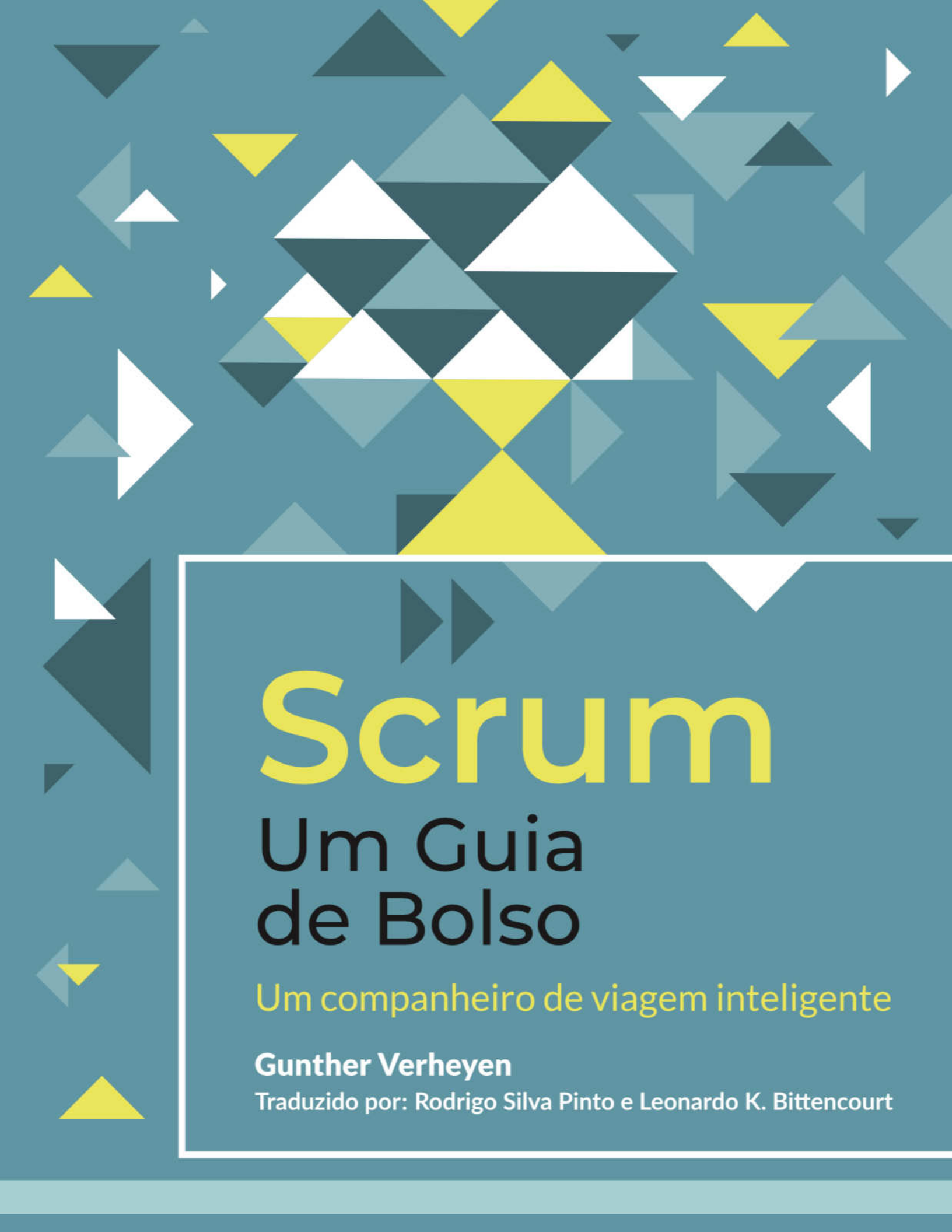




Scrum

Um Guia de Bolso

Um companheiro de viagem inteligente

Gunther Verheyen

Traduzido por: Rodrigo Silva Pinto e Leonardo K. Bittencourt

Scrum - Um Guia de Bolso

OceanofPDF.com

Other editions

English (original):

Scrum – a Pocket Guide (A Smart Travel Companion), 1st edition, 2013

Scrum – a Pocket Guide (A Smart Travel Companion), 2nd edition, 2019

Dutch:

Scrum Wegwijzer (Een kompas voor de bewuste reiziger), 2016

German:

Scrum Taschenbuch (Ein Wegweiser für den bewussten Entdecker), 2017

Portuguese:

Scrum – Um Guia de Bolso (Um companheiro de viagem inteligente), 2019

Italian:

Scrum – Una Guida Tascabile (Un compagno di viaggio smart), 2019

OceanofPDF.com

Scrum

Um Guia de Bolso

Um companheiro de viagem inteligente

OceanofPDF.com

Ficha Técnica

Título:

Scrum - Um Guia de Bolso

Legendas:

Um companheiro de viagem inteligente

Autor:

Gunther Verheyen

Tradução:

Leonardo Bittencourt (Principal Consultant, Ammeon Solutions)

Rodrigo Silva Pinto (Consultant and Trainer, Agile School)

Revisores (versão em inglês):

Ken Schwaber (co-criador do Scrum, Scrum.org)

David Starr (Agile Craftsman, Microsoft)

Ralph Jocham (Agile Professional, Effective Agile)

Patricia M. Kong (Business Agility Enterprise Solutions, Scrum.org)

Blake McMillan (Scrum Master, soulofscrum.com)

Dominik Maximini (Agile Lead, NovaTec Consulting GmbH)

Revisores (tradução):

Leonardo Bittencourt (Principal Consultant, Ammeon Solutions)

Rodrigo Silva Pinto (Consultant and Trainer, Agile School)

Copyright:

Gunther Verheyen

Embora esta publicação tenha sido feita com muito cuidado, nem autor, tradutor ou editora podem aceitar qualquer responsabilidade por danos causados por possíveis erros e/ou incompletude desta publicação.

Nenhuma parte desta publicação pode ser reproduzida em qualquer forma por impressão, impressão fotográfica, microfilme ou outros meios sem permissão por escrito de Gunther Verheyen.

OceanofPDF.com

Prefácio por Ken Schwaber

Um feito extraordinário que ebuli com inteligência.

Scrum – Um Guia de Bolso é um livro extraordinariamente competente. Gunther descreveu tudo sobre o Scrum de forma clara e bem escrita, fluindo com discernimento, compreensão e percepção. No entanto, você não é incomodado por esses atributos. Você simplesmente se beneficia deles, refletindo: “isso foi realmente útil. Eu encontrei o que precisava saber, prontamente entendi o que desejava e não fui incomodado por irrelevâncias.

Eu tenho lutado para escrever este prefácio. Sinto que o prefácio deve ser tão bem escrito quanto o livro o qual descreve. Neste caso, está difícil. Leia o livro de Gunther. Leia em parte ou como um todo. Você ficará satisfeito.

Scrum é simples, mas completo e competente na resolução de problemas complexos. O guia de bolso do Gunther é completo e competente para endereçar a compreensão de um framework simples para resolver problemas complexos, Scrum.

Ken, 22 de agosto de 2013

OceanofPDF.com

Prefácio

O uso de métodos ágeis continua a ganhar força com o Scrum como a definição mais adotada do Ágil. O nível geral de interesse no Scrum já é enorme, mas ainda cresce, dentro e além do desenvolvimento de software.

Transformar a maneira de trabalho de uma organização para o Scrum representa um grande desafio. O Scrum não é um livro de receitas de ‘processo’ com prescrições detalhadas e exaustivas para todas as situações imagináveis. O Scrum é um *framework* de princípios, regras e valores que se desenvolve nas *pessoas* que empregam o Scrum. Um grande potencial do Scrum está na descoberta e *surgimento* de práticas, ferramentas e técnicas e na otimização das mesmas para cada contexto específico.

Os benefícios obtidos através do Scrum dependem da vontade de remover barreiras, atravessar muros e separações e embarcar em uma jornada de descoberta. Scrum é mais sobre comportamento do que sobre processo.

A jornada começa pela compreensão das regras do Scrum para começar a jogá-lo. Este livro aspira a ser seu companheiro ao longo do caminho, todo o caminho. Este livro mostra como Scrum implementa a mentalidade Ágil, quais são as regras do jogo do Scrum e como essas regras deixam espaço para uma variedade de táticas para jogar o jogo. A ambição de introduzir todos esses aspectos é tornar este livro uma leitura válida para pessoas, equipes, gerentes e agentes de mudança, se eles já estão fazendo Scrum ou querem embarcar em sua jornada de Scrum.

Em 2003, minha jornada decolou. Meu caminho de agilidade começou com eXtreme Programming e Scrum. Foi inevitavelmente um caminho de paralelepípedos. Em minha jornada eu tenho usado o Scrum com muitas equipes, em vários projetos e iniciativas, em diferentes escalas e em diferentes organizações. Trabalhei com pequenas e grandes empresas e treinei praticantes individuais e equipes, bem como gerentes executivos. Eu criei a primeira edição deste livro, “Scrum - A Pocket Guide”. Eu estava na posição privilegiada de trabalhar com o Ken Schwaber, co-criador do Scrum, na Scrum.org, enquanto também cuidava da série ‘Profissional’ de

treinamentos, cursos e avaliações do Scrum. Estou satisfeito por poder continuar minha jornada de Scrum como um zelador independente do Scrum.

Esta edição traduzida é baseada na segunda edição do “Scrum - A Pocket Guide”.

Quem teria imaginado que havia demanda para uma segunda edição do meu guia de bolso do Scrum, cinco anos após a criação quase acidental da primeira edição em 2013?

Eu levo em conta como descrevi os valores do Scrum na primeira edição. Em julho de 2016, eles foram adicionados ao Guia do Scrum. Eu descrevi as 3 perguntas tradicionais como uma tática boa, mas opcional, para usar na Daily Scrum. Isso também foi adicionado ao Guia Scrum, em novembro de 2017.

No entanto, mais e maiores desafios surgiram desde 2013. O equilíbrio da sociedade continua drasticamente e rapidamente mudando de trabalho industrial (muitas vezes físico) para trabalho digital (muitas vezes virtual). Em muitos domínios da sociedade, a imprevisibilidade do trabalho aumenta continuamente. O paradigma industrial é proferido inútil, definitivamente. A necessidade do paradigma Ágil é maior do que nunca e, portanto, a necessidade do framework tangível do Scrum para ajudar as pessoas e organizações a aumentar sua agilidade na realização de trabalhos complexos em circunstâncias complexas.

O Scrum está sendo visto como um *framework simples para lidar com desafios complexos*, mais do que uma forma de entregar produtos complexos (software). Mais pessoas pedem orientação e perspectivas sobre sua jornada no Scrum, indo além do desenvolvimento de software. Foi necessária uma descrição mais genérica das regras do Scrum, palavras diferentes, outros ângulos para o mesmo conjunto de regras. As organizações buscam perspectivas claras nas regras simples do Scrum enquanto imaginam reemergir suas estruturas e sua maneira de trabalhar em torno do Scrum. Como a terceira onda do Scrum está crescendo, esta segunda edição introduz ou atualiza a simplicidade do Scrum para aqueles que querem surfar essa onda. Esta segunda edição oferece as perspectivas

fundamentais do Scrum para eles e suas organizações para moldar adequadamente seu Scrum.

Agradeço ao Ken pelo prefácio e sua resenha pela edição original (2013), bem como aos outros revisores Dave Starr, Patricia Kong e Ralph Jocham por seu feedback sobre a primeira edição. Agradeço a Blake McMillan e Dominik Maximini pela muito apreciada revisão desta segunda edição. Agradeço a todos os tradutores pelo esforço passado e presente para espalhar minhas palavras em diferentes idiomas. Agradeço a todos da Van Haren Publishing, e especialmente a Ivo van Haren, por me dar a oportunidade de expressar meus pontos de vista sobre o Scrum com este livro.

Aprecie a leitura e ... keep Scrumming.

Gunther, agosto 2018

OceanofPDF.com

Introdução pelos tradutores

Tenho a oportunidade de formar centenas de pessoas todos os anos em treinamentos de Scrum que vão dos fundamentos a conteúdos mais avançados. Um pedido comum entre os alunos é a indicação de literatura do gênero. Mas a resposta por muito tempo era não satisfatória: “O livro Pocket Guide do Gunther é o melhor, mas só está disponível na língua inglesa”. Havia uma lacuna, faltava uma boa referência literária do Scrum para os falantes de língua portuguesa.

Cansado de esperar, resolvi fazer parte deste projeto, propagando um conteúdo de altíssimo nível e me associando a um dos autores mais influentes do tema, depois dos próprios criadores do framework.

Espero que as horas dispendidas em “Scrum - Um guia de bolso”, possam contribuir com a comunidade Ágil e o mercado brasileiro para juntos construirmos produtos com alto índice de profissionalismo e que gerem impacto necessário para mudar o mundo.

Rodrigo Silva Pinto, Agile School, junho 2019

Tive o prazer de conhecer o Gunther pessoalmente durante Agile Tour Vilnius em 2017. Posteriormente colaborei com a tradução de dois de seus trabalhos para Português, o Glossário Scrum e os Valores do Scrum. Indubitavelmente ele faz juz ao que se auto-intitula, Zelador do Scrum (Scrum Caretaker).

Nesta obra, Gunther usa uma linguagem simples que vai direto ao cerne do Scrum, aborda os pontos cruciais e clarifica o framework Scrum de uma forma cirúrgica. Este livro lhe ajudará a evitar armadilhas, equívocos e adoção de um Scrum mecânico. Você compreenderá o propósito do Scrum Framework bem como os porquês de cada elemento que o compõe.

Manter o conteúdo sem distorções e com a mesma clareza, onde as palavras usadas na versão original foram minuciosamente pensadas, trouxe uma boa dose de desafio extra.

Indico este livro para quem está iniciando e para quem já tem experiência com Scrum. Lhe garanto que durante sua leitura - ou releituras como no meu caso - sempre haverão novas descobertas.

Não perca tempo. Boa leitura e Scrum on!

Leonardo Bittencourt, Principal Lean/Agile Consultant, junho 2019

OceanofPDF.com

Avaliações

Este guia de bolso do Scrum é formidável. É bem organizado, bem escrito e o conteúdo é excelente. Este deve ser o padrão de fato para todos à procura de uma visão completa e clara do Scrum.

(Ken Schwaber, co-criador do Scrum, agosto 2013)

Gunther habilmente embalou o guia certo para equipes que procuram agilidade, sem uma gota de hipérbole. Este é o livro sobre agilidade com Scrum que eu gostaria de ter escrito.

(David Starr, Agile Craftsman, junho 2013)

Durante minhas atividades de treinamento de Scrum muitas vezes me perguntam: “Para Scrum, qual é o livro a ser lido?” No passado, a resposta não era simples mas agora é! Scrum - Um Guia de Bolso é o livro a ser lido ao começar com Scrum. É uma referência concisa, mas completa e apaixonada sobre Scrum.

(Ralph Jocham, Effective Agile, junho 2013)

‘A casa do Scrum é uma casa aconchegante. É uma casa onde as pessoas são bem-vindas. A paixão do Gunther pelo Scrum e seus integrantes é evidente em seu trabalho e em cada capítulo deste livro. Ele explica o paradigma Ágil, expõe o framework Scrum e, em seguida, discute o ‘estado futuro do Scrum’. Intimamente em cerca de 100 páginas.

(Patricia M. Kong, Scrum.org, junho 2013)

Eu recomendo a leitura do “Scrum - Um Guia de Bolso” no início de sua jornada com Scrum para ajudá-lo a ganhar uma compreensão mais profunda do porquê o Scrum funciona e como os valores e os princípios podem impactar positivamente a vida de sua equipe, bem como a saúde de uma organização. Lê-lo mais tarde em sua jornada também é ótimo... exceto pelo sentimento de pesar de não tê-lo lido mais cedo.

(Blake McMillan, Soulofscrum.com, agosto 2018)

É difícil encontrar uma literatura sobre Scrum concisa e direta ao ponto. A maioria dos autores circula em torno dos tópicos centrais em vez de nomeá-los. Gunther optou por quebrar este padrão, esclarecendo-nos com o conhecimento das partes verdadeiramente importantes do Scrum. Ao iniciar a sua jornada no Scrum, certifique-se de levar uma cópia deste guia junto com você.

(Dominik Maximini, NovaTec Consulting GmbH, agosto 2018)

OceanofPDF.com

Índice

[1. O PARADIGMA ÁGIL](#)

[1.1 Mudar ou não mudar](#)

[1.2 As origens do Ágil](#)

[1.3 Definição de Ágil](#)

[1.4 O contínuo iterativo-incremental](#)

[1.5 Agilidade não pode ser planejada](#)

[1.6 Combinando Ágil e Lean](#)

[2. SCRUM](#)

[2.1 A casa do Scrum](#)

[2.2 Scrum, o que há em um nome?](#)

[2.3 É um gorila que eu vejo lá?](#)

[2.4 Framework, não metodologia](#)

[2.5 Jogando o jogo](#)

[2.6 Os princípios fundamentais do Scrum](#)

[2.7 Os Valores do Scrum](#)

[3. TÁTICAS PARA UM PROPÓSITO](#)

[3.1 Visualizando o progresso](#)

[3.2 As perguntas da Daily Scrum](#)

[3.3 Refinamento do Product Backlog](#)

[3.4 Histórias de Usuário](#)

[3.5 Planning Poker](#)

[3.6 Duração da Sprint](#)

[3.7 Como escalar o Scrum](#)

[4. O ESTADO FUTURO DO SCRUM](#)

[4.1 Sim, fazemos Scrum. E...](#)

[4.2 O poder do produto possível](#)

[4.3 A adoção upstream do Scrum](#)

[ANEXO A: GLOSSÁRIO DO SCRUM](#)

[ANEXO B: REFERÊNCIAS](#)

[SOBRE O AUTOR](#)

[SOBRE OS TRADUTORES](#)

1. O Paradigma Ágil

1.1 Mudar ou não mudar

A indústria de software foi por muito tempo dominada por um paradigma *Industrial* de opiniões e crenças. Isso era de fato um copia-e-cola de velhas rotinas e de teorias da manufatura. Um



Figura 1.1 O Paradigma Industrial

elemento essencial nesta perspectiva de conhecimento, visões e práticas foi a convicção Taylorista^[1] de que ‘trabalhadores’ não são confiáveis para realizar seu trabalho de forma inteligente, autônoma e criativa. Espera-se apenas que sejam executores de tarefas pré-definidas. O seu trabalho deve ser preparado, projetado e planejado por funcionários mais graduados. Em seguida, supervisores hierárquicos devem vistoriar a execução dessas tarefas cuidadosamente preparadas. A qualidade é assegurada aceitando os lotes produzidos que são bons e rejeitando os ruins. Recompensas financeiras são usadas para estimular o comportamento desejado. Comportamento indesejado é punido. A velha estratégia de ‘incentivos e penalizações’ (cenoura e vara).

As graves falhas do velho paradigma no desenvolvimento de software são conhecidas e bem documentadas. Em particular, o Chaos Report do Standish Group [Standish, 2011; Standish, 2013] tem repetidamente revelado as baixas taxas de sucesso do desenvolvimento de software tradicional. As deficiências e erros resultantes da aplicação do paradigma

industrial no desenvolvimento de software estão bem além dos níveis razoáveis de tolerância. A resposta infeliz a isso parece ter sido a de diminuir as expectativas. Se tornou aceitável apenas 10-20% dos projetos de software serem bem-sucedidos. A definição de sucesso no paradigma industrial é composta pela combinação no tempo, dentro do orçamento e contemplando todo o escopo. *Embora estes critérios para o sucesso possam ser contestados, é a promessa desse paradigma.* Tornou-se aceitável a qualidade baixa e que mais de 50% das funcionalidades dos aplicativos de software entregues nunca são usadas [Standish, 2002; Standish, 2013].

Embora não seja amplamente e conscientemente admitido, o paradigma industrial colocou a indústria de software em uma crise grave. Muitos tentaram superar esta crise fortalecendo a abordagem industrial. Mais planos foram feitos, mais fases agendadas, mais modelos criados, mais trabalho foi feito antecipadamente, esperando que o trabalho fosse executado mais eficazmente. O trabalho antecipado foi aumentado exaustivamente. Permaneceu a ideia central que os trabalhadores precisavam ser dirigidos, porém com instruções ainda mais detalhadas. A supervisão foi aumentada e intensificada. *Quando as taxas de sucesso não aumentam, o paradigma industrial pressupõe que as instruções não estão claras e detalhadas o suficiente.*

Ainda assim, pouco foi melhorado. Muitas falhas, defeitos e baixa qualidade permaneceram e tiveram de ser tolerados.

Levou algum tempo, mas inevitavelmente novas ideias e percepções começaram a se formar ao se observar as significativas anomalias do paradigma industrial.

As sementes de uma nova visão de mundo já estavam semeadas na década de 1990. Mas foi em 2001 que estas resultaram na nomeação formal de “Ágil”, um ponto de inflexão na história do desenvolvimento de software. Um novo paradigma nasceu no meio da indústria de software mas ao mesmo tempo se expandindo para outros domínios da sociedade. É um paradigma que prospera sobre heurísticas e criatividade, um paradigma que prospera sobre o (restaurado) respeito pela natureza criativa do trabalho e pela inteligência dos “trabalhadores”.

A indústria de software tem boas razões para manter-se movendo para o novo paradigma; as falhas existentes são significativas, amplamente conhecidas e a presença de software na sociedade cresce exponencialmente, tornando-o um aspecto crítico do nosso mundo moderno. No

entanto, por definição, uma mudança para um novo paradigma leva tempo. E o velho paradigma parece ter raízes profundas e um tempo de semivida considerável. A abordagem industrial para o desenvolvimento de software continua a ser ensinada e promovida como a mais adequada.

Muitos dizem que o Ágil é muito radical e, portanto, encorajam uma introdução gradual de práticas ágeis dentro das estruturas tradicionais existentes. No entanto, há motivos para sermos muito céticos acerca de tal evolução gradual, uma progressão lenta do antigo paradigma para o novo, do Cascata ao Ágil.

As chances são altas que uma evolução gradual nunca irá além da superfície, não fará mais do que apenas arranhar a superfície. Novos nomes são introduzidos, novos termos e novas práticas são impostos, mas o pensamento fundamental e o comportamento das pessoas e organizações permanecem os mesmos. As falhas essenciais permanecem intocadas;



Figura 1.2 O Paradigma Ágil

especialmente o desrespeito às pessoas, que leva à uma abordagem a pessoas criativas e inteligentes como trabalhadores ignorantes, como “recursos”.

A preservação dos fundamentos tradicionais manterá os dados, métricas e padrões existentes no mesmo lugar, e o novo paradigma será medido contra esses padrões antigos. Paradigmas diferentes em sua natureza, no entanto, consistem em conceitos e ideias fundamentalmente diferentes, geralmente mutuamente exclusivos. Nenhuma comparação significativa entre o paradigma industrial e o Ágil é possível. É necessária a honestidade de aceitar as falhas graves no formato antigo. São necessários liderança, visão, empreendedorismo e persistência para abraçar a nova forma, abandonando assim o pensamento antigo.

Uma mudança gradual é factualmente uma situação de status quo que mantém o paradigma industrial intacto.

Há evidência esmagadora de que o velho paradigma não funciona. A maioria das evidências sobre o Ágil costumava ser anedótica, pessoal ou relativamente menor. O relatório Chaos de 2011 do Standish Group [Standish, 2011] marcou o ponto da virada, por conter, pela primeira vez, resultados de pesquisa claros e que foram confirmados em todos os relatórios posteriores. Pesquisas extensivas foram feitas na comparação de projetos tradicionais com projetos que utilizavam métodos Ágeis. O relatório mostra que uma abordagem Ágil resulta em um rendimento muito maior, mesmo contra as velhas expectativas de que o software deve ser entregue a tempo, no orçamento e com todo o escopo prometido. O relatório mostra que os projetos Ágeis foram três vezes mais bem-sucedidos, e houve três vezes menos projetos Ágeis que falharam em comparação aos projetos tradicionais. Para grandes projetos no entanto, as mudanças nas taxas de sucesso foram menos expressivas, o que é mais provável estar relacionado às expectativas erradas, como por exemplo a combinação tempo+escopo+preço. Frente às expectativas certas, com foco na colaboração ativa do cliente e entrega frequente de valor, o novo paradigma performa ainda melhor, com fatias verticais de valor, frequentemente entregues, para superar o problema de volume.

No entanto, Ágil é uma escolha, não uma obrigação. É uma maneira de melhorar a indústria de software. Pesquisas mostram que é um paradigma mais bem-sucedido.

! Scrum te ajuda.

As regras distintas do Scrum ajudam a pôr em prática o novo paradigma. O pequeno conjunto de prescrições permite uma ação imediata e resulta em uma absorção mais frutífera do novo paradigma. O Scrum é uma forma tangível de adotar o paradigma Ágil. Usando Scrum, os indivíduos desenvolvem novas formas de trabalho; através da descoberta, aprendizagem baseada em experiências e colaboração. Elas conhecem um novo jeito de ser, um estado de *Agilidade*; um estado de constante mudança, fluxo, evolução e adaptação. Este processo ajuda as organizações a adotar um *estado* de agilidade, liberando tempo, pessoas e energia para serem inovadoras (novamente).

No entanto, apesar de sua praticidade, a experiência mostra que adotar o Scrum muitas vezes representa um salto gigantesco. Isso pode ser devido à incerteza gerada pelo abandono das velhas certezas, mesmo que essas velhas certezas se provaram não confiáveis. Pode ser pelo tempo necessário para se fazer uma mudança substancial. Pode ser dado a determinação e o trabalho duro que são exigidos. Novamente é sabido que o Scrum é simples, mas não é fácil.

1.2 As origens do Ágil

Apesar do domínio da abordagem orientada ao plano e da visão industrial, a abordagem evolucionária para o desenvolvimento de software não é nova. Craig Larman descreveu extensivamente os antecessores históricos do Ágil em seu livro “Agile & Iterative Development, A Manager’s Guide” [Larman, 2004].

Mas o termo oficial “Ágil” data de fevereiro de 2001, quando 17 líderes de desenvolvimento de software se reuniram no resort de esqui Snowbird em Utah. Eles discutiram suas opiniões sobre desenvolvimento de software em tempos que as abordagens cascata falhavam e eram substituídas por pesadas implementações RUP (“Rational Unified Process”), que não levaram a melhores resultados do que os processos tradicionais. Estes líderes de desenvolvimento estavam seguindo diferentes caminhos e métodos, sendo cada um deles uma implementação distinta do novo paradigma; Scrum, eXtreme Programming, Adaptive Software Development, Crystal, Feature Driven Development, etc.

