

Scrum

Gestão Ágil para Projetos de Sucesso



Casa do
Código

RAFAEL SABBAGH

© Casa do Código

Todos os direitos reservados e protegidos pela Lei nº9.610, de 10/02/1998.

Nenhuma parte deste livro poderá ser reproduzida, nem transmitida, sem autorização prévia por escrito da editora, sejam quais forem os meios: fotográficos, eletrônicos, mecânicos, gravação ou quaisquer outros.

Casa do Código

Livros para o programador

Rua Vergueiro, 3185 - 8º andar

04101-300 – Vila Mariana – São Paulo – SP – Brasil

Dedicatória

Para as pessoas mais importantes da minha vida:

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

Agradecimentos

Agradeço aos meus revisores técnicos, Manoel Pimentel, Marcos Garrido e Rodrigo de Toledo, que prontamente aceitaram me ajudar a garantir um livro de alta qualidade e ainda contribuíram diretamente para o seu conteúdo.

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

Prefácio

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

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

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

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

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

Tipicamente, o maior desafio se dá exatamente em função de sua extensibilidade. Na verdade, muito mais difícil do que usar bem o Scrum é estender seus papéis, cerimônias, artefatos e regras de forma a gerar uma congruência do contexto organizacional com os pilares do Scrum e com os valores do Manifesto Ágil. É comum, por exemplo, que, movidos por um desejo de gerar menos conflitos no processo de adoção, sejamos compelidos a retirar algum dos elementos básicos do *framework*. Outro comportamento comum é usarmos de maneira desenfreada a extensibilidade para fazer o Scrum conviver de forma harmônica com todos os outros papéis, cerimônias, artefatos e regras já existentes na organização. Em ambos os casos, os resultados gerados são muito efêmeros e não promovem um uso sustentável do Scrum.

Adotar Scrum traz uma série de ganhos relacionados, mas também traz uma gama de perdas inerentes. “Perdas”, nesse caso, possui o sentido de “coisas que precisamos abrir mão” para usá-lo. Como um *framework* Ágil, ele promove invariavelmente uma reflexão organizacional acerca de “o que” e “por que” melhorar. Logo, é comum que, com seu uso, seja necessário abrir mão de algum papel, de alguma cerimônia, de algum artefato ou de alguma regra existente na organização.

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

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

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

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

usar Scrum para aquela organização. Apesar da aspiração que Rafael já tinha para escrever um livro, gosto de pensar que foi ali que o mesmo começou a se concretizar.

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

Manoel Pimentel de Medeiros

Como este livro está organizado

Este trabalho está dividido em cinco partes: Introdução ao Scrum, Papéis: o Time de Scrum, Artefatos do Scrum, Eventos do Scrum e Final.

Na Parte I – Introdução ao Scrum, começamos com as principais razões para se escolher e utilizar Scrum. Em seguida, ofereço uma definição formal e sigo explicando o que Scrum é em detalhes, indicando também onde melhor o *framework* pode ser aplicado. Em “Como é o Scrum?”, o leitor pode fazer um passeio por um projeto utilizando Scrum e diversas práticas e artefatos associados, muitos deles opcionais. A Parte I se encerra com uma explicação das origens do Scrum e de quais as suas principais influências.

A Parte II – Papéis: o Time de Scrum trata em detalhes do que é, o que faz e como é cada um dos papéis do Time de Scrum, ou seja, o Time de Desenvolvimento, o Product Owner e o ScrumMaster. Mostro também nessa parte diversas ideias e práticas associadas aos papéis, como resolução de impedimentos, motivação e facilitação, entre outras.

Na Parte III – Artefatos do Scrum, explico o que é e como utilizar o Product Backlog, o Sprint Backlog, a Definição de Pronto e o Incremento do Produto. Ao Product Backlog são ainda associados conceitos como Estimativas Ágeis, Story Points, Planning Poker, Velocidade do Time de Desenvolvimento e User Stories. Acrescento aos artefatos básicos do Scrum as metas de negócios (Meta do Sprint, Meta de Release ou de Roadmap e Visão do Produto), os Gráficos de Acompanhamento do Trabalho (Release Burndown, Release Burnup e Sprint Burndown) e a Definição de Preparado, que podem ser muito úteis na execução do projeto.

A Parte IV – Eventos do Scrum descreve as reuniões de Sprint Planning, de Daily Scrum, de Sprint Review e de Sprint Retrospective, indicando em detalhes boas práticas para sua execução e problemas comuns enfrentados por seus participantes. Adiciono também a reunião de Release Planning, a Release e o Refinamento do Product Backlog. O Sprint também é abordado na parte IV, além de fatores determinantes

para a escolha de sua duração, motivos para seu cancelamento e a sua relação com a Definição de Pronto.

Na Parte V - Final, o leitor tem as palavras finais escritas pelo Marcos Garrido e pelo Rodrigo de Toledo, um glossário dos termos relacionados ao Scrum e a bibliografia utilizada neste livro.

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

Sumário

Introdução ao Scrum	1
1 Por que Scrum?	3
1.1 Introdução	3
1.2 Entregas frequentes de retorno ao investimento dos clientes	5
1.3 Redução dos riscos do projeto	5
1.4 Maior qualidade no produto gerado	6
1.5 Mudanças utilizadas como vantagem competitiva	7
1.6 Visibilidade do progresso do projeto	7
1.7 Redução do desperdício	8
1.7.1 Produzir apenas o que os usuários irão utilizar	8
1.7.2 Planejar apenas com o nível de detalhe possível	9
1.7.3 Utilizar apenas os artefatos necessários e suficientes	11
1.8 Aumento de produtividade	12
1.8.1 Trabalho em equipe e autonomia	13
1.8.2 Facilitação e remoção de impedimentos	13
1.8.3 Melhoria contínua	13
1.8.4 Ritmo sustentável de trabalho	14
1.8.5 Motivação	15
2 O que é Scrum?	17
2.1 Introdução	17
2.2 Scrum é Ágil	18
2.2.1 Os valores Ágeis	20
2.2.2 Os princípios Ágeis	24
2.2.3 Os valores do Scrum	27

2.3	Scrum é um framework simples e leve	29
2.4	Scrum se aplica a produtos complexos em ambientes complexos	30
2.4.1	Sistemas Adaptativos Complexos	30
2.4.2	Onde utilizar Scrum?	31
2.5	Scrum é embasado no empirismo	35
2.6	Scrum é iterativo e incremental	35
3	Como é o Scrum?	39
3.1	Início do projeto	39
3.2	Planejamento do Sprint	42
3.3	Desenvolvimento	44
3.4	Encerramento do Sprint	46
3.5	Entregas	47
3.6	Final do projeto	48
4	De onde veio o Scrum?	49
4.1	Introdução	49
4.2	Um novo jogo	50
4.3	Lean	52
4.3.1	Introdução	52
4.3.2	Os princípios do Lean	53
4.3.3	Lean e Agilidade	53
Papéis: o Time de Scrum		59
5	Time de Desenvolvimento	61
5.1	Quem é o Time de Desenvolvimento?	61
5.2	O que faz o Time de Desenvolvimento?	63
5.2.1	Planeja seu trabalho	63
5.2.2	Realiza o desenvolvimento do produto	63
5.2.3	Interage com o Product Owner durante o Sprint	64
5.2.4	Identifica e informa os impedimentos ao ScrumMaster	65
5.2.5	Obtém feedback sobre o produto	68
5.2.6	Entrega valor com frequência	68

5.3	Como é o Time de Desenvolvimento?	69
5.3.1	Multidisciplinar	69
5.3.2	Auto-organizado	71
5.3.3	Suficientemente pequeno	72
5.3.4	Motivado	73
5.3.5	Orientado à excelência técnica	76
5.3.6	Focado nas metas	77
6	Product Owner	79
6.1	Quem é o Product Owner?	79
6.2	O que faz o Product Owner?	82
6.2.1	Gerencia o produto	82
6.2.2	Gerencia as partes interessadas no projeto	82
6.2.3	Mantém a Visão do Produto	83
6.2.4	Gerencia as Releases	84
6.2.5	Colabora com o Time de Desenvolvimento durante o Sprint	85
6.2.6	Aceita ou rejeita a entrega do Time de Desenvolvimento	86
6.3	Como é o Product Owner?	86
6.3.1	Único	86
6.3.2	Disponível para o trabalho no projeto	87
6.3.3	Representativo para o produto	88
7	ScrumMaster	91
7.1	Quem é o ScrumMaster?	91
7.2	O que faz o ScrumMaster?	92
7.2.1	Facilita o trabalho do Time de Scrum	92
7.2.2	Remove impedimentos	98
7.2.3	Promove mudanças organizacionais necessárias	100
7.2.4	Garante o uso do Scrum	101
7.3	Como é o ScrumMaster?	102
7.3.1	Competente em soft skills	102
7.3.2	Presente	103
7.3.3	Neutro	104
8	Onde está o Gerente de Projetos?	107

Artefatos do Scrum	109
9 Product Backlog	111
9.1 O que é o Product Backlog?	111
9.2 Como é o Product Backlog?	113
9.2.1 Ordenado	113
9.2.2 Planejável	116
9.2.3 Emergente	133
9.2.4 Gradualmente detalhado	133
9.3 User Story	134
9.3.1 O que é a User Story?	134
9.3.2 Como é a User Story?	136
9.3.3 Criando boas User Stories	143
9.3.4 Épicos e Temas	144
10 Sprint Backlog	147
10.1 O que é o Sprint Backlog?	147
10.1.1 O quê?	147
10.1.2 Como?	148
10.2 Como é o Sprint Backlog?	148
11 Definição de Pronto	153
11.1 O que é a Definição de Pronto?	153
11.2 Como é a Definição de Pronto?	155
12 Incremento do Produto	157
13 Metas de negócios	159
13.1 O que são metas de negócios?	159
13.2 Meta de Sprint	162
13.2.1 O que é a Meta de Sprint?	162
13.2.2 Como é a Meta de Sprint?	163
13.3 Meta de Release/Roadmap	164
13.3.1 O que é a Meta de Release/Roadmap?	164
13.3.2 Como é a Meta de Release/Roadmap?	165

13.4	Roadmap do Produto	166
13.4.1	O que é o Roadmap do Produto?	166
13.4.2	Como é o Roadmap do Produto?	166
13.5	Visão de Produto	167
13.5.1	O que é a Visão do Produto?	167
13.5.2	Como é a Visão do Produto?	168
14	Gráficos de Acompanhamento do Trabalho	171
14.1	Gráfico de Release Burndown	172
14.1.1	O que é o Gráfico de Release Burndown?	172
14.1.2	Como é o Gráfico de Release Burndown?	174
14.2	Gráfico de Release Burnup	176
14.2.1	O que é o Gráfico de Release Burnup?	176
14.2.2	Como é o Gráfico de Release Burnup?	177
14.3	Gráfico de Sprint Burndown	178
14.3.1	O que é o Gráfico de Sprint Burndown?	178
14.3.2	Como é o Gráfico de Sprint Burndown?	180
14.3.3	Linha ideal	183
15	Definição de Preparado	187
15.1	O que é a Definição de Preparado?	187
15.2	Como é a Definição de Preparado?	190
Eventos do Scrum	193	
16	Sprint	195
16.1	O que é o Sprint?	195
16.2	Como é o Sprint?	197
16.2.1	Duração	197
16.2.2	Incrementos entregáveis	200
16.3	O Sprint pode ser cancelado?	200

17 Sprint Planning	203
17.1 O que é a Sprint Planning?	203
17.1.1 Duração	204
17.1.2 Saídas	204
17.1.3 Preparação	206
17.2 Como é a Sprint Planning?	207
17.2.1 Sprint Planning 1 e Sprint Planning 2	207
17.2.2 Planejamento Intercalado de Sprint	211
17.2.3 Planejamento Just-In-Time de Sprint	213
18 Daily Scrum	215
18.1 O que é a Daily Scrum?	215
18.2 Como é a Daily Scrum?	216
18.3 O que NÃO deve acontecer na Daily Scrum?	217
18.3.1 Informe de impedimentos ao ScrumMaster	217
18.3.2 Reunião de trabalho	217
18.3.3 Prestação de contas a outros	217
18.3.4 Falta de atenção	218
19 Sprint Review	219
19.1 O que é a Sprint Review?	220
19.2 Como é a Sprint Review?	220
19.2.1 Preparação	220
19.2.2 Gestão de expectativas	220
19.2.3 Quem participa	221
19.2.4 Demonstração	221
19.2.5 O que é demonstrado	222
19.2.6 Resultados	222
20 Sprint Retrospective	225
20.1 O que é a Sprint Retrospective?	225
20.1.1 Lições aprendidas em times tradicionais	226
20.1.2 Melhoria incremental contínua em times Ágeis	226
20.1.3 A retrospectiva do Sprint	226
20.2 Como é a Sprint Retrospective?	227

20.2.1	Regularidade, frequência e duração	227
20.2.2	Participação do ScrumMaster	227
20.2.3	Participação do Product Owner	228
20.2.4	Dinâmica básica de uma retrospectiva	229
20.2.5	Algumas variações na retrospectiva	231
20.3	O que NÃO deve acontecer na Sprint Retrospective?	234
20.3.1	Busca de culpados	234
20.3.2	Insegurança e medo de exposição	235
20.3.3	Conflitos da diversidade	236
20.3.4	Pensamento grupal	236
21	Release	239
21.1	O que é a Release?	239
21.2	Como é a Release?	240
21.2.1	Quanto à frequência	241
21.2.2	Quanto a quem a recebe	242
22	Release Planning	245
22.1	O que é a Release Planning?	245
22.2	Plano da Release	246
22.3	Como é a Release Planning?	246
22.3.1	Agendamento e duração	246
22.3.2	A reunião	247
22.3.3	Granularidade dos itens	248
22.3.4	Escopo da Release	249
23	Refinamento do Product Backlog	251
23.1	O que é o Refinamento do Product Backlog?	252
23.2	Como é o Refinamento do Product Backlog?	252
23.2.1	Participantes	252
23.2.2	Quando ocorre	253

Final	255
24 Além do Scrum	257
24.1 Introdução	257
24.2 Facilitação	259
24.3 Management 3.0	260
24.4 Lean Kanban	262
24.5 Técnicas para Product Owners	263
24.6 Lean Startup	264
24.7 Escalando Scrum	265
24.8 Testes automatizados e entrega contínua	266
24.9 Conclusão	268
25 Apêndice - Glossário	271
26 Apêndice - Bibliografia	277

Parte I

Introdução ao Scrum

CAPÍTULO 1

Por que Scrum?

1.1 INTRODUÇÃO

Scrum existe desde o início dos anos 1990, mas foi só na década seguinte que se tornou popular. Scrum ganhou o mundo, desbancou métodos tradicionais e se tornou a forma mais comum de se trabalhar em projetos de desenvolvimento de *software*. Scrum foi apontado como o método de trabalho utilizado por dois em cada três participantes de uma pesquisa realizada em 2011 (VersionOne, 2011).

A adoção mundial de Scrum não significa que todos os problemas estão resolvidos. Longe disso, Scrum é apenas uma ferramenta que pode trazer diversos benefícios em comparação a outras formas de se conduzir projetos, mas somente se bem utilizada. Scrum pode permitir reduzir os riscos de insucesso, entregar valor mais rápido e desde cedo, e lidar com as inevitáveis mudanças de escopo, transformando-as em vantagem competitiva. Seu uso pode também aumentar a qualidade do produto entregue e melhorar a produtividade das equipes.

Scrum vem sendo adotado com sucesso por organizações de diversos tamanhos e tipos. De multinacionais a *startups*, de famosas a desconhecidas. Seu uso não se

limita a projetos de desenvolvimento de *software*, embora tenha sido concebido com essa finalidade. Scrum é hoje também utilizado em diferentes mercados, que incluem empresas de *marketing* e de desenvolvimento de *hardware*, por exemplo.

Scrum é aplicado em projetos com características igualmente variadas. Em projetos críticos de centenas de milhares de dólares e em projetos internos simples. Em projetos para produção de *softwares* comerciais, de *sites* da Internet, de *softwares* embarcados, de aplicativos para dispositivos móveis, de *softwares* financeiros e de jogos.

Ao aprender Scrum, você passará por termos como facilitação, trabalho em equipe, auto-organização, metas de negócios, motivação, relacionamento com os clientes, entre tantos outros. Scrum utiliza-se de poucos conceitos novos, e essa é uma de suas grandes qualidades: juntar práticas de mercado já conhecidas e consagradas de uma forma organizada e que funciona.

Mas por que Scrum é uma boa escolha para o projeto de desenvolvimento de um produto, para a organização que está desenvolvendo esse produto e para os seus clientes? Uma gama de respostas para essa pergunta é dada ao longo deste livro. Neste capítulo, sumarizamos algumas delas.

É importante lembrar que não existe uma solução única para todos os problemas. Scrum é um *framework* simples e pequeno e, assim, funciona bem em cada contexto se for utilizado em conjunto com outras técnicas e práticas a serem experimentadas e adaptadas.

Os benefícios no uso do Scrum incluem:

- entregas frequentes de retorno ao investimento dos clientes;
- redução dos riscos do projeto;
- maior qualidade no produto gerado;
- mudanças utilizadas como vantagem competitiva;
- visibilidade do progresso do projeto;
- redução do desperdício;
- aumento de produtividade.

1.2 ENTREGAS FREQUENTES DE RETORNO AO INVESTIMENTO DOS CLIENTES

Scrum possibilita que se entreguem, desde cedo no projeto e frequentemente, partes do produto funcionando. Cada uma dessas entregas proporciona retorno ao investimento realizado pelos clientes do projeto, além de possibilitar o seu *feedback* rápido sobre o produto para que se realizem as mudanças ou adições necessárias.

Em projetos que utilizam métodos tradicionais, apenas uma ou poucas entregas são realizadas ao final do projeto ou ao final de uma grande etapa. Com Scrum, partes prontas do produto são geradas em ciclos curtos de desenvolvimento, que ocorrem um atrás do outro. As partes entregues são as mais necessárias para seus clientes e usuários no momento da entrega e, por essa razão, serão utilizadas imediatamente. Uma vez que são utilizadas, representam retorno ao investimento realizado.

Da mesma forma, Scrum possibilita um curtíssimo prazo para a introdução do produto no mercado, já que cada uma dessas entregas, somada às anteriores, representa um produto funcional.

1.3 REDUÇÃO DOS RISCOS DO PROJETO

Scrum visa à redução dos riscos de negócios do projeto pela colaboração com os clientes e demais partes interessadas durante todo o seu decorrer. Os riscos também são reduzidos com a produção em ciclos curtos e entregas frequentes de partes prontas do produto, partindo-se das mais importantes em direção às menos importantes.

O produto é construído incrementalmente a partir do *feedback* frequente dos clientes do projeto e demais partes interessadas. Por melhores que sejam as técnicas de levantamento de requisitos utilizadas, não há como se definir de antemão o produto em detalhes. Assim, uma longa fase de detalhamento no início do projeto gera um grande desperdício. Os clientes aprenderão sobre o que está sendo desenvolvido ao longo do trabalho, à medida que veem ou utilizam as partes do produto demonstradas ou entregues. Assim, busca-se, a partir de seu *feedback*, desenvolver e entregar o produto certo. Da mesma forma, quando decisões equivocadas são tomadas, o *feedback* dos clientes e demais partes interessadas rapidamente gera visibilidade sobre o problema e permite a correção do rumo.

Uma vez que as partes mais importantes do produto são produzidas primeiro, raramente é necessário se atrasar uma entrega. Ao se chegar à data pré-estabelecida, as chances são de que valor suficiente já terá sido produzido para se atingir os objetivos

de negócios da entrega, uma vez que restarão apenas as partes menos importantes do que foi inicialmente planejado. Da mesma forma, ao final do prazo estabelecido para todo o projeto, espera-se que a visão inicialmente definida seja atingida por se ter realizado a entrega das funcionalidades mais importantes.

O time que desenvolve o produto é multifuncional, ou seja, possui os conhecimentos necessários para gerar o produto. Esse time produz partes do produto prontas em cada ciclo de desenvolvimento, que serão imediatamente validadas pelo *feedback* dos clientes do projeto e demais partes interessadas. Ao se eliminar ou minimizar dependências externas e reduzir a espera por validação, reduzem-se os riscos de não se ter a entrega pronta e de que essa entrega não represente valor.

1.4 MAIOR QUALIDADE NO PRODUTO GERADO

A parte do produto gerada em cada ciclo de desenvolvimento deve possuir a qualidade necessária para poder ser entregue para os clientes do projeto.

O time que trabalha com Scrum é inteiramente responsável por esse trabalho e, assim, possui todas as habilidades e conhecimentos necessários para fazê-lo. Ele realiza, portanto, dentro do ciclo de desenvolvimento, qualquer validação que seja necessária para garantir a qualidade, ou seja, que a parte do produto esteja funcionando como se espera. Essa validação inclui, entre outros, quaisquer tipos de testes – técnicos e de negócios – que se façam necessários, a integração com o resto do produto já gerado e integrações com outros produtos caso existam dependências externas (no caso de desenvolvimento de *software*, ferramentas de automação para integração contínua auxiliam nessa tarefa). Não existem, a princípio, times externos responsáveis por avaliar ou garantir a qualidade do produto gerado, já que esse trabalho é de inteira responsabilidade do próprio time que desenvolve o produto.

Uma validação de negócios da parte do produto gerada ocorre ao final de cada ciclo de desenvolvimento em uma reunião especialmente reservada para esse propósito, chamada de Sprint Review. A partir de uma demonstração do que foi produzido pelo time, clientes do projeto e demais partes interessadas fornecem seu *feedback*. Um *feedback* ainda mais profundo e valioso é obtido dos usuários do produto a partir da utilização das partes do produto já entregues, o que é potencializado pelas entregas frequentes. Em ambos os casos, o produto será modificado, a partir do *feedback* obtido, para melhor atender aos clientes e usuários do projeto.

Ao se antecipar as validações do que é produzido, há muito mais chances dos problemas ou modificações necessárias serem detectadas cedo. Quando essas vali-

dações são postergadas, os problemas podem se acumular e gerar atrasos no projeto. As consequências podem ser drásticas: não haverá tempo para se realizar todas as validações ou todas as mudanças necessárias, e partes do produto serão entregues sem qualidade.

1.5 MUDANÇAS UTILIZADAS COMO VANTAGEM COMPETITIVA

Em um projeto com Scrum, as mudanças são acolhidas como oportunidades e não como acontecimentos indesejáveis. Elas são parte do ambiente dinâmico em que os projetos estão inseridos. As mudanças também compreendem a própria descoberta dos detalhes do produto, que é um processo incremental que ocorre durante todo o projeto.

As necessidades de negócios e consequentes especificações e planos não são definidos detalhadamente no princípio do projeto. Ao contrário, esses detalhes são descobertos e reavaliados ao longo do desenvolvimento, a partir do *feedback* frequente dos seus clientes e demais partes interessadas. Dessa forma, o produto é incrementalmente construído e modificado, e assim a mudança é utilizada para garantir que o produto entregue realmente signifique valor para esses clientes.

1.6 VISIBILIDADE DO PROGRESSO DO PROJETO

Diversas práticas e artefatos do Scrum ou associados visam garantir a visibilidade ou transparência do progresso do projeto para seus participantes e envolvidos, que incluem tanto os clientes e demais partes interessadas quanto o próprio time.

Os clientes e demais partes interessadas veem o que foi desenvolvido no mínimo ao final de cada ciclo curto de desenvolvimento, o que lhes dá um senso de progresso do projeto, e recebem entregas de partes do produto funcionando com frequência. Em um projeto com Scrum, os clientes estão muito mais próximos e colaboram com o time na evolução do produto.

Diversas ferramentas são utilizadas para garantir que o time tenha grande visibilidade de seu próprio progresso. O *feedback* que o time obtém nas reuniões de Sprint Review diretamente dos clientes e demais partes interessadas, realizadas ao final de cada ciclo, permite descobrir desde cedo se está na direção certa e o quanto eficiente seu trabalho está sendo. A Daily Scrum é uma reunião diária realizada pelo time que desenvolve o produto exatamente para criar visibilidade interna, na qual

cada membro do time explica para seus colegas o que fez desde a última reunião, o que pretende fazer até a próxima e o que está atrapalhando o seu trabalho. Além disso, há outras ferramentas de visibilidade que podem ser utilizadas, como o Quadro de Tarefas e os Gráficos de Acompanhamento do Trabalho, ambos explicados posteriormente neste livro.

As reuniões de Sprint Retrospective, também realizadas ao final de cada ciclo, permitem criar-se visibilidade sobre melhorias que podem ser realizadas em como o trabalho está sendo realizado, de forma que o time possa inspecionar e adaptar suas práticas para se tornar mais eficiente.

