring leaders. The Journal of Virtues & Leadership, v. 1, n. 1, 2010, p. 25-30, 2010.

STACEY, R. *Complexity and creativity in organizations.* San Francisco: Berrett-Koehler, 1996.

SUTHERLAND, J. *Agile can scale*: inventing and reinventing SCRUM in five companies. Cutter IT Journal, v. 14, p. 5-11, 2001.

SUTHERLAND, J. *Agile development*: lessons learned from the first Scrum. Cutter Agile Project Management Advisory Service: Executive Update, v. 5, n. 20, p. 1-4., 2004.

SUTHERLAND, J. *Agile contracts*: Money for Nothing and Your Change for Free. Scrum Inc., oct. 2008. Disponível em: https://www.scruminc.com/agile-contracts-money-for-nothing-and/. Acesso em: 14 mai. 2016.

TAKEUCHI, H.; NONAKA, I. *The new new product development game.* Harvard Business Review, Boston, MA, Estados Unidos, v. 64, n. 1, p. 137-146, jan./fev. 1986.

TAYLOR, F. W. *The principles of scientific management.* 1911. Kindle ebook, domínio público.

THE STANDISH GROUP. *CHAOS Report.* Decision latency theory: it's all about the interval, 2018.

DE TOLEDO, R. *Humanos sim, Recursos não!* Blog da K21, 2019. Disponível em: https://k21.global/blog/humanos-sim-recursos-nao. Acesso em 15 mai. 2020.

DE TOLEDO, R. *Por que usar "story points"?* Blog Visão Ágil, jan. 2009. Disponível em: http://visaoagil.files.wordpress.com/2009/01/storypoints.pdf. Acesso em: 17 set. 2011.

VERSIONONE. *12th annual state of Agile survey.* 2018. Disponível em: https://explore.versionone.com/state-of-

agile/versionone-12th-annual-state-of-agile-report. Acesso em: 21 abr. 2018.

WAKE, W. *INVEST in good stories, and SMART tasks.* XP123, aug. 2003. Disponível em: http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. Acesso em: 14 dez. 2012.

WILKINSON, M. *The secrets of facilitation*: The S.M.A.R.T. Guide to Getting Results With Groups. San Francisco, CA: Jossey-Bass, 2004.

WOMACK, J. P.; JONES, D. T. *A mentalidade enxuta nas empresas*: elimine o desperdício e crie riqueza. Tradução de Ana Beatriz Rodrigues e Priscilla Martins Celeste. 4. ed. Rio de Janeiro: Campus, 1998.

WOMACK, J. P.; JONES, D. T.; ROOS, D. *A máquina que mudou o mundo.* Tradução de Ivo Korytowski. 17. ed. Rio de Janeiro: Campus, 1992.