A reunião resultou em atribuir o rótulo “Ágil” aos princípios, crenças e pensamentos comuns desses líderes e seus métodos. Eles foram publicados como o “Manifesto para Desenvolvimento Ágil de Software” [Beck, et.al., 2001].

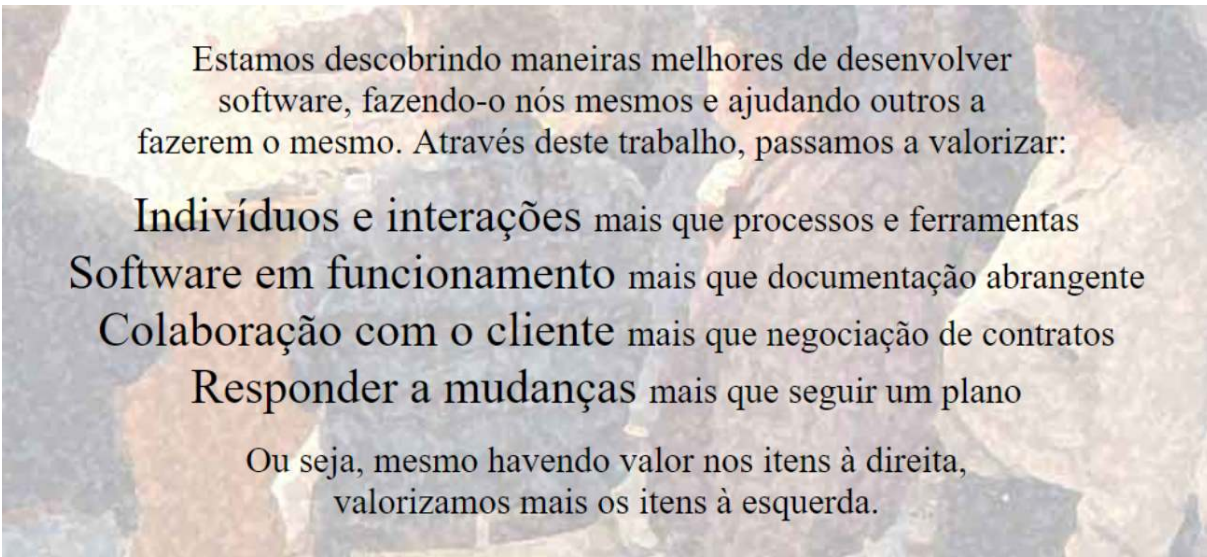


Figura 1.3 O Manifesto Ágil

Frequentemente, o desejo de “fazer Ágil” pode ser ouvido. Muitas vezes é o desejo de uma solução mágica, um outro processo bala de prata que resolve todos os problemas. Isso me faz frequentemente afirmar que “*Ágil não existe*”. Ágil não é um processo fixo, método ou prática. Ágil é a coleção de princípios que os métodos para desenvolvimento de software Ágil têm em comum. Ágil refere-se à mentalidade, às convicções e às preferências expressas no Manifesto para o Desenvolvimento Ágil de Software.

O manifesto ajuda a compreender as ideias subjacentes ao Ágil. Para usá-lo como uma fonte e obter uma compreensão mais profunda do Ágil, eu recomendo fortemente dar olhada nos 12 princípios que estão por trás dos 4 valores, consulte <http://agilemanifesto.org/principles.html>.

1.3 Definição de Ágil

Na ausência de uma definição específica concisa, eu prefiro descrever “Ágil” em três características chave. Estas são as características que são comuns ao portfólio de Métodos Ágeis e são típicas de uma forma Ágil de trabalhar:

- ❖ Orientado à Pessoas;
- ❖ Processo Iterativo-incremental;
- ❖ Valor, a medida do sucesso.

1.3.1 Orientado à Pessoas

Ágil não é orientado à um plano preditivo de como implementar requisitos que foram exaustivamente analisados, projetados e arquitetados de forma antecipada. Ágil atesta que os requisitos não podem ser previstos em todos os detalhes possíveis de maneira antecipada.

Ágil não é um processo de entrega de diferentes tipos de produtos intermediários para diferentes departamentos especializados, onde cada departamento realiza seu trabalho especializado isoladamente.

Ágil é orientado pela *colaboração* contínua das pessoas que trabalham em todos os departamentos necessários; sejam eles negócio, TI, marketing, vendas, atendimento ao cliente, operações ou gestão. Ágil certamente não reconhece a tradicional discórdia negócios versus TI. Ambos são necessários para o sucesso, a partir da perspectiva de criar produtos de software utilizáveis e úteis, software de alto valor.

Colaboração, interação e conversação pedem um estilo de gestão diferente para ser eficaz. Times Ágeis são *facilitados* por meio da liderança servidora. Limites e um contexto para a autogestão existem, sobre eles são dados objetivos e direção às equipes. Um controle sutil emerge dos limites.

Colaboração e facilitação substituem os mecanismos tradicionais de comando e controle de instrução de micro tarefas executáveis à indivíduos

em uma base diária e autoridades totalitárias para controle invasivo.

Isso expressa como as pessoas são respeitadas por sua criatividade, inteligência e capacidades de auto-organização. As pessoas são respeitadas pela sua capacidade de compreender e resolver um problema sem estarem sobrecarregadas com toneladas de cerimônia e burocracia. Uma sobrecarga de cerimônias apenas substitui o pensamento colaborativo, a inovação e a responsabilização das pessoas com burocracia, resultados no papel, transições e desculpas administrativas.

As pessoas são respeitadas no tempo que podem gastar em seu trabalho por meio da ideia de *Ritmo Sustentável*. O trabalho é organizado de tal forma que o tempo é sustentável, indefinidamente.

1.3.2 Processo Iterativo-Incremental

Processos Ágeis não são abordagens desordenadas. Processos Ágeis são *definidos* e requerem alta *disciplina*.

Os produtos são criados peça por peça ('incremental') com cada peça composta de expansões, melhorias, eliminações e modificações. As peças construídas e o produto total são frequentemente revisitados ("iterativo") para assegurar a integridade geral.

Ágil requer atenção explícita de todos os envolvidos em qualidade e excelência. Ágil substitui a ideia de que estes podem simplesmente serem despejados em documentos e descrições de papel que têm propriedades muito diferentes do resultado final previsto, o produto lançável.

A necessidade de um processo iterativo-incremental aumenta pela conclusão que requisitos e implementações são propensos a mudar, não importa quanto tempo, energia e dinheiro são gastos para prevê-los antecipadamente. Mercados e concorrentes evoluem, usuários só sabem o que querem ao utilizarem, as estratégias corporativas mudam, para citar apenas alguns. Isso exige uma consciência extrema e abertura à mudança.

Contrário ao processo preditivo, a mudança não é excluída do processo Ágil nem expulsa para os arredores das cerimônias de desenvolvimento. Novas

ideias, evolução de opiniões e prioridades alteradas formam o coração vivo do Ágil. O Ágil prospera na emergência, a emergência e subsequente evolução gradativa dos requisitos, planos, ideias, arquiteturas e desenhos. A mudança não é prejudicial porque forma uma parte natural da maneira de trabalhar. Ágil *encoraja* a mudança como fonte de inovação e melhoria.

1.3.3 Valor, a medida do sucesso

O progresso no desenvolvimento não pode ser medido e garantido com base no mero cumprimento de planos preditivos e marcos intermediários, documentos, transições, assinaturas, aprovações ou outras obrigações cerimoniais, como acontece no paradigma industrial. O Ágil introduz novas formas de medir o progresso e o sucesso.

O Ágil torna explícito que progresso e sucesso só podem ser determinados pela inspeção frequente de *versões do produto funcionando* (não descrições intermediárias dele) e o real *valor* que o produto representa para as pessoas que virão a usá-lo.

É natural no desenvolvimento de produto que as pessoas que usarão o mesmo só terão certeza sobre usabilidade e utilidade quando realmente colocarem suas mãos nele. Nenhuma documentação de papel ou processo virtual podem substituí-lo. Isso é um convite a um processo iterativo-incremental para fechar o ciclo de feedback com os usuários regularmente e medir o impacto e a apreciação como uma fonte de informações para novas evoluções do produto.

1.4 O contínuo iterativo-incremental

Uma abordagem Ágil fatia o tempo em iterações time-box, períodos com início e fim fixos. Há muitas vantagens na técnica do time-box, com *Foco* sendo uma delas. Esta técnica de gestão do tempo permite a absorção de pivôs ou alterações disruptivas, bem como assegura verificações regulares de modo que as lições aprendidas possam ser incorporadas de uma iteração na seguinte, numa situação de fluxo mais contínuo. O objetivo principal de cada iteração é criar versões valiosas do produto funcionando o mais tardar ao final dela, com intuito de coletar feedback e permitir o aprendizado antecipado.

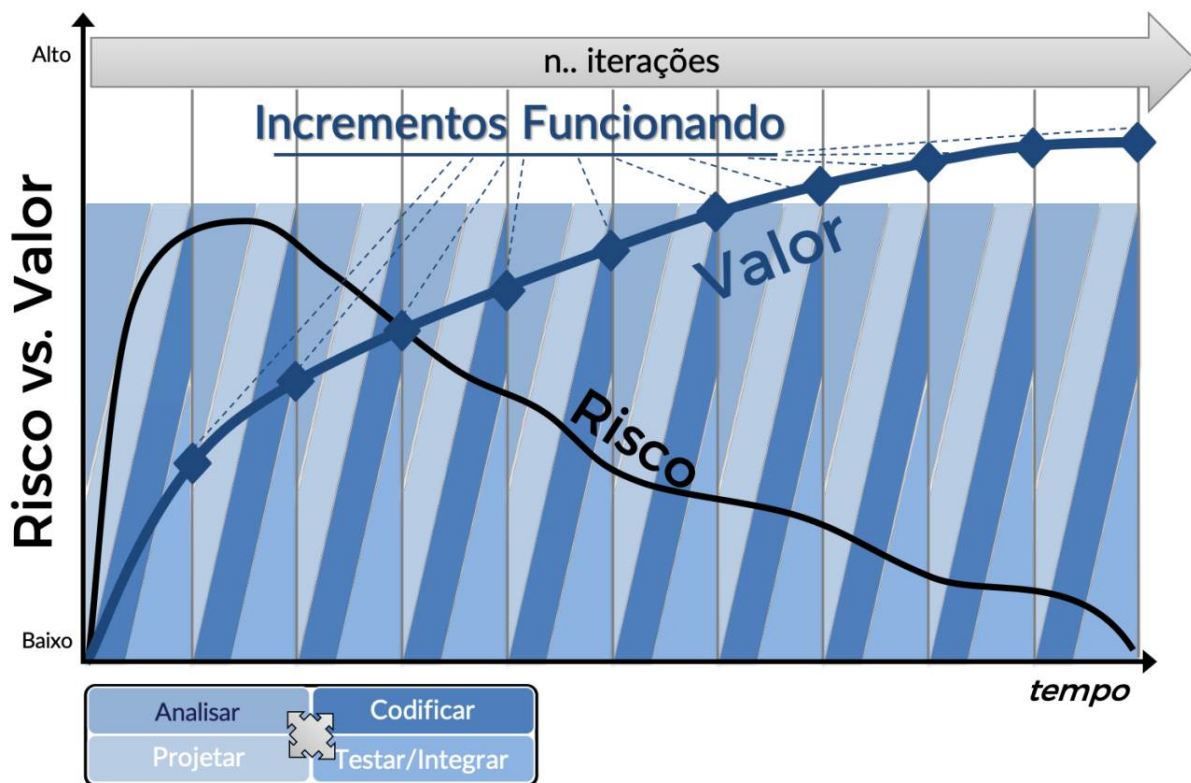


Figura 1.4 A Entrega de Valor Ágil

No Ágil, todo trabalho de desenvolvimento é reorganizado para otimizar a habilidade de responder e capitalizar as *oportunidades de negócio*.

“Valor” é a resposta para demandas do usuário, mercado e negócio e a medida global de progresso e sucesso. Valor é uma suposição interna dentro da organização até que o Produto seja realmente lançado para o mercado. Lançar versões de produtos para o mercado é a única maneira de validar esta suposição de valor. Lançar para o mercado regularmente é a única maneira de se adaptar ao feedback e apreciação, ou falta dos mesmos, do mercado. Isto é feito em evoluções subsequentes do produto. O valor é continuamente otimizado entre as iterações. O risco é controlado pela produção consecutiva de incrementos funcionando com base na definição padrões de desenvolvimento.

“Risco” também se relaciona com a perspectiva de negócios. Tenha em mente que, certamente em um contexto de TI, o risco é normalmente definido como algo técnico (*O sistema irá funcionar? O sistema é escalável?*). Abrange o aspecto de um produto *utilizável* (*Será tecnicamente estável? Vai quebrar?*). Mas uma mera perspectiva sobre o risco que é técnica ou de desenvolvimento, muitas vezes ignora o fato de que o objetivo final do Desenvolvimento Ágil é proporcionar maior satisfação aos clientes e usuários finais para garantir que os produtos *lhes* são *úteis*. Um produto sendo utilizável a partir de uma perspectiva técnica de risco é apenas o começo.

Qualquer processo de desenvolvimento moderno deve resolver o risco de não ser capaz de capitalizar sobre oportunidades de mercado imprevistas e previamente desconhecidas, de não lançar o produto rápido o suficiente, de estar sujeito a insatisfação do cliente, por exemplo lançando uma versão não testada, o risco de entregar funcionalidades que não são o que os usuários esperam ou apreciam, o risco de ficar para trás da concorrência.

O Desenvolvimento Ágil é organizado de forma mitigar esse risco (de negócio) ao máximo. As necessidades de alto valor são respondidas primeiro. Versões de produto e atualizações são lançadas rapidamente e frequentemente. Satisfazem as necessidades existentes, bem como incluem funções inovadoras e inesperadas. Elas fazem com que os usuários paguem pelo produto e otimizem o retorno para os stakeholders. Elas são de alta qualidade, a fim de minimizar a manutenção e suporte, otimizando o Custo Total de Propriedade (“CTP”).

O Ágil reconhece o objetivo principal das atividades ‘normais’ de TI (na Figura 1.4 grosseiramente representado como Análise, Projeto, Codificação e Teste/Integração), mas quebra sua organização sequencial. Para produzir versões do produto lançáveis com o dinamismo certo e obter os benefícios mais cedo, as atividades são estruturalmente reorganizadas. O objetivo é habilitar a flexibilidade e a velocidade em vez de bloqueá-las. No Ágil todas essas disciplinas são executadas de forma incremental, não-linear, concorrente e numa base diária, por equipes multifuncionais com colaboração contínua e negociação sobre ideias, técnicas e práticas emergentes.

O objetivo de uma abordagem integrada e multifuncional é construir qualidade intrínseca e evitar defeitos, em vez de tentar estabelecer qualidade por uma abordagem de caça bugs numa fase pós-criação. É imperativo transformar o desejo de lançar regularmente na real capacidade de fazê-lo. Qualidade não pode ser adicionada a um produto acabado. Atrasos e orçamentos crescem fora de controle quando falta de qualidade é identificada após o processo de criação.

Visando os benefícios reais e duradouros do Desenvolvimento Ágil requer ir além das fronteiras do departamento de TI (ou similares técnicos). A maneira como o Ágil não só abraça e incorpora a mudança, mas a incentiva, provavelmente desafiará grande parte de uma organização. Mas é mais que uma obrigação; é uma *oportunidade*. Uma organização inteira vai prosperar ao adotar a forma Ágil de trabalhar, com seus ciclos curtos, resultados frequentes e adaptações evolutivas. A visão e abordagem Ágil permitem departamentos e partes maiores da organização a finalmente parar de tentar prever o imprevisível. Ágil incorpora lidar com respostas, soluções e ideias concorrentes que emergem *enquanto* se entrega o produto.

Pode levar algum tempo para experimentar o fato de que a aprendizagem contínua inata no Ágil de fato aumenta o controle em meio a organizações, negócios e circunstâncias de mercado turbulentos. Pode levar algum tempo para mudar o foco da gerência em julgar o passado, por exemplo, por meio de registros de efetivos e de tempo. Pode levar algum tempo para ganhar a confiança sobre a otimização e entrega de valor de negócio por meio de incrementos do Processo de Desenvolvimento Ágil.

Pode levar algum tempo para aceitar que a agilidade leva tempo. Pode levar algum tempo para aceitar que a agilidade não precisa ser analisada, projetada e planejada antes que uma transformação possa decolar.

OceanofPDF.com

1.5 Agilidade não pode ser planejada

Agilidade é o *estado* vislumbrado pela transição para uma maneira Ágil de trabalhar. Agilidade é um estado de fluxo contínuo, alta capacidade de resposta, velocidade e adaptabilidade. É um estado necessário para lidar com a imprevisibilidade, tão comum ao trabalho criativo de desenvolvimento de produto e aos mercados dinâmicos que as organizações operam.

Agilidade não tem propósito se as características acima referidas de fluxo, capacidade de resposta, velocidade e adaptabilidade não são expandidas à relação da organização e seus mercados, comunidades e clientes. A adoção de Métodos Ágeis é um fundamento importante para tal agilidade *empresarial* em larga escala. Da adoção do Ágil, novos processos emergem juntamente com uma nova cultura organizacional de aprendizagem, melhoria e constante adaptação, e o respeito restaurado pelas pessoas. Ao longo da adoção, aprendizados importantes são enraizados e injetados no DNA da organização.

Existem algumas verdades básicas que são fundamentais para definir as expectativas corretas para qualquer transformação em direção a um estado crescente de Agilidade. A introdução de Métodos Ágeis sem aceitar essas verdades essenciais fecha a porta para o aumento da agilidade ao invés de transformá-la em uma porta de entrada de oportunidades:

- ❖ Agilidade não pode ser planejada;
- ❖ Agilidade não pode ser ditada;
- ❖ Agilidade não pode ser copiada;
- ❖ Agilidade não tem um estado fim.

Uma maneira planejada para introduzir os Métodos Ágeis induz a expectativas inadequadas e desfavoráveis. A introdução de Métodos Ágeis é a introdução de um novo paradigma e causará um tumulto organizacional significativo. Procedimentos, departamentos e funções existentes são impactados. A mudança é altamente complexa e, portanto, não previsível. Numa transformação para uma maneira Ágil de trabalhar, não há nenhuma

maneira de prever quais necessidades de mudança serão encontradas em que ponto no tempo, como estas serão tratadas, como as novas formas de trabalho serão absorvidas e qual será o resultado exato de forma a planejar e controlar os próximos passos. Não há maneira de prever o ritmo em que a mudança se espalhará e criará raiz.

Agilidade em si é muito mais do que seguir um novo processo. É sobre comportamento, é sobre mudança *cultural*. A decisão de mover-se para Agilidade é uma decisão de deixar o velho formato (industrial) para trás. Não se trata apenas de aceitar, mas também de celebrar o fato de que a Agilidade é viver a arte do possível. Requer coragem, honestidade e determinação de atuar no presente, agindo sobre a realidade que é exposta pelo progresso iterativo-incremental. Agilidade é fazer o melhor possível em todos os momentos, restrito aos meios que temos e enfrentando as limitações que surgem. Uma maneira planejada para uma transformação Ágil ignora a essência da Agilidade, a de lidar com a complexidade por meio de passos bem ponderados de experimentação, aprendizagem e progresso. Planos temporais e tentativas de planejamento da agilidade que vão contra às restrições de tempo simplesmente estendem o pensamento antigo. É até mesmo contraproducente, pois um plano vai ativamente desacelerar o processo de transformação, pois atrasos graves e tempos de espera são incorporados.

Os planos temporais criam a ilusão de prazos e um último estado final. Agilidade não tem estado final. A Agilidade é um estado de melhoria contínua, um estado em que todo o status quo é desafiado por nossa própria vontade ou por turbulência externa.

Agilidade é um estado único e que evolui continuamente, que reflete as lições e aprendizados que uma organização passou e passa, a forma como aborrecimentos e obstáculos específicos foram superados, as muitas inspeções e adaptações que ocorreram ao longo da jornada. Agilidade é uma assinatura única, com impressões das pessoas, relações, interações, ferramentas, processos, práticas, construtos, dentro e por meio dos muitos ecossistemas que existem dentro de uma organização. Nenhum modelo pode prever, esboçar ou capturar tal assinatura única. Agilidade é um caminho que requer visão, crença, persistência e... trabalho duro. Agilidade

como um estado de alta adaptabilidade alcançada por meio de adaptações regulares no trabalho real produzindo resultados observáveis. O que funciona hoje pode não funcionar amanhã. O que funciona para um conjunto de times, tecnologia e negócios podem não funcionar para outra combinação. Inspeção sem adaptação, em um mundo de complexidade, criatividade, concorrência feroz e imprevisibilidade, é inútil. Adaptação sem observação é falha em direção.

Vivendo a arte do possível contra resultados imprevisíveis engaja as pessoas e acelera uma transformação à medida que molda o futuro, prospera sobre o estado indeterminado do futuro e o que ele pode trazer. É um futuro brilhante para as organizações que têm a visão, determinação e dedicação; a coragem de afastar-se de seguir um plano ou copiar um modelo.

Estas verdades básicas devem estar no coração e na mente de cada pessoa que gerencia, guia, facilita ou lidera uma transformação baseada na mentalidade Ágil. E mesmo assim, ainda leva tempo para que a Agilidade se estabeleça nos corações e mentes das pessoas impactadas pela transformação. Afinal, as pessoas têm sido instruídas no comportamento errado por meio do paradigma industrial por décadas ou às vezes mais.

OceanofPDF.com

1.6 Combinando Ágil e Lean

No Lean, bem como no Ágil, é vital estar ciente de que é um conjunto de ferramentas de pensamento, uma coleção de princípios entrelaçados que educam, motivam, valorizam e guiam as pessoas a otimizar continuamente o seu trabalho e a forma como trabalham. Os princípios do Lean formam as alavancas de um sistema que as pessoas podem usar para criar melhores produtos mais rapidamente, porém de forma sustentável e respeitosa. É um sistema que recompensa as pessoas por fazerem o melhor que podem com os meios e ferramentas que são dados em sua situação atual.

Não há um processo Lean unificado, definitivo, totalmente desenvolvido, de tamanho único, para desenvolvimento de produto, nem para manufatura, com fases predefinidas e prescritas, papéis, artefatos, entregáveis, etc. Um processo Lean deve ser projetado sobre princípios e pensamentos subjacentes, e estar constantemente sintonizado com a situação atual. Lean é sobre adaptabilidade. O

documento online ‘Lean Primer’ de Bas Vodde e Craig Larman [Larman & Vodde, 2009] faz um trabalho excelente de introduzir as raízes do Lean junto a seus princípios e pensamentos.

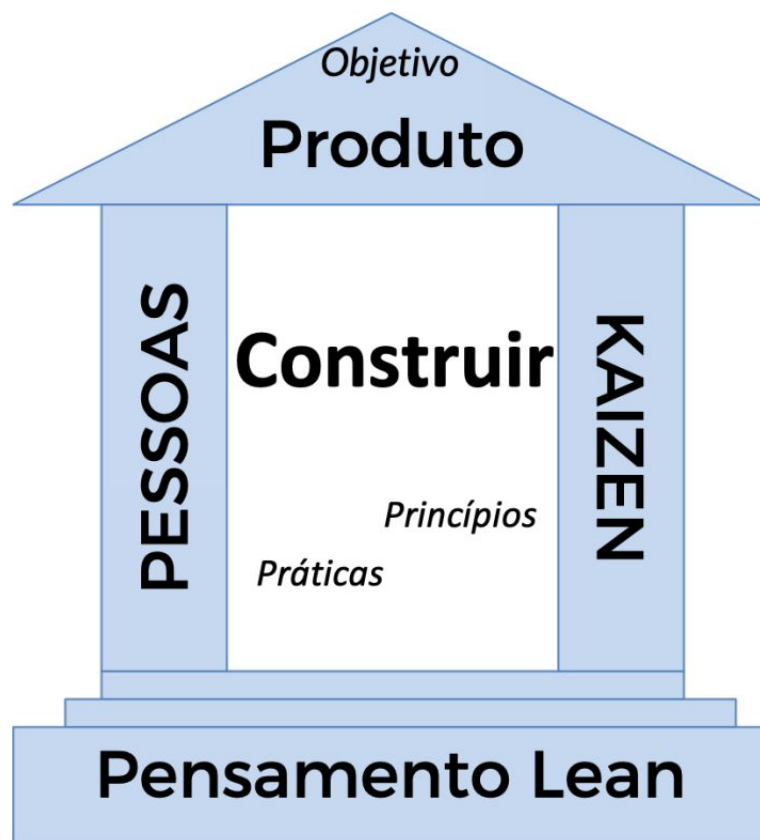


Figura 1.5 O Templo Lean

1.6.1 Aspectos principais do Lean

Pessoas

A alicerce de qualquer sistema que afirma ser Lean são as *Pessoas*. E ‘pessoas’ refere-se a todos os atores possíveis em todo o ecossistema de Desenvolvimento de Produtos Lean: clientes, trabalhadores, times, fornecedores e gerentes; internos e externos.

Todas as pessoas contribuem à sua maneira e por seus próprios meios para fabricação de um produto. Eles colaboram através de habilidades para evitar transições, atrasos e tempo de esperas. Eles tomam decisões autonomamente. Eles se concentram na coleta de conhecimento e aprendizagem constante. Gestores atuam como professores e têm o compromisso de *ir-ver* o chão de fábrica. Promovem o sistema de pensamento Lean; ajudam as pessoas a entender como refletir sobre seu trabalho, os resultados do seu trabalho e como construir produtos melhores. Todo o sistema encarna o espírito do *Kaizen*, a atitude de pensar continuamente sobre o processo, o produto e possíveis melhorias. Cada membro no sistema pode ‘*parar a linha*’ ^[2] se ocorrer algum problema. A raiz do problema será identificada e as contramedidas serão propostas ou instaladas.

Todos os envolvidos na cadeia de valor trabalham de forma integrada. Relacionamentos com fornecedores e parceiros externos não se baseiam na abordagem tradicional de grandes aquisições de volumes, grandes rodadas de negociação e pressão de uns sobre os outros. É tudo sobre a construção de relações de compartilhamento mútuo dos lucros (e riscos). Os contratos Lean incorporam crescimento mútuo.

Desperdícios

Ao considerar o assunto de desperdício, é importante dizer que *evitar* desperdícios, através de melhorias contínuas e otimizações de pequenas etapas, é a opção preferida. Além disso, “desperdícios” refere-se às etapas do processo, não às pessoas como uma desculpa para livrar-se delas.