1.7 REDUÇÃO DO DESPERDÍCIO

O desperdício é reduzido e evitado por meio da busca da simplicidade. A ideia central é que o time que usa Scrum produza e utilize apenas o que é necessário e suficiente.

Identificamos como as regras e práticas do Scrum ajudam a evitar alguns tipos de desperdícios:

- produzir apenas o que os usuários irão utilizar;
- planejar apenas com o nível de detalhes possível;
- utilizar apenas os artefatos necessários e suficientes.

1.7.1 Produzir apenas o que os usuários irão utilizar

Uma pesquisa da Standish Group divulgada em 2002, época em que Scrum e Agilidade eram pouco utilizados, mostra que mais da metade das funcionalidades solicitadas pelos clientes para o projeto e produzidas nunca ou raramente eram utilizadas, caracterizando um enorme desperdício (veja a figura 1.1).

Esse desperdício ocorre quando se busca definir o produto com um alto nível de detalhes no princípio do seu desenvolvimento. Mesmo que se despendam meses em reuniões para levantamento e análises de requisitos, ao se exigir dos clientes que imaginem, formulem e decidam os detalhes de um trabalho a ser realizado durante meses adiante, um grande desperdício será inevitavelmente gerado. Nesse cenário, os clientes adicionam aos requisitos tudo o que puderem imaginar, pois somente depois desses meses de desenvolvimento possivelmente terão uma nova oportunidade de intervir no produto.

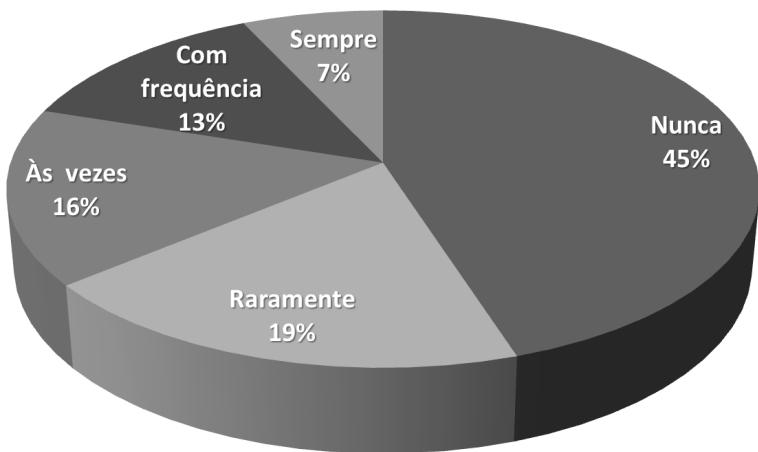


Figura 1.1: Percentual de uso das funcionalidades solicitadas pelo cliente (The Standish Group, 2002)

Com Scrum, os detalhes do produto emergem ao longo de seu desenvolvimento. Mesmo assim, existe um desperdício, uma vez que partes já prontas do produto são modificadas a partir do *feedback* dos seus clientes e demais partes interessadas. Esse desperdício, no entanto, não chega nem perto daquele gerado quando se tenta prever os detalhes antecipadamente, no início do projeto.

Além disso, o trabalho de desenvolvimento do produto com Scrum é direcionado pelas questões de negócios, uma vez que é realizado sobre o que é priorizado a partir das necessidades dos clientes. Baseados em contatos frequentes com os clientes e demais partes interessadas no projeto e seu *feedback*, adicionam-se, detalham-se e ordenam-se funcionalidades que se traduzem em valor para os usuários do projeto e, assim, espera-se que sejam utilizados.

Outro princípio importante é que, ao se produzir a solução mais simples que funcione, evita-se o desperdício. Uma solução complexa e custosa, por mais bela que seja, não é uma boa opção se existir uma solução mais simples que funcione.

1.7.2 Planejar apenas com o nível de detalhe possível

Métodos tradicionais buscam planejar, no início do projeto, todo o trabalho a ser realizado no desenvolvimento do produto (ou, ao menos, da próxima Release) com um alto nível de detalhes. Os planos gerados muitas vezes descrevem cada uma das

funcionalidades do produto, quem as produzirão e quando serão produzidas. O resultado é que esses planos, em geral, não traduzem a realidade futura.

Scrum não possui essa fase inicial de planejamento detalhado, mas isso não significa que o projeto não possa ser descrito desde o seu início até o fim. O princípio básico é, no entanto, que se planeja apenas com o nível de detalhes que é possível de se ter.

Em termos de planejamento para um projeto, pode-se ter, por exemplo, uma visão de produto que traduz seus objetivos e um plano em alto nível que prevê as principais metas de negócios a serem alcançadas para se chegar a essa visão e aproximadamente quando isso acontecerá.

Para o trabalho a ser realizado até a próxima entrega, pode-se criar um plano, no qual serão definidos uma data para a entrega, o trabalho provável a ser realizado e o próximo objetivo ou meta de negócios a ser alcançado a partir desse trabalho, que é um passo para se chegar na visão do produto. Para o ciclo de desenvolvimento atual, o time planeja, com detalhes, que trabalho irá realizar nesse ciclo e como irá realizá-lo, visando alcançar uma meta de negócios que é um passo para se chegar à meta estabelecida para a entrega.

Qualquer um desses planos é modificado sempre que necessário, de forma a refletir a realidade que se enxerga em cada momento.

Analogia do horizonte

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

Ao olhar para o que está mais próximo, você consegue descrever o que vê com um excelente nível de detalhes. O que está um pouco mais distante, você ainda pode descrever com um bom nível de detalhes, ainda que menor. Mas à medida que olha para mais longe, os detalhes que pode utilizar em sua descrição vão gradualmente diminuindo. Se você então tentar descrever o que vê com um alto nível de detalhes, ao caminhar em direção ao horizonte, verá que sua descrição não corresponde inteiramente à realidade, e que você deverá modificar essa descrição. Na verdade, para quanto mais distante você estiver olhando, mais incorreta estará essa sua descrição detalhada, e maior será o desperdício gerado.

Um planejamento tradicional geralmente descreve em detalhes o que será feito em todo o projeto ou, ao menos, todo o trabalho até a próxima entrega. É muito comum ver esses planos descritos em gráficos de Gantt, que mostram tarefas detalhadas e alocação de “recursos” no tempo. Essa prática pode ser comparada a se

descrever detalhadamente todo o caminho, desde o que está mais próximo de você até o que está mais próximo da linha do horizonte. O resultado dessa prática é um plano de altíssima precisão, porém com baixíssima chance de acerto, levando assim à geração de desperdício.

Em um planejamento Ágil, somente se planeja com o nível adequado de detalhes. Para planejar-se um trabalho a ser realizado, por exemplo, até o próximo dia, pode-se utilizar um nível de detalhes bastante alto. Para as próximas duas semanas – um ciclo de desenvolvimento, por exemplo – pode-se utilizar um nível de detalhes ainda razoavelmente alto, porém mais baixo do que para apenas um dia. Ao se planejar uma entrega que acontecerá daqui a dois meses ou ao se planejar o ano inteiro de projeto, a quantidade de detalhes diminui quanto mais longe se olha no tempo, de forma que pouquíssimos detalhes podem ser utilizados para planejar o que está mais distante. À medida que o projeto caminha no tempo, esse plano deve ser refinado e mais detalhes devem ser gradualmente obtidos.

A lista de necessidades de negócios do Scrum, chamada de Product Backlog, reflete esse planejamento Ágil. O alto do Product Backlog possui itens com uma granularidade maior, ou seja, itens menores e que representam mais detalhes. Ao se descer no Product Backlog, os itens vão ficando cada vez maiores e com menos detalhes. Mais detalhes do que o necessário e suficiente significaria geração de desperdício.

1.7.3 Utilizar apenas os artefatos necessários e suficientes

Ferramentas são úteis até o ponto em que começam a gerar desperdício. Aquelas utilizadas no suporte à gestão do projeto e ao desenvolvimento do produto devem cumprir seu papel sem adicionar burocracia ao trabalho. Em geral, as ferramentas mais simples que funcionam geram os melhores resultados.

Ferramentas utilizadas pelo time para esboçar, planejar e monitorar seu trabalho de desenvolvimento do produto são mais bem representadas em meios físicos do que em meios virtuais. Quadros brancos na parede são bons exemplos de meios para essas ferramentas. Eles, enquanto irradiadores de informação, trazem uma visibilidade que não é possível, por exemplo, ao se utilizarem ferramentas de *software*. Essas últimas, quando mal utilizadas, terminam por impor mais empecilhos do que facilitar o trabalho, gerando um desperdício bastante custoso para o projeto.

Documentos que refletem planos, esquemas e especificações também podem ser úteis no suporte à gestão do projeto e ao desenvolvimento. O time, no entanto, evita

o desperdício ao se concentrar em criar e manter apenas aquilo que será, de fato, utilizado.

DOCUMENTAÇÃO SUFICIENTE E NECESSÁRIA

Imagine um projeto em que, por burocracia da organização ou por tentativa de se seguir alguma metodologia específica, é exigida a geração de uma grande quantidade de documentos de especificações. Em muitos casos, esse trabalho é tão grande quanto ou até maior do que o trabalho de se gerar o produto e, na realidade, apenas uma pequena parte dessa documentação é realmente necessária. Assim, um grande desperdício é gerado.

Nesse mesmo cenário, imagine que, em algum momento no decorrer do projeto, você necessita de consultar um item específico dessa documentação. As chances são de que, por haver tanto a ser atualizado, esse item não está mais correspondendo à realidade e, assim, não é mais útil como está. Portanto, um desperdício ainda maior se caracteriza.

Produzir documentação além do suficiente e necessário é desperdício. A documentação em excesso desloca o foco daquela que é realmente necessária, gerando um desperdício ainda maior.

1.8 AUMENTO DE PRODUTIVIDADE

Diversos fatores potencializam a produtividade de times que utilizam Scrum. Entre esses fatores, podemos citar:

- o trabalho em equipe e a autonomia do time na realização desse trabalho;
- a existência de facilitação e de remoção de impedimentos;
- a melhoria continua dos processos de trabalho;
- um ritmo sustentável de trabalho;
- a maior motivação do time.

1.8.1 Trabalho em equipe e autonomia

O time de um projeto com Scrum é pequeno, de forma que seus membros possam interagir e se comunicar durante todo o dia para realizar o trabalho. Trabalhar em equipe é considerado essencial para o sucesso do projeto.

O time gera cada parte do produto em direção a uma meta de negócios específica, que os guia e dá propósito a seu trabalho. O time tem autonomia para decidir o quanto é possível de se fazer em um ciclo e como irá fazê-lo e, assim, ajusta a meta de forma que possa se comprometer com ela. Os membros do time, então, tornam-se igualmente responsáveis por atingir essa meta, o que estimula a cooperação entre eles em busca desse objetivo. Assim, a responsabilidade sobre o trabalho não é individual, e sim coletiva.

Essa autonomia ajuda o time a criar um senso de propriedade sobre seu trabalho, de forma que os próprios membros do time estimulam e ajudam seus colegas a fazerem o seu melhor para alcançarem a meta. Esse comportamento naturalmente leva o time a ser mais produtivo.

1.8.2 Facilitação e remoção de impedimentos

O time conta com a ajuda de um facilitador para se tornar mais produtivo.

Esse facilitador, chamado de ScrumMaster, ensina Scrum ao time, ajuda-o a ganhar autonomia e aprender a se auto-organizar, facilita suas reuniões e remove os impedimentos que ameaçam as metas a serem alcançadas.

1.8.3 Melhoria contínua

Um time que utiliza Scrum está sempre em busca de melhorar a forma como realiza seu trabalho, para se tornar mais eficiente. A reunião de Sprint Retrospective, realizada ao final de cada ciclo de desenvolvimento, busca garantir essa prática. Nessa reunião, pontos a melhorar são levantados e discutidos e planos de ação para colocar as melhorias em prática são definidos.

Na busca por melhorias em seu dia a dia de trabalho, o time é estimulado a experimentar, inspecionar e adaptar-se de acordo. No entanto, tão importante quanto encontrar técnicas ou práticas que melhorem seu trabalho é descobrir cedo as que não funcionam adequadamente para, então, modificá-las ou descartá-las. É melhor falhar cedo para então aprender com a falha ao invés de postergar a validação de uma prática ou técnica em uso que pode não estar funcionando.

A melhoria contínua promovida pelo time o leva a um aumento progressivo da sua eficiência na realização do trabalho, o que por sua vez o leva a um aumento progressivo na sua produtividade.

1.8.4 Ritmo sustentável de trabalho

As práticas de horas-extras e de aceleração forçada do ritmo de trabalho, embora muito utilizadas para se cumprir um trabalho no prazo determinado, geram estresse no time que desenvolve o produto e não tardam em derrubar sua produtividade e a qualidade do produto gerado. Times que utilizam Scrum buscam conseguir o efeito contrário com a obtenção de um ritmo constante e sustentável no trabalho de criação do produto.

Cada ciclo de desenvolvimento possui uma duração fixa. De forma similar, todas as reuniões prescritas pelo Scrum possuem uma duração máxima dentro da qual devem ocorrer. São os chamados *timeboxes* que, quando respeitados com rigor, auxiliam o time a alcançar esse ritmo constante e sustentável de trabalho.

Para um ciclo de desenvolvimento do Scrum, por exemplo, deve-se escolher uma duração fixa entre uma e quatro semanas, durante a qual o time trabalha para atingir uma meta de negócios. Nenhuma pressão deve ser exercida sobre o time para que se comprometa com mais trabalho do que acredita que pode realizar nesse *timebox*. Ao trabalhar sempre em ciclos de desenvolvimento com a mesma duração e ter a autoridade para definir quanto trabalho acredita que realizará, o time naturalmente tende a atingir um ritmo constante e sustentável.

A reunião de Sprint Planning é outro exemplo. Essa reunião tem como objetivo criar um plano para o Sprint e não deve durar mais que oito horas para ciclos com duração de quatro semanas. As outras reuniões do Scrum também têm duração máxima definida.

Mesmo quando o time, perto do final do período estabelecido, tiver certeza que não cumprirá com o objetivo do ciclo ou da reunião, seu *timebox* não será estendido. Da mesma forma, disfunções como horas extras e a aceleração forçada do ritmo de trabalho devem ser evitadas. Ao contrário, é mais importante o time chegar ao final do *timebox* e poder falhar. Essa falha, no entanto, deve gerar um aprendizado que ajude o time a não repeti-la novamente no futuro, o que reforçará a criação de um ritmo sustentável e poderá levar a um aumento de produtividade.

1.8.5 Motivação

Um aumento na motivação dos membros do time é, ao mesmo tempo, consequência de todos os fatores descritos anteriormente e causador de um aumento na produtividade.

A esses fatores que aumentam a motivação soma-se o time trabalhar em um ambiente que apoie seu trabalho, o que inclui a disponibilidade de todas as ferramentas necessárias para realizar esse trabalho e a confiança da organização e de sua gerência na capacidade do time em tomar as decisões que lhe cabem.

CAPÍTULO 2

O que é Scrum?

2.1 INTRODUÇÃO

O capítulo anterior tratou dos benefícios de se utilizar Scrum, ou seja, por que ele pode ajudar a se obterem projetos de sucesso. Mas então o que é Scrum, afinal? Buscamos criar uma definição resumida, que pode ser lida a seguir:

Scrum é um *framework* Ágil, simples e leve, utilizado para a gestão do desenvolvimento de produtos complexos imersos em ambientes complexos. Scrum é embasado no empirismo e utiliza uma abordagem iterativa e incremental para entregar valor com frequência e, assim, reduzir os riscos do projeto.

Vamos entender em seguida o que cada um desses termos significa.

2.2 SCRUM É ÁGIL

Uma definição comum para “ágil” poderia ser: “que se movimenta com facilidade; ligeiro, leve”.

O nome “Ágil” (ou “Agilidade”) foi escolhido para representar um movimento que surgiu em meados dos anos 90 em resposta aos pesados métodos de gerenciamento de desenvolvimento de *software* que predominavam na época, que aqui chamamos de “métodos tradicionais”.

O método tradicional mais conhecido para o desenvolvimento de *software* é o modelo em cascata, ou *waterfall*. Esse modelo foi inicialmente descrito por Royce em 1970 e se caracteriza por uma sequência de fases de desenvolvimento, em que cada fase somente se inicia quando a anterior se encerra, e a saída de uma fase é a entrada da fase seguinte, como mostrado na figura 2.1. Royce, no entanto, criticava o modelo em seu artigo, afirmando que, para o desenvolvimento de *software*, seu uso era arriscado.

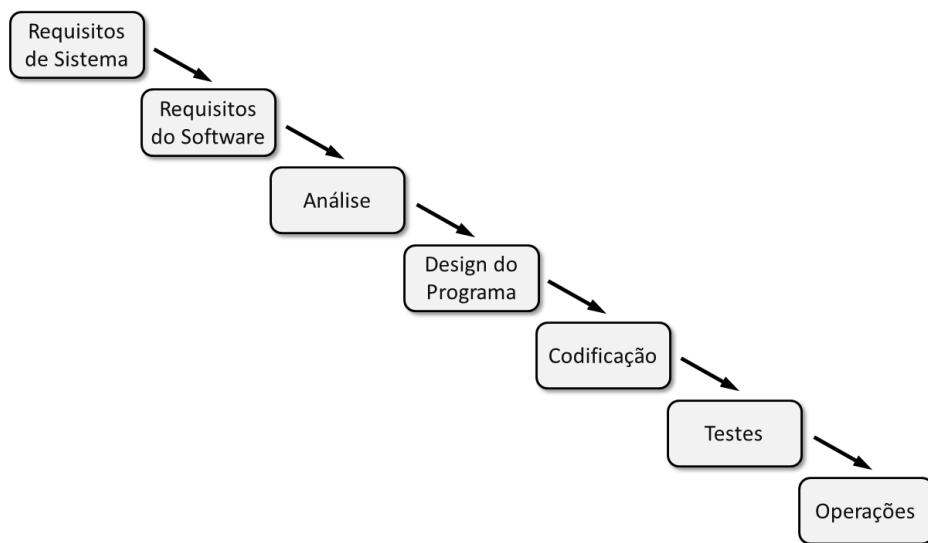


Figura 2.1: O modelo em cascata ou *waterfall*, conforme definido por Royce (1970)

Os métodos tradicionais são fortemente prescritivos e, de forma geral, se caracterizam pelo foco em planos detalhados definidos no princípio do projeto, como custo, escopo e um cronograma detalhado, em microgerenciamento, no poder cen-

tralizado, em processos cada vez mais complicados e em extensa documentação. Mudanças são fortemente indesejadas. Acreditava-se que, pelo uso desses métodos, seria possível tratar o desenvolvimento de *software* como um processo previsível, no que, quanto mais falhavam, mais pesados e complexos esses métodos se tornavam.

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

Nesse encontro histórico, não havia entre os participantes a intenção de unificar suas formas de trabalhar. A expectativa de se chegar a qualquer tipo de consenso era limitada e variada. No entanto, apesar das diferentes práticas defendidas, os participantes encontraram um conjunto de valores em comum e, ao final de três dias, estabeleceram o termo “Ágil” para representar o novo movimento. Neste livro, escrevo “Ágil” ou “Agilidade” com “A” maiúsculo para identificar tudo o que está relacionado ao movimento Ágil.

O Manifesto Ágil, como ficou conhecido, pode ser encontrado em <http://agilemanifesto.org> em diversas línguas, inclusive em português do Brasil (eu mesmo participei desse esforço de tradução). Ao procurar na Internet pelos nomes dos signatários originais do Manifesto e por suas ideias, pode-se ver o quanto representativa a grande maioria deles ainda é.

Discussão: a analogia errada

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

Embora tenham evoluído ao longo dos tempos, projetos de construção existem há eras na história da humanidade e, em linhas gerais, sua forma de execução se manteve a mesma: uma longa fase de definições e especificações no início que tem como saída um plano, seguida de sua fase de execução. Parece natural para o ser humano comparar outros tipos de trabalho com um que lhe seja tão familiar.

Os métodos tradicionais de desenvolvimento de *software* buscaram algo similar com o modelo em cascata e suas fases sequenciais de levantamento e análise de re-

quisitos, especificação, desenvolvimento e testes. Ainda hoje é comum utilizarem-se as expressões “engenharia ou engenheiro de *software*”, “arquitetura ou arquiteto de *software*” e até mesmo “construção de *software*”, todas provindas da analogia com a construção civil.

O livro “*Wicked Problems, Righteous Solutions*” (DeGrace & Stall, 1990 apud. Sutherland, 2004) já descrevia em 1990 as razões por que métodos tradicionais de desenvolvimento de *software* não funcionam, a partir de suas prerrogativas básicas:

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

Sabemos, portanto, que comparar projetos de desenvolvimento de *software* com projetos construção civil não faz sentido. A analogia simplesmente não funciona.

2.2.1 Os valores Ágeis

O Manifesto Ágil reconhece que a utilização de processos, ferramentas, documentação, contratos e planos pode ser importante para o sucesso do projeto, mas são ainda mais importantes os chamados valores Ágeis: os indivíduos e interações entre eles, *software* (ou produto) em funcionamento, colaboração com o cliente e responder a mudanças.

MANIFESTO PARA DESENVOLVIMENTO ÁGIL DE SOFTWARE

Estamos descobrindo maneiras melhores de desenvolver *software* fazendo-o nós mesmos e ajudando outros a fazê-lo. Por meio deste trabalho, passamos a valorizar:

- **Indivíduos e interações** mais que processos e ferramentas
- **Software em funcionamento** mais que documentação abrangente
- **Colaboração com o cliente** mais que negociação de contratos
- **Responder a mudanças** mais que seguir um plano

Ou seja, mesmo havendo valor aos itens à direita, valorizamos mais os itens à esquerda.

Apresentaremos a seguir a visão de dois signatários originais do Manifesto, Alistair Cockburn e Jim Highsmith, e de um autor relevante, Craig Larman, acerca do significado de cada um desses valores.

Indivíduos e interações

Para Jim Highsmith, valorizar indivíduos e interações mais que processos e ferramentas leva em conta que, em última instância, quem gera produtos e serviços são os indivíduos, que possuem características únicas individualmente e em equipe, como talento e habilidade (Highsmith, 2004). Alistair Cockburn afirma que as pessoas na equipe são mais importantes do que seus papéis em diagramas de processos. Assim, embora descrições de processos possam ser necessárias para se começar o trabalho, as pessoas envolvidas nos processos não podem ser trocadas como peças (Cockburn, 2007). Na mesma linha, Craig Larman lembra que a programação de computadores é uma atividade humana e, assim, depende de questões humanas para seu sucesso. O autor cita como exemplo a vida social e familiar dos membros da equipe que são afetadas, por exemplo, pelo excesso de trabalho (Larman, 2003). Highsmith complementa afirmando que são críticas para o sucesso dos projetos as habilidades, personalidades e peculiaridades dos indivíduos, e eles são muitas vezes desorganizados e difíceis de entender, ao mesmo tempo em que são inovadores, criativos, exuberantes e apaixonados (Highsmith, 2002).

Cockburn afirma ainda que a qualidade da interação entre os membros do time é importante para a solução de problemas no desenvolvimento do projeto (Cockburn, 2007). Larman, sobre esse tema, destaca que o uso da comunicação e do *feedback* é uma diretiva essencial para a prática Ágil, especialmente a partir da conversação face a face (Larman, 2003). Highsmith lembra a importância do trabalho em equipe, afirmando que habilidades individuais e trabalho em equipe são inseparáveis em projetos de desenvolvimento de *software* (Highsmith, 2002).

Outro ponto levantado por Larman é que as ferramentas utilizadas para auxiliar o desenvolvimento de *software* devem ser as mais simples possíveis que funcionem (Larman, 2003). Projetos que utilizaram metodologias pesadas em termos de processos e ferramentas, segundo Highsmith, obtiveram sucesso fundamentalmente devido às pessoas envolvidas naqueles projetos (Highsmith, 2002). O autor afirma ainda que processos podem guiar e apoiar o desenvolvimento de *software* e ferramentas podem melhorar sua eficiência, mas é com a capacidade e conhecimento dos indivíduos que se pode contar para tomar decisões críticas para o desenvolvimento do projeto. Bons processos ajudam a equipe ao invés de ditar como seu trabalho deve ser feito, de forma que são os processos que se adaptam à equipe, e não o oposto (Highsmith, 2004). Ainda segundo Highsmith, um processo não é um substituto para uma habilidade, de forma que segui-lo por si só não cria bons programas de computador (Highsmith, 2002).

Software em funcionamento

Para Alistair Cockburn, *software* em funcionamento é o único indicador do que a equipe de fato construiu (Cockburn, 2007). Jim Highsmith afirma que clientes se interessam por resultados, ou seja, *software* em funcionamento que entregue valor de negócio (Highsmith, 2002).