Obviamente, não importa quanta atenção é dedicada para evitá-lo, o desperdício pode e vai se infiltrar. O espírito Kaizen impulsiona todas as pessoas a serem comprometidas, conscientes e críticas em seu trabalho diário. É uma reflexão natural.

Uma prática para identificar desperdícios estruturais é o *Mapeamento de Fluxo de Valor*. Todas as etapas e fases do processo de vão da “ideia” ao “dinheiro” são estabelecidos em uma linha do tempo. As atividades podem ser rotuladas como “de valor” ou “não geradoras de valor”, mas possivelmente também como necessárias, embora não diretamente geradoras de valor. A *Relação de Valor* pode ser calculada como a proporção de tempo gasto em atividades de agregação de valor versus atividades que não agregam valor. É uma figura que pode servir como uma linha de base contra a qual a melhoria pode ser medida. Mas, como em todas as atividades de melhoria, não há objetivo final definitivo, nenhum estado final. A melhoria em si é o objetivo.

Estoque, WIP e Fluxo

O Lean se empenha em continuidade e fluxo. Superprodução e estoque excessivo interrompe o fluxo e pode atrasar a descoberta e resolução de problemas de qualidade. Mas também é desrespeitoso, pois obriga as pessoas a fazer um trabalho que pode de fato nunca ser usado. O estoque é dispendioso e torna a organização suscetível a desperdícios.

O Lean diz para limitarmos o Trabalho em Progresso (e o custoso estoque) produzindo apenas materiais para serem usados mais a frente na linha num modo “Just in Time”, i.e., quando há um sinal de *Puxar* a partir dos próximos passos no processo. Um *kanban* é um cartão de sinal físico com esta função em sistemas de fabricação. Um kanban é anexado a um inventário de peças. Ele está ligado a um nível de estoque. As peças novas são produzidas somente quando muito material for usado e o cartão de sinal aparecer.

1.6.2 Implementação Lean

Bem como com Ágil, muitas organizações têm dificuldades com Lean. E além disso, as organizações têm dificuldades com a combinação de Ágil e Lean.

Em geral, as empresas se referem a problemas organizacionais quando expressam o desejo de serem ‘Lean’. Se eles querem se tornar ‘Ágeis’, por outro lado, é mais provável que estão referindo-se a problemas com Desenvolvimento de Produto. No entanto, nem Ágil nem Lean oferecem soluções mágicas, de prateleira ou balas de prata.

Infelizmente, o Lean é frequentemente limitado a *eliminar desperdício*. Saliento que olhar um elemento da caixa de ferramentas em apenas um aspecto é um indesejável excesso de foco, ao invés de olhar o todo. Fica ainda pior quando o princípio em si é quebrado, e quando ‘eliminação’ é aplicada à (livrar-se de) *pessoas* e não como um meio de melhorar *processos e estruturas*. A prática de gestão altamente popular de ‘corte de custos’ tende a distorcer essa prática importante do Lean, interpretando o trabalho das pessoas como ‘sobrecarga’, ou seja, não-valiosa. O sinal subjacente é que as pessoas que estão fazendo esse trabalho são desperdício e... descartáveis.

A partir desse equívoco popular e sua perspectiva demasiadamente limitada do Lean, há uma longa jornada para construção do entendimento que Lean é principalmente sobre respeito às pessoas a fim de otimizar valor e qualidade. Que Lean é mais sobre o *contexto* em que as pessoas podem prosperar a fim de performar, do que sobre continuamente enfatizar a necessidade de resultados e desempenho. Invoca o difícil exercício de substituir o “comando e controle”, comportamento do grande chefe, micro gerenciamento, excesso de alocação e nano-atribuições por mecanismos de tomada de decisão descentralizada.

É um longo caminho partir deste equívoco para uma compreensão do Lean para além das práticas formais, uma compreensão do Lean como num contexto de mentalidade sem nenhum estado final definitivo, mas com as pessoas refletindo continuamente sobre o seu trabalho diário e auto aprimorando.

! Ágil te ajuda.

Há mais do que apenas algumas semelhanças entre Ágil e Lean que valem a pena explorar. Algumas filosofias de gestão ou de governança não devem ser misturadas porque isso resultará numa mistura difusa e o sabor único dos ingredientes se perderá, assim como os benefícios. Mas, na medida em que Ágil e Lean estão relacionados, eu não só acredito que Ágil e Lean podem ser combinados, mas a combinação de Princípios de Gestão Lean com o Desenvolvimento de Produtos Ágeis, como um resultado total, vai realmente resultar em uma mistura mais poderosa.

Lean e Ágil são filosofias que verdadeiramente se *misturam*. Lean possui uma mentalidade típica e poderosa. Ágil têm visões distintas que não só correspondem aos principais princípios Lean, mas formam uma implementação muito tangível para fins de lançamento de produtos.

1.6.3 A mistura das filosofias Lean e Ágil

‘A mistura das filosofias Lean e Ágil’ é também o título de um artigo mais detalhado que eu publiquei sobre o assunto [Verheyen, 2011]. Aqui eu apresento apenas algumas estratégias do Ágil que se alinham com Lean:

- ❖ *Estoques potencialmente não utilizados*: Requisitos detalhados, planos, projetos, etc. formam uma dívida em software e outras formas de desenvolvimento complexo de produto, e não um ativo, porque eles representam trabalho potencialmente não utilizado. Ágil evita produzi-los de forma antecipada contendo todos os detalhes possíveis. Se o ponto de implementação potencial do trabalho identificado for num futuro muito distante, é considerável a chance deste trabalho não ser realizado. As expectativas exatas podem mudar nesse ínterim, ou a experiência da execução intermediária e dos lançamentos podem indicar melhores maneiras de fazer o trabalho futuro. Apenas o próximo trabalho, mais elevado na ordenação, deve estar mais detalhado pois o mesmo será trabalhado em seguida. E mesmo assim, o time só vai *puxar* a quantidade

de trabalho que considera viável para uma iteração e começará a construir com base na aprendizagem progressiva e melhoria contínua, até mesmo numa base diária.

- ❖ *Trabalho parcialmente pronto*: Trabalho que não está completamente acabado, do tipo “quase lá, eu só preciso de um pouco mais de tempo”, é um conhecido e importante tipo de desperdício. Em um processo Ágil, o objetivo de cada iteração é produzir um pedaço de produto *funcionando*. Nenhum trabalho inacabado é incluído no resultado observável. O pensamento geral do Kaizen e sua implementação explícita no Ágil de *Inspeção & Adaptação* diários, ajuda as equipes a não assumir um novo trabalho enquanto o trabalho inacabado permanece na iteração. Time-boxing é uma técnica de gerenciamento de tempo que ajuda as equipes a se concentrar em finalizar o trabalho.
- ❖ *Uso da funcionalidade*: Pesquisas mostraram que apenas 20% das funcionalidades incluídas num produto (de software) construído de maneira tradicional são usadas regularmente [Standish, 2002; Standish 2013]. As funcionalidades não utilizadas ou subutilizadas constituem assim um enorme desperdício de esforço e de orçamento, tanto em termos de desenvolvê-las como em mantê-las. Colaboração ativa com pessoas que conhecem e representam os clientes e usuários impede a produção de requisitos indesejáveis ou não valiosos, e ajuda o time a focar num conjunto mínimo de funcionalidades que pode realmente ser apreciado. O foco em requisitos ‘desejados’ conserva não somente o orçamento de desenvolvimento, ele igualmente assegura que os custos futuros de manutenção e de suporte possam ser mantidos muito mais baixos. E o processo iterativo-incremental permite que as equipes adaptem regularmente o produto com base em uma apreciação efetiva do valor entregue e também capitalizem sobre novas oportunidades de valor.

O Ágil tem estratégias claras para a melhoria contínua, aproveitando assim o espírito do Kaizen:

- ❖ O plano de trabalho de um time Ágil é verificado e atualizado diariamente;
- ❖ No final de uma iteração, a versão do produto (sendo lançada ou num estado a ser lançada) é verificada para coletar feedback, observações, melhorias e aprimoramentos;

- ❖ O processo, a forma como os times trabalham, colaboram, comunicam e implementam, é verificado em seu último ponto através de uma retrospectiva da iteração.

O Ágil *otimiza o todo* exigindo que clientes ou seus representantes expressem e ordenem o trabalho a ser realizado, e participem ativamente do processo de desenvolvimento para esclarecimentos e negociações funcionais, mesmo durante a implementação. Todas as habilidades para a implementação estão disponíveis dentro do time para transformar ideias, opções e requisitos em versões de produto funcionando dentro de uma única iteração ou menos.

O Ágil encurta os tempos de ciclo otimizando o fluxo de valor por meio da prevenção de atividades de espera tradicionais como as “passagens de bastão” e decisões externas. Não há macro passagens de bastão, ou seja, passagens de bastão entre departamentos e organizações, que normalmente ocorrem em uma organização sequencial de trabalho com grandes blocos de pacotes de trabalho especializados. Mas também não há micro passagens de bastão, ou seja, passagens de bastão entre os indivíduos dentro do time, dada a responsabilidade coletiva do time.

Em geral, as estratégias e os princípios do Ágil são consistentes com, e mesmo alavancam, todos os maiores princípios Lean, como indicado na seguinte representação:

Lean	Ágil
Respeito pelas Pessoas	Times auto-organizados
Kaizen	Inspeção & adaptação, ciclos curtos de feedback
Prevenir/eliminar Desperdício	Sem especificações, arquitetura ou infraestrutura inúteis
Estoque Puxado (Kanban)	Estimativas e trabalho planejado refletem a capacidade do time
Gestão Visual	Irradiador de Informação
Qualidade Embarcada	Definição de Pronto, Padrões de Desenvolvimento
Valor ao cliente	Ativa colaboração com Negócio
Otimização do todo	Todo o time em conjunto (incl. stakeholders)
Entregue rápido	Timebox nas iterações com incremento funcionado
O gerente-professor	O líder-servidor facilitador

Figura 1.6 A consistência entre princípios do Lean e Ágil

2. Scrum

2.1 A casa do Scrum

A casa do Scrum é uma casa acolhedora. É uma casa onde as pessoas são B E M - V I N D A S.

Na casa do Scrum pessoas com diferentes conhecimentos, em diferentes papéis, com diferentes habilidades, talentos e personalidades trabalham, aprendem e melhoram juntas. A casa do Scrum é uma casa inclusiva de relações acolhedoras, abertas e colaborativas.

A casa de Scrum não conhece o ‘contra’. As barreiras são removidas em vez de serem mantidas ou criadas. Não há negócio contra TI na casa do Scrum, equipe contra o resto do mundo, Product Owner contra Development Team, desenvolvimento contra suporte, testadores contra programadores, o ‘meu’ time contra o ‘seu’ time ou Scrum Master contra a organização. A casa de Scrum oferece uma visão aberta sobre o mundo. A casa de Scrum é um lugar excelente e energizante onde o desenvolvimento do produto prospera na combinação da inteligência criativa de pessoas auto organizadas.

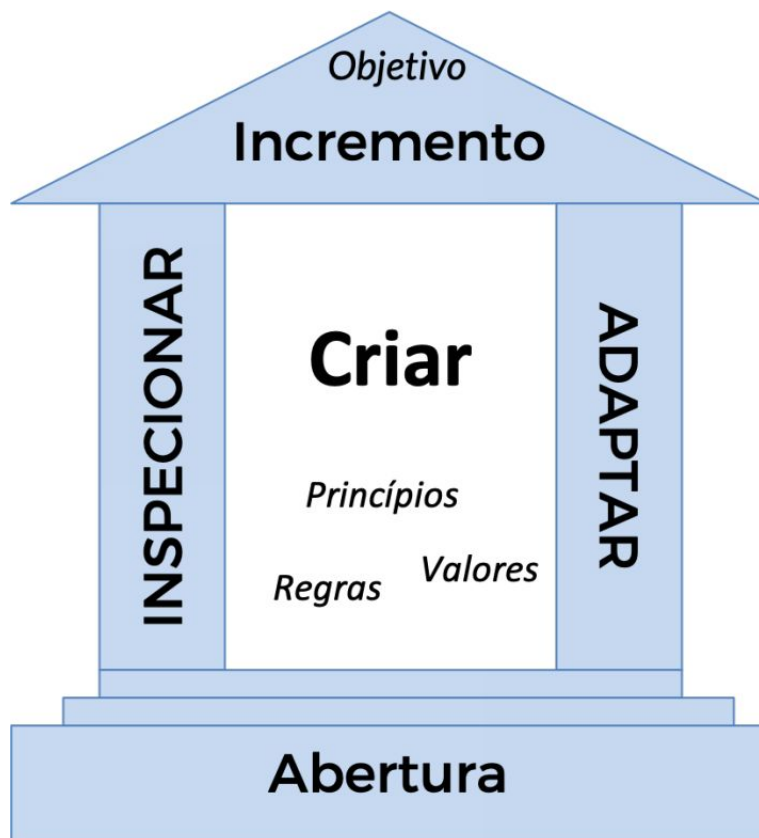


Figura 2.1 A casa do Scrum

A casa do Scrum ajuda a ficar longe do comportamento rígido e de estruturas rígidas. Os habitantes da casa do Scrum, suas equipes e os ecossistemas em que eles operam mostram flexibilidade para melhor lidar com a incerteza, as tensões internas e a pressão externa sobre o ecossistema. Exploram, sentem e adaptam-se em todos os níveis; em níveis estratégicos e táticos, de requisitos para planos, para objetivos, para mercados, para tecnologia.

Scrum é um potencializador para entregar melhores produtos e mais rapidamente. Mas, acima de tudo, a energia e o prazer do trabalho são restaurados para todas as partes envolvidas; desde aqueles que criam os produtos até aqueles que têm interesse sendo stakeholders do produto, àqueles que consomem o produto e seus serviços, e para todos os que co-criam com opiniões, feedback e apreciação. O local de trabalho é humanizado através do Scrum.

OceanofPDF.com

2.2 Scrum, o que há em um nome?

O termo ‘Scrum’ foi usado pela primeira vez por Hirotaka Takeuchi e Ikujiro Nonaka, dois reconhecidos pensadores da área de gestão, em 1986, em seu artigo inovador ‘The New New Product Development Game’ [Takeuchi & Nonaka, 1986].

Com o termo ‘Scrum’, eles se referem em seu artigo ao jogo de rugby para enfatizar a importância do trabalho em equipe bem como usam algumas analogias entre um esporte de equipe, como o rugby, e ser bem-sucedido no jogo de desenvolvimento de novos produtos. A pesquisa descrita em seu artigo mostrou que o desempenho notável no desenvolvimento de produtos novos e complexos é conseguido quando equipes, como unidades pequenas e auto organizadas de pessoas, são providas de objetivos e não de tarefas. As equipes de melhor desempenho são aquelas que recebem direção, porém há espaço para criar suas próprias táticas sobre como progredir em direção ao objetivo comum. Equipes requerem autonomia para alcançar a excelência.

Jeff Sutherland e Ken Schwaber conceberam o processo Scrum para o desenvolvimento Ágil de software no início dos anos 90. Eles apresentaram Scrum pela primeira vez em 1995 na conferência OOPSLA^[3] em Austin, Texas (EUA) [Schwaber, 1995; Sutherland, 1995].



Figura 2.2 Um Scrum no jogo de Rugby

Eles herdaram o nome ‘Scrum’ do artigo do Takeuchi e Nonaka. O framework Scrum implementa os princípios descritos neste artigo para desenvolver e manter produtos complexos. Se as equipes são instruídas apenas para executar tarefas e sua capacidade em horas é preenchido com essas tarefas, os membros da equipe sofrem de um entendimento limitado. Eles são restritos de olhar e pensar além dessas instruções, mesmo que a realidade ou a experiência demonstre que a solução prescrita é difícil de ser atingida ou é inadequada. Eles perdem a abertura para criar soluções melhores, soluções que não são ditadas, mas são mais adequadas para a demanda real dada mudanças, resultados comprovados e circunstâncias atuais. Seu único foco é produzir o que foi instruído sem considerar ideias e opções conflitantes, sem lidar com a instabilidade natural típica do desenvolvimento de produto e descoberta tecnológica. O modo industrial de direcionar as pessoas como se fossem robôs impede o desenvolvimento em cima da inteligência coletiva de uma equipe, assim, antecipadamente limitando seus resultados de trabalho a níveis medíocres.

As semelhanças notáveis entre Lean e Ágil foram apontados na seção 1.6. Há também uma conexão entre Scrum e Lean via ‘The New New Product Development Game’ e o termo ‘Scrum’.

Os autores do artigo ‘The New New Product Development Game’ são muito familiarizados com Lean, e são proponentes do mesmo. Ao longo de suas carreiras e atribuições eles têm estudado e descrito empresas Lean bem conhecidas. No entanto, eles nunca usam o termo ‘Lean’.

Em seu artigo, Takeuchi e Nonaka escolheram descrever o coração pulsante do Lean, e o chamaram de ‘Scrum’, como um diferencial em termos de desenvolvimento de produtos complexos. O ponto de vista é que uma organização provavelmente não se beneficie de qualquer prática chamada ‘Lean’ para o desenvolvimento de produtos complexos se esta batida do coração (Scrum) não esteja presente, mas somente as práticas periféricas a ela. Como este é o caso de muitas implementações Lean, os autores preferiram salientar a necessidade do coração e da alma do sistema e tirar o foco sobre as práticas de gestão circundantes.

Então, eles não mencionam Lean, ao invés focam em seu motor essencial, o Scrum.

Além disso, eles mal falam sobre ‘Lean’ porque se tornou sinônimo apenas das práticas de gestão do Sistema Toyota de Produção.

**“Scrum deve estar no coração de cada implementação de Lean.”
[Sutherland, 2011]**

OceanofPDF.com

2.3 É um gorila que eu vejo lá?

Práticas evolutivas para o desenvolvimento de software têm existido por um longo tempo [Larman, 2004]. O processo Scrum para desenvolvimento de software Ágil está disponível e documentado desde 1995. O movimento Ágil foi formado em 2001 [Beck, et.al., 2001]. Este novo paradigma para a indústria de software rapidamente criou raiz e sua adoção tem aumentado continuamente.

Um modelo amplamente aceito para avaliar e representar o grau de adoção de um produto ou serviço tecnológico é o de Geoffrey Moore, o “Ciclo de Vida da Adoção Tecnológica” (‘Technology Adoption Life Cycle’, TALC) [Moore, 1999; Wiefels, 2002].

Geoffrey Moore baseou seu modelo adaptado na diferença observada no padrão de adoção de produtos ou serviços de *tecnologia* representando um novo paradigma *disruptivo*, um paradigma que causa uma importante descontinuidade na inovação. Moore confirmou que as fases gerais e o público estão de acordo com a adoção mais tradicionais de produtos que representam uma evolução mais contínua. Mas após a fase de *Mercado Precoce* (*Early Market*), Moore observou e acrescentou ao modelo, um período de estagnação. É um período em que a adoção estagna. Um tempo imprevisível decorre antes de entrar na próxima fase de adoção, a *Pista de Boliche* (*Bowling Alley*). Alguns produtos nunca saem deste impasse e simplesmente desaparecem. Moore chamou este período de o *Abismo*.

Durante a fase altamente turbulenta da Pista de Boliche um *Gorila* é formado, um líder de mercado. Nas fases subsequentes, até o desaparecimento do produto do mercado, os gorilas líderes do mercado são difíceis de serem derrotados.

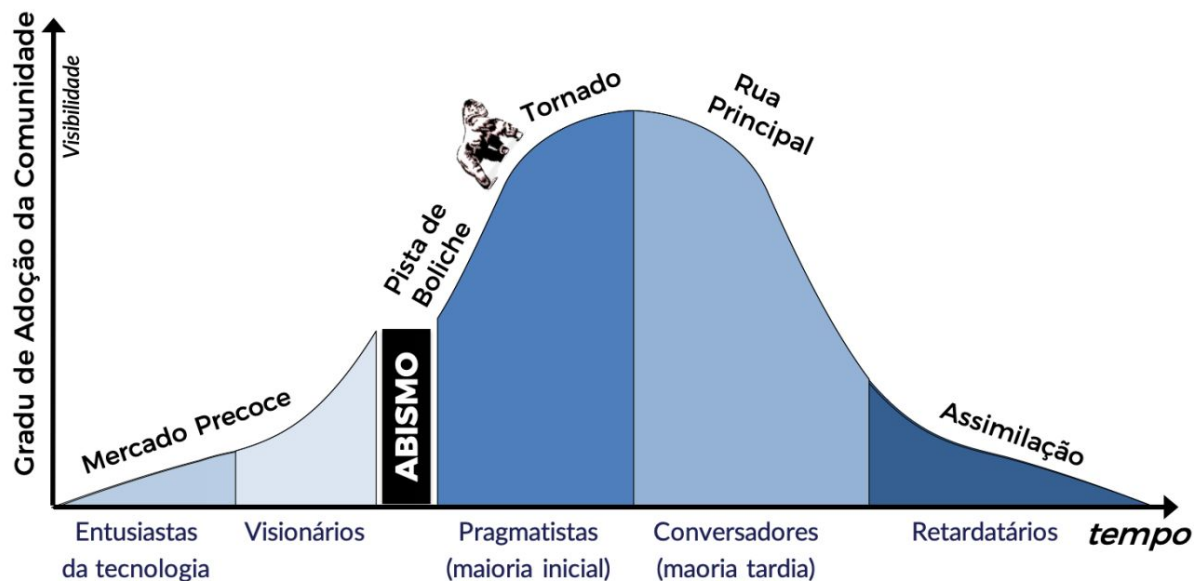


Figura 2.3 O ciclo de vida da adoção da tecnologia

Além do uso do Ágil para a entrega de novos produtos tecnológicos, possivelmente disruptivos, o Ágil em si é um novo e claramente disruptivo paradigma no mercado de tecnologia.

Os anos desde o surgimento dos primeiros processos Ágeis (*avant la lettre*) e o estabelecimento oficial do termo 'Ágil' em 2001 marcaram a fase de *Mercado Precoce* do paradigma ágil.

Por volta de 2007, o Ágil estava atravessando o abismo. Até esse ponto, as evidências sobre o Ágil eram em grande parte anedóticas e geralmente baseadas em adoções corporativas individuais, casos isolados e narrativas pessoais. Isso é típico para essas fases do ciclo de vida de adoção da tecnologia. É igualmente típico que a maioria dos entusiastas e visionários foram atraídos por ele. Mas uma vez que o Abismo foi ultrapassado, o Ágil também se tornou atraente para um público mais amplo, o público de maioria inicial de pragmatistas. Eles geralmente analisam as vantagens comerciais de um paradigma menos comprovado e comparam suas capacidades de resolução de problemas ao paradigma existente. Yahoo! é um exemplo importante de uma grande empresa em transição para o Ágil naqueles anos, documentando suas experiências em 2008 [Benefield, 2008].

No terceiro trimestre de 2009, a Forrester Research e a Dra. Dobb [Hammond & West, 2009] conduziram uma pesquisa entre profissionais de TI em todo o mundo, incluindo uma investigação sobre o tipo de “metodologia que reflete mais de perto o processo de desenvolvimento que você está usando atualmente”. Talvez surpreendentemente, 36% dos participantes indicaram que estavam fazendo Ágil, enquanto apenas 13% confirmaram estar seguindo um processo cascata^[4]. Isso confirmou a percepção comum de que o uso do Ágil tinha, de fato, gradualmente ultrapassado o modelo em cascata, e que o Ágil havia cruzado o abismo.

Em abril de 2012, a Forrester Research [Giudice, 2011] publicou os resultados de uma pesquisa sobre a adoção global do Ágil no desenvolvimento de aplicativos de software observando que “a indústria de TI está (...) amplamente adotando o Ágil” e que a adoção do Ágil não estava limitada às pequenas empresas. Grandes organizações formam uma parte substancial das empresas que se movem para o Ágil. Forrester também descobriu que “iterações mais curtas e práticas do Scrum são as práticas ágeis mais comuns” e confirmou a descoberta generalizada de que o Scrum é o método mais comumente aplicado para o desenvolvimento de software ágil. A Forrester validou assim os resultados das pesquisas anuais ‘State of Agile Development’ realizados pela VersionOne [VersionOne, 2011; VersionOne, 2013].

Embora a adoção do Scrum não seja típica ou limitada a um setor econômico específico, a Forrester descobriu que o setor de serviços financeiros mostra um grau de adoção de métodos Ágeis notavelmente alto. Isso é surpreendente, já que as grandes instituições financeiras são, por natureza, muito avessas ao risco. No rescaldo da crise financeira global de 2008, parece que muitos começaram a adotar o Scrum com sucesso, como eu pude documentar para uma grande organização financeira na Holanda em 2012 [Verheyen & Arooni, 2012].

Os anos do Ágil pós-Abismo mostram muitos movimentos vaivém, impedindo-nos de distinguir claramente a Pista de Boliche do Tornado. Os anos de Ágil pós-Abismo são marcados por um forte redemoinho, tendo o Scrum como uma âncora estável e referência.

Dentro deste turbilhão, se manifestaram três ondas do Scrum.

- ❖ A primeira onda do Scrum foi principalmente uma onda de reconhecimento. As organizações descobriram que as antigas formas industriais não eram mais suficientes para resolver, ou mesmo consertar, os problemas em seus domínios de TI e entrega de software. O Scrum foi adotado como o novo processo de entrega de TI.
- ❖ Durante a segunda onda do Scrum, grandes organizações descobriram que estavam no fim das antigas formas de trabalhar também. Como o Scrum entrou nesse novo segmento de mercado, a ‘escala’ e a divergência se tornaram os temas dominantes. Embora a terminologia de Scrum estivesse em toda parte, subgrupos e movimentos derivativos decolaram. Novos nomes, movimentos e métodos foram inventados, introduzidos, lançados e frequentemente dissolvidos novamente.
- ❖ A terceira onda do Scrum é alimentada por um desejo de simplicidade. As organizações descobrem que combater complexidade com complexidade não as ajuda. Muito desperdício, complexidade organizacional e impedimentos fundamentais permanecem sem solução pelas soluções (muitas vezes complexas) da segunda onda. Organizações renovam sua familiaridade com o Scrum. Eles começam a perceber que o Scrum está bem definido e consolidado, ainda estabelece uma moldura que deixa muito espaço para a diversidade. Eles começam a entender como o Scrum pode envolver uma variedade de estratégias e técnicas dentro dos limites de inspeção e adaptação criados.

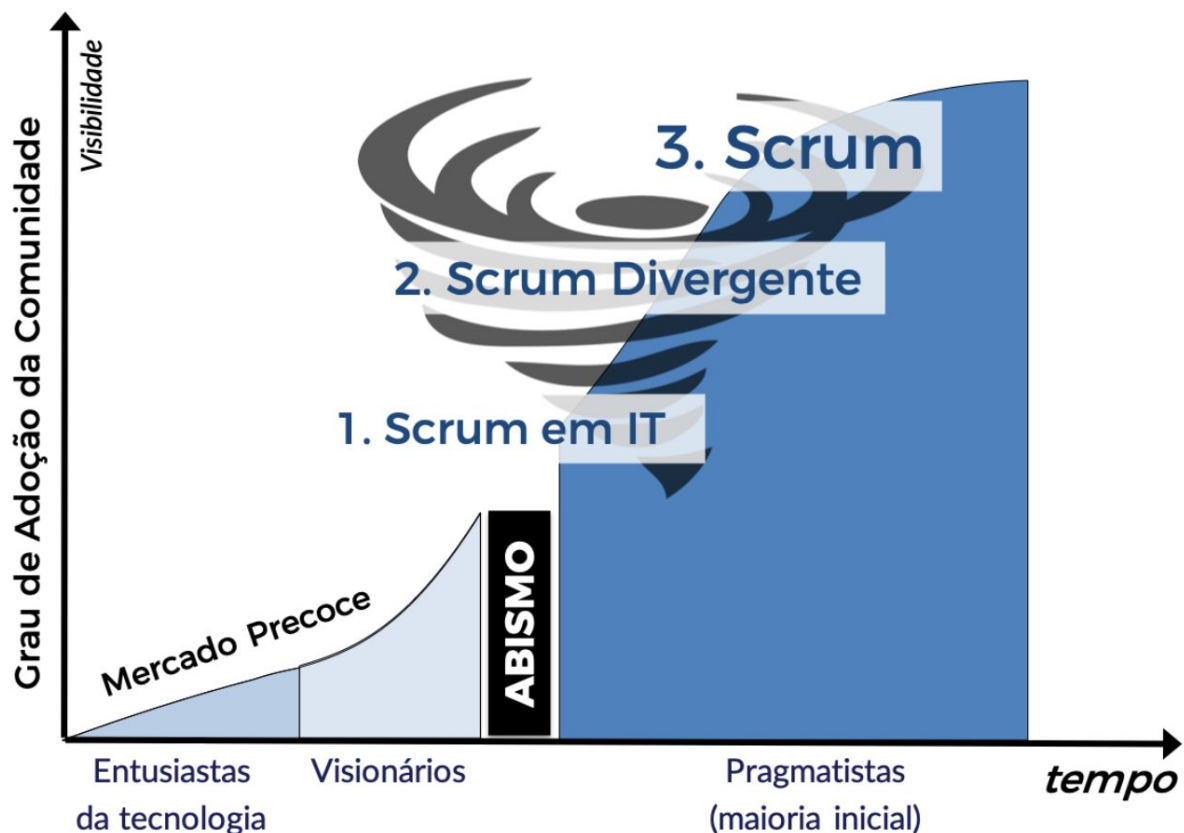


Figura 2.4 A terceira onda do Scrum

Enquanto o Scrum entra em muitos domínios da sociedade fora de software, a convergência aparece no horizonte, onde o redemoinho pode começar a se acalmar. Profissionais Ágeis em todo o mundo plantam as sementes e fertilizam os motivos para muitas organizações começarem a gerar frutos de sua adoção do Scrum, o ponto em que eles começam a pôr o Scrum em prática. O Scrum está afirmado como a definição dominante do Ágil pós-Abismo.

Scrum é o padrão de fato contra o qual medir, se opor ou aderir. Scrum emergiu como o gorila da família dos Métodos Ágeis.

2.4 Framework, não metodologia

O Scrum, com suas raízes na teoria de desenvolvimento de novos produtos, foi projetado para ajudar as equipes a criar, entregar e sustentar produtos complexos em circunstâncias turbulentas por meio da auto-organização. O Scrum implementa o método científico do empirismo para lidar melhor com complexidade e imprevisibilidade. O Scrum substitui a abordagem industrial orientada ao plano por uma experimentação oportunista e bem ponderada. A definição do framework Scrum foi conscientemente limitada a um conjunto mínimo de elementos obrigatórios, tornando cada elemento essencial. Quebrar o design básico do Scrum deixando de fora um ou mais elementos muito provavelmente irá encobrir problemas, em vez de revelá-los.

O objetivo do empirismo via Scrum é ajudar as pessoas a realizar inspeções e adaptações em cima da transparência do trabalho que está sendo realizado e dos resultados produzidos. O Scrum prevê frequentes verificações da realidade para assegurar as melhores decisões possíveis. O Scrum ajuda a ajustar, adaptar, mudar e ganhar flexibilidade. As regras, princípios e valores do framework, conforme descrito no Guia do Scrum [Schwaber & Sutherland, 2017], servem para esse propósito.

O Scrum, por meio de seu design minimalista, não tem prescrições exaustivas e formais sobre como projetar e planejar o comportamento de todos os atores, nem define seu comportamento esperado em relação ao tempo, muito menos como o trabalho deve ser documentado, mantido e armazenado. O Scrum não possui regras para instruções iniciais de tipos de documentos e entregas a serem produzidos. O Scrum também não instrui a hora exata de sua produção. Em vez de implantar, desenvolver e depender de passagens de bastão, checagens e reuniões de controle, o Scrum os expõe como uma grande fonte de atrasos, desperdício e desrespeito, deixando para a organização removê-los (ou não).

As metodologias, por design, são compostas de sequências de etapas, processos e procedimentos rigorosos e obrigatórios, implementando

algoritmos e executores pré-definidos para cada etapa, processo ou procedimento. Isso mantém a promessa de sucesso quando essas prescrições são seguidas. Assim, as ‘metodologias’ pretendem substituir a criatividade, a autonomia e os poderes intelectuais de pessoas por componentes como fases, tarefas, práticas e padrões obrigatórios, técnicas e ferramentas executáveis. A prática e a pesquisa mostram que a obediência a uma metodologia em um ambiente cognitivo serve apenas para garantir uma cobertura formal de culpa, não o sucesso dos resultados do trabalho. Metodologias dependem de altos graus de previsibilidade para ter um alto rendimento. O desenvolvimento complexo de produtos não possui esse alto grau de previsibilidade. É ainda mais imprevisível do que previsível.

O Scrum é o oposto à uma grande coleção de componentes obrigatórios interligados e a um conjunto máximo de prescrições. Scrum não é uma metodologia. O Scrum substitui uma abordagem algorítmica pré-programada por uma heurística, com respeito pelas pessoas e auto-organização, a fim de lidar com a imprevisibilidade e resolver problemas complexos.

Se, e quando o Scrum é referido como um ‘processo’, certamente não é um processo do tipo repetível. Isso é um desafio para explicar, no sentido de que o termo ‘processo’ tipicamente invoca um senso de etapas algorítmicas e previsíveis, ações repetíveis e controle executável de cima para baixo; o tipo de expectativas que são típicas para ... uma metodologia.

Se for referido como um ‘processo’, então o Scrum é um processo *servidor*, não um processo de *comando*. O que melhor funciona para todos os membros envolvidos e seus processos no trabalho, emerge do uso do Scrum, não de uma definição ditada do Scrum. Os membros descobrem o trabalho necessário para fechar a lacuna entre um resultado intermediário observável e um resultado previsto, em que o resultado intermediário inibe as características do resultado final previsto. O Scrum é um processo que ajuda a revelar os processos, práticas e estruturas mais eficazes. O Scrum cria os limites para ajudar a descobrir uma maneira de trabalhar que é continuamente adaptável ao contexto atual de todos e às circunstâncias atuais. O Scrum é ... um framework.

O framework Scrum define os ambientes delimitados para a ação e deixa as pessoas agirem, decidindo, assim, por si mesmas qual é a melhor ação possível dentro desses limites.

OceanofPDF.com

2.5 Jogando o jogo

Scrum, como um *framework* para o desenvolvimento Ágil, foi projetado para otimizar e controlar a criação de produtos valiosos em circunstâncias empresariais, organizacionais, de negócio e de mercado turbulentas.

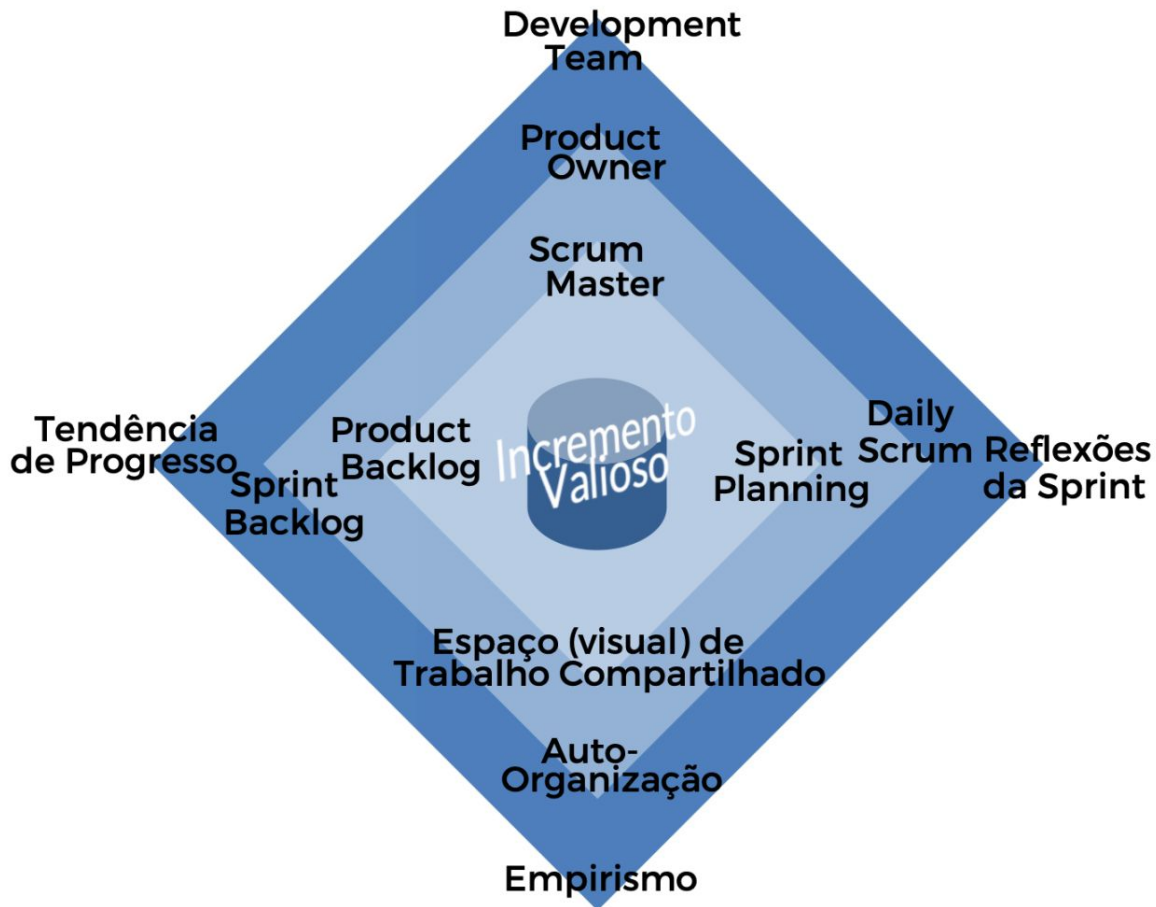


Figura 2.5 O Tabuleiro do Scrum

O tabuleiro de jogo do Scrum mostra os elementos essenciais e princípios do Scrum, tudo o que é necessário para jogar o jogo.

O Scrum requer muita disciplina de seus jogadores, enquanto deixa muito espaço para criatividade pessoal e táticas específicas do contexto. As regras do jogo são baseadas no respeito pelos jogadores através de uma

distribuição sutil e equilibrada de responsabilidade. Respeitar as regras do jogo, sem pegar atalhos nas regras e papéis, nem causar um curto-circuito nos fundamentos empíricos do jogo, proporciona os melhores prazeres e os maiores benefícios para os jogadores, assim como resultados.

O tabuleiro de jogo Scrum mostra os jogadores, os artefatos, os eventos e os principais princípios do jogo do Scrum. As regras do Scrum unem esses elementos.

2.5.1 Jogadores e responsabilidades

Métodos Ágeis são movidos por um senso de oportunismo de negócio. A técnica de gerenciamento de *time-boxing* (limitar o tempo) de todo o trabalho permite que os jogadores respondam rapidamente às novas oportunidades e se adaptem a quaisquer mudanças e evoluções.

Scrum organiza seus jogadores em torno três responsabilidades afins, cada uma complementando as outras, tornando a colaboração a chave para o sucesso:

- ❖ Product Owner;
- ❖ Development Team;
- ❖ Scrum Master.

O *Product Owner* é um papel de um só jogador, injetando a perspectiva de negócio do produto dentro do processo de entrega. Um Product Owner representa todos os stakeholders, internos e externos, para o *Development Team*, um papel composto por várias pessoas. Embora um *Product Owner* possa ter tarefas estratégicas de gerenciamento de produto além do Scrum Team, é importante que o *Product Owner* se envolva ativamente com os outros jogadores da equipe regularmente e repetidamente.

O Product Owner garante ao Development Team que existe um *Product Backlog*. O Product Owner gerencia o Product Backlog baseado na visão do produto como uma perspectiva de longo prazo do caminho a seguir. Uma visão do produto captura o porquê o produto existe.

O Product Backlog representa todo o trabalho atualmente imaginado para o produto que está sendo criado e mantido. Este trabalho pode incluir expectativas funcionais e não funcionais, aprimoramentos, correções, reparos, ideias, atualizações e outros requisitos. Se alguém quiser saber qual trabalho é identificado e planejado para o produto, basta consultar o Product Backlog.

O Product Owner expressa as expectativas de negócios e as ideias capturadas no Product Backlog para a equipe e ordena os itens no Product Backlog a fim de otimizar o valor entregue. O Product Owner gerencia o orçamento para otimizar o equilíbrio de valor, esforço e tempo para os *stakeholders* que ele representa.

O *Development Team* se auto organiza para executar todas as atividades de desenvolvimento de ponta a ponta necessárias para transformar itens do Product Backlog, expressos e ordenados pelo Product Owner, em versões de produto lançáveis. ‘Desenvolvimento’ aplica-se a todo o trabalho realizado pelo Development Team dentro de uma Sprint. Dependendo do contexto, pode incluir a criação de casos de teste, atividades de teste, programação, documentação, integração, atividades de release, etc. Abrange todo o trabalho necessário para garantir que o Incremento do produto esteja em um estado utilizável até no máximo ao final de uma Sprint, e que tecnicamente pode ser liberado para os usuários e consumidores do produto ou serviço. Esse estado do Incremento é chamado de “Pronto”. As qualidades e critérios que precisam ser atendidos para alcançar esse estado, também direcionando o trabalho de desenvolvimento a ser realizado pelo Development Team, são capturados em uma “Definição de Pronto”.

O Development Team tem um conjunto de ‘Padrões de Desenvolvimento’ para descrever como a implementação está sendo realizada. Isso é necessário para garantir o nível de qualidade necessário para lançar regularmente. E fornece a transparência certa para o modo como o jogo está sendo jogado.

O Development Team define a indicação de custo ou esforço nos itens do Product Backlog. O Development Team seleciona a quantidade de trabalho que supõe que pode realizar em uma Sprint no início da mesma. As

indicações de esforço emergentes no Product Backlog podem ser comparadas com alguma experiência comprovada anteriormente para fazer uma projeção do Product Backlog para uma Sprint.

O *Scrum Master* é um papel de uma única pessoa para facilitar a interação do Product Owner e o Development Team, dentro da organização, durante o jogo. Um Scrum Master ensina, treina e orienta o time e a organização, em entender, respeitar e saber como jogar o jogo do Scrum. O Scrum Master certifica-se de que as regras do jogo são bem compreendidas e que quaisquer elementos que atrapalhem ou bloqueiem a equipe em seu progresso sejam removidos. Tais elementos são chamados Impedimentos no Scrum.

O Scrum Master induz o desejo contínuo de se tornarem jogadores melhores. O Scrum Master implementa o Scrum ajudando os outros a usar melhor o Scrum.

2.5.2 Tempo

As iterações com time-boxe no jogo do Scrum são chamadas de *Sprints*. Sprints permitem que o Development Team se concentre em alcançar o próximo nível de jogo, o *Objetivo da Sprint*, com o mínimo de interrupções externas.

Todo o trabalho no Scrum é organizado em Sprints. O Scrum não tem Sprints estereotipadas já que o objetivo de cada Sprint é entregar um pedaço valioso de produto funcionando, um *Incremento* (de produto). A duração de uma Sprint nunca é superior a quatro semanas e normalmente dura de uma a quatro semanas.

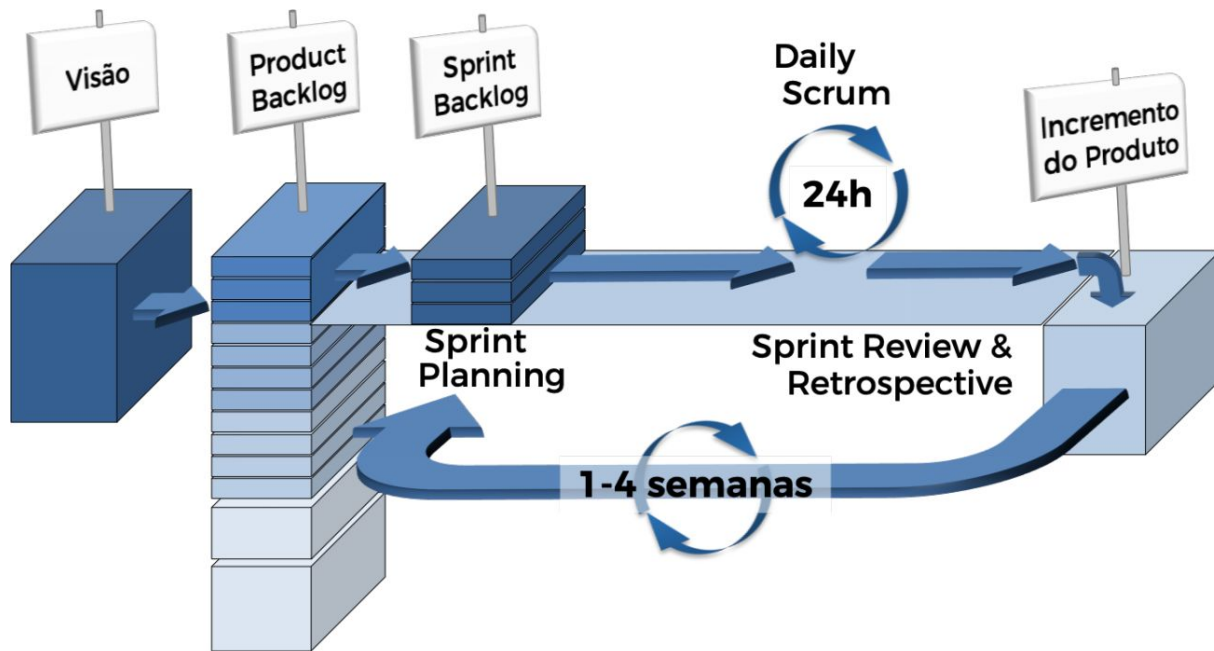


Figura 2.6 Visão geral de uma Sprint do Scrum

Como um evento contêiner, a Sprint encapsula os outros eventos do Scrum. Cada evento tem um time-box (duração máxima) e é uma oportunidade para mudar de rumo ou se adaptar às condições de mudança:

- ❖ Sprint Planning;
- ❖ Daily Scrum;
- ❖ Sprint Review;
- ❖ Sprint Retrospective.

Cada Sprint começa com a *Sprint Planning*, na qual o Development Team obtém trabalho para a Sprint a partir do Product Backlog atual. A equipe seleciona a quantidade de trabalho que considera viável para a Sprint em relação às expectativas do que é necessário para torná-la lançável. O trabalho selecionado é uma *projeção* que representa as informações que a equipe tem no momento da seleção. O Development Team pode analisar a quantidade de trabalho que eles realizaram, em média, em Sprints anteriores e compará-la com sua capacidade para a próximo Sprint, para aumentar ligeiramente a precisão da projeção. As opiniões do Product Owner são respeitadas e detalhes adicionais são discutidos com o Product Owner.

O trabalho selecionado, *projeção*, é desenhado, analisado e elaborado em um plano de trabalho implementável para a Sprint (time-box), o *Sprint Backlog*. Após o término do time-box da Sprint Planning, ou possivelmente antes, o Development Team começa a trabalhar nesse plano que foi criado de forma colaborativa. A Sprint Planning nunca leva mais de oito horas.

Para gerenciar e acompanhar seu trabalho de desenvolvimento, o Development Team organiza um curto evento diário de 15 minutos, chamado de Daily Scrum. É um evento de planejamento no tempo oportuno. O plano de trabalho do time é otimizado para atingir o objetivo da Sprint com base no progresso real dentro da Sprint. A adaptação é capturada em uma atualização do Sprint Backlog. O progresso real no Sprint Backlog é visualizado, com base na quantidade de trabalho restante. Se o progresso atual impacta a projeção, o Development Team consultará o Product Owner.

À medida que a Sprint avança, um Incremento do produto surge do trabalho colaborativo da equipe. Se o Product Owner, como único representante dos stakeholders, considerar o Incremento útil, então o Incremento poderá ser lançado sem demora. No final da Sprint, o Incremento é inspecionado em uma Sprint Review; para verificar a adequação funcional para lançá-lo ou o uso real, se já lançado.

O Product Owner mantém um alto nível de transparência apresentando as evoluções do Product Backlog durante a Sprint em relação à visão do produto a longo prazo na Sprint Review. Ao analisar o Incremento do produto, todos os jogadores provavelmente descobrirão mudanças e receberão feedback e insights evolutivos da inspeção. Elas são processadas no Product Backlog para implementação futura, e a compreensão de que a data exata da implementação depende da ordenação do Product Owner e do progresso sustentável da equipe. O Sprint Review nunca leva mais de quatro horas.

A Sprint é concluída com uma Sprint Retrospective, na qual o Scrum Team inspeciona e reflete sobre o ‘processo’ completo. O evento abrange todos os aspectos do trabalho, por exemplo, conveniência no lançamento do produto, tecnologia, aspectos sociais, o processo Scrum, práticas de desenvolvimento, colaboração, qualidade do produto, etc. O evento é

basicamente sobre estabelecer o que foi bem, onde há espaço para melhoria e quais experimentos podem ser realizados de forma útil para aprender e construir um produto melhor.

Como parte da melhoria contínua, o Scrum Team entra em acordo sobre o que manter, ajustar, experimentar e melhorar para a próxima Sprint. Uma Sprint Retrospective nunca leva mais de três horas.

O Scrum só conhece Sprints, e o objetivo de cada Sprint é entregar uma versão de produto funcional, um incremento do produto. As versões do produto em funcionamento são consideradas a única medida de progresso.

A duração da Sprint é mantida estável ao longo de várias Sprints por razões de cadência. É a pulsação do desenvolvimento e é útil para a equipe entender quanto trabalho pode ser feito durante uma Sprint.

A quantidade de trabalho que pode ser feita em uma Sprint é, às vezes, chamada de *Velocidade*. A velocidade é uma indicação da quantidade de trabalho que uma equipe conseguiu criar em Sprints anteriores. A velocidade é a soma das unidades de esforço ou custo que foram concluídas em uma Sprint e é típica de uma equipe e da composição da mesma.

A duração da Sprint é adequada para aproveitar oportunidades de negócios emergentes e não previstas anteriormente. A Sprint Review colaborativa fornece ao Product Owner as melhores informações possíveis sobre como as Sprints adicionais podem melhorar ainda mais o valor do produto, levando em consideração um equilíbrio entre risco, esforço, orçamento e coesão.

A duração da Sprint também pode depender de quanto tempo uma equipe pode trabalhar sem consultar os stakeholders na Sprint Review. A Sprint Review é uma oportunidade para se adaptar a novas direções estratégicas ou de mercado.

Uma equipe sofrerá com oportunidades reduzidas de aprendizado e adaptação quando não estiver consultando os stakeholders e conhecendo mercados, evoluções de negócios e novas estratégias pelo menos a cada

quatro semanas. Sprints podem ser mais curtas do que quatro semanas, mas nunca mais longas.

2.5.3 Acompanhamento do progresso

O progresso geral do trabalho é rastreado e visualizado, a fim de conhecer as *tendências* do progresso para fins de previsões, olhando para o futuro incerto com base no passado comprovado e observável.

A fim de medir continuamente e adaptar à realidade e alcançar as melhores previsões possíveis, sempre levando em conta a complexidade, o trabalho restante é reestimado honestamente e regularmente:

- ❖ *Progresso da Sprint*: Dentro de uma Sprint, o progresso é rastreado diariamente. O Sprint Backlog sempre mantém o plano mais preciso e realista do restante do trabalho para implementar o objetivo da Sprint.
- ❖ *Progresso do produto*: A indicação de progresso em nível de Product Backlog é atualizada e revisada no mínimo na Sprint Review. O Product Owner talvez empacote itens do Product Backlog e incrementos em *versões* candidatas. O progresso comprovado das últimas Sprints fornece ao Product Owner e aos seus stakeholders uma data de entrega prevista para *releases*, funcionalidades individuais ou conjuntos de recursos.