Segundo Cockburn, a documentação pode ser muito útil para o desenvolvimento do projeto, mas deve-se produzir somente a documentação necessária e suficiente para a realização do trabalho (Cockburn, 2007). Highsmith por sua vez afirma que *software* em funcionamento não exclui a necessidade de documentação, pois ela pode permitir a comunicação e colaboração, melhorar a transferência de conhecimento, preservar informações históricas, ajudar a melhorias em progresso e satisfazer a necessidades legais e regulatórias. Segundo o autor, a documentação não é desimportante, ela simplesmente é menos importante do que versões em funcionamento do produto. Ainda segundo o autor, um erro comum em projetos é a crença de que a documentação substitui a interação, servindo como um meio de comunicação.

A documentação, na realidade, facilita a interação, funcionando como seu subproduto na forma de documentos, rascunhos, desenhos, anotações etc., que podem (ou não) ser utilizados como um registro permanente (Highsmith, 2004).

Highsmith afirma ainda que a entrega iterativa de versões do *software* em funcionamento possibilita um *feedback* confiável no processo de desenvolvimento de formas que simplesmente com a documentação é impossível (Highsmith, 2004). Craig Larman denomina essa prática como “entrega evolutiva”, na qual há um esforço em se obter o máximo de *feedback* possível sobre o que foi entregue, de forma que a próxima entrega seja planejada dinamicamente, baseada nesse *feedback* (Larman, 2003).

Colaboração com o cliente

Para Alistair Cockburn, no desenvolvimento Ágil não há “nós” e “eles”, há simplesmente “nós”, o que significa que clientes e desenvolvedores estão do mesmo lado, colaborando para produzir o *software* que traga valor para esses clientes. Nessa dinâmica, ambos são necessários para que se produza *software* de boa qualidade. Segundo o autor, essa colaboração envolve companheirismo, tomada de decisão conjunta e rapidez na comunicação, de forma que muitas vezes pode até tornar contratos desnecessários (Cockburn, 2007).

Segundo Jim Highsmith, no desenvolvimento de novos produtos como *software* em que há alta volatilidade, ambiguidade e incertezas, a relação cliente-desenvolvedor deve ser colaborativa, ao invés de marcada por disputas de contrato antagônicas (Highsmith, 2004).

Responder a mudanças

De acordo com Jim Highsmith, todo projeto deve balancear o planejar com o mudar, dependendo do nível de incerteza inerente a ele. Segundo o autor, em projetos onde há alta incerteza, a investigação e a concepção mental predominam sobre o planejamento e a execução restrita de tarefas planejadas (Highsmith, 2004). Esse conceito se aplica ao desenvolvimento de *software*, já que, de acordo com Craig Larman, a incerteza é inerente e inevitável em projetos e processos de *software* (Larman, 2003). Alistair Cockburn afirma que construir um plano é útil, mas seguir o plano só é útil até o momento em que ele ainda está próximo o suficiente da situação atual. Assim, manter-se preso a um plano ultrapassado não funciona em favor do sucesso (Cockburn, 2007).

Highsmith destaca ainda que empresas que buscam prosperar na turbulenta economia atual devem alterar tanto seus processos quanto suas perspectivas em consideração à mudança, já que praticamente tudo, exceto a visão do produto, pode mudar em pouquíssimo tempo, como escopo, funcionalidades, tecnologia, arquitetura etc. (Highsmith, 2004). Segundo Cockburn, iterações curtas de desenvolvimento permitem que mudanças possam ser mais rapidamente inseridas no projeto, de forma que atendam às novas necessidades dos clientes (Cockburn, 2007).

2.2.2 Os princípios Ágeis

Os doze princípios Ágeis foram cunhados a partir do Manifesto por parte de seus autores e podem ser vistos em seguida, cada um com uma pequena explicação adicional:

- 1) **Nossa maior prioridade é satisfazer o cliente por meio da entrega cedo e frequente de software com valor.**
 - O foco no desenvolvimento do produto está na satisfação dos clientes. Geralmente, desde cedo e frequentemente, retorno ao investimento dos clientes no projeto a partir da entrega de partes do produto que atendam às suas necessidades. Esse princípio se opõe à prática de se seguir um plano detalhado, sugerindo que a prioridade está em se adaptar e buscar, em cada momento, o que de fato trará valor aos clientes, para entregar-lhes o mais cedo e frequentemente possível.
- 2) **Mudanças de requisitos são bem-vindas, mesmo em fases tardias do desenvolvimento. Os processos Ágeis utilizam a mudança em favor da vantagem competitiva para o cliente.**
 - Aceitar a mudança como natural no processo de desenvolvimento do produto, para melhor atender às necessidades dos clientes. Ao se trabalhar em ciclos curtos de *feedback*, permite-se aos clientes evoluirem o produto à medida que melhor entendem suas necessidades e adaptarem às mudanças de mercado, tornando-se mais competitivos. Esse princípio se opõe a se tratar o processo de desenvolvimento do produto como previsível, cenário irreal no qual a necessidade de mudança poderia e deveria ser prevenida, já que ela seria considerada indesejada e custosa.

3) Entregar *software* em funcionamento com frequência, desde a cada duas semanas até a cada dois meses, com uma preferência por prazos mais curtos.

- Entregar a seus clientes e usuários, com frequência, partes do produto prontas gera, a cada entrega, retorno ao investimento dos clientes e permite obter-se *feedback* sobre o que foi produzido. Assim, pode-se adaptar o produto às necessidades dos clientes incrementalmente, reduzindo os riscos do projeto. Esse princípio se opõe a realizar poucas ou, no limite, uma entrega de valor única, apenas ao final do projeto.

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

- Pessoas de negócio e desenvolvedores do produto possuem o objetivo comum de garantir a geração de valor para os clientes e, para atingir esse objetivo, cooperam continuamente durante todo o projeto, interagindo com frequência. Esse princípio se opõe ao cenário de antagonismo comum em projetos de desenvolvimento de *software*, nos quais pessoas de negócios – que frequentemente incluem os próprios clientes do projeto – e desenvolvedores raramente se comunicam e, muitas vezes, estão em lados opostos.

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

- O produto é construído por pessoas. O ambiente, o suporte e a confiança necessários para realizar seu trabalho são fatores fundamentais para sua motivação. Esse princípio se opõe à crença de que o produto se constrói em torno das melhores ferramentas e processos, e não das pessoas, apoiando-se nos melhores instrumentos de monitoração e controle externos, por exemplo.

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

- A melhor forma de comunicação entre membros do time que desenvolve o produto e entre esse time e o mundo externo é a comunicação face a face, que é direta, síncrona e enriquecida pela entonação de voz, olhar e linguagem corporal, entre outros fatores. Quando a comunicação presencial não é

viável (em um projeto distribuído, por exemplo) é uma boa prática fazer-se o melhor uso possível da tecnologia disponível para se aproximar da comunicação face a face. Esse princípio se opõe à utilização de documentos, emails, telefone e teleconferência, entre outros, como formas padrão de comunicação em um projeto.

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

- O progresso do projeto ocorre à medida que partes do produto que signifiquem valor são entregues aos clientes do projeto. Esse princípio se opõe à prática de se gerar artefatos como protótipos e extensos documentos de planos e especificações e, assim, acreditar que se progrediu no projeto. Isso também se opõe à geração de quaisquer artefatos e partes do produto – inclusive documentação – que não gerem valor para os clientes do projeto.

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

- Busca-se promover um ritmo constante e sustentável para o trabalho do time que desenvolve o produto, o que se torna possível quando esse ritmo é apoiado por toda a cadeia, incluindo usuários e patrocinadores. No entanto, ao se exigir do time um compromisso com mais trabalho do que ele é capaz de produzir, são muitas vezes adotadas as horas extras, o trabalho em fins de semana e a pressa exacerbada para se cumprir o prazo de entrega, por exemplo. Essas práticas podem levar à insatisfação dos membros do time de desenvolvimento, a uma menor produtividade e a uma menor qualidade no produto gerado.

9) *A atenção contínua à excelência técnica e a um bom projeto aumentam a agilidade.*

- O produto projetado com qualidade e produzido com excelência técnica permite que seja facilmente ser modificado e, assim, aceite a mudança como natural no processo de seu desenvolvimento. Assim, a alta qualidade no produto gerado é essencial para se manter a Agilidade. Esse princípio se opõe à crença de que, para se obter velocidade e flexibilidade no desenvolvimento do produto, a qualidade deveria ser sacrificada. Na realidade, é exatamente o oposto.

10) Simplicidade - a rate de se maximizar a quantidade de trabalho não feito - é essencial.

- Evita-se o desperdício no desenvolvimento do produto ao não se realizar trabalho que não é necessário. Exemplos comuns de desperdícios incluem desenvolvimento de funcionalidades de que os clientes não precisam ou de soluções desnecessariamente complexas, planejamento com nível de detalhes maior do que se pode ter em um determinado momento e uso ou geração de artefatos desnecessários.

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

- Equipes com maior autonomia são mais eficientes. Essas equipes auto-organizadas trabalham em direção a metas acordadas, mas têm a liberdade de decidir qual a melhor forma de realizar esse trabalho e, assim, são responsáveis e responsabilizadas por seus resultados. Dessa forma, geram um melhor produto.

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

- Para se tornar cada vez mais efetiva, a equipe regularmente inspeciona suas formas de trabalho, identifica pontos de melhoria e se adapta de acordo, promovendo a melhoria incremental contínua. É a inspeção e adaptação que o time realiza em seus processos de trabalho.

2.2.3 Os valores do Scrum

Além dos valores e princípios Ágeis, Scrum também tem o seu próprio conjunto de valores a serem seguidos. Eles complementam as regras do Scrum descritas em seus papéis, artefatos e eventos. Assim, os valores do Scrum representam os pilares para todo o trabalho realizado pelas pessoas que trabalham no projeto, ou seja, o time que desenvolve o produto e a pessoa de negócios que o define, chamada no Scrum de Product Owner.

Os cinco valores do Scrum são: foco, coragem, franqueza, compromisso e respeito (Scrum Alliance, 2012).

- **foco:** os times mais produtivos trabalham em apenas um projeto de cada vez, evitando a multitarefa. O time trabalha focado em metas de negócios claras e alcançáveis com as quais se comprometem. Ao mesmo tempo, essas metas, definidas e negociadas com o Product Owner, mantêm o foco do trabalho nas necessidades de negócios mais urgentes em cada momento. Além disso, para auxiliar o time a manter seu foco no seu trabalho, há no Scrum um facilitador que remove impedimentos e protege o time de distrações e interferências externas, chamado de ScrumMaster;
- **coragem:** as pessoas que trabalham no projeto têm coragem para aceitar a mudança como parte natural do processo de desenvolvimento do produto. O Product Owner e a organização têm coragem para confiar no time e deixá-lo livre para realizar o trabalho necessário para atingir metas acordadas. O time tem coragem para falhar e criar visibilidade sobre os problemas e, assim, aprender com essas falhas e problemas o mais cedo possível. Tanto time quanto Product Owner têm coragem para mostrar e entregar com frequência as partes do produto criadas. O ScrumMaster tem coragem para remover impedimentos e realizar as mudanças necessárias na organização, de forma a ajudar a equipe a se tornar mais produtiva;
- **franqueza:** a franqueza ou transparência é necessária para que se possa realizar a inspeção e adaptação. Assim, o time busca melhorar sua forma de trabalhar a partir do *feedback* frequente de seus membros, o que cria visibilidade sobre os problemas e estimula a busca por soluções. O time também busca validar os resultados de seu trabalho a partir do *feedback* frequente sobre o que produzem, o que cria visibilidade sobre as mudanças necessárias. Ele mantém visível o progresso de seu trabalho, para poder tomar medidas de correção necessárias. Ao mesmo tempo, sente-se seguro para estabelecer e informar suas estimativas de quanto trabalho acredita que pode ser realizado. O time também contacta o Product Owner para esclarecer dúvidas e o ScrumMaster para remover impedimentos o mais cedo possível, sempre que necessário;
- **compromisso:** o time determina como seu trabalho será realizado, monitora seu progresso e realiza as correções de rumo que achar necessárias. Assim, é responsável e responsabilizado pelos seus resultados. Esse controle maior que ele tem sobre seu trabalho o torna mais comprometido com o sucesso do projeto. Em cada ciclo de desenvolvimento, o time se compromete em alcançar a meta de negócio acordada com o Product Owner e, assim, se compromete em

dar o seu melhor para fazê-lo. Por sua vez, o Product Owner se compromete a estabelecer as prioridades de trabalho de forma a satisfazer as necessidades de negócios do projeto, interagindo com quem for necessário para fazê-lo;

- **respeito:** os membros do time trabalham juntos, compartilhando responsabilidades, e assim ajudam-se uns aos outros em seu trabalho. Todos que trabalham no projeto respeitam as opiniões uns dos outros, ouvem e buscam entender os diferentes pontos de vista. O Product Owner respeita as decisões técnicas dos membros do time, que respeitam suas decisões de negócios. Por fim, são respeitados o bem-estar e o direito das pessoas que trabalham no projeto a uma vida privada.

2.3 SCRUM É UM FRAMEWORK SIMPLES E LEVE

O Scrum é um *framework* para gestão de projeto, e não uma metodologia ou um conjunto de processos. Mas o que isso quer dizer exatamente?

Um *framework* ou arcabouço é uma estrutura básica que pretende servir de suporte e guia para a construção, a partir da expansão dessa própria estrutura, de algo com uso prático. Enquanto *framework*, o Scrum não define práticas específicas e detalhadas a serem seguidas. Scrum não prescreve, por exemplo, como o time deve desenvolver o produto, como se deve facilitar seu trabalho ou remover impedimentos, nem como devem ser levantadas as necessidades de negócios ou se deve lidar com os clientes do projeto.

Scrum entende que as práticas necessárias para o sucesso do projeto são muito específicas para cada contexto, não podendo assim ser prescritas. Não existe aquela solução ou conjunto de soluções que funcionam para todas as situações. Ao contrário, as pessoas gerando o produto, a partir dos papéis, eventos, artefatos e regras do Scrum, avaliam, adquirem e desenvolvem um conjunto de práticas que melhor lhe servirão, o que é constantemente reavaliado. Esse é um trabalho contínuo de descoberta, inspeção e adaptação.

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

Extreme Programming (XP), por exemplo, é uma metodologia. Embora seja considerado leve, XP prescreve um conjunto mais completo de práticas que devem ser seguidas para se obter sucesso no projeto. Ao usar Scrum, muitos times de desen-

volvimento pegam emprestadas diversas práticas do XP, como por exemplo a programação em par, a integração contínua, o desenvolvimento dirigido por testes etc.

Scrum é simples e leve. Com ele, não se gera ou aplica nada que não será efetivamente útil e utilizado. Prescrevendo apenas três papéis (Product Owner, Time de Desenvolvimento e ScrumMaster), seis eventos (Sprint e reuniões de Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective) e três artefatos (Product Backlog, Sprint Backlog e o Incremento do Produto), Scrum pode ser facilmente explicado e compreendido (neste livro adicionamos alguns eventos e artefatos opcionais).

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

Analogia do esqueleto

Podemos imaginar que um *framework* se opõe a uma metodologia assim como o esqueleto está para o corpo humano.

O esqueleto – ou seja, os ossos – fornece uma estrutura, mas não se movimenta sozinho. Ele necessita que sejam agregados músculos, tendões e outros tecidos para poder se movimentar. O corpo humano, por outro lado, já possui tudo o que necessita para se movimentar.

O *framework* fornece uma estrutura de trabalho, mas não é útil se aplicado sozinho. Ele necessita de ser complementado por outras técnicas e práticas para trazer valor a quem o está utilizando. Uma metodologia já prescreve grande parte do que é necessário para que sua aplicação seja bem sucedida.

2.4 SCRUM SE APLICA A PRODUTOS COMPLEXOS EM AMBIENTES COMPLEXOS

2.4.1 Sistemas Adaptativos Complexos

Scrum foi criado para auxiliar no desenvolvimento de produtos complexos imersos em ambientes complexos. O desenvolvimento de diversos tipos de produtos pode ser entendido como um sistema adaptativo complexo. Podemos ver a seguir duas das definições mais comuns para esse tipo de sistema:

Sistemas adaptativos complexos são sistemas que envolvem um grande número de componentes, frequentemente chamados de agentes, que se adaptam ou aprendem à medida que interagem.

-- Holland, 2006

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

-- Stacey, 1996

Características desse tipo de sistema incluem, entre outras, a auto-organização de seus agentes, a alta adaptabilidade, a sensibilidade a *feedback* e a emergência de padrões de comportamentos como consequência das interações entre seus agentes. Sistemas adaptativos complexos operam à “beira do caos”, um estado de instabilidade ordenada. A ordem se dá pela emergência de padrões, que aumentam a previsibilidade de comportamentos do sistema em curto prazo, e a instabilidade é exatamente o que possibilita a adaptabilidade.

Diversos líderes do Movimento Ágil basearam suas ideias de como gerenciar o desenvolvimento de *software* em teorias que estudam os sistemas adaptativos complexos (Highsmith, 2002). Scrum foi criado nesse contexto.

2.4.2 Onde utilizar Scrum?

Cynefin (Snowden & Boone, 2007) é um modelo criado por Dave Snowden que auxilia na tomada de decisões. O modelo categoriza os contextos em que sistemas operam, de forma que se possa selecionar a abordagem mais apropriada. Nesse modelo, os contextos ordenados são aqueles em que se pode estabelecer a relação entre causa e efeito, e são divididos em simples e complicado. Os contextos não ordenados, em que não há relação imediata aparente entre causa e efeito, são o complexo e o caótico. No contexto de desordem, é difícil de reconhecer com o que se está lidando (veja a figura 2.2).

Utilizaremos o Cynefin para determinar em que sistemas (tipos de projetos) Scrum melhor se aplica. Vamos analisar o significado de cada domínio de contextos definido no modelo, com relação a projetos.

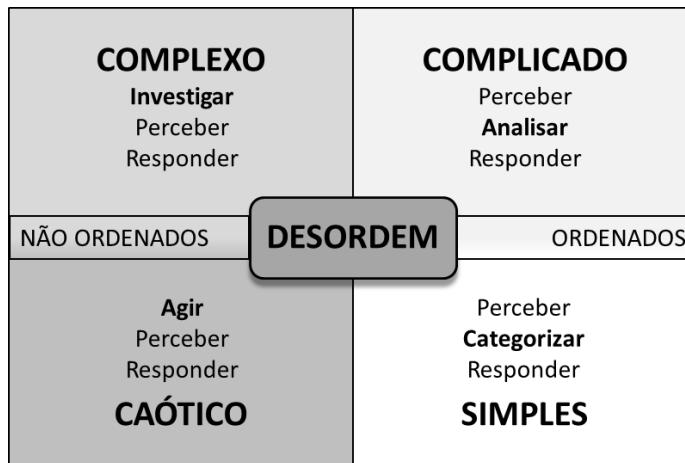


Figura 2.2: O modelo Cynefin

Contextos simples

Esse é o domínio das melhores práticas, onde há estabilidade e as relações entre causa e efeito são evidentes para todos, previsíveis e repetíveis. Assim, para uma dada situação, devem-se avaliar seus fatos (perceber) para definir de que tipo de situação se trata (categorizar) e então aplicar a melhor prática conhecida (responder).

Para um projeto que predominantemente opera nesse domínio, pode-se utilizar conhecimento do passado para realizar um planejamento completo e detalhado do projeto, e então monitorar o seu andamento de acordo com esse plano.

A troca da tubulação de um banheiro com a planta desse banheiro em mãos, por exemplo, é um projeto simples, uma vez que existe uma significativa base de conhecimento sobre o assunto. Realizar para um cliente a digitação dos dados dos produtos de uma loja na Internet também caracteriza um contexto simples.

Scrum não é geralmente muito útil em projetos desse domínio, pois pode trazer uma sobrecarga de regras, eventos e artefatos que apenas são úteis para ambientes de mudança e imprevisibilidade.

Contextos complicados

Esse é o domínio das boas práticas, onde existe a relação entre causa e efeito, mas nem todos podem vê-la. Diferentemente dos contextos simples, pode haver várias respostas viáveis para um mesmo problema e especialistas podem ser necessários

para investigar e escolher a que melhor se aplica ao problema em questão. Assim, para uma dada situação, devem-se avaliar seus fatos (perceber) para definir qual das possíveis soluções é a mais adequada (analisar) e então aplicá-la (responder).

Para um projeto que predominantemente opera nesse domínio, podem-se realizar análises suficientes dos problemas e escolher, entre as possíveis soluções, aquelas que se julguem mais adequadas. Pode-se então realizar um planejamento detalhado do projeto e monitorar o seu andamento de acordo com esse plano.

Definir a planta de uma nova casa, por exemplo, é um projeto complicado, uma vez que escolhas entre diferentes soluções deverão ser adotadas para cada uma das questões do projeto.

Da mesma forma que para contextos simples, Scrum pode trazer uma sobrecarga de regras, eventos e artefatos úteis apenas onde há mudança e imprevisibilidade.

Contextos complexos

Esse é o domínio da emergência, onde a relação entre causa e efeito somente se torna evidente em retrospecto, e não é reproduzível. Não é possível prever o que irá acontecer e, assim, em vez de se impor um caminho, uma abordagem empírica e adaptativa deve ser utilizada, a partir da qual as práticas irão emergir. A tolerância a falhas é um aspecto essencial da aprendizagem empírica. Assim, para uma dada situação, deve-se investigar por meio de experimentos (investigar) para entender seus impactos (perceber) e então promover o que funciona e abandonar o que não funciona (responder).

Abordagens tradicionais de gestão com comando-e-controle não são eficientes nesses contextos de mudança e imprevisibilidade. Nos contextos complexos, aplicam-se a experimentação, a inovação, a criatividade e a emergência de novas formas de se trabalhar. É aqui que operam os Sistemas Adaptativos Complexos, descritos anteriormente, em que se enquadra a grande maioria dos projetos de desenvolvimento de *software*, assim como tantos outros tipos de projetos. É nesses contextos que Scrum melhor funciona.

Contextos caóticos

Esse é o domínio das práticas novas, onde as relações entre causa e efeito são impossíveis de serem determinadas, pois mudam constantemente. Não há padrões, apenas turbulência, de modo que qualquer prática utilizada será completamente nova. Quando possível, deve-se agir rapidamente de modo a reestabelecer a ordem

(agir), para se entender onde existe a estabilidade (perceber) e então trabalhar para trazer a situação do caos para a complexidade (responder), que é um contexto mais gerenciável.

As ações necessárias em uma situação de crise, como o resgate de reféns, por exemplo, caracterizam um projeto caótico. Projetos com foco em pesquisa, no qual o que se descobre em um momento pode mudar todo o trabalho a ser realizado em seguida, são exemplos de projetos que operam em contextos caóticos. Um projeto de desenvolvimento de *software* sobre um sistema legado mal estruturado e repleto de problemas também pode tomar conformações de um contexto caótico.

Scrum não funciona bem para projetos que operam em contextos caóticos. Para que Scrum seja útil, é necessário estabelecer-se ao menos um horizonte de curto prazo no qual os objetivos a serem alcançados sejam bem definidos e estáveis. Nos contextos caóticos, no entanto, os objetivos podem mudar a qualquer momento.

Desordem

Nesse domínio, múltiplas perspectivas se apresentam e não é possível determinar qual dos quatro contextos é o predominante. Assim, nada se sabe sobre a relação entre causa e efeito e não se pode determinar qual a melhor forma de se trabalhar.

Na desordem, as pessoas competem de forma destrutiva por soluções que estejam em sua zona de conforto, ou seja, em algum dos outros quatro contextos. A saída para essa situação é quebrar o problema em partes e definir o contexto de cada uma delas – simples, complicado, complexo ou caótico – para então agir de acordo.

Conclusão

De forma geral, métodos tradicionais de gerência de projetos tratam todos os tipos de projeto como se fossem ordenados, ou seja, pertencendo ao domínio de contextos simples ou complicados. Essa prerrogativa, quando falsa, pode levar a grandes e custosos fracassos.

A grande maioria de projetos de desenvolvimento de *software*, assim como tantos outros tipos de projetos, pertence ao domínio de contextos complexos. Gerenciar projetos que requerem uma abordagem empírica e adaptativa como se fossem ordenados simplesmente não é uma boa solução. Scrum foi criado exatamente para lidar com projetos complexos.

2.5 SCRUM É EMBASADO NO EMPIRISMO

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

-- Ogunnaike & Ray, 1994

Scrum entende que, por ser complexo e possuir um alto grau de incerteza, o desenvolvimento de produtos como *software* deve ser realizado de forma empírica.