A abordagem clássica do Scrum para visualizar o progresso é um gráfico de *Burn-down*, um gráfico mostrando a evolução do trabalho remanescente total:

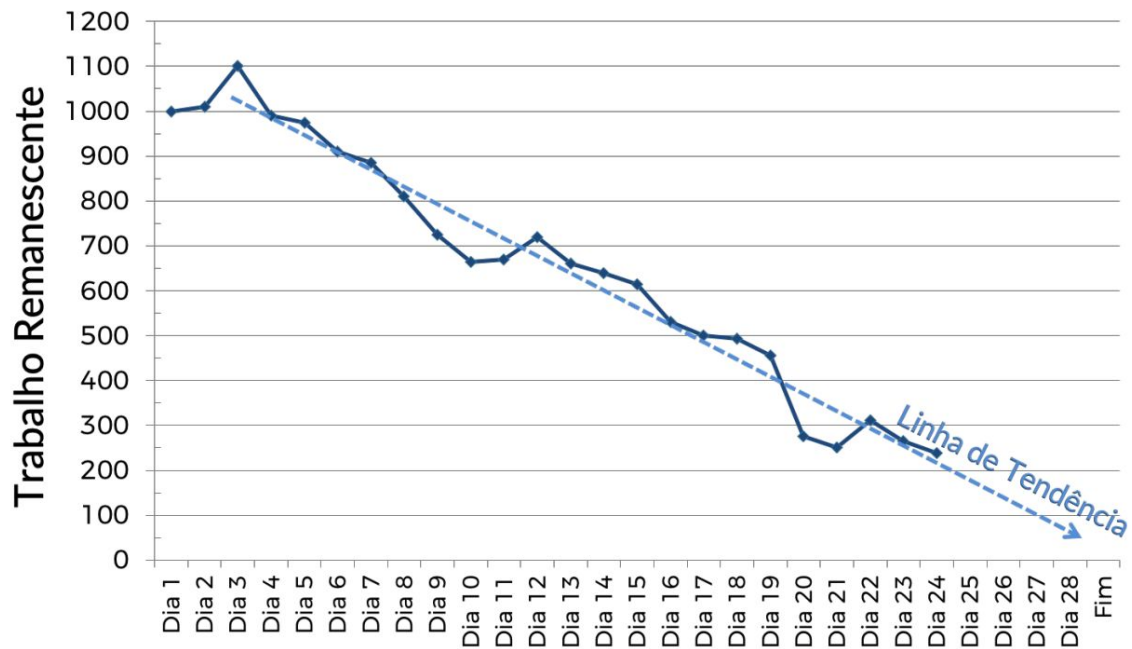


Figura 2.7 Exemplo de uma Sprint Burn-down chart

No entanto, a equipe decide a melhor maneira de representar o progresso. Este pode ser um gráfico de burn-down, um quadro físico de Scrum, um gráfico de burn-up (por exemplo, em valor), ou pode ser um diagrama de fluxo cumulativo para visualizar melhor o fluxo de trabalho:

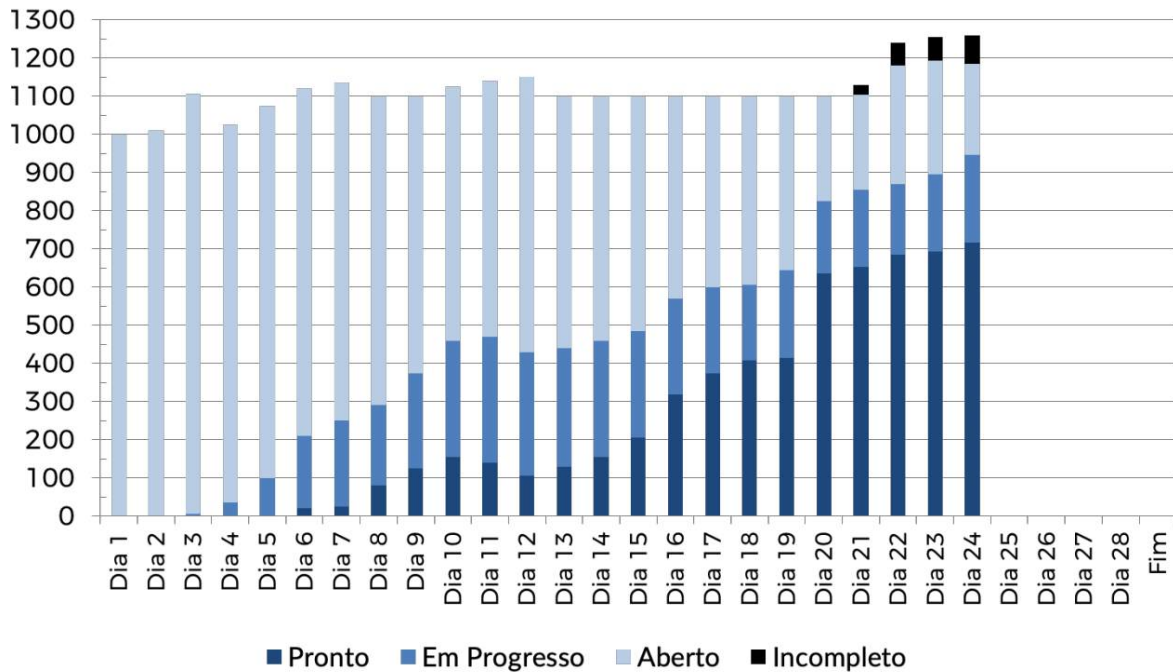


Figura 2.8 Exemplo de um Diagrama de Fluxo Cumulativo

2.5.4 O valor do Product Backlog

O valor do Product Backlog não está na integralidade, precisão, detalhe ou perfeição, nem na captura de todos os requisitos possíveis em todos os detalhes possíveis e em todos os possíveis horizontes de tempo. O valor do Product Backlog está na transparência, em deixar claro que trabalho precisa ser feito para criar um produto minimamente viável e valioso (ou Incremento do produto). O Product Backlog traz à tona todo o trabalho, desenvolvimento, conformidade e restrições com as quais uma equipe precisa lidar para criar versões de produto que podem ser liberadas.

O Product Backlog é uma lista ordenada de ideias, funcionalidades e opções para dar vida a um produto imaginado, para entregá-lo e subsequentemente sustentá-lo e expandi-lo. A lista é obrigada a incluir todas as funcionalidades e recursos, mas naturalmente também inclui correções, trabalho de manutenção, trabalho de arquitetura, trabalho a ser gasto em segurança, escalabilidade, estabilidade e desempenho, etc. No momento da

criação de um item no Product Backlog, o item é supostamente valioso para um cliente ou contribui para a capacidade de agregar valor.

Cada item no Product Backlog contém apenas detalhes suficientes para deixar claro qual valor que ele representa. Um item é intencionalmente incompleto para incentivar discussões adicionais e explícitas sobre ele. Cada item do Product Backlog serve como um espaço reservado para discussão no momento apropriado.

O Product Owner tem responsabilidade sobre o Product Backlog. Um Product Owner, no entanto, ainda levará em conta a contribuição técnica e de desenvolvimento do Development Team. Um Product Owner ainda levará em conta dependências, requisitos não funcionais e expectativas organizacionais.

O Product Backlog é gradualmente refinado, introduzindo assim o gerenciamento incremental dos requisitos do produto:

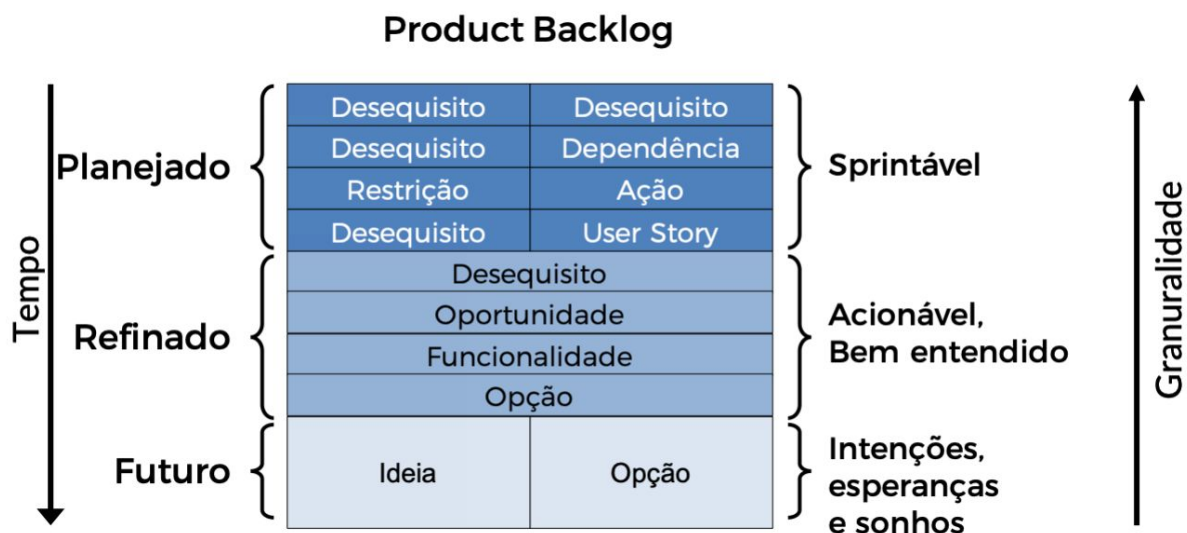


Figura 2.9 Evolução incremental do Product Backlog

Conforme o desenvolvimento avança, o Product Backlog é refinado, ajustado e atualizado. O Product Backlog é continuamente ordenado e reordenado pelo Product Owner. Os itens são regularmente refinados em conjunto com o Development Team. O Product Backlog é um artefato vivo.

O Product Owner procura equilibrar as necessidades de todos os stakeholders, internos e externos, que devem ser representadas para o Scrum Team. Continuamente adere às descrições e design do trabalho apenas ‘suficientes’, ou seja, deixando de fora detalhes desnecessários, garantindo que nenhum tempo e dinheiro excessivos serão desperdiçados se o item não for criado ou for implementado bem mais tarde ou de maneira diferente.

O nível de descrição e detalhe de um item do Product Backlog está em algum lugar entre o que é desejo e o que costumava ser um requisito. Um “desejo” é muito confuso para se trabalhar e um “requisito” (tradicional) é super especificado e super detalhado. A especificação excessiva no desenvolvimento impede o uso ideal da tecnologia, bloqueia a capacidade de capitalizar sinergias entre diferentes funções e é um desperdício de dinheiro em situações de turbulência ou mudança mínimas. Portanto, o termo ‘desequisito’ é adequado para descrever a granularidade de um item do Product Backlog.

‘Desequisitos’ são transferidos do Product Backlog via Sprint Backlog para Incrementos de produto funcionando. Embora a ordem do Product Backlog dependa de uma combinação complexa de fatores como custo/esforço, dependências, prioridade, coesão e consistência, é essencial também ter uma visão do valor assumido de um item do Product Backlog.

Os fatores centrais para um item do Product Backlog são custo e valor:

- ❖ *Custo*: O custo ou o esforço de um item do Product Backlog geralmente é expresso em uma unidade relativa. Sprints passadas mostram à equipe quanto trabalho, expresso na unidade escolhida, pode ser convertido, em média, em um Incremento durante uma Sprint. Com base nesse dado empírico, uma expectativa pode ser criada quando um item do Product Backlog atual pode se tornar disponível como parte de um produto em evolução. É uma projeção, sem mover os jogadores para o campo das previsões exatas, entendendo que qualquer expectativa desse tipo é limitada pelo conhecimento e pelas circunstâncias atuais.
- ❖ *Valor*: Um princípio importante do Ágil é “satisfazer o cliente por meio da entrega adiantada e contínua de software de valor”. [Beck, et. Al, 2001] Sem um atributo para o valor (de negócio) nos itens do Product

Backlog, um Product Owner não faz ideia de quanto valor uma funcionalidade, uma ideia ou um conjunto de funcionalidades leva presumivelmente ao cliente que ele representa dentro do Scrum Team. O valor dependerá do tipo de empresa, do tipo de produto e do mercado. O valor de um item do Product Backlog pode ser indireto, pois em não escolher um item do Product Backlog poderá reduzir o valor do sistema ou até mesmo da organização, ou que não ordenando o item no topo do Product Backlog poderá produzir um valor negativo ou prejudicar a capacidade futura de criar valor.

A noção de valor ajuda os Product Owners e seus stakeholders a se afastarem da (falsa ideia de) perfeição de um produto total que deve ser completamente construído antes mesmo de considerar seu lançamento. O foco muda para um lançamento de produto mínimo comercializável e o mínimo trabalho necessário para efetivamente trazer valor ao mercado. O Product Backlog pode ser usado para agrupar itens, funcionalidades e requisitos não funcionais em conjuntos de funcionalidades coesas.

O Product Backlog é todo o plano que o Scrum necessita, seus ‘desequisitos’ contêm todas as informações para fazer previsões sobre escopo e tempo, se preciso. Um item do Product Backlog precisa dos atributos certos para ser ordenado, mais do que apenas ser priorizado.

2.5.5 A importância do Pronto

Em uma definição de ‘Pronto’, as qualidades e os critérios são expressos e precisam ser atendidos por um Incremento do produto para que ele seja ‘lançável, para poder lançá-lo em produção. A definição de Pronto possui as qualidades do produto e, muitas vezes, as atividades, critérios, tarefas e trabalho que precisam ser executados em um pedaço funcional de um produto para que ele esteja em um estado de qualidade de produção. ‘Pronto’ é uma qualidade do Incremento e, portanto, faz parte dos artefatos do Scrum.

A definição de Pronto é essencial para entender e planejar o trabalho necessário para criar um Incremento liberável durante a Sprint Planning e para a inspeção desse Incremento durante a Sprint Review. A definição de Pronto serve à transparência exigida no Scrum em termos do trabalho a ser feito e do trabalho realmente realizado.

No entanto, o prefixo ‘potencial’ é adicionado ao ‘Incremento lançável’. Isso se refere à responsabilidade do Product Owner em decidir sobre o lançamento real de um Incremento; uma decisão que provavelmente será baseada na coesão do negócio e na utilidade funcional, conforme observado durante a Sprint ou na Sprint Review. No entanto, a decisão de lançamento pelo Product Owner não deve ser restringida pelo trabalho de ‘desenvolvimento’, portanto, todo o trabalho necessário para atingir o nível de Pronto é executado antes da Sprint Review dentro da Sprint.

O empirismo do Scrum só funciona bem com transparência. A transparência exige padrões comuns para referência de trabalho e inspeção. A definição de Pronto define o padrão para *lançável* e deve ser conhecida por todos os jogadores. Transparência não significa apenas visível, mas também compreensível. O conteúdo da definição de Pronto deve ser autoexplicativo.

Espera-se que uma organização que prospera e dependa de produtos e serviços tenha uma definição vigente das qualidades do produto e as expresse por meio de padrões, diretrizes, regras, níveis de serviço e outras expectativas. Uma organização profissional define qualidade. Os Development Teams do Scrum, formados por desenvolvedores profissionais de produto, são parte integrante de uma organização, em vez de serem gangues isoladas de codificadores arruaceiros dentro da organização. Espera-se que os tais Development Teams sigam os padrões gerais do produto como definido pela organização.

Se, como seria de esperar, a definição mínima de Pronto é fornecida pela organização, um Development Team irá ainda complementá-la com elementos específicos do contexto; o produto, uma versão ou tecnologia. Na ausência de tais diretrizes organizacionais, um Development Team deve, como profissionais, criar uma definição apropriada de Pronto para o seu trabalho.

Através de uma definição acordada de Pronto, a qualidade está no centro do Scrum. Nenhum trabalho incompleto (não Pronto) faz parte de um Incremento. Nenhum trabalho incompleto é colocado em produção. *Nunca*. A partir da inspeção do Incremento com base na definição de Pronto, a conversa colaborativa durante a Sprint Review pode incluir qualidade e requisitos com relação à definição de qualidade na organização. Isso ajuda a equipe a considerar a definição de Pronto conveniente na subsequente Sprint Retrospective. A condução auto organizada do Development Team fará com que eles incluam tudo o que for realmente possível e, mais, levará em conta o feedback dos stakeholders.

A propriedade primária da definição de Pronto está com o Development Team, da mesma forma que a propriedade primária do Product Backlog está com o Product Owner. O Development Team é responsável pelo trabalho árduo relacionado ao fornecimento de versões funcionais de produtos que atendem à definição de Pronto. Uma definição de Pronto não pode ser reduzida por forças externas ao Development Team. O Development Team basear-se-á nas expectativas e definições gerais da organização (incluindo as áreas de desenvolvimento, engenharia, qualidade ou operações). O Development Team adicionará qualidades específicas do produto e, obviamente, incorporará as expectativas de qualidade funcionais ou de negócio do Product Owner.

Decisões sobre a definição de Pronto talvez dependam da presença de habilidades, autorizações e disponibilidade de sistemas, serviços e interfaces. Embora dependências em sistemas externos ou interfaces possam levar a uma reordenação do trabalho no Product Backlog, um Development Team prefere progredir. Uma equipe pode usar stubs, mock-ups ou simuladores para sistemas não disponíveis ou dependências técnicas não resolvidas. Mas todas as partes continuam conscientes de que o trabalho não está realmente pronto, pois a definição de Pronto não se reflete trabalho lançável. Isso aumenta a independência da equipe, mas não elimina a dependência. Há uma quantidade imprevisível de trabalho oculto no sistema e ele deve ser executado em algum momento para lançar o produto. Enquanto isso, o Product Owner está impedido de tomar essa decisão. Felizmente, a Sprint Review em última instância revela essas informações

para os stakeholders também, então as chances são maiores de que ações apropriadas sejam tomadas dentro da organização.

A Definição de Pronto enfatiza a importância de capturar a oportunidade de lançamento, fazendo todo o trabalho necessário para capitalizar uma oportunidade dentro de uma Sprint.

OceanofPDF.com

2.6 Os princípios fundamentais do Scrum

O tabuleiro do jogo Scrum na figura 2.5 não apenas mostra os elementos formalmente prescritos do Scrum, mas também três princípios básicos subjacentes ao Scrum:

- ❖ Espaço de trabalho (Visual) compartilhado
- ❖ Auto-organização;
- ❖ Empirismo (conhecido como ‘controle de processo empírico’).

2.6.1 Espaço de Trabalho (Visual) Compartilhado

As equipes, para funcionar adequadamente e crescer em termos de eficácia e desempenho, precisam de um espaço de trabalho compartilhado para suas interações e colaboração. O time organiza seu próprio espaço de trabalho para otimizar diálogo, comunicação e colaboração. Isso inclui a remoção de barreiras - físicas ou mentais - que obstruem o fluxo de informações. O espaço de trabalho compartilhado facilita que o time e seus membros tomem decisões rápidas e engajadas. Embora não seja obrigatório, a colocação física acaba sendo mais ideal do ponto de vista da dinâmica de equipe. Mas, mesmo quando não está trabalhando colocada, um time precisa de um espaço de trabalho compartilhado, incluindo todos os recursos de comunicação modernos necessários para superar a distância física da melhor maneira possível.

Dentro do espaço de trabalho, um time deve ser capaz de se concentrar em atividades de valor agregado. Todo o trabalho administrativo e indireto é mantido a um mínimo. Isso inclui o armazenamento de informações. As equipes precisam de acesso rápido a todas as informações do time, para criar, manter e compartilhar, e agilizar todas as decisões que serão tomadas baseadas nessas informações. É por isso que as equipes preferem técnicas de gerenciamento visual. Um espaço de trabalho compartilhado tem muitos radiadores de informação [Cockburn, 2002]. Os radiadores de informação

limitam o tempo de transmitir informações e concentram-se no time como uma unidade, o que é crucial quando se realiza um trabalho complexo.

Uma visão geral de tarefas, definições de equipe, padrões e acordos, artefatos de processo e tendências de progresso são disponibilizados e visíveis dentro do espaço de trabalho compartilhado, publicando-os nas paredes da sala; em quadros brancos, flip charts ou outros meios. Isso inclui todas as informações que o time julgar apropriadas para visualizar, como desenhos e modelos, análises de impacto, impedimentos, a Definição de Pronto, os padrões de desenvolvimento, etc.

Ao entrar na sala toda a informação está prontamente disponível, a sala a irradia para o leitor interessado. O leitor não é forçado a entrar em sistemas eletrônicos, obter autorizações, autenticá-las, procurá-las, procurar a versão mais recente ou até mesmo perguntar sobre elas. As equipes mantêm todas as informações cruciais dessa forma visualizada para *compartilhá-las* dentro e além dos membros do time, e usá-las para inspecionar e adaptar.

A informação não é estática. Ela reflete constantemente o atual estado de coisas, onde o estado atual pode ser usado para projetar previsões para o futuro.

Um espaço de trabalho visual compartilhado otimiza a transparência e reduz drasticamente o tempo de troca de informações.

2.6.2 Auto-organização

O Scrum prospera na colaboração diária das três funções pares. Cada papel tem claras responsabilidades dentro do time bem como em sentido da organização. O Scrum Team e, dentro dele, o Development Team são unidades de pessoas auto organizadas.

A auto-organização não é apenas o grau de liberdade que é permitido. A auto-organização não é sobre delegação. A auto-organização é, acontece. A auto-organização não é sobre habilitar ou empoderar, não existe autoridade

superior que a conceda. A capacidade de se auto organizar realmente requer a remoção de muitas barreiras existentes que impedem as pessoas de se comunicar, interagir, obter insights e colaborar. É assim que autoridades externas são mais eficazes; removendo barreiras organizacionais ou processuais para melhor facilitar as equipes.

A auto-organização não é sobre anarquia ou liberdade ilimitada. A auto-organização tem e exige limites, limites dentro dos quais a auto-organização acontece. As regras do Scrum são um dos principais limites dentro dos quais uma equipe auto organiza seu trabalho:

- ❖ O Development Team seleciona colaborativamente o trabalho conforme ordenado e expresso pelo Product Owner, cria colaborativamente atividades acionáveis para a sua projeção da Sprint e replaneja o trabalho diariamente em uma Sprint com time-box para otimizar o resultado.
- ❖ O Product Owner interage com usuários, stakeholders e gerentes de produtos para identificar o trabalho mais valioso e conta com Development Team(s) multifuncional para a entrega real disso em Incrementos de produto. Os stakeholders ajudam a moldar o futuro produto em cada Sprint Review.
- ❖ O Scrum Master não tem interesse direto no escopo, orçamento, entrega ou tarefas, mas treina e facilita o ecossistema completo no uso do Scrum para gerenciar esses fatores.

Pessoas organizadas em equipes têm maior coesão, confiança mais profunda e interconexões mais eficazes quando o tamanho do time totaliza em torno de cinco a sete membros. Embora o Scrum estabeleça uma expectativa de que o tamanho do Development Team seja entre três e nove membros, não há necessidade de um processo formal para impor isso. Através da auto-organização, uma equipe ajustará seu tamanho de forma autônoma até que o desempenho ideal seja atingido. Isso até acontecerá entre equipes trabalhando juntas. Não existe um órgão externo que saiba organizar melhor o trabalho do que as pessoas que realmente realizam o trabalho.

Em seu livro ‘Drive’ [Pink, 2009], Daniel Pink elabora a evidência científica do que motiva as pessoas. Ele descreve como a ‘auto-dirigência’, a capacidade das pessoas dirigirem seu próprio trabalho, é um dos três

motivadores cruciais no trabalho criativo e cognitivo. ‘Maestria’ e ‘Propósito’ completam a lista. Juntos, eles decifram o que Pink identifica como o terceiro impulso (*drive*), o modelo de motivação humana que segue o primeiro impulso de sobrevivência e o segundo impulso de esquemas tayloristas industriais que implementam recompensas por ‘incentivos e penalização’. A auto-organização do Scrum é, assim, confirmada como crucial para a motivação dos jogadores no trabalho criativo, exigindo habilidades cognitivas.

No entanto, autonomia e auto-organização não resolvem todos os problemas. Alguns problemas vão além da auto-organização do Development Team. O Scrum os chama de ‘impedimentos’.

A definição geral de um impedimento é uma ‘*obstrução; empecilho; obstáculo*’. Um impedimento no Scrum é um fator que bloqueia o time na criação de uma versão valiosa do produto em uma Sprint, ou que restringe a equipe a atingir seu nível intrínseco de progresso. É a responsabilidade do Scrum Master em remover impedimentos.

Vamos ilustrar isso com o exemplo de um conflito de equipe, um conflito entre os membros do time.

Uma equipe pode ter problemas para resolver um conflito interno e chamar o conflito de impedimento, esperando que o Scrum Master o remova para eles. Em outras palavras, eles esperam que o Scrum Master resolva o conflito.

No entanto, trabalhar em equipe inclui, inevitavelmente, conhecer uns aos outros, encontrar formas de construir versões dos produtos, explorar formas diferentes de colaborar, encontrar consenso sobre ideias diferentes, superando o desejo de heroísmo pessoal. Em seu livro, ‘Coaching Agile Teams [Adkins, 2010], Lyssa Adkins elabora sobre ‘desacordo construtivo’ como uma necessidade para as equipes. Este nível mais baixo de conflito conecta-se com a ‘instabilidade embutida’ observada e descrita por Takeuchi e Nonaka [Takeuchi & Nonaka, 1986] como o solo fértil para o desenvolvimento bem-sucedido de produtos complexos. É uma parte natural da liberdade dada a um grupo de pessoas descobrir conjuntamente

as melhores maneiras de seguir em frente, na ausência de uma autoridade externa que prescreva a solução.

Conflitos são uma parte natural do trabalho com pessoas, do trabalho em equipe. É uma parte essencial da auto-organização. Se um membro traz um conflito interno para ciência do Scrum Master, este deve se perguntar qual é o problema real. É o papel do Scrum Master resolver o conflito? Ou seria uma intervenção indesejada no ecossistema de auto-organização, minando a honestidade futura, a aprendizagem e o auto aperfeiçoamento? Como o Scrum Master pode facilitar a auto-organização? Oferecendo às equipes uma desculpa, uma decisão externa de se esconder?

Um Scrum Master, como promotor do Scrum e auto-organização, considera como ajudar um time a resolver seus próprios problemas e oferecer quaisquer ferramentas, treinamento e insights sobre a melhor forma de fazer isso, sobre como se autodesenvolver.

2.6.3 Controle de processo empírico

O desenvolvimento de novos produtos é uma atividade complexa por si e serve para fornecer produtos complexos em circunstâncias complexas.

Uma perspectiva sobre o grau de ‘complexidade’ refere-se ao número de parâmetros, variáveis e eventos que influenciam uma atividade e seu curso. No desenvolvimento de produtos, alguns dos parâmetros mais conhecidos são as expectativas e requisitos do usuário, habilidades, disponibilidade e experiência de pessoas organizadas em equipes, tecnologia, integrações técnicas, condições de mercado e concorrência, regulamentos e dependências.

No entanto, não é apenas o número de parâmetros conhecidos que é importante, mas também o conhecimento *disponível*, bem como o conhecimento *necessário* desses parâmetros. Qual é o nível de detalhe necessário para compreender uma variável, bem como o comportamento futuro dessa variável? Mesmo que um parâmetro seja conhecido, o nível de detalhe pode ser muito profundo para poder gerenciá-lo e controlá-lo. E então, é claro, o comportamento do parâmetro ainda não é necessariamente

previsível. Uma variável conhecida pode ainda se comportar de maneira completamente diferente do esperado.

‘Complexidade’ também é dependente da natureza da atividade em si. Os resultados exatos e detalhados do desenvolvimento do produto são difíceis de descrever e prever antes ou até mesmo no início do desenvolvimento real. As etapas, tarefas e atividades combinadas que compõem o trabalho atual não são previsíveis com qualquer grau de alta precisão. Pessoas as executam e o envolvimento das pessoas depende de muitas circunstâncias. Além disso, também há o trabalho com a tecnologia em si, em que a tecnologia evolui constantemente e é dependente da particularidade do ambiente de uma organização.

As etapas, tarefas e atividades de desenvolvimento de produtos não são previsíveis em um alto grau de precisão porque não são repetíveis. Cada ‘produto’ sendo desenvolvido, que não é mecanicista ou industrial, é único. Novas tecnologias surgem, novas interfaces precisam ser construídas, novos plug-ins são usados, novas integrações precisam ser configuradas, novos insights e técnicas para o desenvolvimento são descobertas regularmente, enquanto o trabalho já está sendo realizado.

O grau de dinamismo de um problema ou atividade requer que o processo correto esteja em vigor para se ter o controle correto sobre a atividade:

- ❖ Em um *sistema de ciclo aberto*, todas as variáveis são coletadas antecipadamente porque elas precisam ser apresentadas ao sistema como entrada, antes que uma única das etapas seja executada, resultando em um resultado previsto. Para poder prever o resultado e o tempo decorrido, esse tipo de controle de processo assume um alto grau de previsibilidade das variáveis que influenciam o processo, bem como das próprias atividades do processo.



Figura 2.10 Sistema de ciclo aberto

Para obter controle sobre problemas grandes ou complexos, usando o pensamento de ciclo aberto, os subsistemas são criados onde cada subsistema é um sistema de ciclo aberto separado. A entrada de cada subsistema é a saída do subsistema anterior. Em situações de maior turbulência e mudança frequente, desvios e variações se acumularão ao longo da cadeia de subsistemas, inerentemente muito além de níveis aceitáveis, e somente serão detectados no fim do subsistema final.

Planos preditivos são expressões do paradigma industrial e das implementações da corrente de pensamento de ciclo aberto. Planos preditivos podem incluir apenas variáveis conhecidas e seu comportamento *esperado*. Planos preditivos criam a ilusão de que o comportamento das variáveis conhecidas é entendido com precisão e que outras variáveis são inexistentes. Planos preditivos convidam a uma longa consideração inicial de todos os elementos que devem fazer parte do plano e, em um contexto complexo, tentam prever o imprevisível. Para controlar variáveis não previstas ou comportamento inesperado, são necessários procedimentos pesados para verificar, manter e atualizar o plano preditivo.

- ❖ Em um *sistema de ciclo fechado*, o resultado real do sistema é regularmente comparado ao resultado desejado, a fim de eliminar ou diminuir gradualmente quaisquer variações indesejáveis. Nem todas as variáveis e parâmetros precisam ser conhecidos com precisão e em detalhes antecipadamente, pois o processo aplica a autocorreção e leva em consideração parâmetros novos ou variáveis. Essa técnica de inspeções regulares requer e cria transparência. A situação real é inspecionada e exposta, de modo que as adaptações mais apropriadas possam ser realizadas para preencher a lacuna entre os resultados efetivos e esperados. As pessoas que realizam as inspeções, os inspetores-jogadores, precisam de padrões claros e acordados para realizar suas inspeções. Daí a necessidade de transparência do processo e de todas as suas variáveis para todos os membros envolvidos.

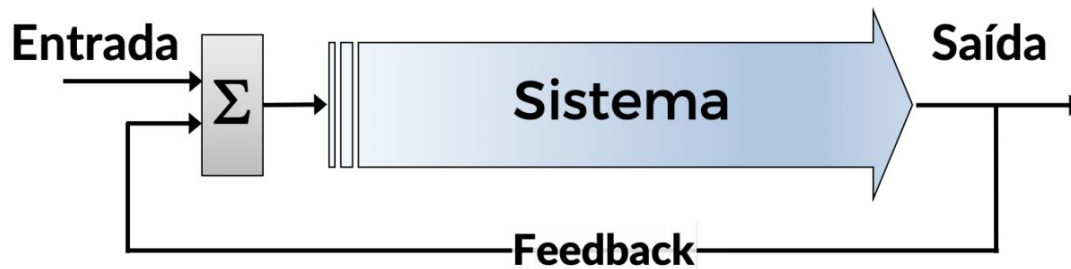


Figura 2.11 Sistema de ciclo fechado

O Scrum reconhece que a complexidade do desenvolvimento de produto requer o processo correto, ou seja, feedback em ciclo fechado. O Scrum substitui os ciclos abertos dos processos tradicionais pelo *empirismo* dos sistemas de ciclo fechado. O Scrum implementa oportunidades regulares de inspeção e adaptação, para que os jogadores possam aprender da inspeção, coletar feedback dos resultados e melhorar. O Scrum traz o controle baseado na realidade para o trabalho complexo.

O Scrum implementa dois ciclos específicos de feedback de ciclo fechado. Uma Sprint forma um ciclo de ‘inspecionar e adaptar’ que envolve as ‘inspeções e adaptações’ a cada 24 horas na Daily Scrum:

- ❖ Na *Daily Scrum*, o Development Team inspeciona seu progresso e planeja seu próximo trabalho dentro do contêiner da Sprint. Eles usam o Sprint Backlog e o Objetivo da Sprint, como inicialmente definidos na Sprint Planning e uma tendência de progresso para considerar o trabalho restante. Isso garante que eles não saiam de sincronia entre si no time e com o Objetivo da Sprint por mais de 24 horas.
- ❖ Uma *Sprint* é um ciclo que começa com a projeção do trabalho e termina com reflexões sobre o que realmente foi construído, o Incremento do produto, e como ele foi construído, o processo, as interações do time e a tecnologia.

Os eventos do Scrum definem a frequência da inspeção e adaptação, onde os artefatos contêm as informações a serem inspecionadas e adaptadas:

Evento	Inspeção	Adaptação
Sprint Planning	• Product Backlog (Performances passadas) (Disponibilidades) (Comprometimentos da Retrospectiva) (Definição de Pronto)	• Previsão • Sprint Backlog • Objetivo da Sprint
Daily Scrum	• Progresso da Sprint (em direção ao Objetivo da Sprint)	• Plano Diário • Sprint Backlog
Sprint Review	• Incremento do Produto • Product Backlog (& progresso) • Condições de Mercado & Negócio	• Product Backlog
Sprint Retrospective	• Equipe & colaboração • Tecnologia & desenvolvimento • Definição de Pronto	• Ações de melhorias (próxima Sprint)

Figura 2.12 Empirismo no Scrum

O Scrum prevê esses eventos formais como oportunidades para inspecionar e adaptar-se à situação real, de modo que a arte do empirismo seja realizada no mais tardar durante esses eventos. Isso não deve impedir que os membros da equipe busquem melhorias e progridam sempre que necessário. Em um mundo do alto dinamismo que leva ao uso do framework Scrum, seria muito estranho que as equipes não aproveitassem novas informações e insights que melhorassem sua vida profissional tão logo possível.

Nenhum desses eventos foi projetado para fins de relatar status. Todos os eventos no Scrum são projetados para serem voltados ao futuro. A capacidade de adaptação define o nível de agilidade alcançado.

Inspeção sem adaptação é inútil no Scrum. Todos os eventos do Scrum são voltados para o futuro, como oportunidades para moldar o futuro.

2.7 Os Valores do Scrum

Scrum, como já foi demonstrado, é um *framework* sobre o qual pessoas e organizações desenvolvem um processo de trabalho que é específico e apropriado para o seu momento e contexto. Todas as regras e princípios do Scrum servem ao empirismo ou ao controle de processo empírico, como o mais ideal em tratar com desafios complexos em circunstâncias complexas.



Figura 2.13 Os valores do Scrum

Há, no entanto, mais do que as regras e os princípios. Scrum é mais sobre comportamento do que sobre processo. O framework Scrum é baseado em cinco valores principais [Schwaber & Beedle, 2001]. Embora esses valores

não tenham sido concebidos como parte do Scrum e não sejam exclusivos do Scrum, eles direcionam o trabalho, o comportamento e as ações no Scrum.

Em um contexto Scrum, nossas decisões, os passos que damos, a forma como jogamos o jogo, as práticas que incluímos e as atividades que empreendemos dentro do Scrum, devem reforçar todos esses valores, não diminuí-los ou miná-los.

Scrum é um framework de regras, princípios e ... valores.

2.7.1 Comprometimento

A definição geral de ‘comprometimento’ é “*o estado ou qualidade de estar dedicado a uma causa, atividade, etc*”. Isso pode ser ilustrado por um treinador de equipe declarando “Eu não poderia culpar os meus jogadores pela falta de comprometimento” (embora talvez eles tenham perdido um jogo).

Isso descreve exatamente como comprometimento é pretendido no Scrum. Comprometimento se refere à dedicação e se aplica às ações e à intensidade do esforço. Não se refere ao resultado final porque isso por si mesmo é incerto e imprevisível para desafios complexos em circunstâncias complexas.

Ainda, havia uma má interpretação amplamente espalhada da palavra comprometimento no contexto do Scrum. Isso se origina principalmente da antiga expectativa do framework Scrum que dizia que times deveriam ‘se comprometer’ com a Sprint. Através das lentes do paradigma industrial tradicional, isso estava traduzido incorretamente dentro da expectativa de que todo o escopo selecionado no Sprint Planning seria finalizado até o final da Sprint, impreterivelmente. ‘Comprometimento’ estava equivocadamente convertido em um contrato pré-definido.

No complexo, criativo e altamente imprevisível mundo de desenvolvimento de um novo produto, a promessa de um escopo mediante tempo e

orçamento não é possível. Muitas variáveis influenciando o resultado são desconhecidas ou talvez se comportem de modo imprevisíveis.

Para melhor refletir a intenção original e conectar mais efetivamente ao empirismo, ‘comprometimento’ no contexto do escopo de uma Sprint foi substituído por ‘projeção’.

Contudo, comprometimento ainda é e permanece como um valor fundamental do Scrum:

Os membros se comprometem com o time. Comprometem-se com qualidade. Comprometem-se em colaborar. Comprometem-se em aprender. Comprometem-se em fazer o melhor que podem, todos os dias. Comprometem-se com o Objetivo da Sprint. Comprometem-se em agir como profissionais. Comprometem-se em auto organizar. Comprometem-se com excelência. Comprometem-se com os valores e princípios Ágeis. Comprometem-se em criar versões operacionais do produto. Comprometem-se em buscar por melhorias. Comprometem-se com a Definição de Pronto. Comprometem-se com o framework Scrum. Comprometem-se em focar em valor. Comprometem-se em finalizar o trabalho. Comprometem-se em inspecionar e adaptar. Comprometem-se com transparência. Comprometem-se em desafiar o contexto atual (status-quo).

2.7.2 Foco

As balanceadas, porém distintas responsabilidades do Scrum possibilitam que todos os membros foquem em suas competências.

O time-box do Scrum incentiva os membros a focarem no que é mais importante naquele momento sem serem interrompidos por análises do que possa vir ter a chance de se tornar importante em algum momento futuro. Eles focam no que eles conhecem agora. YAGNI (Você não vai precisar disso. Em inglês: ‘You Ain’t Gonna Need It’), um princípio da eXtreme Programming, ajuda a manter esse foco. Os membros focam no que é imediato já que o futuro é altamente incerto e eles querem aprender com o presente para ganhar experiência para o trabalho futuro. Eles focam no

trabalho necessário para terminar as coisas. Eles focam na coisa mais simples que possivelmente funcione.

O Objetivo da Sprint dá o foco para um período de 4 semanas ou menos. Dentro desse período, a Daily Scrum ajuda que as pessoas colaborativamente foquem no trabalho diário imediato necessário para progredir o melhor possível em direção ao Objetivo da Sprint.

2.7.3 Abertura

O empirismo do Scrum exige transparência, abertura e honestidade. Os membros-inspetores querem verificar a situação atual para realizar adaptações sensatas. Os membros são francos sobre o seu trabalho, progresso, aprendizado e problemas. Mas eles também estão abertos para pessoas e para trabalhar com pessoas; reconhecendo que pessoas são pessoas e não recursos, robôs, engrenagens ou peças de reposição de um maquinário.

Os membros estão abertos para colaborar através de disciplinas, habilidades e funções. Eles estão abertos para colaborar com os stakeholders e com o ambiente ao seu entorno. Estão abertos em compartilhar sua opinião (feedback) e aprendizado uns com os outros.

Eles estão abertos a mudanças assim que a organização e o mundo em que eles operam mudem; imprevisivelmente, inesperadamente e constantemente.

2.7.4 Respeito

O mais amplo ecossistema Scrum prospera no respeito às pessoas, suas experiências e seu histórico pessoal. Os membros respeitam diversidade. Eles respeitam opiniões diferentes. Eles respeitam as habilidades de cada um, suas competências e visões.

Eles respeitam o ambiente ao seu entorno em não se comportando como uma entidade isolada no mundo. Eles respeitam o fato que consumidores

mudam sua opinião. Eles mostram respeito com os patrocinadores (sponsors) em não criar e manter funcionalidades que nunca serão usadas e que aumentam o custo do produto. Eles mostram respeito em não desperdiçar dinheiro em coisas que não trazem valor, depreciadas ou que talvez nunca sejam implementadas ou usadas de qualquer maneira. Eles demonstram respeito aos usuários corrigindo seus problemas.

Todos os membros respeitam o framework Scrum. Eles respeitam as responsabilidades do Scrum.

2.7.5 Coragem

Os membros demonstram coragem em não criar coisas que ninguém quer. Coragem em admitir que requisitos nunca serão perfeitos e que nenhum plano pode capturar realidade e complexidade.

Eles mostram a coragem em considerar mudanças como recurso de inspiração e motivação. Coragem em não entregar versões de produto inacabadas. Coragem em compartilhar todas as possíveis informações que possam ajudar o time e a organização. Coragem em admitir que ninguém é perfeito. Coragem em mudar de direção. Coragem em compartilhar riscos e benefícios. Coragem em deixar para trás as certezas do passado.

Os membros demonstram coragem em promover Scrum e empirismo para tratar com complexidade.

Eles mostram coragem em apoiar os valores do Scrum. A coragem para tomar uma decisão, agir e progredir, sem travar. E ainda mais coragem para mudar aquela decisão.

3. Táticas para um propósito

Desde a primeira descrição formal em 1995, a definição do Scrum evoluiu gradualmente através de pequenas atualizações funcionais. Os elementos básicos ainda são os mesmos, assim como os princípios e regras que os unem. Mas as prescrições obrigatórias do Scrum se tornaram... mais leves, como a evolução do Guia do Scrum [Schwaber & Sutherland, 2017] mostra.

A evolução serve para descrever “o que” é esperado, ou seja, o propósito das regras a partir de uma compreensão do seu “porquê”, ao contrário de instruir “como” aplicar exatamente as regras.

O capítulo anterior descreve as regras para jogar o jogo do Scrum. Aquelas regras do Scrum Framework deixam espaço para diferentes *táticas* para jogar o jogo, táticas que são do tamanho correto a qualquer momento e podem ser implementadas em diferentes contextos e circunstâncias. É como em todos os jogos e esportes, todos jogam pelo mesmo conjunto de regras, mas alguns são mais bem-sucedidos do que outros. O sucesso depende de muitos fatores e nem todos são igualmente controláveis pelos próprios jogadores, mas o sucesso é certamente influenciado pelas táticas escolhidas para jogar o jogo.

É como selecionar boas práticas de uma coleção de práticas muito boas e transformá-las em melhores práticas, aplicando-as e ajustando-as a um contexto específico. Quando referido como um “processo”, Scrum é um *processo-servente*, não um *processo-comandante*. O Scrum não diz quais práticas devem ser aplicadas ou não. O Scrum ajuda a descobrir se elas funcionam, mas deixa aos jogadores a escolha de continuar fazendo-as ou alterá-las.

Há muitas táticas para usar no Scrum. Boas táticas servem o objetivo do Scrum. Boas táticas reforçam os valores do Scrum ao invés de diminuí-los. Quando bem aplicado, o sistema resultante ainda é reconhecível como... Scrum.

Vamos dar uma olhada mais detalhada em alguns exemplos para esclarecer a diferença entre táticas e regras:

OceanofPDF.com

3.1 Visualizando o progresso

Uma boa ilustração da evolução da estrutura do Scrum para uma maior leveza é a eliminação de gráficos burn-down como obrigatórios.

Olhando para as regras do Scrum, incluindo a necessidade de transparência, que é crucial para o processo de inspeção e adaptação, e auto-organização, é claro o “porquê” é importante visualizar o progresso. A autocorreção é difícil de acontecer sem rastreamento e visualização do progresso.

A antiga obrigação, no entanto, de usar gráficos burn-down para esse propósito (o “como”) foi removida da definição do Scrum. A forma ou formato da visualização não é mais prescrito. Ele é substituído pela mera, mas explícita, expectativa de que o progresso nos artefatos obrigatórios do Scrum do Product Backlog e Sprint Backlog é visualizado (o “o quê”).

Observe um exemplo de um gráfico burn-down de Sprint (Figura 2.7) no capítulo 2.5.3 (“Acompanhamento de progresso”). Gráficos de Sprint burn-down são um grande instrumento de autogestão para um Development Team dentro de uma Sprint.

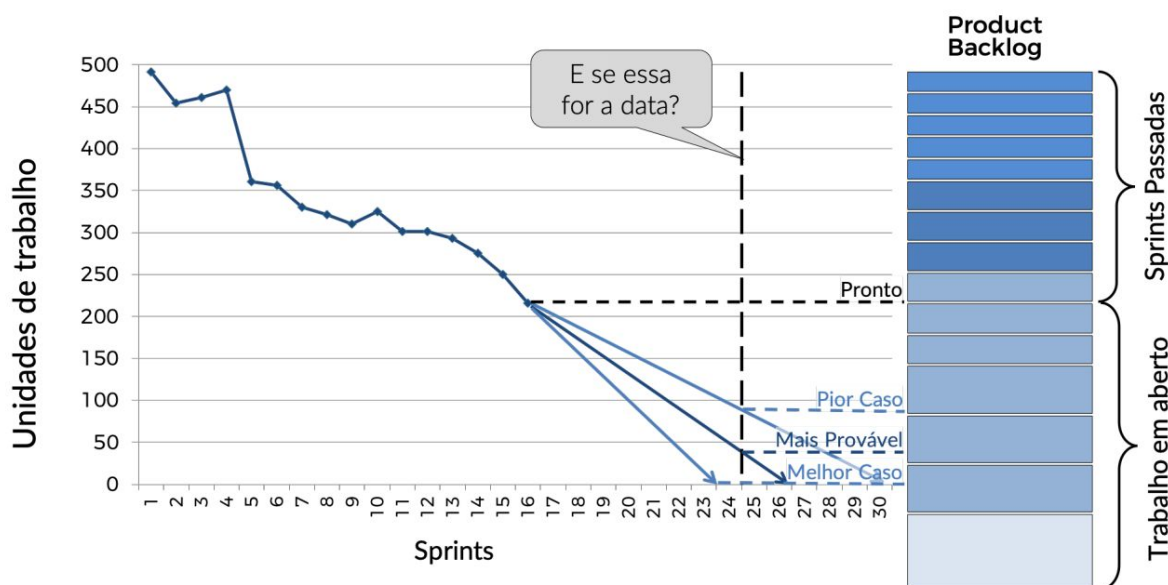


Figura 3.1 Exemplo de um gráfico Burn-down de Release

A seguir um exemplo de um gráfico Burn-down de uma Release, uma maneira de acompanhar e visualizar o progresso de um Product Backlog, ou parte de um Product Backlog, servindo a uma meta específica. A visualização suporta o Product Owner na comunicação com os stakeholders, os usuários e a mais ampla organização de gerenciamento do produto. Um gráfico Burn-down de uma Release e a previsão visual inclusa, permite conversas sobre o balanceamento de tempo e escopo com base nos resultados atuais entregues.

Gráficos Burn-down são uma ótima maneira de jogar o jogo e são adequados em muitas situações. No entanto, eles foram transformados em uma prática boa não obrigatória.

Sim, é Scrum se ambos Product e Sprint Backlog existem e uma visualização de seus progressos estão disponíveis, acessíveis e claras. Mas existem várias boas práticas para tal visualização. Pode ser um gráfico burn-down com esforço aberto. Pode ser um Diagrama de Fluxo Cumulativo. Pode ser tão simples como um Quadro Scrum. Para o progresso do Product Backlog, pode ser também um gráfico de burn-up, e.g. em valor criado.

3.2 As perguntas da Daily Scrum

O Scrum sugere que na Daily Scrum cada membro do time responda a três perguntas no que diz respeito ao andamento da equipe em direção ao seu objetivo da Sprint (*Feito? Planejado? Impedimentos?*).

Mas mesmo que os membros respondam às perguntas, eles ainda podem limitá-las a uma atualização individual de status. Eles podem usar as paredes ou o Quadro Scrum para fins de reportar. Eles podem apenas se certificar de simplesmente responder as três perguntas. Isso talvez aconteça pela incapacidade das pessoas em olhar além da sugestão do Scrum de responder às perguntas. As regras observadas são formalmente seguidas sem entender o “porquê”.

O time está vendo o Scrum apenas como uma metodologia? Ou está usando o Scrum como uma estrutura para descoberta e colaboração? Não ajuda muito se eles formalmente responderem às três perguntas ou não, se eles realmente não ouvirem e falarem uns com os outros. Não ajuda muito se eles não revelarem as informações para otimizar seu plano de trabalho compartilhado para as próximas 24 horas em direção ao objetivo da Sprint. Talvez usem o evento apenas como uma obrigação formal, um remanescente mental do paradigma industrial. Talvez se sintam pressionados para garantir que todas as suas micro tarefas sejam registradas e se cubram contra possíveis culpas. Em fazendo assim, no entanto, eles perdem a oportunidade de obter discernimento da situação real, para inspecioná-la e se adaptar de forma rápida e fluente.

Inspeção sem adaptação é inútil em ambientes complexos e em mudança. O objetivo da Daily Scrum é compartilhar informações, e replanejar o trabalho coletivo do Development Team para que eles avancem da melhor maneira possível para o Sprint Goal. A equipe define seu próprio formato para fazê-la, dentro do limite de mantê-la em 15 minutos ou menos. Essa deve ser a base a partir da qual a Daily Scrum é executada, independente das perguntas serem endereçadas, não percorra cegamente por elas do ponto de vista das “melhores práticas”.

Você sabia que uma Daily Scrum não é necessariamente uma Daily Stand-up?

A Daily Stand-up é uma prática descrita em eXtreme Programming [Beck, 2000] e que serve ao mesmo propósito da Daily Scrum no Scrum. Mas o eXtreme Programming diz aos participantes para fazê-la de pé.

No Scrum não há obrigação de fazê-la de pé. No entanto é uma boa tática, especialmente para manter o time-box dentro de 15 minutos.

OceanofPDF.com

3.3 Refinamento do Product Backlog

O refinamento do Product Backlog é uma atividade corrente durante a Sprint na qual o Development Team e o Product Owner olham para o Product Backlog atualmente ordenado para as próximas Sprints. Há uma certeza crescente de que os itens serão realmente implementados.

Com os itens se aproximando para serem implementados, os times podem querer estudar dependências, entender melhor o que é esperado do trabalho, decidir sobre uma abordagem compartilhada para seu desenvolvimento ou ajudar um Product Owner a compreender o impacto do desenvolvimento num nível funcional. O refinamento colaborativo do Product Backlog e o conhecimento adicional que emerge da conversação do refinamento, aumentam as possibilidades que o trabalho possa realmente, ou mais facilmente, ser puxado para uma Sprint quando apresentado numa Sprint Planning futura.

Refinamento do Product Backlog não é um evento oficial com time-box. A ambição do Scrum é permanecer simples, mas suficiente. A ambição do Scrum é ajudar as pessoas e os times a descobrir práticas específicas que podem ou não serem apropriadas em seu contexto específico. Refinamento do Product Backlog é uma atividade que muitas equipes executam para suavizar suas Sprints, e reduzir turbulência nos primeiros dias de uma Sprint. Uma atividade típica de refinamento do Product Backlog é a estimativa de esforço ou custo ser criada ou revisada. Outras equipes podem demandar menos precisão na Sprint Planning ou ter um relacionamento com o Product Owner que é menos sobre a precisão. Eles lidam bem sem isso, ou o fazem menos formalmente, sem nomear ou planejar de forma explícita e consciente esta atividade. Eles iriam perceber como opcional ou até mesmo como uma sobrecarga se fosse um evento obrigatório tendo um time-box e uma hora definida de ocorrência.

Refinamento do Product Backlog é uma grande atividade dentro da Sprint, uma boa tática para gerenciar colaborativamente o Product Backlog. Alguns podem fazer sem, entretanto.

OceanofPDF.com

3.4 Histórias de Usuário

Em eXtreme Programming [Beck, 2000] os requisitos eram capturados em “Histórias de Usuário”. As Histórias de Usuário eram escritas em cartões de índice e descreviam expectativas funcionais da perspectiva de um usuário. Bill Wake, um praticante precoce de eXtreme Programming, sugeriu o “INVEST”^[5] acrônimo como uma maneira simples de lembrar e avaliar se ou não uma História de Usuário está bem formada: Independente, Negociável, Valiosa, Estimável, de Tamanho adequado, Testável.

Histórias de Usuário geralmente descrevem uma funcionalidade como uma “História” a partir da perspectiva do “Usuário”. A vantagem de tomar a perspectiva do usuário para descrever o requisito de sistema ou aplicativo é o foco no valor do trabalho para esse usuário.

Os cartões são fáceis de mover, ou serem removidos de um quadro de planeamento, como um radiador da informação. Outra vantagem de usar cartões físicos para as histórias é o espaço limitado para descrições textuais e detalhes. Assegura-se que ele está incompleto por design e desta maneira certifica-se de que uma conversa ocorra. Quando uma história de usuário chega mais próxima de ser feita e a chance de ela ser implementada cresce, isso necessariamente requer uma conversa para descobrir detalhes adicionais. Mais informações podem ser adicionadas ao cartão e algumas destas podem ser expressas como critérios de aceitação para a história de usuário, a confirmação de que a história está bem implementada. Esses critérios de aceitação são tipicamente escritos na parte de trás do cartão.

O Product Backlog no Scrum serve para dar transparência de *todo* trabalho que um time precisa fazer. Isto compreende mais do que apenas requisitos funcionais. Embora o formato de história de usuário possa ser usado para outros tipos de requisitos além do funcional, não há um encaixe natural. Ele tende a colocar o foco na sintaxe em vez da informação a ser transmitida.