Na abordagem empírica, aprende-se tanto sobre os meios de produção utilizados quanto sobre o produto que está sendo gerado ao se trabalhar em ciclos sucessivos de *feedback*, experimentando-se, verificando-se o que é válido e o que não é, e adaptando-se de acordo.

Ao contrário do que pregam métodos tradicionais, não é viável definir as necessidades de negócios com grande nível de detalhes no início do projeto. Embora se trabalhe sempre em direção a uma visão de produto suficientemente bem definida, as necessidades de negócios e seus detalhes irão mudar ao longo do projeto, seguindo as mudanças no ambiente e a melhor compreensão das necessidades dos clientes. Scrum busca maximizar a habilidade do time que desenvolve o produto em responder rapidamente a essas mudanças. Ao mesmo tempo, o time buscará tornar-se cada vez mais produtivo, avaliando quais práticas serão mantidas e quais práticas serão modificadas ou acrescentadas.

Dessa forma, podemos afirmar que o Scrum baseia-se na visibilidade, inspeção e adaptação frequentes tanto do produto quanto dos processos de desenvolvimento para que, pela realização de melhorias incrementais contínuas em ambos, busque-se sempre gerar o maior retorno possível para os clientes.

2.6 SCRUM É ITERATIVO E INCREMENTAL

Scrum é iterativo. O produto é desenvolvido em ciclos ou iterações sucessivas. Em cada um desses ciclos é gerado um incremento no produto, que se soma e modifica o que já se tem pronto do produto até o momento. Cada ciclo, assim, é como se fosse um pequeno projeto autocontido, com todas as atividades necessárias para se gerar uma parte do produto estável e funcionando, ainda que incompleta e que não seja imediatamente entregue para seus usuários (Larman, 2003).

No Scrum, os ciclos são chamados de Sprints. Eles têm tamanho fixo e acontecem um depois do outro, sem intervalos entre eles, ou seja, tudo em um projeto que utiliza Scrum acontece dentro dos Sprints. Cada Sprint é composto por:

- reunião de Sprint Planning, onde se realiza o planejamento do Sprint;
- trabalho de desenvolvimento do produto, que inclui tudo que é necessário para se entregar uma parte do produto pronta ao final do Sprint;
- reunião de Daily Scrum, contida no dia a dia de desenvolvimento, que é uma reunião diária para gerar visibilidade e planejar o próximo dia de desenvolvimento;
- reunião de Sprint Review, onde os resultados do trabalho realizado no Sprint são apresentados a clientes e demais partes interessadas, que oferecem seu *feedback*;
- reunião de Sprint Retrospective, onde o time avalia o que foi bem e o que precisa ser melhorado na sua forma de trabalho;
- quaisquer reuniões ou sessões programadas adicionais que se fizerem necessárias, como por exemplo a reunião de Release Planning, as sessões de Refinamento do Product Backlog etc.

A figura 2.3 representa a iteração do Scrum, indicando a sucessão de elementos presentes em todos os ciclos (ou Sprints) e a sucessão de ciclos. A reunião de Daily Scrum e quaisquer reuniões adicionais acontecem no dia a dia de desenvolvimento.

O time que desenvolve o produto gera, em cada ciclo, uma parte do produto que se soma ou modifica o que já foi produzido. São funcionalidades prontas de ponta a ponta, de forma que é possível entregar o produto para uso, caso se decida fazê-lo. Nesse trabalho em iterações, mais do que gerar apenas funcionalidades prontas, o time visa frequentemente entregar um produto que represente valor suficiente para ser utilizado por seus usuários.

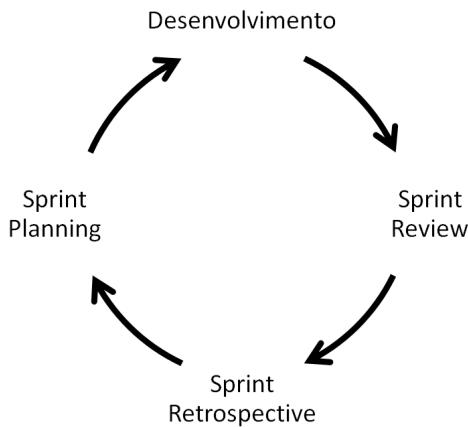


Figura 2.3: Iteração ou sucessão de ciclos (Sprints) do Scrum

Analogia do bolo

Vamos supor que dois times de confeiteiros irão assar um bolo com diversas camadas, cada um utilizando um método diferente (veja a figura 2.4).

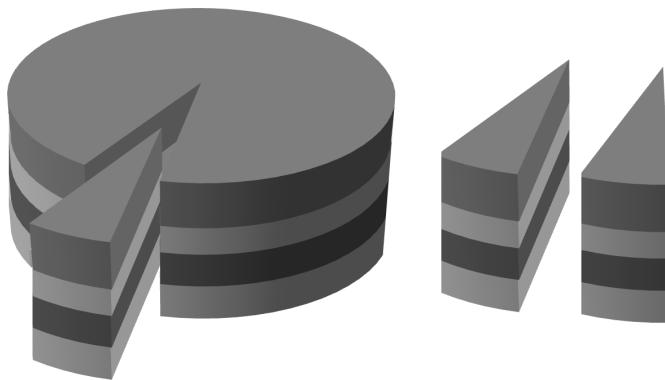


Figura 2.4: A analogia do bolo: o produto com Scrum é gerado fatia a fatia

O primeiro time irá usar um método tradicional, em que se produz o bolo camada por camada. Assim, seus membros misturam os ingredientes da primeira camada do bolo e então a assam. Em seguida, misturam os ingredientes da segunda ca-

mada, assam-na e colocam sobre a primeira. Fazem o mesmo com a terceira, quarta e quinta camadas, colocando cada uma sobre a anterior. Somente ao final uma fatia pode ser cortada do bolo para ser consumida.

O segundo time tem um objetivo ousado. Deve-se permitir que o bolo seja consumido o mais rápido possível, e não apenas no final. Para atingir esse objetivo, o time utiliza uma forma de trabalho bem diferente: um método iterativo e incremental. Eles produzem o bolo fatia por fatia, sendo que cada uma já possui todas as camadas necessárias. Assim, o bolo já pode ser consumido desde a sua primeira fatia. Mesmo que cada uma possa ser bem fina.

É claro que nenhum confeiteiro irá utilizar o método do segundo time para produzir um bolo, mas o trabalho de desenvolvimento do produto com Scrum se assemelha muito com essa forma de se trabalhar. Diferentemente do trabalho com métodos tradicionais, o time de desenvolvimento não trabalha em camadas do produto. A parte do produto, gerada pelo time em cada ciclo de desenvolvimento, é uma fatia do produto de ponta a ponta, pronta para ser utilizada. Essa entrega de fatias funcionando do produto gera desde cedo retorno ao investimento dos clientes e possibilita o *feedback*, reduzindo os riscos do projeto.

CAPÍTULO 3

Como é o Scrum?

3.1 INÍCIO DO PROJETO

O projeto para o desenvolvimento de um produto é contratado ou simplesmente se inicia para suprir algum objetivo ou necessidade de negócios, seja de um cliente específico, de um grupo de clientes ou visando uma oportunidade de mercado. A **Visão do Produto** é uma forma de traduzir esse objetivo a ser alcançado.

O **Product Owner** é o responsável por definir, comunicar e manter a Visão do Produto relativamente constante ao longo do projeto. O Product Owner é único. Ele trabalha com os clientes do projeto e com quaisquer outras partes interessadas que possam contribuir para o entendimento e definição da Visão do Produto. O grupo de partes interessadas do projeto também inclui os próprios **usuários do produto**, que receberão ao longo do desenvolvimento partes prontas do produto para serem utilizadas.

A partir da Visão do Produto, o Product Owner pode criar um plano de como se espera que o produto evolua ao longo do tempo, chamado de **Roadmap do Produto**.

Esse plano é geralmente expresso por uma linha do tempo com as metas esperadas para cada momento no futuro, que aqui chamamos de **Meta de Roadmap**.

Antes do desenvolvimento do produto começar, o projeto geralmente passa por uma fase inicial na qual definições e preparativos básicos são realizados. Essa fase possui uma duração que depende do que e de quanto é necessário se definir e preparar. É chamada por alguns de “pré-jogo” ou, por outros, equivocadamente de “Sprint Zero”, já que o trabalho realizado nessa fase, como será visto mais adiante, de forma alguma caracteriza um Sprint.

Embora seu uso seja comum em projetos Ágeis, tanto a fase de “pré-jogo” quanto os artefatos Visão do Produto e Roadmap do Produto não são parte do *framework* Scrum.

Ainda nessa fase inicial, decide-se quem serão as pessoas a trabalhar no projeto, que formarão o **Time de Scrum**: além do Product Owner, são elas os membros do Time de Desenvolvimento e o ScrumMaster. Esse processo de escolha varia de organização para organização. Entre diferentes possibilidades, pode haver um departamento responsável pela seleção de pessoal, pode-se partir de um Product Owner ou ScrumMaster que escolhe na organização o resto do time, ou se pode designar para o projeto um time que já trabalha junto, por exemplo.

O **Time de Desenvolvimento** realiza o trabalho de desenvolvimento do produto. Ele é multidisciplinar, o que significa que possui, em seus membros, todo o conhecimento necessário para realizar esse trabalho. O Time de Desenvolvimento é também auto-organizado, ou seja, ele próprio define como irá realizar o trabalho e gerenciar seu progresso em direção a metas de negócios acordadas com o Product Owner.

O **ScrumMaster** é o responsável por garantir que os **impedimentos** que o Time de Desenvolvimento encontre em seu trabalho sejam removidos, atuando quando necessário como um agente de mudança na organização. Esses impedimentos geram o risco de não se atingirem os objetivos. O ScrumMaster está presente e age como um facilitador em todas as reuniões do Scrum, facilita o trabalho do dia a dia do Time de Desenvolvimento e facilita as interações entre o Time de Desenvolvimento e o Product Owner. Ele também ensina Scrum ao Time de Scrum e a se auto-organizarem. O ScrumMaster é tão neutro quanto possível e possui *soft skills*, ou seja, competências comportamentais e pessoais, para realizar seu trabalho.

Ainda nessa fase de pré-jogo, pode ser necessário especificar uma arquitetura básica de produto, de forma a reduzir os riscos de decisões tardias que invalidariam o que já foi produzido. Pode ser também necessário criar ou adaptar uma infraestrutura que servirá de suporte para o desenvolvimento do produto. A ideia é gerar

apenas o mínimo necessário e suficiente para reduzir os riscos, mas sem engessar o desenvolvimento do produto nem gerar grandes desperdícios.

Antes do início do desenvolvimento, o Product Owner inicia, a partir da Visão do Produto, a criação de uma lista ordenada, incompleta e dinâmica de itens que representam o que ele acredita que será produzido ao longo do projeto. Essa lista é chamada de **Product Backlog**.

Os itens do topo do Product Backlog são os mais importantes naquele momento e, por essa razão, possuem mais detalhes para serem desenvolvidos primeiro. Os itens mais abaixo têm gradativamente menos detalhes.

O Product Backlog inicial pode ser longo, contendo desde itens pequenos e bem detalhados até itens grandes e vagos. Mas ele também pode conter apenas uma quantidade de itens necessária para se iniciar o desenvolvimento. O Product Backlog evoluirá ao longo de todo o projeto e será frequentemente modificado com a adição, subtração, reordenamento e modificação de seus itens.

User Story é a forma preferida de times Ágeis para representar cada um dos itens do Product Backlog que tratam de necessidades ou objetivos de negócios, descrevendo-os sob o ponto de vista dos usuários do produto e de uma forma concisa, simples e leve. A User Story não é, no entanto, parte do *framework* Scrum, e cabe ao Time de Scrum definir a forma que melhor lhe serve para representar os itens do Product Backlog.

O Time de Scrum então está pronto para iniciar o primeiro de vários ciclos do projeto, nos quais o trabalho de desenvolvimento do produto será realizado. Esses ciclos são chamados de **Sprints**. O projeto com Scrum acontece Sprint após Sprint. Assim, ao terminar um Sprint, inicia-se imediatamente o seguinte.

Os eventos do Scrum – o próprio Sprint e as reuniões de Sprint Planning, de Daily Scrum, de Sprint Review e de Sprint Retrospective – possuem uma duração máxima ou fixa definida, chamada de **timebox**. Os *timeboxes* são importantes, pois evitam o desperdício, limitando o tempo em que um objetivo deve ser alcançado, além de ajudar a criar um ritmo ou uma regularidade no trabalho realizado.

Pode-se ver o ciclo completo do Scrum na figura 3.1 (Schwaber & Beedle, 2002; Schwaber, 2004).

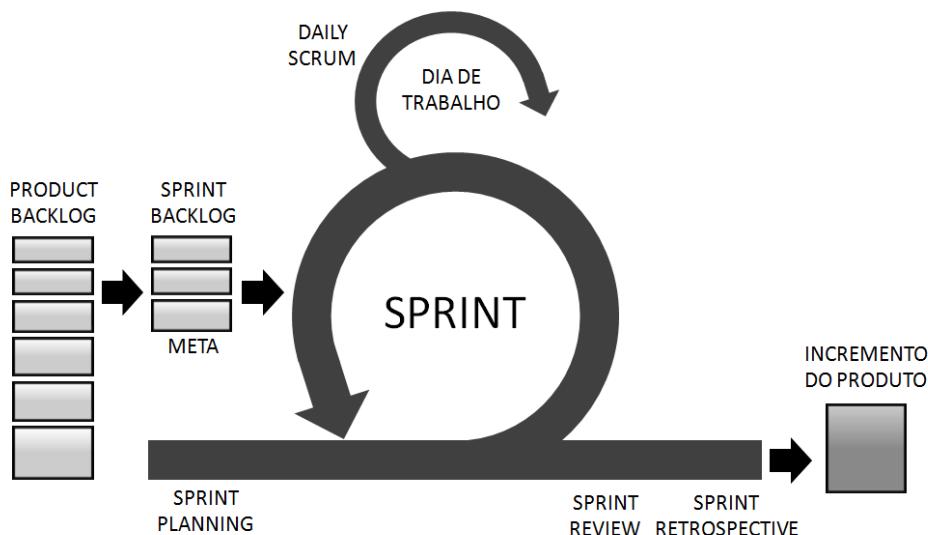


Figura 3.1: O ciclo do Scrum

3.2 PLANEJAMENTO DO SPRINT

O Sprint se inicia com a reunião de **Sprint Planning**, na qual se planeja o trabalho a ser realizado no próprio Sprint. Nessa reunião, Time de Desenvolvimento e Product Owner negociam, a partir dos itens do alto do Product Backlog, o que será desenvolvido. Ou seja, facilitados pelo ScrumMaster, eles selecionam um conjunto de itens do alto do Product Backlog que julgam ser capazes de desenvolver na duração do Sprint, o que é apenas uma previsão, e estabelecem um objetivo ou meta de negócios a ser alcançada com o desenvolvimento desses itens, chamada de **Meta do Sprint**. O Time de Desenvolvimento, então, se compromete com atingir essa **Meta do Sprint**.

É importante que os itens do alto do Product Backlog estejam preparados para que a reunião de Sprint Planning seja eficiente e produtiva. Itens que chegam à reunião sem detalhes suficientes, por exemplo, podem colocar todo o Sprint em risco.

Para garantir que esses itens estejam preparados para serem discutidos na reunião de Sprint Planning, pode-se criar e utilizar uma **Definição de Preparado**. São critérios claros que definem o que é necessário para um item estar preparado para ser colocado em desenvolvimento. Caso um Time de Scrum opte pelo uso de uma Definição de Preparado, o Time de Desenvolvimento passa a ter a prerrogativa de re-

cusar, na a reunião de Sprint Planning, um item que não esteja preparado de acordo com essa definição.

Além do detalhamento necessário, por exemplo, outros critérios da Definição de Preparado podem incluir o item ser pequeno o suficiente e possuir os **Critérios de Aceitação** definidos, entre outros.

No primeiro Sprint, o Time de Desenvolvimento ainda não tem dados para gerar métricas sobre sua capacidade de trabalho em um Sprint. Ele pode estimar os itens do Product Backlog individualmente para possibilitar a futura obtenção dessas métricas úteis para o planejamento. **Story Point** é uma unidade muito utilizada por times Ágeis em suas estimativas. A partir dos Sprints seguintes, o Time de Desenvolvimento pode utilizar como parâmetro a média da quantidade de trabalho entregue nos últimos Sprints, que pode ser medida em Story Points.

Outra forma de se obter essas métricas é dividir e deixar sempre os itens do alto do Product Backlog bem pequenos, com pouca variação de tamanho entre eles, para então contar o número de itens entregues nos últimos Sprints e calcular a média. Em qualquer caso, essa quantidade de trabalho que se espera realizar por Sprint é chamada de **Velocidade do Time de Desenvolvimento**.

Além de, juntamente com o Product Owner, selecionar os itens e definir uma meta, o Time de Desenvolvimento também cria um plano de como o que foi selecionado será desenvolvido. Esse plano é geralmente expresso por tarefas a serem realizadas durante o Sprint. O conjunto de itens selecionados e seu respectivo plano é chamado de **Sprint Backlog** e é geralmente representado na forma de um **Quadro de Tarefas** (veja a figura 3.2).

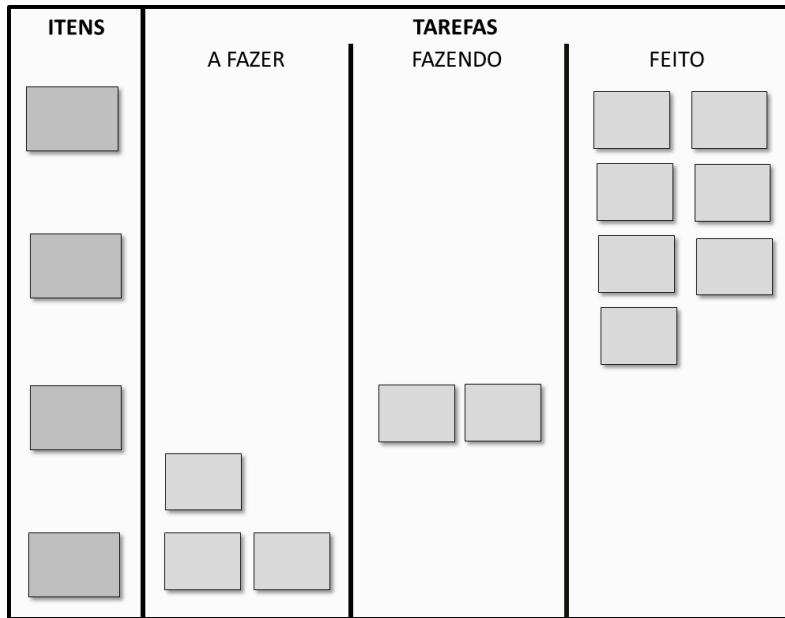


Figura 3.2: Exemplo de Quadro de Tarefas representando o Sprint Backlog

3.3 DESENVOLVIMENTO

Após a reunião de Sprint Planning, inicia-se o trabalho de desenvolvimento propriamente dito dos itens do Sprint Backlog. Os membros do Time de Desenvolvimento definiram, durante a reunião de Sprint Planning, como o trabalho será realizado. Agora, são os responsáveis por se auto-organizar para realizar esse trabalho e por monitorar seu progresso em direção à Meta do Sprint.

O desenvolvimento inicia-se a partir do primeiro item do Sprint Backlog, que é o item mais importante para se atingir a Meta do Sprint. Os diferentes membros do Time de Desenvolvimento realizam as diferentes tarefas necessárias, comunicando-se e colaborando sempre que necessário. Ao completar o primeiro item, seguem juntos para o item seguinte, e assim por diante.

Em cada dia do trabalho de desenvolvimento, os membros do Time de Desenvolvimento se encontram por no máximo quinze minutos, preferencialmente no mesmo horário e no mesmo local, para a reunião de **Daily Scrum**. O objetivo da reunião é garantir a visibilidade de seu trabalho entre eles e planejar, informalmente, o próximo dia de trabalho. O ScrumMaster pode, se necessário, facilitar essa reunião,

mas sua presença não é obrigatória.

A reunião de Daily Scrum é um ótimo momento para se atualizar o **Gráfico de Sprint Burndown**, uma ferramenta que o Time de Desenvolvimento pode utilizar para monitorar seu progresso no Sprint. Esse gráfico possui no eixo x os dias de trabalho no Sprint e no eixo y a quantidade de trabalho restante, que pode ser medida pela quantidade de tarefas restantes ou por algum tipo de estimativa realizada sobre as tarefas, por exemplo. O Time de Desenvolvimento marca no gráfico a quantidade de trabalho restante no dia corrente (veja a figura 3.3).

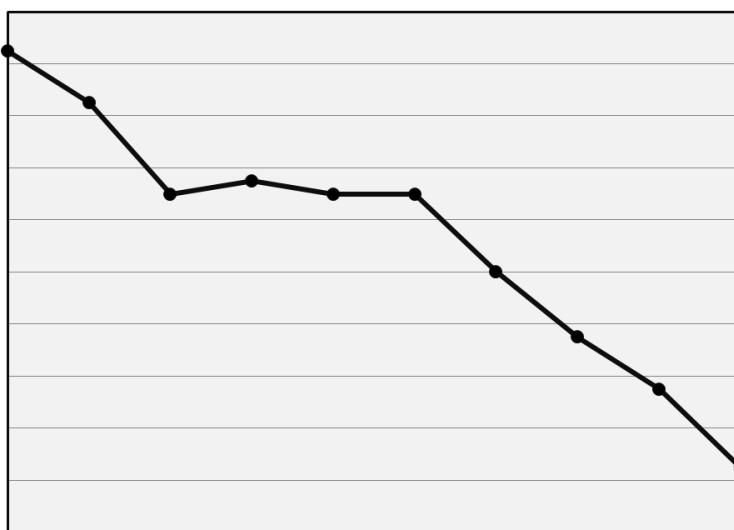


Figura 3.3: Exemplo de Gráfico de Burndown

Em seu dia a dia de trabalho, é comum os membros do Time de Desenvolvimento se depararem com desafios. Sempre que encontram algo que os impede de realizar seu trabalho, eles avisam ao ScrumMaster, que inicia imediatamente o trabalho de remoção desse impedimento.

Um Product Owner acessível ao longo do Sprint permite que o Time de Desenvolvimento tire dúvidas de negócios sobre os itens em desenvolvimento que naturalmente surgem. Ao mesmo tempo, o Product Owner mantém contato constante com os clientes do projeto e demais partes interessadas para entender suas necessidades ou objetivos de negócios mais urgentes, colher *feedback* e, assim, atualizar o Product Backlog de acordo, em um trabalho contínuo de **Refinamento do Product Backlog**.

O Product Owner também busca interagir com o Time de Desenvolvimento para preparar itens para o Sprint seguinte. Para realizar esse Refinamento do Product Backlog, eles podem se encontrar em sessões agendadas, ou esse pode ser um trabalho contínuo ao longo do Sprint. Em qualquer caso, pode-se utilizar uma Definição de Preparado como critério sobre o que significa um item estar preparado.

3.4 ENCERRAMENTO DO SPRINT

Ao final normal do Sprint, o Time de Desenvolvimento deverá ter gerado, a partir dos itens do Sprint Backlog, um **Incremento do Produto** entregável, que representa valor visível para os clientes do projeto. Ser entregável significa que nenhum trabalho adicional é necessário para que esse Incremento do Produto possa ser entregue aos clientes do projeto. Entregar ou não o Incremento ao final do Sprint, no entanto, é uma decisão de negócios e, mesmo que já seja possível, cabe ao Product Owner decidir quando fazê-lo.

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

No último dia do Sprint, Product Owner e Time de Desenvolvimento se encontram para duas reuniões consecutivas de inspeção e adaptação, ambas facilitadas pelo ScrumMaster. Na primeira reunião, de **Sprint Review**, faz-se a inspeção e adaptação do produto, enquanto que na reunião de **Sprint Retrospective**, faz-se a inspeção e adaptação da forma de trabalhar do Time de Scrum.

O Product Owner convida para a reunião de Sprint Review os clientes do projeto e demais pessoas relevantes que possam prover *feedback* sobre o que foi produzido durante o Sprint. Nessa reunião, o Time de Desenvolvimento e o Product Owner apresentam e demonstram para os presentes o que foi produzido no Sprint, para então deles obter *feedback*.

A **Definição de Pronto** é um acordo entre Product Owner e Time de Desenvolvimento sobre o que é necessário para que um item ou o Incremento do Produto como um todo seja considerado pronto e, assim, passe a fazer parte do produto em desenvolvimento. A Definição de Pronto é a mesma para todos os itens do Product Backlog que representem funcionalidades a serem desenvolvidas. Idealmente, ela é utilizada para garantir que o Incremento do Produto gerado no Sprint seja entregável.