No Scrum, não há nenhuma obrigação em usar o formato de história de usuário para itens do Product Backlog. Isso até cria o risco de negligenciar outros trabalhos importantes que precisam ser executados, ou podem forçar

os times a gastar mais tempo e energia usando o formato “certo”, criando assim desperdício. No entanto, para itens funcionais do Product Backlog, as histórias de usuário podem ser muito boas, uma ótima tática.

OceanofPDF.com

3.5 Planning Poker

Planning Poker é uma técnica de estimativa inventada por James Grenning durante um projeto com eXtreme Programming onde ele sentia que muito tempo era gasto apenas produzindo estimativas.

No Planning Poker, um time tem uma discussão sobre um requisito, após a qual cada membro do time decide sobre uma estimativa para o requisito, escolhendo um valor de um conjunto de cartas do poker. As cartas desse poker tipicamente usam uma escala exponencial como a sequência de Fibonacci (1, 2, 3, 5, 8, 13, 21, 34, 55, ...). Os membros do time mantêm o valor escolhido para si até que todos tenham escolhido um valor. Então, todos revelam suas estimativas ao mesmo tempo, seguida de uma conversa sobre possíveis diferenças. Este ciclo é repetido até que o acordo seja alcançado, uma compreensão conjunta do requisito. As estimativas são relativas umas às outras e são expressas em uma unidade abstrata como Story Points, ou mesmo Ursinhos Gummy como nos primeiros projetos com eXtreme Programming.

No Scrum, as estimativas de itens do Product Backlog são de responsabilidade do Development Team. Como parte da transparência e colaboração, é necessário haver estimativas honestas e imparciais de todo o Development Team, uma perspectiva coletiva, incluindo todas as habilidades e expertise presentes na equipe.

Embora não seja obrigatório, o Planning Poker é uma boa tática para esse princípio. Mas não se esqueça que a intenção final é invocar uma discussão honesta sobre as estimativas, porque isso resulta em uma boa compreensão do trabalho referente à implementação do item discutido.

3.6 Duração da Sprint

O Scrum define somente a duração máxima de uma Sprint, ou seja, não mais de quatro semanas. Esta duração máxima assegura que ninguém é privado do direito para adaptar os planos futuros para o produto pelo menos a cada 4 semanas. Além disso, a equipe não fica trancada num contêiner por muito tempo, correndo o risco de perder o controle sobre um mundo em mudança.

A duração da Sprint mantém um equilíbrio entre retenção de foco para obter uma quantidade substancial de trabalho pronto e adaptabilidade oportunista, pesados contra outros fatores, como incertezas tecnológicas e oportunidades de aprendizado.

Em um processo empírico como Scrum, os objetivos de controle são apresentados ao sistema e, por meio de um ciclo de feedback fechado, os resultados são regularmente inspecionados contra estes objetivos para adaptar materiais de trabalho, tarefas e processos. Inspetores qualificados, os papéis previstos no Scrum, realizam inspeções em uma frequência apropriada, de modo que o foco e o tempo exigidos para criar um resultado valioso são balanceados junto ao risco de permitir uma variação demasiada na saída criada.

Além da transparência, a frequência é um fator importante no empirismo. Os eventos do Scrum determinam a frequência das inspeções e adaptações, com o Sprint sendo um evento contêiner, o ciclo de feedback externo que envolve o ciclo de feedback da Daily Scrum, e os ciclos de feedback das práticas de desenvolvimento aplicadas na Sprint.

Há uma clara tendência a passar a usar Sprints mais curtas. Embora não seja uma obrigação formal, Sprints de uma semana parecem um mínimo aceitável.

Vamos dar uma olhada nisso presumindo que um time faz Sprints de um dia.

Todos os eventos Scrum, como oportunidades para inspecionar e adaptar, ocorrem na duração de um dia, e são organizados numa frequência muito alta. Na verdade, ao tentar realizar todos os eventos, a equipe passa uma quantidade de tempo razoável inspecionando e adaptando o menor dos pacotes de trabalho. A frequência fica no caminho da produção de valor.

O perigo é ainda maior pois o time irá se focar meramente no seu trabalho diário e progresso, perdendo de vista o panorama geral. Eles não terão tempo para inspecionar e adaptar o processo geral, ou provar maneiras de melhorar a qualidade ou se conectar a metas e objetivos abrangentes. Eles apenas tentarão empurrar mais produto da porta pra fora, todos os dias.

A duração da Sprint também determina a frequência na qual o Product Owner e o Development Team consultam os stakeholders sobre uma versão funcionando do produto. A finalidade é compartilhar toda informação necessária para ajudar o Product Owner a tomar decisões sobre o futuro do produto. No caso de Sprints de um dia, será mais difícil conseguir o apoio dos stakeholders, e também a permissão para captar e adaptar às mudanças corporativas, de mercado e estratégicas.

A duração da Sprint deve levar em conta o risco de perder uma oportunidade de negócio porque as Sprints são demasiadas longas. Se seu negócio é tão volátil que você arrisca perder oportunidades gastando mais de um dia no contêiner de uma Sprint, por favor faça Sprints de um dia. Mas tenha cuidado para não queimar os mecanismos de inspeção numa frequência tão alta.

Evite que o Scrum se transforme num novo nome para a velha corrida dos ratos, onde não há lugar para humanização no local de trabalho. Organize o trabalho de forma que seja sustentável, indefinidamente.

Considere a duração da Sprint como uma tática para rodar o Scrum. Veja como funciona e se adapte de acordo, tendo em mente a estabilidade, cadência e ritmo sustentável. E se você quiser fazer uma *release* mais cedo que no fim da Sprint, nada no Scrum diz que você não pode. A finalidade das Sprints e do evento da Sprint Review permanecem intactos da mesma maneira.

OceanofPDF.com

3.7 Como escalar o Scrum

Os elementos obrigatórios do Scrum foram descritos, bem como as regras para jogar o jogo do Scrum, as regras que unem coesamente esses elementos. As regras permanecem consistentes e são independentes da escala em que o Scrum é organizado.

Scrum promove a simplicidade. O Scrum promove a responsabilidade clara e a colaboração de pares para lidar com a imprevisibilidade e formular respostas a problemas complexos.

A simplicidade, a responsabilidade e a colaboração *bottom-up* não estavam no cerne de muitas empresas ao escalar suas organizações e seu trabalho sobre o paradigma industrial. O principal desafio ao escalar o Scrum não está no encaixe do Scrum nas estruturas existentes, mas sim na revisão das estruturas existentes por meio de uma compreensão, implementação e crescimento *bottom-up* do Scrum, mantendo as regras básicas do jogo intactas e respeitando-as. Adaptar a organização ao Scrum, não o contrário.

Existem algumas táticas que permitem que o Scrum seja jogado em uma escala maior, dependendo do contexto.

3.7.1 Scrum de Um time (“Single-team Scrum”)

A situação mais simples para entregar produto com Scrum é ter um Product Backlog expressando os desejos para o produto e ter um time entregando incrementos de produto em Sprints com time-box.

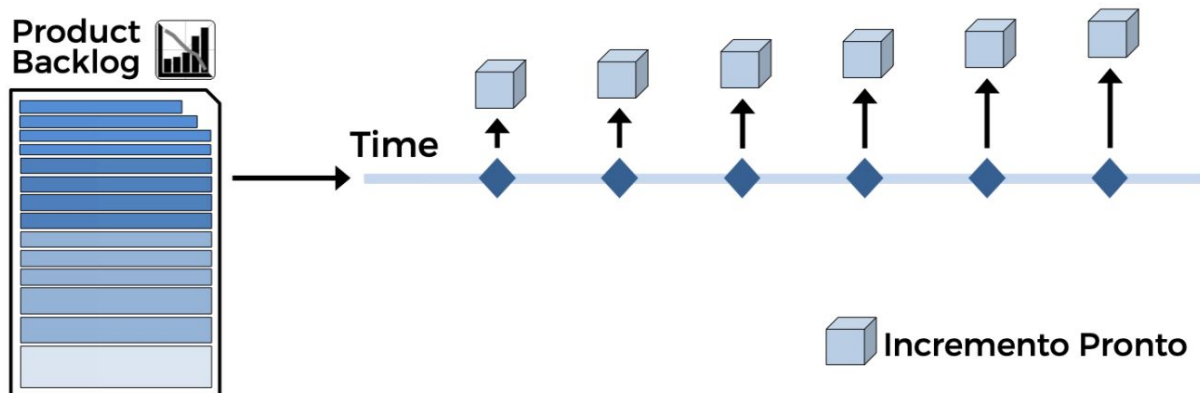


Figura 3.2 Scrum de Um Time

O Development Team tem todas as habilidades para transformar vários itens de Product Backlog em um incremento lançável de produto por Sprint, guiado pela Definição de Pronto. O Development Team gerencia seu trabalho de forma autônoma através do Sprint Backlog e garantem direção e alinhamento por meio da Daily Scrum. O Product Owner fornece esclarecimentos funcionais e de negócios no tempo certo. O Scrum Master treina, facilita e serve o time e a organização.

O único time é o Scrum Team, tendo um Product Owner, um Development Team e um Scrum Master. A fim de prover as funções do produto que fornecem valor ao usuário de ponta-a-ponta, o Development Team é o que normalmente é chamado de um “feature team”.

O maior desafio reside em ter todas as habilidades de desenvolvimento colaborando como um time. Mas se esse problema estiver superado, a Sprint Review é totalmente transparente, um importante pré-requisito para fazer a abordagem empírica do Scrum funcionar. O time usa a Sprint Retrospective para melhorar a si mesmo.

3.7.2 Scrum de Vários Times (“Multi-team Scrum”)

Para produtos maiores ou resultados mais rápidos, a necessidade de criar e lançar um produto com vários times pode emergir.

Os vários times entregam um só produto. Eles trabalham no mesmo Product Backlog. O sistema coletivo possui um Product Owner, vários Development Teams e um ou mais Scrum Masters. Cada Development Team cria seu Sprint Backlog para trabalhar a partir da previsão. Cada Development Team se auto adapta através da Daily Scrum e assegura a integração do trabalho com os outros Development Teams.

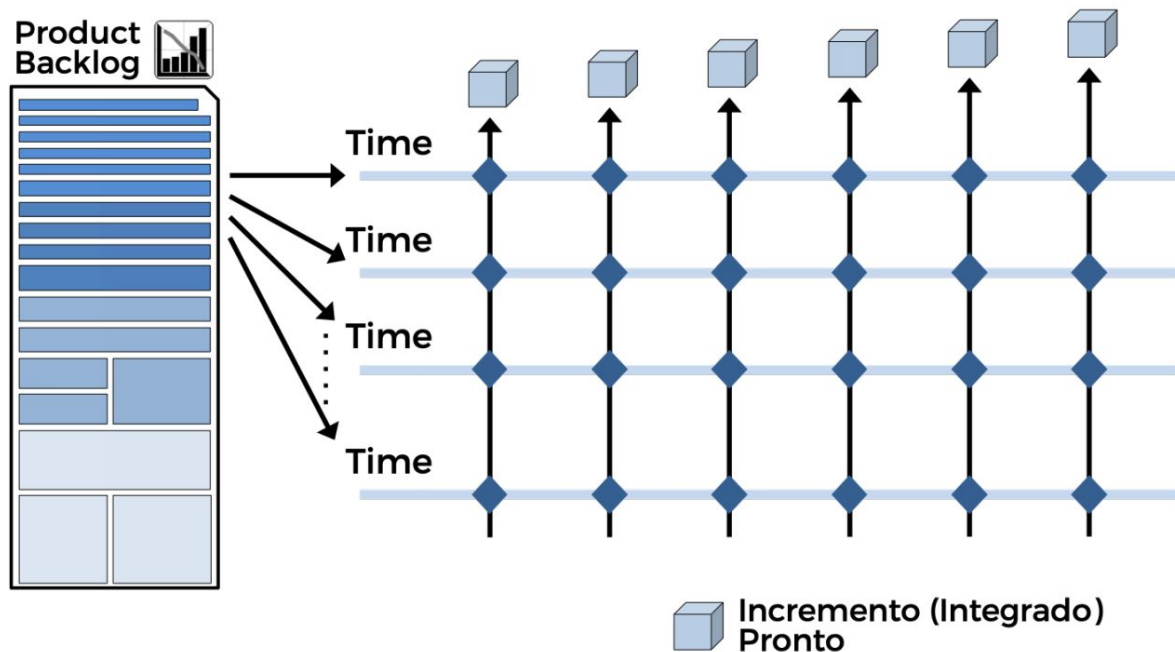


Figura 3.3 Scrum de Vários Times

A necessidade por transparência total na Sprint Review permanece, transparência completa sobre a versão do produto como lançado ou potencialmente lançável. Os Incrementos de Produto não podem ter trabalho não-finalizado escondido deixado para trás. No entanto, os múltiplos times estão construindo conjuntamente o mesmo produto. Apenas um Incremento totalmente *integrado* assegura ao Product Owner e aos stakeholders total transparência.

Os múltiplos times se auto organizam, dentro dos limites das regras e princípios do Scrum. Ao trabalhar num conceito de Múltiplos Times Scrum, ou seja, várias equipes que criam e sustentam o mesmo produto, os times se organizam na expectativa de criar um incremento integrado até o fim da Sprint.

Comunicação regular através dos múltiplos times é necessária, durante toda a Sprint, para alinhar os planos de trabalho do time com o objetivo de criar um Incremento integrado. Os times entendem o princípio e propósito da Daily Scrum a um nível inter-equipes e organizam eventos *Scrum-of-Scrums*.

A fim de manter o todo otimizado, o Scrum-of-Scrums acontece antes das Daily Scrums individuais. Os representantes mais apropriados dos Development Teams se reúnem para trocar informações de desenvolvimento, centrando-se principalmente no estado de integração do produto. Posteriormente, cada Development Team pode otimizar o planejamento e ajustar os Sprint Backlogs individuais dentro do ecossistema multi-time. Como resultado, os múltiplos times otimizam o seu progresso conjunto para um incremento integrado de produto, o mais tardar ao final de cada Sprint. O Incremento tecnicamente saudável pode ser liberado após a avaliação do Product Owner considerando se tem o nível funcional correto, não prejudicado por demandas de desenvolvimento em aberto ou desconhecidas, ainda a serem realizadas.

Os múltiplos times trabalham pelos mesmos critérios de qualidade para o produto como expressos na Definição de Pronto. Os múltiplos times podem achar mais fácil trabalhar em Sprints do mesmo tamanho, para simplificar planejamento, integração, lançamento e revisão do trabalho. Serão previstas tarefas adicionais nos Sprint Backlogs individuais para manter o trabalho integrado e saudável.

A fim de trabalhar em Sprints de diferentes tamanhos, um acordo claro e políticas são cruciais para assegurar que todo o trabalho será mantido integrado continuamente, uma política da árvore verde que tem precedência em todos os momentos. Se alguma coisa quebra o todo, é imprescindível que isso seja corrigido primeiro. Como resultado, cada time individual ou cada combinado de times (potencialmente independente) pode derivar um incremento lançável da base de código compartilhada. *Sprint Reviews compartilhadas ainda fazem muito sentido.*

Todas as responsabilidades definidas pelo Scrum são cumpridas. O todo do sistema é reconhecidamente Scrum. Para entregar incrementos de produto que proveem o valor ao usuário do início ao fim, o todo é um “feature

system”. Independente da composição dos times individuais, eles coletivamente entregam incrementos integrados de produto, o mais tardar até o fim de uma Sprint.

3.7.3 Scrum de Vários Produtos (“Multi-product Scrum”)

De acordo com a interdependência funcional ou técnica de vários produtos, pode haver a necessidade do planejamento e implementação dos mesmos serem alinhados e sincronizados.

Cada produto tem um Product Owner. Para cada produto existe um Product Backlog com um ou vários times para criá-lo, entregá-lo e mantê-lo. Cada ecossistema de produto é um Scrum de time único ou uma instância Scrum de vários times.

Das responsabilidades do Scrum, está claro que alinhamento e sincronização primariamente acontecem ao nível dos Product Backlogs por meio dos respectivos Product Owners. Product Backlogs individuais são adicionalmente ordenados sobre a otimização da linha de produtos, a suíte de produtos, considerações de programa ou portfólio. Os Product Owners, auxiliados pela organização, gerenciam incrementalmente seus Product Backlogs com base em informações compartilhadas e em progresso compartilhado.

As dependências técnicas e de desenvolvimento são tratadas dentro da Sprint pelos Development Teams dos diferentes hubs de produtos num espírito em rede, não hierárquico.

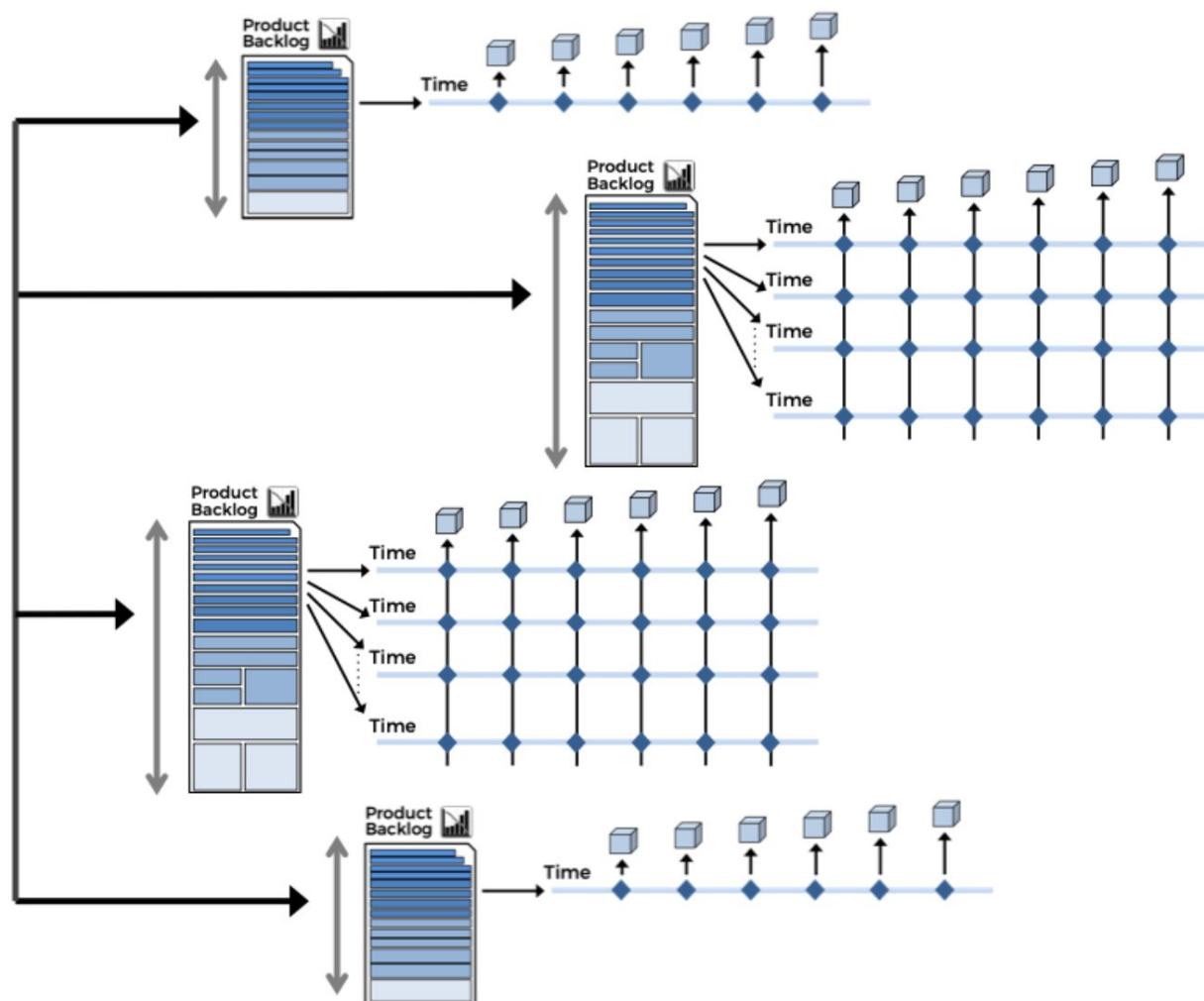


Figura 3.4 Scrum de Vários Produtos

Muitos outros problemas de escala e, portanto, cenários, existem. Não há uma solução bala-de-prata. O Scrum promove um pensamento bottom-up com o suporte top-down para descobrir e emergir o que melhor funciona para você, sua organização e seu contexto.

4. O estado futuro do Scrum

4.1 Sim, fazemos Scrum. E...

O Scrum surgiu na década de 1990 do trabalho e descoberta de Ken Schwaber e Jeff Sutherland. Eles analisaram e refletiram criticamente sobre práticas que na época eram consideradas comuns no desenvolvimento de software, sua própria experiência profissional, estratégias bem-sucedidas de desenvolvimento de produtos [Takeuchi & Nonaka, 1986] e teoria de controle de processos. A soma de suas descobertas se tornou Scrum [Schwaber, 1995] e foi formalmente apresentada ao público em 1995 [Sutherland, 1995]. Nos anos que se passaram desde a publicação do ‘Manifesto para o Desenvolvimento Ágil de Software’ [Beck, et. Al., 2001] em 2001, o Scrum se tornou o framework Ágil mais aplicado em todo o mundo.

No entanto, o Scrum continua sendo uma maneira leve e simples de organizar um trabalho complexo e abordar desafios complexos baseados nos princípios e ideias Ágeis. A combinação única entre a definição clara e a natureza pouco prescritiva do Scrum é primordial para o seu sucesso, agora e no futuro previsível. O Scrum, como um framework organizacional, pode envolver várias práticas de desenvolvimento de produtos existentes ou pode tornar as práticas existentes supérfluas. O Scrum provavelmente revela a necessidade de novas práticas. Os benefícios do Scrum serão maiores quando complementados por desenvolvimento, gerenciamento de produtos, pessoas e práticas organizacionais aprimoradas ou revisadas. Quando bem feito, a soma ainda será Scrum.

Nas duas primeiras décadas de sua existência, as organizações utilizaram o Scrum principalmente para os aspectos de TI e tecnologia do desenvolvimento de software. Para muitas pessoas de TI em todo o mundo, o framework Scrum se tornou uma solução comprovada. Apesar dos ótimos resultados, como o Ágil ultrapassando o Cascata (desenvolvimento em cascata) e a posição de gorila do Scrum, há espaço para melhorias. Há necessidade de ir mais longe.

Desafiar o status quo do paradigma industrial melhorou o aprendizado contínuo ao lidar com muitas incertezas *tecnológicas* no domínio de TI e ajudou muitas organizações a deixar de organizar o trabalho em projetos e recuperar o foco nos produtos. Em muitas organizações, o entendimento foi restaurado de que o desenvolvimento de software e produtos é uma atividade criativa e complexa. Mas o foco muitas vezes ainda é ‘como’ os produtos são construídos. É hora de elaborar os resultados alcançados e levar seu Scrum ao próximo nível.

Existe uma infinidade de possibilidades para ‘rodar’ Scrum, uma miríade de práticas que podem ser envolvidas pelo Scrum, e os resultados e desempenho do trabalho gerenciado com o Scrum são influenciados por muitos fatores. A co-localização das pessoas influencia isso. A energia, dedicação e alegria dos membros-jogadores influenciam isso. O nível de auto-organização influencia isso. O fato de as pessoas trabalharem em multitarefa influencia isso. A disponibilidade de automação de desenvolvimento e teste, ferramentas, plataformas e práticas influenciam isso.

Um aspecto crucial é o pensamento *multifuncional* que vai além das paredes do departamento de TI, através de toda a empresa. Lembre-se de que o Ágil é impulsionado por oportunidades de negócios. Em ter implementado o Scrum para o ‘como’ da entrega do produto, adicionar mais foco agora ao ‘o que’ precisa ser construído, é crucial. Essa mudança ajudará as organizações a descobrir o poder do produto possível, sobre as restrições do produto previsto, reduzir a quantidade de produto a ser desenvolvido, em vez de simplesmente otimizar a maneira como o produto está sendo desenvolvido. Veja o *capítulo 4.2*.

Existe uma infinidade de técnicas e práticas para “rodar” Scrum e que o Scrum pode envolver. Mas mais do que processos e técnicas, passando do antigo paradigma industrial para o novo paradigma Ágil, trata-se de uma atualização organizacional. O entusiasmo comum que surge nas equipes ao adotar Scrum provavelmente não será suficiente para uma profunda transformação organizacional. Para um efeito duradouro, o entusiasmo bottom-up comum precisa ser apoiado e facilitado pela adoção upstream. Veja o *capítulo 4.3*.

OceanofPDF.com

4.2 O poder do produto possível

O valor que os produtos trazem pode ser grandemente aumentado se lidarmos melhor com o *que* está sendo entregue; os requisitos, funcionalidades e funções.

Em empresas, negócios e mercado em circunstâncias turbulentas, a certeza e a estabilidade dos requisitos são mínimas. A colaboração aprimorada e ativa com proprietários de negócios e gerentes de produtos é um próximo passo natural na otimização da entrega de produtos. Somente as pessoas responsáveis pelo valor de negócio dos produtos podem ajudar a superar a inevitável ausência de pleno acordo sobre funcionalidades e requisitos. E mais do que nunca, essas pessoas de produtos precisam da flexibilidade para aproveitar oportunidades imprevistas para criar o melhor produto possível no momento certo. O que é apreciado hoje pode não ser o que as pessoas estão procurando amanhã.

O framework Scrum permite que as pessoas desistam de tentar prever o imprevisível, pois o framework suporta a lidar com respostas, soluções e ideias concorrentes que surgem ao construir e desenvolver um produto, enquanto o produto já está sendo usado. O Scrum processa a questão de saber se problemas foram inicialmente considerados irrelevantes. Não se espera mais que as funcionalidades como entrada para o ecossistema de entrega de produtos sejam completos, finais e exaustivos. O Scrum ajuda a aceitar - e abraçar - o fato de que o acordo final sobre o '*o quê*' do produto só fica resolvido *enquanto* o criamos. O Scrum ajuda a validar frequentemente decisões internas em comparação ao uso do produto no mercado. O Scrum abre as portas para, e promove, lançamentos funcionais frequentes como uma maneira melhor de progredir, uma vez que eles incorporam feedback regular dos clientes e não apenas acumulam suposições via sistemas sequenciais de ciclo aberto. Quando a conexão com o mercado é estreita, o feedback real do usuário pode ser facilmente incorporado como requisitos emergentes ao Product Backlog que é um artefato vivo.

No Scrum, o Product Owner é o único que decide o que construir (em seguida). O Product Owner consolida o trabalho do(s) Development Team(s) em um próximo lançamento ou versão do produto para o mercado. A autoridade do Product Owner influencia o nível de melhoria e agilidade que uma organização alcança com o Scrum. Além da autoridade, um Product Owner do Scrum precisa ter uma conexão próxima com todos os domínios relacionados ao gerenciamento de produtos: marketing, comunicação, vendas, jurídico, pesquisa, finanças, suporte, etc. Por fim, o Product Owner atua como um CEO do produto.

É nesse sentido igualmente essencial que promove a colaboração multidisciplinar entre as paredes organizacionais dentro do gerenciamento de produtos. A capacidade de adaptar essas partes de uma organização aproveita o uso do Scrum para agilidade *corporativa*. Em um mundo globalizado de imprevisibilidade interna e externa sempre crescente, adotar uma mentalidade de empirismo e adaptabilidade é benéfico para as organizações como um todo.

Empregar o Scrum não é renomear ou refazer um pouco as antigas técnicas que estão enraizadas no paradigma industrial. O Scrum não exige que as pessoas do produto preencham o Product Backlog como faziam com os documentos de requisitos tradicionais. Tampouco é suficiente para os analistas atuarem como proxies para as pessoas de produto, se lhes faltam autoridade, apoio dos stakeholders, responsabilidade orçamentária e representação real do usuário.

As Sprints do Scrum são o núcleo da agilidade do negócio na geração de um fluxo regular de melhorias, aprendizados e várias outras fontes de valor; valor para os stakeholders, valor para os usuários e consumidores dos produtos, valor para as pessoas que o criam e mantêm.

Em última instância, uma empresa e seus mercados se tornam um contínuo auto equilibrado, com os membros contribuindo atravessando barreiras, domínios, habilidades e funções. As organizações podem descobrir, experimentar e oferecer oportunidades de uma perspectiva de ponta a ponta da maneira mais rápida possível com Scrum.

4.3 A adoção upstream do Scrum

Ao adotar o Scrum, a organização mais ampla é impactada. Ponto final.

Questões que vão além dos times Scrum surgirão e precisam ser atendidas para obter todos os benefícios do Scrum, para facilitar melhor os times Scrum e, assim, melhorar toda a entrega de produto. Oportunidades organizacionais e áreas de melhoria são descobertas através da aplicação do Scrum.

As organizações que querem usar o Scrum para progredir em sua jornada de agilidade devem estar cientes de que isso não será alcançado apenas por implementar o Scrum. O Scrum tem o potencial de ser uma *ferramenta* para ser ágil em nível organizacional ou corporativo. O Scrum não foi projetado para ser apenas um novo processo de TI, mas sim uma estrutura de regras, princípios e valores para permitir que as organizações desenvolvessem o imprevisível. O Scrum permite uma adaptação rápida para acompanhar o mercado e promover vantagem competitiva (novamente).

A grande maioria das organizações infelizmente agem como se ainda residisse na terra do *Mediocristão*. As características desse ‘estado’ da sociedade, como descrito por Nassim Nicholas Taleb em seu livro ‘The Black Swan’ [Taleb, 2007], mostram que o sucesso tem relação direta com as horas ou esforços gastos em trabalho não escalável e repetitivo. Taleb descreve como o Mediocristão tornou-se uma ilusão do passado e foi substituído pelo *Extremistão*, onde o sucesso depende da ‘produção de ideias e da elaboração de singularidades imprevisíveis. O Scrum tem tudo para levar os habitantes do Mediocristão ao Extremistão, para que eles se tornem, pelo menos, ‘um membro’ no Extremistão, e potencialmente um líder, um gigante. O Scrum pode ser o mecanismo de adaptação tão rápido que cabe aos seus concorrentes responder à mudança que você causa. A liderança do jogo é atingida, superando os concorrentes em campo, sendo o gigante.

Mas começa com a aceitação, ou melhor, aceitando que nós operamos em um estado de mercado do Extremistão. Começa com aceitar, ou melhor,

provocar, que nossas organizações devem mudar para não desaparecer. Os fundamentos industriais em que a maioria delas são construídas foram invalidados. *Nosso iceberg está derretendo*, é a metáfora do conto de Holger Rathgeber e do especialista em mudança John P. Kotter [Kotter & Rathgeber, 2006]. Ignorar ou desprezar a grande mudança em direção à complexidade é uma importante causa da falta de adoção upstream para o Scrum e isso limita os benefícios obtidos com o jogo do Scrum. Isso prejudica sua futura liderança e até sobrevivência.

Em organizações maiores, times Scrum e seus Scrum Masters têm controle limitado ou nenhum controle sobre as obrigações burocráticas relacionadas à entrega e ao lançamento de produtos. Muitas vezes, as equipes precisam operar com base nas expectativas de conformidade e nas regras cerimoniais que foram postas em prática como parte do paradigma industrial. Elas estão sendo mantidas apesar da experiência real do desenvolvimento de produtos e da falta de sucesso. Em muitos casos, eles cresceram em descompasso com as rápidas evoluções que são tão típicas para os mercados atuais, circunstâncias externas e evoluções internas da organização.

No entanto, a experiência com o Scrum em uma grande maioria desses casos é excelente e faz jus ao senso comum. Os habitantes da casa do Scrum apreciam o Scrum porque ele prospera e cria muito entusiasmo. Não é surpresa que seja exatamente por isso que a *adoção downstream* é geralmente enorme.

Seria de esperar que resultados comprovados, números melhorados e aumento da entrega de valor levassem mais interesse ao upstream tendo maior apoio para sua adoção real. A experiência contradiz essa expectativa.

Sua organização merece suporte ativo e explícito do upstream bem como a promoção do Scrum. Pense em gerenciamento operacional de TI, divisões de vendas, gerentes de entrega, departamentos de gerenciamento de produtos e gerenciamento hierárquico CxO.

É necessário um senso de urgência e a aceitação de que de fato é urgente. Começa aceitando a verdade inconveniente de que nenhum conforto, certeza e controle vêm de previsões e planos tradicionais. Conforto vem da realidade, da experiência comprovada, de resultados de trabalho

observáveis e dados empíricos, em vez de informações estáticas e manipuladas. Os formalismos tradicionais do paradigma industrial não resultaram em melhor execução. Requisitos mudam, requisitos inesperados aparecem, prioridades mudam.

A adoção upstream é uma questão de gerenciamento. A gerência não é inútil só porque o Scrum não tem um papel explícito para eles. O Scrum não possui prescrições, artefatos, eventos ou funções para muitas outras tarefas ou atividades interessantes e úteis dentro das organizações.

O objetivo de uma transformação Scrum duradoura na jornada da Agilidade é envolver os gerentes no jogo por meio de uma abordagem estruturada e iterativa-incremental para a mudança organizacional. O objetivo dos gerentes é se reinventar, reinventar o seu papel e seu trabalho. Tal abordagem prospera na urgência por melhorias, enquanto capitaliza no entusiasmo bottom-up que existe sobre o Scrum. Essa mudança não endereça áreas organizacionais em uma produção em massa ou em um modelo cascata. Uma transição em cascata típica começa com uma empresa introduzindo o Scrum e resolvendo primeiro o ‘problema’ das equipes multifuncionais. Isso muitas vezes revela uma falta de recursos de engenharia e suporte, de modo que esse domínio é abordado em seguida. Depois de endereçar a área de engenharia, uma empresa pode querer aumentar o envolvimento do pessoal de negócios. E assim por diante. Dependendo do tamanho da empresa, pode facilmente levar de um a três anos por área. No final, trata-se de uma série de sistemas de ciclo aberto sequenciados linearmente, que não oferecem chances de sucesso em alta complexidade.

Uma transformação organizacional baseada no Scrum endereça domínios corporativos simultaneamente, produzindo Incrementos de mudança. Uma transformação muda a forma como uma organização trabalha, em vez de adicionar trabalho ao que as pessoas já fazem. Uma transformação Ágil na adoção do Scrum simplifica a forma como o trabalho é feito, maximiza o trabalho não realizado. Uma transformação não segue uma direção reta; é de baixo para cima, de cima para baixo, da esquerda para direita, da direita para esquerda, de dentro para fora, de fora para dentro. Ao mesmo tempo. Uma transformação Ágil baseia-se na premissa de que as pessoas são

naturalmente Ágeis, têm a capacidade natural de se adaptar. O processo do Scrum segue as pessoas. Estrutura deve seguir o processo.

Equipes de mudança multifuncionais implementam pequenos passos nos domínios em paralelo, enquanto os efeitos gerais das etapas incrementais são medidos. As inspeções regulares das medições em nível de empresa ou produto formam a base para decisões fundamentadas sobre os próximos passos e práticas em vários domínios. Os departamentos parecidos com silos verticais ficam ofuscados. Barreiras são removidas. Comunidades emergem. Autoridade arrefece. Responsabilidade cresce. Estruturas organizacionais e processos ressurgem. Agilidade ocorre.

E lembre-se, a agilidade não pode ser planejada, ditada ou copiada, pois a agilidade é única e não tem um estado final.

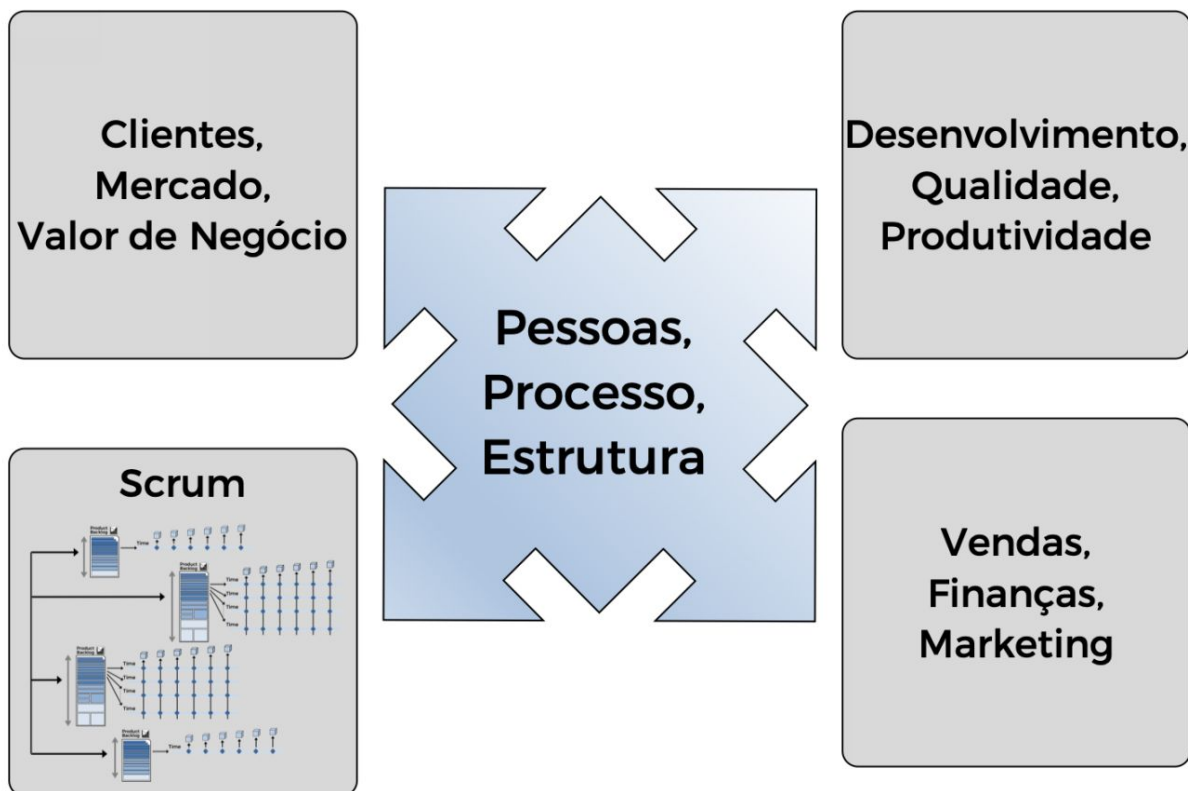


Figura 4.1 Transformação Scrum empresarial

OceanofPDF.com

O futuro estado do Scrum não será mais chamado de 'Scrum'. O que agora chamamos de Scrum terá se tornado a norma e as organizações reemergirão ao seu redor.

OceanofPDF.com

Anexo A: Glossário do Scrum

Daily Scrum: um evento diário, limitado a 15 minutos ou menos, para replanejar o trabalho de desenvolvimento visando atingir o objetivo da Sprint. O evento serve para o Development Team compartilhar o progresso diário, planejar o trabalho das próximas 24 horas e atualizar o Sprint Backlog adequadamente.

Definição de Pronto (English: *Definition of Done*): o conjunto de expectativas de qualidade que um incremento do produto deve conter para ser liberável, por exemplo, estar adequado para uma liberação aos usuários do produto.

Development Team: um grupo de pessoas responsáveis por todo o trabalho de desenvolvimento progressivo para criar um incremento liberável até, no máximo, o final do Sprint.

Duração de uma Sprint (English: *Sprint Length*): a duração de uma Sprint, de um mês ou menos.

Emergência (English: *Emergence*): o processo de surgimento ou proeminência de fatos imprevistos ou a ciência de um fato anteriormente desconhecido, ou o reconhecimento de um fato que se torna visível inesperadamente.

Empirismo (English: *Empiricism*): o tipo de controle do processo pelo qual decisões são baseadas em resultados observados, experiência e experimentação. Empirismo implementa inspeções regulares e adaptações, demandando e criando transparência. Também é referido como ‘controle de processo empírico’.

Gráfico de Burn-down (English: *Burn-down Chart*): um gráfico mostrando a diminuição de trabalho restante em comparação ao tempo.

Gráfico de Burn-up (English: *Burn-up Chart*): um gráfico mostrando o aumento de um parâmetro, por exemplo valor, em comparação ao tempo.

Impedimento (English: *Impediment*): qualquer barreira ou obstáculo que está bloqueando ou desacelerando o Development Team e que não pode ser resolvido pela própria auto-organização do time. Deve ser apontada mais tardar na Daily Scrum, onde o Scrum Master é responsável por removê-lo.

Incremento (English: *Increment*): um candidato de trabalho liberável que adiciona ou muda incrementos criados anteriormente, e - como um todo - forma um produto.

Objetivo da Sprint (English: *Sprint Goal*): uma descrição concisa que expressa o propósito abrangente de uma Sprint.

Padrões de Desenvolvimento (English: *Development Standards*): conjunto de normas e práticas que um Development Team identifica como necessárias para criar um incremento liberável do produto até, no máximo, o final do Sprint.

Produto (English: *Product*): Qualquer bem ou serviço tangível ou intangível consistindo de uma combinação coesa de características e funções que provém valor de ponta a ponta (do começo ao fim) solucionando um problema específico para pessoas que usam o produto (usuários). *Define a abrangência de: Product Owner, Product Backlog e Incremento.*

Product Backlog: uma lista emergente e ordenada de todo o trabalho considerado necessário pelo Product Owner para criar, entregar, suportar e manter um produto.

Product Owner: a pessoa responsável por otimizar o valor que um produto entrega, gerenciando incrementalmente e expressando todas as ideias e expectativas no Product Backlog; ele é o representante único de todos os stakeholders.

Projeção (English: *Forecast*): a antecipação da tendência futura baseada em observações do passado, como a seleção do Product Backlog considerado entregável na Sprint atual ou, em Sprints futuras para Product Backlog futuros.

Refinamento do Product Backlog (English: *Product Backlog Refinement*): uma atividade recorrente durante uma Sprint pela qual o Product Owner e o Development Team adicionam granularidade ao futuro Product Backlog.

Scrum (n): um simples conjunto de ferramentas (*framework*) para a entrega de produtos complexos (1); um simples *framework* para endereçar desafios complexos (2).

Scrum Master: a pessoa responsável pelo fomento de um ambiente Scrum através da orientação, treinamento e facilitação de um ou mais Scrum Teams bem como do seu entorno em entender e empregar o Scrum.

Scrum Team: o conjunto de responsabilidades do Product Owner, Development Team e do Scrum Master.

Sprint: um evento que serve como um contêiner para os outros eventos do Scrum, limitado a 4 semanas ou menos. O evento serve para produzir um montante suficiente de trabalho pronto ao mesmo tempo que garante oportuna inspeção, reflexão e adaptação em nível estratégico e de produto. Os outros eventos do Scrum são: Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective.

Sprint Backlog: um plano emergente de todo o trabalho julgado necessário pelo Development Team para atingir o objetivo da Sprint.

Sprint Planning: um evento marcando o início da Sprint, limitado a 8 horas ou menos. O evento serve para o Scrum Team inspecionar o Product Backlog considerado mais valioso naquele momento bem como para projetar um Sprint Backlog inicial, a fim de atingir o objetivo da Sprint.

Sprint Retrospective: um evento marcando o encerramento de uma Sprint, limitado a 3 horas ou menos. O evento serve para o Scrum Team inspecionar a Sprint que está acabando e estabelecer o modo de trabalho para a próxima Sprint.

Sprint Review: um evento marcando o encerramento de um desenvolvimento realizado em uma Sprint, limitado a 4 horas ou menos. O evento serve para o Scrum Team e os stakeholders inspecionar o incremento, o progresso geral e as mudanças estratégicas a fim de possibilitar que o Product Owner atualize o Product Backlog.

Stakeholder: uma pessoa externa ao Scrum Team com interesse específico ou com conhecimento de algum produto que é necessário para a evolução do produto em questão.

Time-box: um contêiner de tempo com máxima duração, potencialmente uma duração definida. Em Scrum todos os eventos têm uma duração máxima, exceto a própria Sprint que tem uma duração fixada.

Valores do Scrum (English: *Scrum Values*): um conjunto de 5 valores e qualidades fundamentais que sustentam o *framework* Scrum: comprometimento, foco, transparência, respeito e coragem.

Velocidade (English: *Velocity*): popular indicação da quantidade média de Product Backlog convertidos em incremento de produto liberável durante uma Sprint por um específico (ou da composição de um) time. A Velocidade serve primariamente como um auxílio para o Development Team do Scrum Team projetar suas Sprints.

OceanofPDF.com

Anexo B: Referências

Adkins, L. (2010). *Coaching Agile Teams, A Companion for ScrumMasters, Agile Coaches, and Project Managers in Transition*. Addison-Wesley.

Beck, K. (2000). *Extreme Programming Explained – Embrace Change*. Addison-Wesley.

Beck, K., Beedle, M., v. Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (February 2001). *Manifesto for Agile Software Development*. <http://agilemanifesto.org/>

Benefield, G. (2008). *Rolling Out Agile at a Large Enterprise*. HICSS'41 (Hawaii Conferência Internacional em Sistemas de Software).

Cockburn, A. (2002). *Agile Software Development*. Addison-Wesley.

Giudice, D. L. (November 2011). *Global Agile Software Application Development Online Survey*. Estudo Forrester.

Hammond, J., West, D. (October 2009). *Agile Application Lifecycle Management*. Estudo Forrester.

Kotter, J., Rathgeber, H. (2006). *Our Iceberg Is Melting, Changing and Succeeding Under Any Conditions*. MacMillan.

Larman, C. (2004). *Agile & Iterative Development, A Manager's Guide*. Addison-Wesley.

Larman, C., Vodde, B. (2009). *Lean Primer*. <http://www.leanprimer.com>

Moore, G. (1999). *Crossing the Chasm, Marketing and Selling Technology Products to Mainstream Customers* (second edition). Wiley.

Pink, D. (2009). *Drive: The Surprising Truth About What Motivates Us*. Riverhead books.

Schwaber, K. (October 1995). *SCRUM Development Process*.
<http://www.jeffsutherland.org/oopsla/schwaber.html>

Schwaber, K., Beedle, M. (2001). *Agile Software Development with Scrum*. Prentice Hall.

Schwaber, K., Sutherland, J. (November 2017). *The Scrum Guide*.
<http://www.scrumguides.org>.

Standish Group (2002). Keynote sobre o Uso de funcionalidades em um Sistema Típico no Congresso XP2002 por Jim Johnson, presidente do Standish Group. Resultados de um estudo de 2000 projetos em 1000 empresas.

Standish Group (2011). *Chaos Manifesto (The Laws of Chaos and the Chaos 100 Best PM Practices)*. The Standish Group International.

Standish Group (2013). *Chaos Manifesto 2013: Think big, act small*. Standish Group. Sutherland, J. (-) Oopsla '95 - Workshop de Design e Implementação de Objetos de Negócios.

Sutherland, J. (October 2011). *Takeuchi and Nonaka: The Roots of Scrum*.
<https://www.scruminc.com/takeuchi-and-nonaka-roots-of-scrum/>

Taleb, N. N. (2007). *The Black Swan - The Impact of the Highly Improbable*. Random House.

Takeuchi, H., Nonaka, I. (January-February 1986). *The New New Product Development Game*, Harvard Business Review.

Verheyen, G. (December 2011). *The Blending Philosophies of Lean and Agile*. Scrum.org (<https://www.scrum.org/resources/blending-philosophies-lean-and-agile>)

Verheyen, G., Arooni, A. (December 2012). *ING, Capturing Agility via Scrum at a large Dutch bank*. Scrum.org (<https://www.scrum.org/resources/ing-capturing-agility-scrum-large-dutch-bank>)

VersionOne (2011). State of Agile Survey. 6th Annual. VersionOne Inc.

VersionOne (2013). 7th Annual State of Agile Development Survey. VersionOne Inc.

Wiefels, P. (2002). The Chasm Companion. A Fieldbook to Crossing the Chasm and Inside the Tornado. Wiley.

OceanofPDF.com

Sobre o autor



Gunther Verheyen (Gunther.Verheyen@mac.com) é um praticante de Scrum de longa data (2003). Depois de uma carreira como consultor, tornou-se parceiro de Ken Schwaber e diretor da série Professional Scrum na Scrum.org (2013-2016). Gunther se envolve com pessoas e organizações como um zelador independente do Scrum (independent Scrum Caretaker).

Gunther aventurou-se em TI e desenvolvimento de software após graduar-se em 1992. Sua jornada ágil começou com eXtreme Programming (XP) e Scrum em 2003. Anos de dedicação se seguiram, anos empregando Scrum em diversas circunstâncias. A partir de 2010 Gunther se tornou a força inspiradora por trás de algumas transformações corporativas em grande escala. Em 2011 ele se tornou um treinador profissional de Scrum (Professional Scrum Trainer - PST).

Gunther deixou a consultoria em 2013 para fundar a Ullizee-Inc e fez parceria exclusiva com Ken Schwaber, co-criador do Scrum. Ele representou Ken e sua organização Scrum.org na Europa, conduziu a série 'Professional Scrum' e guiou a rede global de treinadores profissionais (PST) Scrum da Scrum.org. Gunther é co-criador do Agility Path, do EBMgt[™] (Gerenciamento Baseado em Evidências de Software) e do framework Nexus para Scrum Profissional Escalado (Scaled Professional Scrum).

Desde 2016, Gunther continua sua jornada como um zelador independente do Scrum; um conector, escritor, palestrante, humanizador. Seus serviços se baseiam em 15 anos de experiência, ideias, crenças e observações do Scrum, expressas em *re-vers-ify* (re-versificar), um ato de pessoas reimaginando seu Scrum para ressurgir sua organização.

Em 2013, Gunther publicou a primeira edição do “Scrum - Um Guia de Bolso”, o qual Ken Schwaber recomendou como ‘a melhor descrição do Scrum atualmente disponível e ‘extraordinariamente competente. Em 2016, a tradução holandesa do livro foi publicada como “Scrum Wegwijzer”. Em 2017, a tradução alemã foi lançada como “Scrum Taschenbuch”. Essa segunda edição é a base para traduções adicionais.

Quando não está viajando pelo Scrum e humanizando o local de trabalho, Gunther vive e trabalha na Antuérpia (Bélgica).

Encontre Gunther no LinkedIn em <https://www.Linkedin.com/in/ullizee>, no Twitter como <https://Twitter.com/ullizee> ou para ler mais de suas reflexões sobre Scrum em seu blog, <https://guntherverheyen.com/>.

[OceanofPDF.com](https://www.oceanofpdf.com)

Sobre os tradutores



Rodrigo Silva Pinto é formado em TI, pós-graduado em Administração de Empresas e Mestre em Gestão de Sistemas Produtivos. Trabalha com métodos ágeis a mais de 10 anos, nas funções de desenvolvedor, Product Owner, Scrum Master, Agile Coach e Agile Chapter Lead. Possui a designação Professional Scrum Trainer pela Scrum.org e é sócio fundador da Agile School. Seu propósito profissional é que por meio da execução, coaching, mentoring e ensino, promova eficácia, eficiência e um ambiente sustentável nas organizações.



Leonardo Bittencourt possui graduação em Ciências da Computação, MBA em Gerenciamento de Projetos e Especialização em Gerenciamento de Produtos. Tendo iniciado sua carreira como Desenvolvedor Java também atuou como Líder Técnico, Gerente de Desenvolvimento, Scrum Master e Agile Coach. Desde 2013 está estabelecido na Irlanda onde tem trabalhado como Consultor Principal Lean/Ágil em organizações mundiais, sediadas na Europa e Estados

Unidos. Através de treinamentos customizados, coaching, mentoria e muita mão na massa, facilita o desenvolvimento de profissionais, times e organizações a fim de criar ambientes de trabalho produtivos bem como soluções relevantes aos usuários.

OceanofPDF.com

“Scrum é livre. As funções, artefatos, eventos e regras do Scrum são imutáveis e, embora a implementação de apenas partes do Scrum seja possível, o resultado não é Scrum. O Scrum só existe **na sua totalidade** e funciona bem como um contêiner para outras técnicas, metodologias e práticas.”

(Ken Schwaber, Jeff Sutherland, O Guia do Scrum)

[1] Frederick Taylor (1856-1915) foi um engenheiro norte-americano que ficou conhecido por sua pesquisa em maneiras de otimizar a produtividade, eficiência e o custo do trabalho. Promoveu a padronização imposta e a adoção imposta de métodos e práticas sistemáticas. O controle pertence exclusivamente à gestão, com os trabalhadores apenas para realizar o trabalho.

[2] Isto refere-se às origens da fabricação Lean da Toyota, onde cada pessoa na linha de produção tem o direito de parar a linha quando os problemas, defeitos ou uma falta de qualidade são detectados.

[3] ‘programação orientada a objetos, sistemas, linguagens e aplicativos’

[4] 31% indicaram não seguir nenhuma metodologia. 21% indicaram que estavam fazendo o desenvolvimento iterativo.

[5] Ver <http://xp123.com/articles/Invest-in-Good-Stories-and-Smart-Tasks/> para mais detalhes.

OceanofPDF.com