O Time de Desenvolvimento utiliza a Definição de Pronto ao criar o plano para

o desenvolvimento dos itens na reunião de Sprint Planning e o Product Owner a utiliza para verificar, no final do Sprint, se os itens estão realmente prontos. Apenas os itens prontos de acordo com a Definição de Pronto podem ser apresentados na reunião de Sprint Review.

O *feedback* obtido dos clientes e demais pessoas relevantes presentes na reunião de Sprint Review é utilizado pelo Product Owner como matéria-prima para alterações no Product Backlog, ou seja, para modificar o produto que está sendo gerado de forma a melhor atender às necessidades dos clientes.

Após a reunião de Sprint Review, Time de Desenvolvimento e Product Owner realizam a reunião de Sprint Retrospective, facilitados pelo ScrumMaster. Nela, ambos identificam o que foi bem no Sprint corrente, e que por essa razão pode ser mantido no próximo Sprint, e o que pode melhorar, buscando formas práticas e traçando planos de ação para fazê-lo. O objetivo desse processo de melhoria contínua é tornar o Time de Scrum cada vez mais efetivo.

Uma vez terminada a reunião de Sprint Retrospective, está encerrado o Sprint. Um novo Sprint se inicia imediatamente após o término da anterior (respeitando, claro, os fins de semana e feriados).

3.5 ENTREGAS

Sempre que o Incremento do Produto ou a soma de Incrementos do Produto representar valor suficiente e já puder ser utilizada, é importante que chegue a seus usuários o mais rápido possível.

Por meio de **Releases** frequentes, o Time de Scrum pode obter *feedback* dos usuários do produto e, assim, reduzir os riscos e produzir o produto certo. O Time de Scrum pode também dar um senso de progresso do projeto aos seus clientes e demais partes interessadas e pode prover retorno ao investimento realizado pelos clientes do projeto.

A estratégia de Releases a ser utilizada no projeto é definida pelo Product Owner. Ele decide quando ou com que frequência as Releases serão realizadas e quem irá recebê-las – algum grupo de usuários que pode experimentar o produto e prover *feedback* ou, sempre que possível, o usuário final. Assim, o Product Owner decide se a Release será realizada cada vez que um item estiver pronto (em entrega contínua), ao final de cada Sprint, sempre que ele julgar adequado ou após alguns Sprints, visando um objetivo ou meta de negócios definida.

Pode-se realizar, nesse último caso, uma reunião de **Release Planning** para cada

Release. Essa reunião é realizada em algum momento antes do início dos trabalhos para a Release correspondente e assim, portanto, deve acontecer durante o último Sprint do Release anterior (ou durante o pré-jogo, para a primeira Release).

Na reunião de Release Planning, Product Owner e Time de Desenvolvimento estabelecem o **Plano de Release**, que contém uma data aproximada para a Release, um objetivo ou meta a ser atingida, chamada de **Meta da Release**, e um conjunto de itens selecionados a partir do alto do Product Backlog. Essa reunião não substitui as reuniões de Sprint Planning que serão realizadas para cada Sprint da Release.

O progresso em direção à data da Release e, portanto, ao cumprimento da Meta da Release é inspecionado Sprint a Sprint, e a ferramenta mais utilizada com esse propósito é o **Gráfico de Release Burndown**. Esse gráfico possui no eixo x os Sprints da Release e no eixo y a quantidade de trabalho restante, que pode ser medida pela quantidade de itens restantes entre os previstos para a Release ou por estimativas realizadas sobre os itens, por exemplo. O Product Owner marca no gráfico a quantidade de trabalho restante ao final de cada Sprint, momento em que o Plano da Release é revisto. Outras ferramentas, como o **Gráfico de Release Burnup**, podem ser utilizadas em seu lugar.

3.6 FINAL DO PROJETO

Um projeto, em geral, possui uma duração determinada. Na realidade, um projeto pode ser encerrado por diversas razões, dependendo do contexto: o tempo de contrato expirou, valor suficiente foi entregue para os clientes do projeto e o contrato permite que o projeto seja interrompido, o orçamento do projeto acabou ou o contrato foi cancelado, entre outras.

No momento em que o projeto passa a ser considerado terminado, espera-se que as necessidades ou objetivos de negócios de maior importância já tenham sido entregues. O projeto terá sido bem-sucedido se, por meio de suas entregas frequentes, a Visão do Produto tiver sido alcançada.

Após um término normal de projeto, é geralmente necessária sua manutenção, em uma fase que podemos chamar de “pós-jogo”. Embora não haja fórmula definida para esse trabalho, sabe-se que o uso de Scrum, por suas características, não costuma ser a melhor opção para esse trabalho. Métodos orientados a pedidos ou tickets como o **Kanban**, criado por David Anderson, são geralmente mais apropriados para essa fase do projeto.

CAPÍTULO 4

De onde veio o Scrum?

4.1 INTRODUÇÃO

O primeiro Time de Scrum foi criado por Jeff Sutherland em 1993. Jeff trabalhava na empresa Easel Corporation, uma empresa de *software* de Massachussets (Estados Unidos) e aplicou Scrum no projeto mais crítico da organização. Resultados excepcionais foram obtidos com essa nova forma de se trabalhar.

Em 1995, Jeff apresentou o primeiro Time de Scrum a Ken Schwaber, que era CEO da empresa Advanced Development Methods e vinha trabalhando com ideias similares, focadas em abordagens empíricas para o desenvolvimento de *software*. Ken e Jeff trabalharam juntos para criar uma definição formal de Scrum, apresentada por Ken no OOPSLA de 1995, um congresso de orientação a objetos.

Em 2001, tanto Jeff Sutherland quanto Ken Schwaber foram signatários do Manifesto Ágil. Esse manifesto deu início ao chamado movimento Ágil, do qual Scrum faz parte. Nesse mesmo ano, Ken Schwaber publicou “*Agile software Development with Scrum*”, o primeiro livro a descrever o *framework* Scrum.

4.2 UM NOVO JOGO

Scrum não é uma sigla ou um acrônimo. Assim, não se deve escrever SCRUM, com as letras maiúsculas, ou S.C.R.U.M., com um ponto após cada letra. Scrum é, na verdade, uma das formações em que as equipes se colocam para a reposição de bola em um jogo de rúgbi. Mas como esse nome veio parar em um *framework* Ágil?

Os criadores do Scrum se inspiraram em um artigo dos autores Hirotaka Takeuchi e Ikujiro Nonaka, intitulado “*The New New Product Development Game*” (ou “O Novo Jogo no Desenvolvimento de Novos Produtos”). Esse artigo foi publicado em 1986 na renomada revista Harvard Business Review (Takeuchi & Nonaka, 1986).

Takeuchi e Nonaka são conhecidos por suas importantes contribuições na área de gestão de conhecimento e aprendizado organizacional. Nesse artigo, esses autores estudaram empresas do Japão e Estados Unidos, como a 3M, Xerox, Honda, Epson e Hewlett-Packard, que estavam utilizando uma abordagem diferente e com excelentes resultados para equipes de desenvolvimento de novos produtos. Essas equipes possuíam seis características fundamentais:

- instabilidade embutida, ou seja, a alta gerência indica objetivos ou estratégias amplos e desafiadores para criar um elemento de tensão, ao mesmo tempo em que dá à equipe uma grande liberdade para alcançar esses objetivos;
- equipes de projeto auto-organizadas, que possuem:
 - autonomia no seu dia a dia do trabalho, com pouquíssima interferência da alta gerência;
 - autotranscendência, ou seja, uma busca interminável por um limite inalcançável, em que a equipe estabelece objetivos cada vez mais altos durante o processo de desenvolvimento;
 - fertilização cruzada, ou seja, a equipe deve possuir uma variedade de especializações, comportamentos e personalidades, de forma que a integração entre seus membros promova a distribuição do conhecimento e o trabalho em equipe, já que cada um entende melhor a posição do outro;
- sobreposição nas fases de desenvolvimento, o que possibilita uma melhor integração entre as diferentes fases e o melhor tratamento de gargalos. Embora os membros da equipe possuam horizontes iniciais diferentes, pois seu trabalho começa em fases diferentes do projeto, a equipe acaba por produzir um ritmo comum de trabalho, ou um pulso, que funciona como uma força motora;

- múltiplo aprendizado, que permite à equipe responder rapidamente às mudanças de mercado. Por ficarem expostos a diversas fontes de informação, os membros da equipe adquirem um amplo conhecimento e habilidades diversas, gerando uma equipe versátil capaz de resolver uma miríade de problemas rapidamente. Esse aprendizado se dá em duas dimensões: em múltiplos níveis, isto é, o aprendizado individual, o aprendizado do grupo e o aprendizado da organização; e em diversas funções, nas quais especialistas são encorajados a acumular experiências em áreas fora de sua especialidade;
- controle sutil exercido pela alta gerência, que estabelece pontos de verificação para prevenir que a instabilidade, ambiguidade e tensão levem ao caos. A gerência evita o tipo de controle rígido que prejudica a criatividade e espontaneidade. O controle sutil é exercido das seguintes formas:
 - selecionando as pessoas certas para o projeto, enquanto monitoram-se as dinâmicas do grupo e adicionam-se ou removem-se membros quando necessário;
 - criando um ambiente aberto de trabalho;
 - encorajando os engenheiros a interagir com os clientes e fornecedores;
 - estabelecendo um sistema de avaliação e recompensa baseado no desempenho do grupo, e não no individual;
 - gerenciando as diferenças de ritmo ao longo do processo de desenvolvimento;
 - tolerando erros, buscando que sejam cometidos o mais cedo possível e que sempre se aprenda com eles;
 - encorajando fornecedores a também se auto-organizarem, e envolvendo-os desde cedo no projeto.
- transferência organizacional de aprendizado, ou seja, a transferência do conhecimento acumulado para outras divisões da organização e para outros projetos de desenvolvimento de novos produtos. Essa transferência ocorre ao se designar indivíduos-chave para projetos subsequentes ou ao se criarem padrões a partir de atividades do projeto. No entanto, a transferência de conhecimento funciona bem quando o ambiente externo é estável. Mudanças no ambiente, no entanto, podem fazer com que essas lições aprendidas sejam inúteis. Nesses cenários, desaprender pode ajudar o time de desenvolvimento

a se manter alinhado à realidade de fora de seu ambiente e a realizar melhorias incrementais.

Takeuchi e Nonaka afirmam nesse artigo que a forma dessas equipes de alto desempenho trabalharem em muito lembra times de rúgbi, que se movem em direção ao objetivo como um só bloco, passando a bola entre seus membros para trás e para frente – daí o nome Scrum. Essa visão opõe-se à forma tradicional de se trabalhar no desenvolvimento de produtos, em que o projeto progride em fases sequenciais e as funções são altamente especializadas e segmentadas.

4.3 LEAN

4.3.1 Introdução

O Sistema Toyota de Produção (STP) foi criado nos anos 1950 pela empresa japonesa Toyota Motor Corporation, com a ambiciosa missão de competir com as gigantes Ford e General Electric (GE), dos Estados Unidos. Essas empresas dominavam o mercado mundial produzindo automóveis da forma mais barata possível, utilizando-se da produção em massa. No cenário do Japão no pós-guerra, com uma economia devastada e um mercado interno muito limitado, a Toyota necessitava de alternativas. A empresa concluiu que deveria produzir pequenos volumes de diferentes modelos, utilizando a mesma linha de montagem (Liker, 2003; Womack et al., 1992).

Os propósitos da Agilidade e do Sistema Toyota de Produção são diferentes, a princípio. Enquanto STP foi originalmente concebido para manufaturas, a Agilidade foi criada no contexto do desenvolvimento de *software*. No entanto, Scrum, assim como todo o movimento Ágil, possui várias de suas ideias enraizadas no STP. Assim, faz-se importante entender esses conceitos e compará-los ao Scrum e à Agilidade em geral.

Nos anos 1990, autores americanos escreveram livros e divulgaram, a partir de um ponto de vista norte-americano, o Sistema Toyota de Produção para o mundo. O sistema japonês foi generalizado e rebatizado para “*Lean Production*” (ou Produção Enxuta), com a justificativa de que nele se utilizam menores quantidades de tudo em comparação com a produção em massa.

Chamaremos tanto o Sistema Toyota de Produção quanto o Lean Production nas seções a seguir apenas de “*Lean*” (Womack & Jones, 1998; Womack et al., 1992).

4.3.2 Os princípios do Lean

Os cinco princípios do Lean são:

- **definir valor:** especificar o que significa valor na perspectiva de seus clientes, e não do produtor. Ou seja, o produto deve atender às necessidades do cliente;
- **identificar a cadeia de valor:** para cada produto (ou família de produtos), identificar todos os passos que são realizados para que se possa oferecer ao cliente o produto final. Os passos que não geram valor para o cliente constituem desperdício e devem ser eliminados;
- **criar fluxo:** os passos que geram valor devem fluir continuamente, sem interrupções nem fronteiras entre departamentos ou entre a organização e fornecedores;
- **estabelecer a produção “puxada”:** os passos devem ser realizados de forma que o produto final seja o que o cliente quer, produzido e entregue quando ele necessita. A produção, assim, é “puxada” pelo cliente, e não “empurrada” pelo produtor;
- **buscar a perfeição:** a organização deve sempre buscar a perfeição, eliminando os desperdícios e melhorando continuamente seus processos.

4.3.3 Lean e Agilidade

Para que se compreendam as influências do Lean na Agilidade, traçamos a seguir um paralelo entre princípios de ambos, além de destacar como Scrum trata dos mesmos.

Definir valor

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

O valor Ágil “*colaboração com o cliente mais que negociação de contratos*” e o princípio Ágil “*pessoas de negócio e desenvolvedores devem trabalhar em conjunto diariamente por todo o projeto*” estabelecem que a contribuição dos clientes na produção de valor é essencial para o sucesso do projeto.

No Scrum, o Product Owner é responsável por interagir frequentemente com os clientes e definir de forma incremental um produto que lhes signifique valor, ordenando a produção a partir das funcionalidades de maior importância para os clientes. Entregas frequentes são realizadas. *feedback* é obtido dos clientes em cada ciclo de desenvolvimento e em cada entrega, e o produto é ajustado de acordo.

Identificar a cadeia de valor

Os passos na produção que não geram valor para os clientes constituem desperdício (ou *muda*, em japonês) e devem ser eliminados. Evitar o desperdício, uma das principais preocupações do Lean, permeia praticamente todos os valores e princípios Ágeis. Em particular, o princípio Ágil “*simplicidade - a arte de se maximizar a quantidade de trabalho não feito - é essencial*” defende a redução do desperdício pela aplicação da simplicidade, ou seja, a geração, como acessório e como produto do trabalho, de somente o que é realmente necessário e suficiente e da forma mais simples possível.

Podemos definir a cadeia de valor no Scrum como todas as atividades realizadas para que se possa oferecer aos clientes um Incremento do Produto pronto ao final de um ciclo de desenvolvimento (Sprint), de acordo com a Definição de Pronto. Entre essas atividades, aquelas que não agregam valor ao produto (e que, portanto, são desnecessárias) são desperdício e devem ser eliminadas imediatamente (chamadas de *muda* Tipo 2 no Lean). No entanto, aquelas que não agregam valor ao produto, mas são necessárias ou inevitáveis, também são desperdício, mas não podem ser eliminadas, ao menos naquele momento (chamadas de *muda* Tipo 1 no Lean). Algumas documentações burocráticas, exigidas pelos clientes ou pela organização, são exemplos desse último tipo de desperdício.

O time trabalha sempre a partir dos itens mais importantes em cada momento. Assim, ao se chegar ao final de um prazo dado – uma data de entrega ou mesmo o final de um Sprint – os itens mais importantes estarão prontos, restando apenas os menos importantes. A priorização desses itens é continuamente revista e atualizada a partir do *feedback* dos clientes sobre o que foi produzido, de forma a garantir que o desenvolvimento de itens de pouca importância seja postergado ou até mesmo não realizado, o que ajuda a evitar o desperdício.

Nas práticas Ágeis, a busca pela maximização da comunicação dentro da equipe e entre os membros da equipe e os clientes, por meio das entregas frequentes e das reuniões programadas, pretende propiciar inspeções frequentes do processo e do produto. Assim, com o *feedback* obtido por essas inspeções, busca-se permitir a

adaptação do processo e do produto com as melhorias, que dessa forma reduzem o desperdício. Outra questão é a valorização dos indivíduos e de suas interações mais que processos e ferramentas, pois observa-se que o maior foco nos dois últimos, ou seja, no uso excessivo de ferramentas e de processos complexos, gera desperdícios.

Já o princípio Ágil “*software em funcionamento é a medida primária de progresso*” é uma resposta a uma das grandes fontes de desperdício em projetos, que é a geração de funcionalidades ou artefatos (como documentos, planos e relatórios) que não criam qualquer valor para os clientes. Pode-se afirmar também que o princípio Ágil “*a atenção contínua à excelência técnica e a um bom projeto aumenta a agilidade*” pretende reduzir o desperdício buscando evitar produtos mal projetados e realizados por indivíduos sem o conhecimento e habilidades necessárias.

Criar fluxo e estabelecer a produção “puxada”

Lean define que os passos necessários para a geração de valor devem fluir continuamente, um após o outro, sem paradas, lotes, filas ou fronteiras entre departamentos de forma que o produto seja produzido e entregue quando o cliente necessita, em uma produção “puxada” – a aplicação do chamado *just-in-time* ou JIT.

Diferentemente do sistema em lote, no JIT cada processo produz somente o que é necessário para o processo seguinte, em um fluxo contínuo. Esse sistema inclui todos os fornecedores no fluxo de produção e elimina a necessidade de estoques.

De forma semelhante, no desenvolvimento Ágil de software com Scrum, Incrementos do Produto prontos são gerados por equipes multidisciplinares em iterações curtas, uma após a outra, que objetivam entregas frequentes. Cada Incremento entregue deve suprir as necessidades de negócio mais importantes para os clientes em cada momento, e sua descoberta é um trabalho contínuo de interação do Product Owner com esses clientes.

Kanban, palavra que significa “sinalização” em japonês, é um sistema do Lean utilizado para auxiliar na criação do fluxo utilizando-se da visibilidade. Os operadores utilizam sinais visuais no fluxo de produção para determinar que uma ação deva ser tomada, como quando há a necessidade de reposição de uma peça que serve de entrada em um processo ou quando há necessidade de algum tipo de ajuda. Sinais visuais comuns nesse sistema são os cartões de sinalização com diferentes cores (e significados) e até mesmo recipientes vazios, indicando que devem ser preenchidos novamente (Gross & McInnis, 2003).

O Sprint Backlog do Scrum é frequentemente representado na forma de um Quadro de Tarefas. Embora não seja um *kanban*, esse quadro funciona de forma parecida

criando visibilidade sobre o fluxo de trabalho do Time de Desenvolvimento.

Lean traz também o conceito de *jidoka*, que significa “automação com um toque humano” em japonês. O *jidoka* estabelece que os próprios trabalhadores devem realizar o controle de qualidade, paralisando imediatamente a produção caso algum defeito seja encontrado, para se buscar a causa raiz. A qualidade, assim, deve estar embutida no próprio processo de produção.

Da mesma forma, as equipes multidisciplinares do Scrum são responsáveis pela qualidade do produto, o que significa que a habilidade de testar ou garantir a qualidade deve estar dentro da equipe, não existindo assim equipes externas de qualidade.

Buscar a perfeição

O *kaizen*, ou “mudar para melhor” em japonês, é a busca da perfeição por meio de etapas infinitas e frequentes de melhorias contínuas, prática essencial do Lean. Uma das chaves para se viabilizar a realização do *kaizen* é o *hansei*, que significa “reflexão profunda” em japonês. O *hansei* é uma auto-reflexão implacável e obrigatória que é utilizada para identificar e reconhecer os erros cometidos ou melhorias que precisam ser feitas, para então se criar um plano de ação que coloque as melhorias em prática. O *hansei* parte da crença de que há sempre algo que pode ser melhorado e seu objetivo nunca é apontar culpados, mas sim tornar os problemas visíveis para ajudar a equipe a realizar melhorias (Liker, 2003).

OS CINCO PORQUÊS

Para se descobrir as raízes dos problemas, pode-se fazer uso no Lean de uma ferramenta chamada de Os Cinco Porquês. Nessa técnica, uma vez que um problema ocorre, pergunta-se “por quê?” iterativamente cinco vezes, sempre questionando a resposta anterior. Assim, inicialmente pergunta-se o porquê do problema e obtém-se uma causa imediata. Essa causa é questionada perguntando-se novamente “por quê?”, para o que se obtém outra causa, que por sua vez é questionada novamente, e assim sucessivamente até que “por quê?” tenha sido perguntado cinco vezes, quando então se pretende ter chegado à raiz do problema. A partir daí, propõem-se ações de melhoria (Liker, 2003; Osono et al., 2008).

Essa técnica pode ser utilizada por times Ágeis, já que descobrir a causa raiz dos problemas para, então, resolvê-los é uma prática essencial da Agilidade, onde busca-se evitar o desperdício e realizar as melhorias contínuas.

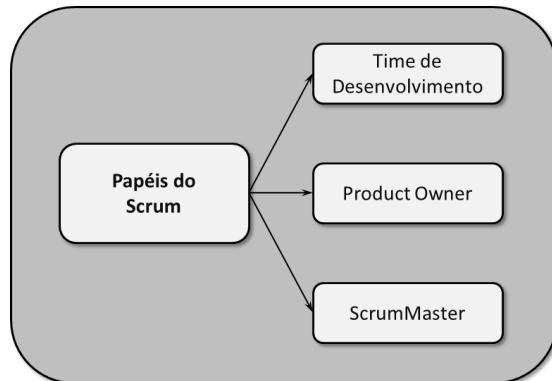
Na Agilidade, a melhoria contínua é expressa no princípio “*em intervalos de tempo regulares, a equipe reflete sobre como se tornar mais efetiva, para aprimorar e ajustar seu comportamento de acordo*”. Um dos importantes mecanismos de melhoria contínua no Scrum são as reuniões de Sprint Retrospective, que é basicamente a aplicação do *hansei*.

Parte II

Papéis: o Time de Scrum

Apenas três papéis são definidos pelo Scrum: o Time de Desenvolvimento, o Product Owner e o ScrumMaster. As pessoas que desempenham esses papéis são igualmente responsáveis e responsabilizadas pelos resultados do trabalho e, assim, se comprometem com o projeto.

Os membros do Time de Desenvolvimento, o Product Owner e o ScrumMaster formam o Time de Scrum. Dessa forma, eles são membros de um mesmo time, e trabalham juntos, de forma colaborativa, para alcançarem seus resultados.



Ao longo do livro, chamaremos de “clientes” as pessoas, grupos ou organizações que solicitam o projeto ou apenas o patrocinam e recebem o retorno ao investimento

uma vez que o que é produzido lhes é entregue. Chamaremos de “usuários” aqueles que, de fato, recebem e utilizam o produto. Clientes podem também ser usuários.

As demais partes interessadas no projeto contribuem de alguma forma para o projeto ou simplesmente são informados do seu progresso. Além dos próprios usuários do produto, exemplos incluem patrocinadores do projeto, executivos e gerentes da organização.

CAPÍTULO 5

Time de Desenvolvimento

5.1 QUEM É O TIME DE DESENVOLVIMENTO?

O Time de Desenvolvimento é um grupo multidisciplinar de pessoas, responsável por realizar o trabalho de desenvolvimento do produto. A partir das prioridades definidas pelo Product Owner, o Time de Desenvolvimento gera, em cada Sprint, um Incremento do Produto pronto, de acordo com a Definição de Pronto, e que significa valor visível para os clientes do projeto.

O Time de Desenvolvimento gerencia o seu trabalho de desenvolvimento do produto. É ele que determina tecnicamente como o produto será desenvolvido, planeja esse trabalho e acompanha seu progresso. Para tal, tem propriedade e autoridade sobre suas decisões e, ao mesmo tempo, é responsável e responsabilizado por seus resultados.

Para realizar esse trabalho, o Time de Desenvolvimento:

- planeja seu trabalho, definindo com o Product Owner o que será realizado

no decorrer do Sprint, para então detalhar, de forma autônoma, como esse trabalho será realizado;

- realiza as tarefas de desenvolvimento do produto para atingir a Meta do Sprint, garantindo a qualidade do que é produzido, além de acompanhar seu progresso no Sprint em direção a essa Meta;
- interage com o Product Owner, sempre que necessário, para ter dúvidas esclarecidas ou solicitar decisões quanto ao produto, e colabora com ele para refinar e aprimorar o Product Backlog, preparando-o para o próximo Sprint;
- identifica e informa ao ScrumMaster sobre impedimentos que obstruam seu trabalho e previne-se deles, quando possível;
- obtém *feedback* dos clientes do projeto e demais partes interessadas sobre o trabalho realizado durante o Sprint, ao apresentar e demonstrar os resultados desse trabalho ao final do Sprint;
- entrega valor com frequência para os clientes do projeto.

O Time de Desenvolvimento é:

- multidisciplinar, possuindo todas as habilidades e conhecimentos necessários para gerar, em cada Sprint, o Incremento do Produto pronto, de acordo com a Definição de Pronto;
- auto-organizado, planejando e executando seu trabalho com autonomia, propriedade e responsabilidade;
- suficientemente pequeno, de forma que seus membros se comuniquem efetivamente e se auto-organizem, sendo capazes de produzir Incrementos do Produto prontos que representem valor visível para os clientes;
- motivado, uma vez que possua o ambiente, apoio e a confiança necessários para realizar seu trabalho;
- orientado à excelência técnica, buscando sempre aprender e realizar seu trabalho com qualidade e consciênci;a;
- focado nas metas estabelecidas junto ao Product Owner.

5.2 O QUE FAZ O TIME DE DESENVOLVIMENTO?

5.2.1 Planeja seu trabalho

Na reunião de Sprint Planning, o Time de Desenvolvimento colabora e negocia com o Product Owner para decidirem o que será realizado no decorrer do Sprint que se inicia. Ou seja, a partir das prioridades definidas pelo Product Owner e a partir da capacidade de trabalho do Time de Desenvolvimento, eles decidem juntos quais e quantos itens estarão previstos para serem desenvolvidos nesse Sprint e definem a Meta do Sprint que o Time de Desenvolvimento irá se comprometer a atingir por meio desses itens selecionados.

Para auxiliá-lo com o planejamento, o Time de Desenvolvimento pode estimar o tempo necessário para o desenvolvimento de cada item do Product Backlog, e usar essas estimativas como parâmetros. Uma vez que se meça, a partir das estimativas, o quanto em média o Time de Desenvolvimento entregou nos últimos Sprints, esse valor pode ser utilizado como um balizador de sua capacidade. É importante notar que essas estimativas, quando utilizadas, são sempre feitas pelo Time de Desenvolvimento e nunca pelo Product Owner, ScrumMaster ou qualquer outra pessoa.

O Time de Desenvolvimento também planeja como irá desenvolver esses itens selecionados, o que em geral é feito quebrando-os em tarefas e, possivelmente, estimando-as. O conjunto formado pelos itens selecionados, suas tarefas correspondentes e o status delas é chamado de Sprint Backlog (veja “[10. Sprint Backlog](#)”).

5.2.2 Realiza o desenvolvimento do produto

O Time de Desenvolvimento realiza, durante o Sprint, as tarefas necessárias para transformar cada item do Sprint Backlog em uma funcionalidade pronta do produto, de acordo com a Definição de Pronto estabelecida em conjunto com o Product Owner. O Time de Desenvolvimento é responsável por garantir a qualidade dos resultados do seu trabalho, o que em geral é uma parte explícita da Definição de Pronto acordada.

É importante entender que, embora tenha a intenção de completar todo o trabalho planejado, o objetivo do Time de Desenvolvimento é o de atingir a Meta do Sprint, o que pode ocorrer sem que necessariamente tenham sido realizadas todas as tarefas de todos os itens selecionados do Product Backlog.

Aqui, consideramos **desenvolvimento do produto** toda e qualquer atividade necessária no trabalho de produção do Incremento do Produto, de acordo com a Definição de Pronto. No caso de desenvolvimento de *software*, por exemplo, esse trabalho pode incluir programação, diferentes tipos de testes e de documentação etc.

O Time de Desenvolvimento possui em seus membros todos os conhecimentos e habilidades necessários para realizar o trabalho de desenvolvimento do produto, evitando assim dependências externas. O Time de Desenvolvimento também possui autonomia para realizar seu trabalho da forma que considerar mais apropriada, planejando-o, acompanhando seu progresso e buscando suas próprias soluções para os problemas que surgirão pelo caminho.

5.2.3 Interage com o Product Owner durante o Sprint

Durante o Sprint, o Time de Desenvolvimento, sempre que necessário, comunica-se com o Product Owner para que ele esclareça ou tome uma decisão rápida sobre algum item do Sprint Backlog. Para tornar essa comunicação possível, o Product Owner coloca-se acessível e disponível no decorrer do Sprint. Caso contrário, o Time de Desenvolvimento pode ter que pausar o desenvolvimento do item enquanto aguarda por essa interação, o que se torna um impedimento. Outra possível consequência é que o Time de Desenvolvimento termine por tomar decisões sobre o produto que não lhe cabem.

O Product Owner ter acesso a uma prévia e poder dar *feedback* antecipado sobre o trabalho em andamento pode ser saudável e reduzir os riscos de um possível fracasso do Sprint. O Product Owner, no entanto, de forma alguma interfere no planejamento realizado pelo Time de Desenvolvimento sobre como transformar os itens selecionados do Product Backlog em um Incremento do Produto, pois essas decisões não lhe cabem.

Também é saudável Time de Desenvolvimento e Product Owner interagirem para refinar e preparar itens no alto do Product Backlog para o Sprint seguinte. Diversos Times de Scrum realizam essa atividade durante a reunião de Sprint Planning, no início do Sprint onde os itens serão desenvolvidos. Outra forma programada e mais efetiva de realizar essa atividade é utilizar sessões de Refinamento do Product Backlog no Sprint anterior e nelas preparar os itens do Product Backlog para a próxima reunião de Sprint Planning, de acordo com uma Definição de Preparado acor-

dada. Ambos os termos, “Refinamento do Product Backlog” e “Definição de Preparado”, não são parte do Scrum básico (veja “[23. Refinamento do Product Backlog](#)” e “[15. Definição de Preparado](#)”).

5.2.4 Identifica e informa os impedimentos ao ScrumMaster

Um impedimento é um obstáculo ou barreira que dificulta significativamente ou impede que o trabalho do Time de Desenvolvimento seja realizado, bloqueando um ou mais itens do Sprint Backlog de forma a ameaçar a Meta do Sprint. A resolução de um impedimento não está ao alcance do Time de Desenvolvimento, está fora do seu contexto de trabalho ou lhe tomará muito tempo. Problemas do dia a dia que o próprio Time de Desenvolvimento pode resolver, sem que ameacem a Meta da Sprint, não caracterizam impedimentos.

O Time de Desenvolvimento busca proteger-se e prevenir-se de impedimentos em seu trabalho. Uma vez que o Time de Desenvolvimento encontra um impedimento que não pôde ser evitado, ele é imediatamente informado ao ScrumMaster, que então toma as ações necessárias para removê-lo o mais rapidamente possível. Assim, os membros do Time de Desenvolvimento não esperam até a reunião de Daily Scrum seguinte para comunicar impedimentos ao ScrumMaster, mas sim o fazem ativamente durante o seu trabalho.

Sinalização de impedimentos

Além da comunicação verbal, uma boa prática que pode ser utilizada pelo Time de Desenvolvimento é a sinalização visual do impedimento no seu Quadro de Tarefas, junto ao item ou à tarefa afetada. Essa sinalização não substitui a ação de se informar o impedimento imediatamente ao ScrumMaster, mas ajuda tanto o ScrumMaster quanto o Time de Desenvolvimento a não negligenciarem e a melhor lidarem com o impedimento.

Um exemplo típico desse tipo de sinalização é a marcação do item ou tarefa impedidos com uma cor diferente, seja colando-se outra nota adesiva sobre o item ou tarefa (quando o Time de Desenvolvimento utiliza um Quadro de Tarefas real), mudando-se a cor do próprio item ou tarefa (no caso de um quadro virtual) ou utilizando-se algum outro tipo de marcação. A figura [5.1](#) mostra um exemplo de como se pode sinalizar um impedimento em uma tarefa.

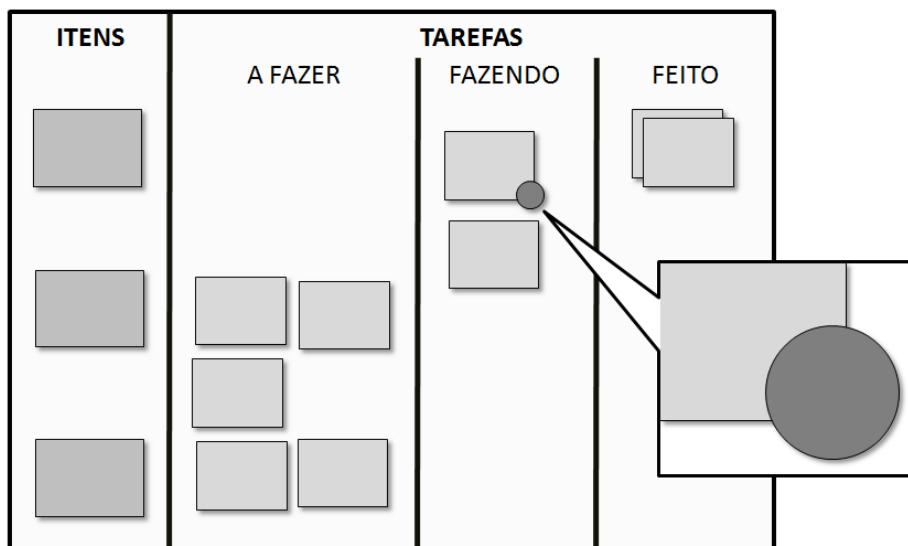


Figura 5.1: Exemplo de sinalização de um impedimento no Quadro de Tarefas

Prevenção de impedimentos

Em geral, impedimentos podem evidenciar problemas sérios na estrutura da organização, na qualidade dos processos ou na qualidade dos produtos gerados. Assim, é importante que o Time de Desenvolvimento não entenda os impedimentos como um fator natural e aceitável do processo.

Impedimentos também geram desperdícios, pois além de provocar pausas e esperas, aumentam a quantidade de trabalho em progresso, podem provocar a execução de diversas atividades ao mesmo tempo e estimulam um consumo inapropriado de energia do Time de Desenvolvimento. Esse consumo de energia ocorre desde a identificação do impedimento como tal, o que pressupõe um esforço inicial para resolvê-lo, até a interrupção do trabalho, a sinalização do impedimento ao Scrum-Master e as mudanças de contexto decorrentes de trocas do item em execução.

Considerando esses efeitos extremamente negativos dos impedimentos, é saudável que o Time de Desenvolvimento, antes de tudo, busque evitar a própria ocorrência do impedimento. Para que seja possível fazê-lo, o Time de Desenvolvimento pode utilizar alguns importantes recursos, como os exemplos descritos abaixo (Pimentel, 2009).

Um verdadeiro compromisso com as metas. Quanto menor for o compromisso e, portanto, a motivação do Time de Desenvolvimento acerca da Meta do Sprint, mais facilmente um problema cuja resolução está ao alcance do Time de Desenvolvimento será entendido como impedimento e aceito como tal. Assim, esse problema poderá, mesmo que involuntariamente, ser utilizado como razão para atrasos ou mesmo para a não realização do trabalho.

O uso da dinâmica da iteração e seu *timebox* a seu favor. Solicitações externas ou iniciativas do próprio Time de Desenvolvimento que desviam da Meta do Sprint o foco de seu trabalho também podem transformar-se em impedimentos. Assim, é importante que o Time de Desenvolvimento seja capaz de questionar se a urgência da interrupção é real ou se ela pode esperar o próximo Sprint, onde o trabalho necessário será devidamente planejado. A Meta do Sprint pode ficar seriamente ameaçada, por exemplo, quando um pedido da alta gerência fizer com que o Time de Desenvolvimento pare o que está fazendo para atendê-lo, e assim abandone o planejamento. O mesmo problema ocorre quando, no decorrer de um Sprint, o Time de Desenvolvimento resolve testar ou implantar novas ferramentas para melhorar sua produtividade, mas gasta tempo demais nessas atividades não planejadas. É importante lembrar aqui que, caso a Meta do Sprint perca o sentido, o Sprint provavelmente será cancelado (veja “[16.3. O Sprint pode ser cancelado?](#)”).

O acionamento preventivo do ScrumMaster. Assim que identifica um impedimento iminente, o Time de Desenvolvimento aciona o ScrumMaster para ajudá-lo a rejeitar esse impedimento antes mesmo que ele se concretize e, assim, proteger as atividades que estão sendo executadas. Por exemplo, assim que identificar ser futuramente necessário o envolvimento pontual de uma pessoa ou equipe externa de difícil disponibilidade, o Time de Desenvolvimento pode imediatamente solicitar uma ação do ScrumMaster.

A identificação de dependências em tempo de planejamento. Ao se identificarem, em tempo de planejamento (Release Planning, Sprint Planning e Daily Scrum), dependências internas e externas que poderão impactar o trabalho do Time de Desenvolvimento durante o Sprint, podem-se buscar meios de evitá-las ou antecipar o envolvimento e a ação de atores que possam ajudar a endereçá-las. Por exemplo, não se recomenda colocar em desenvolvimento uma funcionalidade que depende de

uma ferramenta ou de uma pessoa externa que somente poderá se tornar disponível no decorrer do Sprint.

É importante, portanto, que o Time de Desenvolvimento busque se proteger e se prevenir dos impedimentos, tratando-os não somente como algo que prejudica suas pequenas tarefas do dia a dia, mas sim como algo que pode prejudicar e até mesmo comprometer as suas metas.

5.2.5 Obtém feedback sobre o produto

Na reunião de Sprint Review, o Time de Desenvolvimento, em colaboração com o Product Owner, apresenta e demonstra aos clientes e demais pessoas relevantes o trabalho desenvolvido no decorrer do Sprint, ou seja, o Incremento do Produto pronto de acordo com a Definição de Pronto.

Além do Product Owner e membros do Time de Desenvolvimento, as demais pessoas presentes nessa reunião são preferencialmente as mais adequadas para prover *feedback* sobre o trabalho do Time de Desenvolvimento realizado no Sprint, como por exemplo clientes ou usuários diretos dos itens que foram desenvolvidos. O *feedback* dessas pessoas é valioso, pois permite a adaptação do produto de acordo com as necessidades dos clientes, e assim serve de entrada para o Product Owner atualizar o Product Backlog.

A partir do que foi produzido durante o Sprint, o Product Owner determina se a Meta do Sprint foi atingida ou não.

5.2.6 Entrega valor com frequência

A partir das prioridades definidas pelo Product Owner para trazer valor para os clientes, o Time de Desenvolvimento gera, em cada ciclo de desenvolvimento, um Incremento do Produto que é um conjunto de funcionalidades prontas de acordo com a Definição de Pronto, o que significa que já poderiam ser entregues e utilizadas pelos usuários do projeto.

A decisão de fazer uma Release do produto, ou seja, de fato entregar o Incremento ou um conjunto deles para os clientes pertence ao Product Owner. No entanto, ao gerar valor em cada Sprint, o Time de Desenvolvimento possibilita que essa entrega de valor seja frequente.

5.3 COMO É O TIME DE DESENVOLVIMENTO?

5.3.1 Multidisciplinar

O Time de Desenvolvimento é multidisciplinar, e possui em seus membros todos os conhecimentos e habilidades necessários para gerar o Incremento do Produto pronto em cada Sprint, de acordo com a Definição de Pronto. A Definição de Pronto, portanto, está intimamente relacionada com a formação do Time de Desenvolvimento.

Dependências externas, que não estão sob o controle do Time de Desenvolvimento, constituem um risco muito alto para o trabalho realizado durante o Sprint. Se, por exemplo, for necessário que o Time de Desenvolvimento espere o resultado do trabalho de pessoas ou times externos para poder realizar o desenvolvimento de algum de seus itens, pode ser-lhe impossível atingir a Meta do Sprint. Um Time de Desenvolvimento multidisciplinar reduz esses riscos, já que é capaz de realizar todo o trabalho sem, a princípio, depender de ninguém.

Polinização cruzada

O fato de um Time de Desenvolvimento ser multidisciplinar, no entanto, não necessariamente implica em possuir membros multidisciplinares. As pessoas naturalmente possuem suas especialidades, conhecimentos e áreas de interesse específicos. Mas, como com Scrum a responsabilidade sobre os resultados do trabalho recai sobre todo o Time de Desenvolvimento e não sobre determinado indivíduo, os membros do Time de Desenvolvimento são estimulados a trabalhar juntos, trocar conhecimentos e desenvolver habilidades secundárias.

Essa “polinização cruzada” reduz a dependência nas especialidades de determinados membros e, assim, diminui o risco de não se atingir a Meta do Sprint. A ausência ou sobrecarga de um de seus membros em um determinado momento, por exemplo, tem menos chances de se tornar um impedimento para o trabalho do Time de Desenvolvimento, pois normalmente haverá outros ao menos minimamente capazes de realizar o trabalho.

O compartilhamento de conhecimentos também reduz as chances de que, em alguns momentos, membros do Time de Desenvolvimento fiquem sem ter o que fazer e, assim, procurem trabalhar em partes específicas de itens de menor importância ou até mesmo se vejam obrigados a trabalhar em mais de um time ao mesmo tempo. Um especialista que só saiba lidar com bancos de dados, por exemplo, enfrentaria esse problema, já que só teria trabalho em momentos específicos do Sprint.

Em um Time de Desenvolvimento ideal, cada membro seria capaz de utilizar suas diferentes habilidades e conhecimentos para, em conjunto com os outros membros, realizar os diversos tipos de atividades necessárias durante o Sprint. Não haveria distinção entre essas atividades por sua natureza, ou seja, todos os membros do Time de Desenvolvimento seriam capazes de trabalhar em questões como arquitetura de produto, os diferentes níveis de desenvolvimento, os testes necessários, documentação etc.

Aprendizado

De forma geral, espera-se que uma parte significativa dos conhecimentos e habilidades necessários já exista no Time de Desenvolvimento desde o momento de sua formação, ou seja, selecionam-se as pessoas mais adequadas para o projeto. No entanto, novos desafios que exigem aprendizado surgem naturalmente ao longo do projeto. Por meio da “polinização cruzada”, os conhecimentos e habilidades novos e já existentes são disseminados entre os membros do Time de Desenvolvimento.

A forma como conhecimentos e habilidades são adquiridos e compartilhados entre os membros de um Time de Desenvolvimento varia de time para time. As atividades que promovem esse aprendizado podem ser classificadas em formais e informais.

Muitas organizações contratam treinamentos formais para seus funcionários, o que é sempre limitado pelo orçamento disponível. É comum, por outro lado, os próprios membros de Times de Desenvolvimento da organização promoverem, periódica ou eventualmente, sessões de treinamento para seus colegas. Essas sessões são geralmente em torno de assuntos em que esses indivíduos são especialistas ou sobre os quais aprenderam recentemente, seja por conta própria ou por meio de treinamentos contratados (Armony, 2010).

Típico entre desenvolvedores de *software*, o *coding dojo* é outro método formal de treinamento técnico para membros de Times de Desenvolvimento. Por seu aspecto lúdico, a prática do dojo ganha ares informais, embora seja uma forma extremamente disciplinada e eficiente de se cultivar habilidades para o desenvolvimento de *software*.

CODING Dojo

Coding Dojo é um encontro onde um grupo de programadores se reúne para trabalhar em um desafio de programação. Eles estão lá para se divertir e se engajarem em práticas deliberadas, ou seja, atividades direcionadas a melhorar suas habilidades, proporcionando *feedback* (tanto sobre o código produzido quanto sobre as técnicas utilizadas) que permite o aprendizado.

Mais informações em: <http://codingdojo.org/>

O treinamento informal se dá a partir do compartilhamento de conhecimento com colegas de trabalho ou com outras pessoas de fora da equipe ou organização, o que pode acontecer no dia a dia de trabalho ou, por exemplo, com a participação em feiras e eventos.

Oriunda do Extreme Programming, a prática da programação em par tem se mostrado uma forma muito eficiente de treinamento informal em projetos de *software*. Entre outros benefícios que ela traz, ao se colocar juntos, em um mesmo computador, um desenvolvedor menos ou não capacitado em uma determinada matéria e outro mais capacitado, o segundo ensina ao primeiro enquanto ambos executam o trabalho. Essa prática pode acontecer em tempo integral, sistematicamente durante algumas horas por dia ou eventualmente.

5.3.2 Auto-organizado

Ao contrário do que se espera de times tradicionais, não há gerentes de projeto dizendo para o Time de Desenvolvimento como ele deve realizar seu trabalho, nem pressionando ou cobrando informações sobre o andamento de cada tarefa. Com Scrum, o microgerenciamento realizado por um agente externo dá lugar à auto-organização.

O Time de Desenvolvimento, por ser aquele que realiza o trabalho de desenvolvimento, é o mais indicado para planejar e gerenciar esse trabalho. Mas essa auto-organização não deve ser confundida com anarquia ou falta de controle. Ainda que de forma auto-organizada, os membros do Time de Desenvolvimento trabalham seguindo as necessidades dos clientes quanto ao produto a ser gerado, traduzidas pela Meta do Sprint que negociaram com o Product Owner e com a qual se comprometeram. Além disso, o Time de Desenvolvimento está inserido no contexto de uma

organização e, assim, deve estar alinhado com as regras e objetivos dessa organização.

Para facilitar a auto-organização, Scrum possui pontos-chave onde o andamento do trabalho ou seus resultados são inspecionados, como as reuniões de Daily Scrum, onde o próprio Time de Desenvolvimento inspeciona seu andamento em direção à Meta do Sprint, e de Sprint Review, onde os resultados do trabalho no Sprint são inspecionados pelos clientes e partes interessadas. Scrum também possui ferramentas que aumentam a visibilidade do andamento do trabalho, como o Sprint Backlog e gráficos que indicam o progresso ou o trabalho restante.

Para que a auto-organização torne-se possível, é imprescindível que o Time de Desenvolvimento realize o trabalho de forma colaborativa, maximizando a comunicação entre seus membros. Essa é uma das grandes justificativas do porquê do Time de Desenvolvimento dever se manter pequeno em número de membros. Além disso, é importante que o Time de Desenvolvimento possua o espaço necessário para criar e desenvolver, dentro de sua capacidade e ritmo, seus próprios meios para atingir Metas estabelecida para cada Sprint, tornando-se o principal responsável por essas Metas.

O Time de Desenvolvimento conta ainda com o apoio do ScrumMaster para, entre outras coisas, remover impedimentos que atrapalhem seu trabalho, para ensiná-lo a usar Scrum e a se auto-organizar e para facilitar seu acesso ao meios necessários para realizar o seu trabalho.

5.3.3 Suficientemente pequeno

O Time de Desenvolvimento deve ser suficientemente pequeno. De forma geral, recomenda-se que o tamanho do Time de Desenvolvimento seja algo entre 3 e 9 membros. Esses números não incluem o ScrumMaster e o Product Owner, a menos que também exerçam o papel de membro do Time de Desenvolvimento.

Esses parâmetros servem apenas como orientação. Times de Desenvolvimento menores do que três pessoas e não muito maiores do que nove também podem se beneficiar do uso do Scrum. Na realidade, ao escolher o número de membros do Time de Desenvolvimento espera-se que:

- os membros do Time de Desenvolvimento sejam capazes de se comunicar efetivamente para se auto-organizarem. O número de canais de comunicação cresce exponencialmente com o número de pessoas envolvidas. Assim, times

grandes têm maiores dificuldades de comunicação e coordenação, o que dificulta e até pode impossibilitar a auto-organização;

- o Time de Desenvolvimento possua todos os conhecimentos e habilidades necessários para produzir um Incremento do Produto pronto, de acordo com a Definição de Pronto. Esse trabalho se traduz em funcionalidades prontas de ponta a ponta, que representam valor para os clientes, ao invés de partes de funcionalidades ou camadas. Pode ser mais difícil encontrar todos os conhecimentos e habilidades necessários em um número muito pequeno de pessoas;
- o Time de Desenvolvimento seja capaz de produzir valor visível suficiente em cada Sprint, para que se possa obter *feedback* dos clientes e demais partes interessadas na reunião de Sprint Review.

Para times pequenos demais, é possível que o Scrum gere uma sobrecarga com suas regras, papéis e eventos que supere seus benefícios.

Quando o projeto é muito grande, pode-se utilizar, de forma coordenada, mais de um Time de Desenvolvimento trabalhando no mesmo produto. Nesse caso, cada um desses Times de Desenvolvimento do projeto deve ser suficientemente pequeno, de acordo com a definição dada nesta seção.

5.3.4 Motivado

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

Ao buscar fatores que influenciam a motivação dos membros de um Time de Desenvolvimento, deve-se levar em conta que os seres humanos são complexos por natureza e, assim, são em cada instante influenciados por suas necessidades, desejos, aspirações, preferências, problemas, frustrações e toda a sorte de emoções. O homem, portanto, possui momentos bons e momentos ruins na realização do seu trabalho.

Em um Time de Desenvolvimento, a motivação no trabalho é essencial para gerar compromisso de seus membros com as metas e com a qualidade do produto de seu trabalho. A motivação também pode aumentar a estabilidade da composição do Time de Desenvolvimento, já que pode levar a uma menor rotatividade de pessoal, ajudando a não se perder o conhecimento e ritmo de trabalho adquiridos.

Os princípios Ágeis apontam explicitamente o ambiente, o suporte e a confiança necessários para realizar o trabalho como essenciais para a motivação do indivíduo. Outro fator que pode ser entendido como motivador, também apontado pelos princípios Ágeis, é a existência de um ritmo sustentável de trabalho, segundo o qual se evita a prática de horas extras e a aceleração forçada do ritmo de trabalho, por exemplo, diante da perspectiva de atraso em uma entrega prevista.

O ScrumMaster, enquanto uma liderança motivadora, também exerce influência sobre a motivação dos membros do Time de Desenvolvimento, facilitando a auto-organização e encorajando-os à auto-observação, autoavaliação e autorreforço (veja “Liderança no time auto-organizado” em “[[7.2. O que faz o ScrumMaster](#)]”).

A partir de reconhecidas teorias sobre a motivação no trabalho, podemos também destacar exemplos de fatores motivacionais relevantes a Times de Desenvolvimento (Armony, 2010).

Teoria da Fixação de Objetivos

De acordo com essa teoria, a fixação de objetivos desafiadores, claros e atingíveis, o compromisso com esses objetivos e a disponibilidade de *feedback* mostrando o progresso em direção a esse objetivo são poderosos mecanismos motivacionais (Locke, 1996), tanto para indivíduos quanto para grupos (Locke & Latham, 2006).

Os objetivos no Scrum são as Metas estabelecidas para os Sprints. Adicionalmente, pode-se estabelecer metas para as Releases (ou marcos no Roadmap do Produto) e uma Visão do Produto. O *feedback* sobre o progresso do Time de Scrum em direção aos objetivos vem, por exemplo, por meio dos Gráficos de Burndown ou Burnup de Trabalho e da reunião de Sprint Review. De acordo com o princípio Ágil “software em funcionamento é a principal medida de progresso”, podemos também afirmar que os *feedbacks* dos clientes e demais partes interessadas sobre Incrementos do Produto entregues também indicam o progresso em direção aos objetivos estabelecidos.

Teoria das Características do Trabalho

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

Para se chegar a esses estados psicológicos, é importante que determinadas ca-

racterísticas estejam presentes no trabalho do indivíduo. A primeira delas é o indivíduo necessitar de uma variedade de habilidades e conhecimentos para executar seu trabalho, o que ocorre naturalmente em um Time de Desenvolvimento multidisciplinar. Outra dessas características é o indivíduo executar uma parte inteira e identificável do trabalho do início ao fim, característica importante do Scrum em que se produzem, em cada Sprint, funcionalidades prontas e de ponta a ponta. O indivíduo deve também perceber seu trabalho como tendo um impacto significativo sobre a vida ou trabalho de outras pessoas, o que ocorre no Scrum a partir do *feedback* provindo das reuniões de Sprint Review e das entregas frequentes. Esse *feedback*, somado à autorregulação do time, dá uma noção ao indivíduo sobre a sua eficácia e desempenho, mais uma característica necessária. Por fim, a autonomia na realização do trabalho, outra dessas características, é natural em um time auto-organizado.

Teoria ERC

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

A existência de condições básicas de trabalho, como salários em dia, ambiente de trabalho adequado, suporte técnico efetivo e infraestrutura suficiente é uma necessidade de existência de membros do Time de Desenvolvimento.

As necessidades de relacionamento do indivíduo podem ser expressas pela intensa comunicação e pelo bom relacionamento entre os membros do Time de Scrum, que leva ao compartilhamento de pensamentos e sentimentos relevantes.

As necessidades de crescimento do indivíduo podem ser atendidas a partir da perspectiva de evolução de carreira na organização, ou seja, a possibilidade clara de ascensão profissional advinda do reconhecimento do trabalho realizado ou do potencial percebido.

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

AVALIAÇÃO E RECOMPENSA

As avaliações e recompensas relativas ao trabalho realizado (bônus, por exemplo), quando dirigidas ao indivíduo, estimulam-no a trabalhar como indivíduo. Quando dirigidas ao time, estimulam seus membros a trabalharem como um time. Deixar a critério de um gerente julgar a contribuição individual de membros do time somente gera conflitos e desentendimentos, reduzindo a efetividade do time, já que cada membro passa a trabalhar em detrimento do outro. Ao recompensar o indivíduo, o que se está incentivando é a competição entre os membros do time, ao invés da colaboração para se alcançar os objetivos comuns (Hackman, 1987). Assim, prefere-se que times Ágeis sejam avaliados e recompensados como um time, e não como indivíduos.

É importante destacar, no entanto, que algumas abordagens de avaliação cruzada entre membros do Time de Desenvolvimento, ou seja, em que cada membro avalia seus colegas, têm se mostrado efetivas. Nessas abordagens, o conjunto das avaliações serve de base para o sistema de recompensas, sendo em geral cruzado com a avaliação externa da contribuição do Time de Desenvolvimento como um todo. Esse tipo de avaliação funciona melhor quando são realizadas quanto à contribuição do indivíduo para a efetivação de valores como colaboração, comunicação, foco etc. ao invés da quantidade de trabalho realizada por cada um.

5.3.5 Orientado à excelência técnica

“A atenção contínua à excelência técnica e a um bom projeto aumenta a Agilidade”. Esse princípio Ágil reforça a ideia de que um Time de Desenvolvimento deve possuir pessoas tecnicamente qualificadas que trabalhem de forma a produzir com consciência e entregar Incrementos do Produto de alta qualidade. Não existe mágica: não há método ou processo que faça com que maus profissionais produzam bons resultados.

Problemas técnicos podem se acumular no projeto devido à falta de qualificação, ou até mesmo por falta de compromisso com a qualidade. Times de Desenvolvimento são orientados à excelência técnica e, assim, não apenas possuem os conhecimentos e habilidades necessários, mas também continuamente se preocupam

em não gerar dívidas técnicas ou, ao menos, em mantê-las de forma controlada e planejar sua resolução.

Ainda que se considere suficientemente qualificado, o Time de Desenvolvimento está sempre buscando melhores formas de fazer seu trabalho, de se tornar mais efetivo e de produzir com mais qualidade. Diversas práticas do Scrum reforçam essa ideia de melhoria contínua. Entre elas, a reunião de Sprint Retrospective.

5.3.6 Focado nas metas

Uma meta no Scrum é uma necessidade ou objetivo de negócios a ser atendido como resultado de um trabalho. Essas metas são estabelecidas pelo Product Owner e ajustadas em conjunto com o Time de Desenvolvimento. Nessa colaboração, o Product Owner sabe que forçar o Time de Desenvolvimento além de sua capacidade de produção não trará os resultados que ele almeja e que um Time de Desenvolvimento comprometido buscará dar o melhor de si para entregar valor para os clientes do projeto.

Cada Sprint possui uma meta, chamada de Meta do Sprint. Além dessas metas, pode se definir uma Visão do Produto e uma Meta para cada Release ou marco do Roadmap do Produto. Essas metas provêm um foco e uma direção para o trabalho do Time de Desenvolvimento, incentivando a auto-organização. Além disso, elas motivam e oferecem ao Time de Desenvolvimento alguma flexibilidade sobre as entregas correspondentes.

O Time de Desenvolvimento mantém seu foco de trabalho nas metas estabelecidas, e assim dedica o máximo do seu tempo útil ao projeto. Um Time de Desenvolvimento que pratica multitarefa, ou seja, trabalha em mais de um projeto ao mesmo tempo ou em outras atividades além do projeto, encontra grandes desafios para conseguir manter seu foco nessas metas. Para ajudar a manter seu foco na Meta do Sprint, o Time de Desenvolvimento se compromete formalmente com a mesma na reunião de Sprint Planning, tornando-se responsável por atingi-la e sendo responsabilizado por conseguir ou não tal resultado.

CAPÍTULO 6

Product Owner

6.1 QUEM É O PRODUCT OWNER?

O Product Owner, também chamado de P. O., é a pessoa responsável por garantir e maximizar, a partir do trabalho do Time de Desenvolvimento, o retorno sobre o investimento no produto para os clientes do projeto.

O Product Owner define o produto e toma as decisões de negócios relativas a seu desenvolvimento a partir das necessidades dos clientes do projeto e demais partes interessadas, alinhando com ou em direção aos objetivos da organização.

Para realizar esse trabalho, o Product Owner:

- gerencia o produto, inserindo, detalhando, removendo e priorizando as necessidades de negócios do produto no Product Backlog, a partir do contato frequente com os clientes do projeto e demais partes interessadas;
- gerencia os clientes e demais partes interessadas em sua relação com o projeto,

descobrindo quem são essas partes interessadas que devem influenciar as decisões sobre o produto, balanceando suas necessidades com relação ao produto, comunicando-se com elas para descobrir essas necessidades e influenciando-as para maximizar sua colaboração com relação ao projeto;

- mantém a Visão do Produto, definindo-a junto aos clientes do projeto e demais partes interessadas, comunicando-a a todos os envolvidos e garantido que o trabalho seja realizado em direção a ela;
- gerencia as Releases do produto para os clientes, definindo a melhor estratégia para a realização das Releases e estabelecendo a Meta e a data de cada Release;
- realiza, com o Time de Desenvolvimento, o planejamento do Sprint na reunião de Sprint Planning, para que juntos definam uma Meta para o Sprint e uma previsão do que será feito durante o Sprint;
- colabora com o Time de Desenvolvimento, sempre que necessário, para esclarecer dúvidas ou tomar decisões quanto aos detalhes do produto, e para refinar e aprimorar o Product Backlog, preparando-o para o próximo Sprint;
- aceita ou rejeita as entregas do Time de Desenvolvimento, verificando na reunião de Sprint Review se a Meta estabelecida para o Sprint foi atingida.

O Product Owner é:

- único para um Time de Scrum, pois é importante haver apenas um foco de decisões sobre o produto para o Time de Desenvolvimento;
- disponível para tirar dúvidas e esclarecer itens do Product Backlog para o Time de Desenvolvimento, para estar presente nas reuniões de Sprint Planning, Sprint Review e Sprint Retrospective, para interagir com os clientes e demais partes interessadas e para manter e priorizar o Product Backlog;
- representativo com relação ao produto, com conhecimento e poder suficiente para tomar decisões rápidas e adequadas.

Para o projeto, pode-se escolher como Product Owner alguém do próprio cliente ou alguém da organização contratada para gerar o produto.

A escolha para Product Owner de alguém do cliente (ou o próprio cliente, se for apenas uma pessoa) é adotada por muitas organizações. Nesse caso, o Product Owner definirá o produto a ser desenvolvido a partir das suas próprias necessidades e da sua organização. Naturalmente, essa escolha simplifica a gestão da participação do cliente no projeto e todo o processo de obtenção de *feedback*.

Em diversos cenários, no entanto, a escolha de alguém designado pelo cliente como Product Owner pode não ser a melhor opção, especialmente se essa pessoa não souber exercer esse papel. Esse problema é mais comum do que parece. Um Product Owner do cliente pode simplesmente não ter as habilidades e conhecimentos necessários, que obrigatoriamente incluem o próprio Scrum e a gestão de produtos. Embora talvez treinamentos sejam uma opção, é mais comum que o foco do cliente esteja quase que inteiramente nos problemas e muito pouco nas soluções, enquanto que se espera que um Product Owner seja capaz de oferecer soluções de negócios a partir do produto que está definindo.

Além disso, mesmo que esse Product Owner do cliente possua os conhecimentos e habilidades, é possível que ele não possua a disponibilidade necessária para executar o seu papel, pois provavelmente estará envolvido em outras questões de seu próprio dia a dia de trabalho no cliente. O resultado seria um Product Owner que pouco interage com o Time de Desenvolvimento e que não consegue dedicar tempo para pensar no produto e refinar o Product Backlog.

Um outro exemplo simples em que a escolha de um Product Owner no cliente não funciona ocorre quando o projeto possui vários clientes, que podem até mesmo pertencer a diferentes organizações. Pode haver apenas um Product Owner no Time de Scrum e, assim, caso um desses clientes fosse o escolhido para o papel, ele poderia tomar decisões que o favorecessem em detrimento dos outros. Há também a categoria de produtos para os quais não há clientes bem definidos, como por exemplo no desenvolvimento de produtos de prateleira ou de *sites* na Internet com serviços oferecidos ao público.

Nesses casos, uma boa opção pode ser tal que o Product Owner seja designado pela própria organização contratada para gerar o produto. Ele realizará seu trabalho a partir do contato frequente com os clientes do projeto e com as demais partes interessadas. Mas, para ser de fato um Product Owner, essa pessoa deve ter poder e conhecimento suficientes para tomar as decisões necessárias sobre o produto. Ou seja, mesmo que não seja alguém do cliente, o Product Owner é sempre quem de fato define o produto.

6.2 O QUE FAZ O PRODUCT OWNER?

6.2.1 Gerencia o produto

O Product Owner mantém um contato frequente com os clientes e demais partes interessadas ao longo de todo o projeto para fazer o levantamento dos objetivos ou necessidades de negócios mais prioritárias do produto em cada momento. Ele decide quais dessas necessidades farão parte do produto e as insere como itens em uma lista, chamada de Product Backlog. Ele ordena, atualiza e reordena esses itens. Detalha os itens progressivamente à medida que ganham importância e remove aqueles que não mais são necessários e que não mais deverão ser desenvolvidos. Ele também garante a visibilidade do Product Backlog ao Time de Desenvolvimento.

O Product Owner realiza a gestão do produto e tem a palavra final sobre o Product Backlog. Ele é o único que pode alterar o Product Backlog. Nesse trabalho de gestão do produto, é importante que o Product Owner desenvolva algum tipo de estratégia de negócios para o desenvolvimento do produto em direção à Visão do Produto, e utilize-se dessa estratégia para tomar as decisões necessárias quanto ao que será desenvolvido. Essa estratégia, assim como o próprio desenvolvimento do produto, evoluí de forma iterativa e incremental. Um formato possível de estratégia de negócios é o Roadmap do Produto, que descreve o futuro do produto em alto nível, apontando uma data provável para cada Release ou para marcos do projeto e a Meta a ser alcançada em cada uma dessas Releases ou marcos.

Além dos clientes do projeto e demais partes interessadas, o Product Owner também ouve o próprio Time de Desenvolvimento para tomar as decisões quanto ao produto. O Time de Desenvolvimento oferece uma visão do que é tecnicamente possível, do custo de desenvolvimento de cada item do Product Backlog e de que trabalho técnico adicional poderá ser necessário. Com relação a essa última questão, cabe ao Time de Desenvolvimento identificar itens de trabalho técnico – como o teste de uma nova ferramenta ou a melhoria de alguma parte do produto, por exemplo – e sugerir ao Product Owner sua inserção no Product Backlog. Cabe ao Product Owner decidir, após entender o que for necessário junto ao Time de Desenvolvimento, se esses itens serão realmente desenvolvidos e quando.

6.2.2 Gerencia as partes interessadas no projeto

Além dos clientes, as partes interessados no projeto são os usuários, patrocinadores e quaisquer pessoas que tenham algum tipo de influência nas definições do projeto ou tenham interesse direto no seu andamento e sucesso.

O Product Owner faz a gestão dos clientes do projeto e demais partes interessadas em sua relação com o produto que está sendo desenvolvido. Este trabalho inclui:

- identificar corretamente quem são os clientes e pessoas relevantes do projeto, para interagir com as pessoas certas e assim definir as necessidades de negócios;
- entender as necessidades dos diferentes clientes e demais partes interessadas com relação ao produto;
- gerenciar as expectativas dos clientes e demais partes interessadas com relação ao produto, garantindo que entendam o que está para ser demonstrado ou entregue e o que será realizado em seguida;
- balancear e gerenciar conflitos entre as necessidades e expectativas dos diferentes clientes e demais partes interessadas;
- ajudar os clientes e demais partes interessadas a descobrir e entender o que lhes trará maior retorno em cada momento do projeto;
- obter a colaboração direta dos clientes e demais partes interessadas sempre que necessária como, por exemplo, para ajudar o ScrumMaster na remoção de impedimentos organizacionais ou relacionados ao ambiente do cliente;
- comunicar-se frequentemente e proativamente com os clientes e demais partes interessadas para realizar esse trabalho.

6.2.3 Mantém a Visão do Produto

Embora não faça parte do *framework* Scrum, a definição de uma Visão do Produto é uma prática importante para o trabalho no projeto de desenvolvimento de um produto. A Visão do Produto responde à pergunta: “por que esse produto está sendo desenvolvido?”. Ou, ainda melhor, “que problema será resolvido com o desenvolvimento desse produto?”. Ela serve de guia para o trabalho do Time de Scrum e alinha o entendimento e as expectativas quanto ao produto entre as diferentes partes interessadas do projeto e o Time de Scrum. Para tornar esse alinhamento possível, o Product Owner comunica a Visão a todos os envolvidos, assegurando que ela seja compreendida da mesma forma por todos.

O Product Owner estabelece a Visão do Produto junto aos clientes e demais partes interessadas do projeto antes do início do desenvolvimento do produto e, portanto, antes do uso do Scrum. Se possível, é importante também envolver o Time de Desenvolvimento nas atividades necessárias para se chegar a essa definição.

O Product Owner é o responsável por manter a Visão do Produto, gerenciando o Product Backlog de forma a garantir que todo o trabalho realizado pelo Time de Desenvolvimento caminhe em direção a ela.

6.2.4 Gerencia as Releases

A gestão das Releases do produto é uma atribuição do Product Owner. Ele decide qual é a melhor estratégia para se realizarem as Releases no projeto, e a modifica quando necessário. Ele leva em conta também que os princípios Ágeis pregam as entregas desde cedo e frequentes para possibilitar o *feedback* rápido e, assim, reduzir os riscos do projeto.

A estratégia de Releases é condicionada por questões de negócios e questões técnicas. Por um lado, as Releases são alinhadas com a estratégia de negócios do produto de forma a contribuir com ela. Ao mesmo tempo, para decidir como e quando serão realizadas, o Product Owner também considera questões técnicas, que são apontadas pelo Time de Desenvolvimento e, em muitos casos, pelos próprios clientes ou usuários.

Existem diferentes possibilidades para a estratégia para a realização das Releases de um projeto. Por exemplo, elas podem ser realizadas quando o Product Owner julgar adequado, após alguns Sprints, ao final de cada Sprint ou em entrega contínua. As Releases também podem ser realizadas diretamente para seus usuários finais ou para um grupo selecionado desses usuários ou de usuários intermediários, geralmente internos, dependendo da estratégia do produto.

Planeja o Sprint com o Time de Desenvolvimento

Na reunião de Sprint Planning, o Product Owner colabora e negocia com o Time de Desenvolvimento para decidirem o que será realizado durante o Sprint que se inicia. Ou seja, a partir das prioridades definidas pelo Product Owner e a partir da capacidade de trabalho do Time de Desenvolvimento, eles decidem juntos quantos itens estarão previstos para serem desenvolvidos nesse Sprint, desde o item de maior importância, e qual será a Meta do Sprint a qual o Time de Desenvolvimento irá se comprometer a atingir a partir do desenvolvimento dos itens selecionados.

De forma geral, o Product Owner chega a uma reunião de Sprint Planning já com uma boa ideia do que ele pretende obter como Meta do Sprint, mas essa Meta é sempre negociada e refinada com o Time de Desenvolvimento.

Enquanto o Time de Desenvolvimento planeja como irá desenvolver esses itens selecionados (em geral, quebrando-os em tarefas), surge um número de questões sobre o Incremento do Produto a ser gerado. Assim, é importante que o Product Owner, caso não esteja presente nessa parte da reunião, ao menos se apresente disponível e acessível para esclarecer essas dúvidas.

6.2.5 Colabora com o Time de Desenvolvimento durante o Sprint

Durante o Sprint, para permitir ao Time de Desenvolvimento solicitar o esclarecimento ou uma decisão rápida sobre algum item do Sprint Backlog sempre que necessário, o Product Owner se coloca disponível e acessível. De outra forma, ele estaria gerando um impedimento ou estimulando o Time de Desenvolvimento a tomar decisões sobre o produto que não lhe cabem.

É também saudável o Product Owner interagir com o Time de Desenvolvimento durante o Sprint para, juntos, refinarem e prepararem itens no alto do Product Backlog para o Sprint seguinte. Enquanto diversos Times de Scrum realizam essa atividade durante a reunião de Sprint Planning, cada vez mais times utilizam as sessões de Refinamento do Product Backlog, realizadas durante o Sprint, para essas atividades (veja “[23. Refinamento do Product Backlog](#)”).

No entanto, interferências do Product Owner no planejamento já realizado para a Sprint desviam o foco do Time de Desenvolvimento e podem comprometer a realização do trabalho. Assim, o Time de Desenvolvimento, amparado pelo ScrumMaster, rejeita durante o Sprint alterações no Sprint Backlog que modifiquem a Meta do Sprint já acordada.

Da mesma forma, o Product Owner que busca informações sobre o andamento do trabalho em direção à Meta do Sprint pode ter o intuito de exercer pressão sobre o Time de Desenvolvimento, interferindo em sua auto-organização. No entanto, o Time de Desenvolvimento pode buscar o *feedback* do Product Owner no decorrer do Sprint sobre itens em andamento ou já prontos, o que pode ajudá-lo a melhorar a sua entrega e a atingir a Meta da Sprint.

6.2.6 Aceita ou rejeita a entrega do Time de Desenvolvimento

O Product Owner tem a responsabilidade de aceitar ou rejeitar os entregáveis que serão demonstrados para os clientes do projeto e demais partes interessadas no final de cada Sprint, durante a reunião Sprint Review. Essa atividade pode ser realizada na própria reunião, mas o Time de Scrum pode preferir preparar-se antes e chegar à reunião já alinhado sobre os resultados do Sprint.

Apenas os itens prontos de acordo com a Definição de Pronto serão aceitos pelo Product Owner, o que inclui, entre outros critérios, que cada item passe pelos Critérios de Aceitação acordados especificamente para ele e tenha qualidade suficiente para ser entregue. Na reunião de Sprint Review, o Time de Desenvolvimento demonstra para os presentes apenas esses itens prontos.

O Product Owner verifica se esses entregáveis são suficientes para cumprir a Meta estabelecida junto ao Time de Desenvolvimento para o Sprint, e comunica claramente se essa Meta do Sprint foi cumprida ou não.

6.3 COMO É O PRODUCT OWNER?

6.3.1 Único

O papel de Product Owner é exercido por apenas uma pessoa em um Time de Scrum. A existência de mais de um Product Owner interagindo com o Time de Desenvolvimento geraria dúvidas e conflitos sobre quem tem o poder de decisão sobre o produto em desenvolvimento. Quando houvesse discordâncias, por exemplo, o Time de Desenvolvimento não saberia que decisões sobre o produto seguir, e assim o processo decisório sobre o produto se tornaria desnecessariamente lento e confuso.

De forma similar, a existência de Product Owners substitutos, que compareceriam quando o Product Owner não pudesse estar presente, confundiria o Time de Desenvolvimento e o próprio trabalho de definição do Produto. Como na situação anterior, as expectativas e interpretações diversas de diferentes Product Owners poderiam tornar as Metas dos Sprints difíceis de serem compreendidas e, finalmente, atingidas. Um revesamento de Product Owners ao longo do projeto também poderia confundir o Time de Desenvolvimento e trazer problemas parecidos.

Ausências do Product Owner por doença ou férias, no entanto, são situações de exceção e, nesses casos, dificilmente há alguma solução diferente de simplesmente substituí-lo provisoriamente.

Não existem também, ao mesmo tempo, um Product Owner do cliente e outro

da organização que está gerando o produto, trabalhando como um representante ou como um canal de comunicação. O Product Owner é único: ou ele é uma pessoa escolhida no cliente ou ele é uma pessoa escolhida na organização.

O Product Owner também não é um comitê ou um departamento da organização. Grupos de pessoas, mesmo que tomem decisões únicas e centralizadas, em geral demoram um tempo demasiadamente longo para chegar a elas.

Com o Product Owner único, o Time de Desenvolvimento enxerga apenas um foco para a tomada de decisões sobre o produto, para o esclarecimento de dúvidas e perante o qual se comprometerá com a Meta do Sprint.

O Product Owner é a única pessoa responsável por gerenciar o Product Backlog. Suas decisões sobre as necessidades de negócios com relação ao produto devem ser respeitadas, de forma que ninguém além dele pode mudar as prioridades por ele estabelecidas. Para realizar seu trabalho, no entanto, o Product Owner é aconselhado, influenciado e conta com o apoio de outras pessoas ou equipes envolvidas no projeto além dos próprios clientes do projeto e Time de Desenvolvimento. Pode também existir, por exemplo, uma equipe de analistas de negócios que auxilie o Product Owner em seu trabalho.

Em projetos maiores, pode haver a necessidade de haver mais de um Time de Scrum trabalhando no mesmo produto. Nesses casos, apenas um Product Owner dificilmente terá a capacidade e disponibilidade suficientes para realizar todo trabalho necessário para fazer parte de todos os Times de Scrum do projeto. Ele conseguirá, talvez, participar de dois ou três times. Nesses casos, o Product Owner continua sendo único para um Time de Scrum: cada Time de Scrum do projeto possuirá apenas um Product Owner, que em geral será responsável por uma área definida do produto em desenvolvimento, e todos os Times de Scrum trabalharão no mesmo Product Backlog. Os diferentes Product Owners do projeto coordenarão seu trabalho e tomarão decisões sobre o produto em conjunto. Esse assunto, no entanto, está fora do escopo deste livro e não será aqui discutido em detalhes.

6.3.2 Disponível para o trabalho no projeto

O trabalho do Product Owner envolve principalmente:

- estar presente na reunião de Sprint Planning. O Product Owner define, junto com o Time de Desenvolvimento, qual a Meta a ser atingida no Sprint e qual o trabalho a ser realizado para se atingir essa Meta;

- colocar-se acessível e disponível para tomar decisões e esclarecer dúvidas do Time de Desenvolvimento sobre o produto, quando solicitado;
- interagir com o Time de Desenvolvimento para que, juntos, preparem itens para o próximo Sprint;
- estar presente na reunião de Sprint Review para, junto com o Time de Desenvolvimento, obter *feedback* dos clientes do projeto e demais partes interessadas sobre o trabalho realizado no Sprint, para verificar se esse trabalho está pronto de acordo com a Definição de Pronto e para estabelecer se a Meta do Sprint com a qual o Time de Desenvolvimento se comprometeu foi atingida ou não;
- estar presente na reunião de Sprint Retrospective para ajudar o Time de Scrum a melhorar seu trabalho e a se tornar mais produtivo;
- interagir frequentemente com os clientes e demais partes interessadas ao longo de todo o projeto para entender suas necessidades de negócios, obter seu *feedback* sobre o trabalho já entregue e atualizar o Product Backlog com as novas informações.

Todas essas atividades podem demandar do Product Owner uma dedicação considerável de tempo, embora não necessariamente exclusiva. Assim, um Product Owner ocupado e ausente pode não ter disponibilidade suficiente para realizar seu trabalho, o que poderá trazer consequências desastrosas para o projeto.

Observadas todas essas questões envolvidas em seu trabalho, o Product Owner pode trabalhar, ao mesmo tempo, em mais de um projeto. Ou seja, ele pode fazer parte de mais de um Time de Scrum. No entanto, dependendo da complexidade do produto e do projeto, essa pode se tornar uma tarefa muito difícil.

6.3.3 Representativo para o produto

O Product Owner define qual o produto a ser desenvolvido, Incremento a Incremento. Ele é, de fato, o gerente do produto em um projeto que utiliza Scrum.

Para tomar as decisões necessárias, não se espera que o Product Owner saiba tudo sobre o produto ou sobre o negócio dos clientes. Afinal, os detalhes de um novo produto são “alvos em movimento”, já que ele se desenvolve a partir do *feedback* frequente dos clientes do projeto e das demais partes interessadas. Espera-se que o Product Owner possua apenas o conhecimento necessário para tomar decisões

suficientemente boas, que de qualquer forma serão logo revistas na próxima reunião de Sprint Review. Um Product Owner que tome decisões erradas, no entanto, pode ser muito prejudicial para o projeto, gerando um grande desperdício com grandes correções de rumo. A qualidade das decisões do Product Owner torna-se visível rápida e frequentemente por meio das reuniões de Sprint Review.

O Product Owner não é um intermediário para os clientes do projeto, mas sim aquele que de fato define o produto a ser desenvolvido, com o propósito de atender as necessidades desses clientes. Assim, ele deve possuir o poder de tomar decisões que considere as mais adequadas em cada momento.

Quando há alguma dúvida ou questão a ser resolvida, o Product Owner, se considera necessário, consulta os clientes do projeto ou quaisquer outros especialistas relevantes. No entanto, é importante que essa consulta seja feita antecipadamente, sempre que possível, para não que não se crie um impedimento para o Time de Desenvolvimento, que em outro caso ficaria esperando por uma decisão.

CAPÍTULO 7

ScrumMaster

7.1 QUEM É O SCRUMMASTER?

O ScrumMaster trabalha para facilitar e potencializar o trabalho do Time de Scrum. Ou seja, utilizando-se de seu conhecimento de Scrum, habilidade de lidar com pessoas, técnicas de facilitação e outras técnicas, o ScrumMaster ajuda o Product Owner e Time de Desenvolvimento a serem mais eficientes na realização do seu trabalho.

Para realizar esse trabalho, o ScrumMaster:

- facilita o trabalho do Time de Scrum, de forma que seus membros se auto-organizem para que juntos desenvolvam o produto, comuniquem-se efetivamente e busquem continuamente melhorar seus processos de trabalho, realizando-o com qualidade e produtividade; além de facilitar os eventos do Scrum;
- remove ou gerencia a remoção dos impedimentos que atrapalham o trabalho

do Time de Desenvolvimento e ajuda a prevenir que os impedimentos aconteçam, quando possível;

- promove as mudanças organizacionais necessárias para que o Time de Scrum possa realizar seu trabalho com efetividade;
- assegura que o Scrum seja compreendido e adequadamente utilizado pelo Time de Scrum;

O ScrumMaster é:

- competente em *soft skills*, ou seja, possui competências comportamentais e pessoais como comunicação, facilitação e política. Ele é também corajoso, proativo e autoconfiante para realizar as mudanças necessárias, remover impedimentos e proteger o trabalho do Time de Desenvolvimento;
- presente sempre que o Time de Scrum necessitar dele, para observar, identificar, criar visibilidade e ajudar a resolver problemas, para remover impedimentos e para proteger o Time de Desenvolvimento de interrupções;
- suficientemente neutro, visando aumentar a responsabilidade e capacidade do Time de Scrum de resolver seus próprios problemas e chegar a suas próprias soluções.

7.2 O QUE FAZ O SCRUMMASTER?

7.2.1 Facilita o trabalho do Time de Scrum

O ScrumMaster é um facilitador para o trabalho do Time de Scrum, ou seja, dos membros do Time de Desenvolvimento e do Product Owner. Ele promove a autonomia, a boa relação de trabalho e a comunicação entre os membros do Time de Scrum no seu dia a dia, de forma a se tornarem cada vez mais efetivos.

Facilitador hábil

Na definição clássica de Roger Schwarz, “*facilitação de um grupo é um processo pelo qual uma pessoa cuja escolha é aceitável para todos os membros do grupo, que é suficientemente neutra e que não possui autoridade considerável no processo decisório*

do grupo, diagnostica e intervém para ajudar o grupo a melhorar como ele identifica e resolve problemas e toma decisões, para aumentar a efetividade do grupo” (Schwarz, 2002). A partir dessa definição, Schwarz criou o modelo do “facilitador hábil”.

Listamos a seguir algumas características do trabalho do ScrumMaster, enquanto facilitador, utilizando o modelo criado por Schwarz. Em particular, adotamos aqui a abordagem de facilitação desenvolvimentista, na qual o facilitador ensina o time a melhorar seus processos de trabalho e a depender cada vez menos do facilitador.

O facilitador habilita o time a aumentar sua efetividade. A principal atribuição do ScrumMaster enquanto facilitador é ajudar o Time de Scrum a aumentar a sua efetividade. Para tal, ele ajuda a habilitar os membros do Time de Desenvolvimento e o Product Owner a melhorarem:

- **seu processo:** como trabalham juntos, ou seja, como se comunicam, como é o seu processo criativo, como identificam e resolvem problemas, como lidam com conflitos e como se relacionam com seu entorno;
- **sua estrutura:** características relativamente estáveis importantes para o funcionamento do grupo, como a existência de metas claras a serem alcançadas, de uma composição do time adequada para a realização de suas tarefas, de tarefas motivadoras, de valores e crenças consistentes com um time efetivo, de normas ou acordos explícitos derivados desses valores e crenças e de tempo suficiente para realizar seu trabalho e para melhorar sua forma de trabalhar;
- **seu contexto:** como o contexto no qual está inserido o time (por exemplo, sua organização ou setor) influencia a sua efetividade, como a existência de objetivos organizacionais claros, de uma cultura organizacional que apoie o uso do Scrum e de práticas de gestão que tornem o time mais efetivo, da premiação de comportamentos do time (e não do indivíduo) consistentes com os objetivos do time, de informações necessárias para realizar o trabalho, de *feedback* sobre o seu trabalho, de acesso a treinamento e consultoria que se façam necessários para possibilitar a resolução de problemas, aumentar seus conhecimentos e desenvolver habilidades necessárias para realizar seu trabalho, de acesso ao material e tecnologia necessários e de um ambiente físico adequado).

O ScrumMaster, enquanto facilitador, intervém diretamente no processo, estrutura e contexto do Time de Scrum para ajudá-lo a se tornar mais efetivo. Mas, ao

mesmo tempo, o ScrumMaster estimula o Time de Scrum a refletir sobre como pode melhorar seu processo, estrutura e contexto e o ensina a desenvolver as habilidades necessárias para atuar por si só nos mesmos, pois a responsabilidade de aumentar sua efetividade pertence ao próprio time.

Essas mudanças, no entanto, só podem ser realizadas se o Time de Scrum possuir a autoridade necessária para modificá-los e se todos os seus membros compartilharem a responsabilidade por essas mudanças. Embora o Time de Scrum não possua controle direto sobre seu contexto, ele pode influenciá-lo e provocar mudanças que o possibilitem a se tornar mais efetivo.

O facilitador é neutro. O ScrumMaster, enquanto facilitador, é uma pessoa suficientemente neutra, que tem como objetivo aumentar a responsabilidade e capacidade do grupo de resolver seus próprios problemas. Assim, ele não interfere diretamente no conteúdo das discussões do grupo. Para que essa neutralidade seja possível, o ScrumMaster preferencialmente não exerce também o papel de membro do Time de Desenvolvimento ou Product Owner, que têm suas opiniões e interesses e, assim, são invariavelmente parciais. O ScrumMaster, portanto, trabalha para o Time de Scrum como um todo.

O facilitador habilita o time a aumentar sua autonomia. Uma parte importante do trabalho do facilitador é habilitar o time a realizar suas próprias escolhas (e decisões) a partir de informações suficientes: é o que chamamos de escolhas livres e informadas. Ao realizar escolhas livres e informadas, os membros do time naturalmente tornam-se comprometidos com essas escolhas e, ao mesmo tempo, revisam-nas e as mantêm atualizadas. O trabalho do ScrumMaster, enquanto facilitador, é chave para possibilitar a auto-organização do Time de Desenvolvimento.

Um elemento importante para possibilitar as escolhas livres e informadas é a transparência. Para assegurá-la, o ScrumMaster nunca deve fazer arranjos com determinados membros ou com terceiros para ocultar informações do Time de Scrum ou de alguma parte dele, ou para se comportar de uma determinada forma ou de outra. Ao contrário, o ScrumMaster deve ajudar o Time de Scrum a garantir a visibilidade das informações disponíveis.

A princípio, o time é mais efetivo se é internamente comprometido com suas escolhas. No entanto, mesmo que o ScrumMaster esteja realizando a facilitação de forma efetiva, o time pode realizar escolhas ruins, já que suas escolhas são livres, ainda que informadas. Mesmo que o ScrumMaster identifique que uma determi-

nada escolha não é apropriada, sua intervenção não é recomendável. Seu papel é o de estimular o time a refletir sobre o resultado de suas escolhas e as ajustar de acordo. Ao intervir com suas próprias opiniões e tomar a decisão pelo time, o ScrumMaster estará reduzindo tanto a autonomia do time quanto sua credibilidade enquanto facilitador.

Para aumentar sua autonomia do Time de Scrum e desenvolver sua efetividade, as intervenções do ScrumMaster, enquanto facilitador, têm sempre o objetivo de reduzir a dependência do Time de Scrum no próprio ScrumMaster. Assim, podemos afirmar que o ScrumMaster trabalha em direção a se tornar cada vez menos necessário, até mesmo habilitando os membros do Time de Scrum a atuarem como facilitadores eles mesmos.

O facilitador não é intermediário ou representante. O ScrumMaster também não atua como um intermediário entre membros do Time de Scrum ou entre o Time de Scrum e pessoas externas a ele. Ao contrário, ele estimula o time a desenvolver as habilidades de se comunicar e lidar diretamente com quem se fizer necessário.

O ScrumMaster também não atua como representante do Time de Scrum ou de qualquer de seus membros diante de outros membros ou de pessoas externas. Dessa forma, não é o ScrumMaster, por exemplo, que reporta ao resto da organização os resultados do trabalho ou o desempenho do Time de Scrum ou de algum de seus membros. Ele não tem autoridade de utilizar qualquer informação obtida a partir da facilitação para influenciar decisões externas sobre membros do time como, por exemplo, bonificações ou punições. Caso o fizesse, a confiança do Time de Scrum no ScrumMaster ficaria comprometida, assim como sua neutralidade e consequentemente sua eficiência como facilitador. Esse tipo de informação, caso necessária, é fornecida diretamente pelo Time de Scrum.

Da mesma forma, o ScrumMaster não atua como intermediário entre os membros do Time de Desenvolvimento e Product Owner. Ele, ao contrário, estimula que se comuniquem diretamente.

O facilitador é um especialista no processo. Como um especialista no processo, o ScrumMaster sabe quais elementos mais contribuem para tornar o Time de Scrum mais eficiente. Uma parte importante desses elementos são as regras do Scrum, que devem ser corretamente compreendidas e utilizadas pelo Time de Scrum. Assim, o ScrumMaster identifica quaisquer comportamentos disfuncionais do time e desvios no uso do Scrum utilizando sua expertise no processo.

Embora não interfira diretamente no conteúdo das discussões do grupo, quando uma discussão envolve processos do time ou organizacionais, o ScrumMaster pode nesse momento deixar sua neutralidade de lado – justamente por ser um especialista no processo – e então interferir nesse conteúdo.

Facilitador de eventos

Michael Wilkinson, no livro “*The Secrets of Facilitation: The S.M.A.R.T. Guide for Getting Results With Groups*”, define uma sessão facilitada como “*um encontro altamente estruturado no qual o facilitador guia os participantes por meio de uma série de passos pré-definidos para que cheguem a um resultado que é criado, compreendido e aceito por todos os participantes*”. Ainda segundo o autor, técnicas de facilitação são apropriadas sempre que um grupo necessita de compreensão comum e aceitação para obter sucesso em se chegar a um determinado resultado (Wilkinson, 2004).

Além de facilitar o dia a dia de trabalho do Time de Scrum, o ScrumMaster tem a importante função de atuar como um facilitador nos eventos do Scrum. Ele estimula os envolvidos a participarem ativamente das discussões, ajuda-os a manter o foco nos objetivos do evento e a chegarem a suas próprias conclusões.

Podemos interpretar qualquer um dos eventos do Scrum como uma sessão facilitada, que possui um caminho específico a se percorrer e um resultado específico a ser alcançado. O resultado esperado, por exemplo, da reunião de Sprint Planning é um plano para o Sprint, ou seja, a Meta do Sprint e o Sprint Backlog. O resultado esperado da reunião de Sprint Review é o *feedback* dos clientes e demais partes interessadas sobre o Incremento do Produto gerado pelo Time de Desenvolvimento durante o Sprint.

As características de um facilitador de eventos descrito por Wilkinson são bastante similares a do “facilitador hábil”, descrito anteriormente. De acordo com o autor, o facilitador de eventos não interfere no conteúdo das discussões, nem oferece soluções ou suas próprias opiniões. Ele, na realidade, serve ao grupo guiando os participantes por meio de passos pré-definidos para que alcancem resultados específicos, utilizando seu conhecimento de dinâmicas de grupo e dos passos do processo.

O facilitador cria um ambiente em que os participantes se sentem à vontade para contribuir. O facilitador em geral responde a questionamentos com perguntas, e sabe realizar as perguntas certas, que estimularão os participantes a chegarem a suas próprias conclusões. O facilitador procura prevenir comportamentos disfuncionais que possam ocorrer durante a sessão, ajudando o grupo, por exemplo, a estabelecer regras básicas de conduta. Ele reconhece e lida com comportamentos disfuncionais

quando ocorrem, buscando soluções a partir de suas causas. O facilitador ajuda o grupo a manter seu foco nos resultados da reunião e a construir um consenso em torno das soluções geradas.

Ao final, os resultados da sessão facilitada devem ter sido criados, compreendidos e aceitos por todos os participantes.

Liderança no time auto-organizado

Times auto-organizados são utilizados em diferentes áreas de trabalho, e não apenas no desenvolvimento de *software*.

Embora possa parecer paradoxal, a prática mostra que times auto-organizados muitas vezes possuem líderes. Sua função primária, no entanto, é a de facilitar a auto-organização, e não a de gerenciar a equipe. O líder de um time auto-organizado, portanto, tem o importante papel de *lidar o time em direção a que ele lidere a si mesmo*.

O ScrumMaster possui muitas das características de um líder de time auto-organizado. O líder facilita a auto-organização principalmente ao encorajar o time à auto-observação e à autoavaliação, de forma que o time monitore, torne consciente e avalie seu desempenho; e do encorajamento do time ao autorreforço, de forma que seus membros demonstrem entre si que valorizam um bom trabalho feito por um colega e valorizam o alto desempenho do time. Essas ações de encorajamento são mais efetivas quando o líder consegue criar uma relação de confiança e de atenção com os membros do time, o que facilita que consiga levantar informações que determinem os tipos de ações que devem ser encorajadas.

Para serem efetivos, líderes de times auto-organizados realizam tanto funções voltadas internamente quanto voltadas externamente ao time, atuando na organização que o contém e atravessando constantemente a fronteira entre os dois lados. O líder efetivo é capaz de construir relacionamentos tanto com membros do time quanto com outros indivíduos ou unidades da organização que podem afetar o desempenho do time (como a alta gerência, a manutenção e outros grupos), e buscar entender ambas as perspectivas.

Nesse trabalho, o líder de um time auto-organizado não dá ordens: ele na verdade obtém acesso a informações-chave de ambos os contextos por meio do bom relacionamento, da percepção política e da habilidade em realizar perguntas. Ele utiliza essas informações-chave para alinhar as necessidades do time com as da organização e vice-versa, de forma a fazer com que tanto o time quanto a organização se comportem de modo a facilitar a efetividade de ambos. Esse trabalho envolve,

por exemplo, evitar que uma decisão tomada pelo time, que de fato cabe ao time, seja rejeitada pela organização. Esse trabalho permite ao líder promover o poder de ação e de decisão do time, o que o leva a assumir uma maior responsabilidade por seus resultados.

Dos trabalhos de Cummings (1978), Manz & Sims (1987) e Druskat & Wheeler (2003).

7.2.2 Remove impedimentos

Impedimentos ameaçam o cumprimento da Meta do Sprint, pois dificultam significativamente ou obstruem o trabalho do Time de Desenvolvimento. O ScrumMaster é o responsável pela resolução desses impedimentos, removendo-os ele mesmo ou mobilizando as pessoas e os recursos necessários para tal.

Um impedimento tipicamente acontece em um trabalho que já foi iniciado pelo Time de Desenvolvimento e, assim, gera uma espera sobre um item ou itens que têm prioridade de desenvolvimento naquele momento. Por essa razão, uma vez identificado pelo Time de Desenvolvimento, o impedimento é imediatamente sinalizado para o ScrumMaster, que por sua vez toma ações rápidas e efetivas para a sua remoção. É importante que as soluções adotadas atuem diretamente sobre as possíveis causas raízes, ao invés de se utilizarem soluções paliativas.

O ScrumMaster estimula o Time de Desenvolvimento a identificar se há impedimentos por vir. Assim, quando o Time de Desenvolvimento se antecipa a um impedimento, o ScrumMaster trabalha imediatamente para ajudar a evitar que o mesmo aconteça.

No entanto, deve-se saber distinguir problemas e questões do dia a dia de impedimentos no trabalho, sobre os quais atua o ScrumMaster. A resolução de problemas está ao alcance e no contexto de atuação do Time de Desenvolvimento e por essa razão são tratados pelo próprio Time de Desenvolvimento, ainda que com a ajuda do ScrumMaster.

Natureza dos impedimentos e sua resolução

De acordo com sua natureza, os impedimentos podem ser classificados em organizacionais, básicos, administrativos ou de nível de serviço (adaptado de Pimentel, 2009).

Organizacionais. Ocorrem quando a própria estrutura ou funcionamento da organização os impõe ao Time de Desenvolvimento. Podem acontecer em pelo menos três situações distintas:

- o Time de Desenvolvimento necessita mas não obtém a ajuda ou intervenção de outra pessoa, equipe ou área da organização. Nesses casos, o ScrumMaster utiliza o acesso e influência que construiu e está construindo na organização para buscar, o mais rapidamente possível, o apoio necessário para o Time de Desenvolvimento realizar seu trabalho, e monitorar se esse apoio foi realmente obtido.

Esse tipo de impedimento se configura, por exemplo, quando o equipamento de trabalho de um membro do Time de Desenvolvimento apresenta defeito, mas a equipe de suporte não resolve o problema em tempo hábil;

- outra pessoa, equipe ou área da organização intervém no trabalho do Time de Desenvolvimento e demanda sua ajuda ou de algum de seus membros, assim desviando seu foco da Meta do Sprint. O ScrumMaster tem o importante papel de proteger o Time de Desenvolvimento de interferências externas que coloquem em risco a Meta do Sprint. De forma geral, o modo mais efetivo de se realizar essa proteção em ambientes hierárquicos é utilizando política e o ensino do Scrum e de seus benefícios.

Esse tipo de impedimento ocorre, por exemplo, quando um membro da organização em uma posição hierarquicamente superior aborda diretamente um membro do Time de Desenvolvimento para realizar alguma tarefa externa ao projeto;

- questões de cunho político ou burocrático cultivadas pela organização obstruem o trabalho do Time de Desenvolvimento. Como um agente de mudanças organizacional, o ScrumMaster trabalha para modificar as práticas organizacionais em benefício da eficiência do Time de Desenvolvimento em seu trabalho. Ele, por exemplo, pode se unir a ScrumMasters de outros times para realizar as mudanças organizacionais necessárias.

A adoção de métodos ou práticas pesadas que conflitem com os valores e princípios Ágeis, como as que geram uma produção excessiva de documentação, constitui um exemplo desse tipo de impedimento.

Básicos. São impedimentos que ocorrem pela falta de um ou mais elementos essenciais à realização do trabalho em algum determinado momento. O ScrumMaster trabalha para garantir que o Time de Desenvolvimento possua acesso aos meios necessários para realizar suas tarefas.

Esse tipo de impedimento inclui, por exemplo, a falta de ferramentas ou equipamentos necessários para a realização do trabalho, a falta de um ambiente adequado ou de elementos essenciais nesse ambiente (como cadeiras ou mesas, por exemplo) e até mesmo a impossibilidade de acesso a dados ou informações necessários.

Administrativos: São impedimentos provenientes de ocorrências administrativas, como absenteísmo (atrasos, licenças, folgas, faltas por motivos diversos etc.), demissões e restrições de horários de trabalho. Essas questões estão muitas vezes ligadas a áreas em que atua o ScrumMaster, como políticas organizacionais ou problemas intra-equipe que podem estar gerando desmotivação.

Nível de serviço: Aqui são identificados impedimentos oriundos de problemas na sustentação de algum serviço em operação. O ScrumMaster trabalha para criar visibilidade para o problema e monitora sua resolução junto à equipe que tem a atribuição de resolvê-lo, sendo muitas vezes necessário estimular o envolvimento do Time de Desenvolvimento nesse trabalho.

Exemplos desse tipo de impedimento incluem, para projetos de desenvolvimento de *software*, os erros em ambientes de produção e problemas com o servidor de aplicações, com o servidor de integração ou com ferramentas diversas de apoio.

7.2.3 Promove mudanças organizacionais necessárias

O ScrumMaster atua como um agente de mudanças na organização onde está inserido o Time de Scrum. Ou seja, ele trabalha para promover as mudanças no contexto do Time de Scrum necessárias para que a equipe possa realizar seu trabalho com mais efetividade. Essas mudanças podem envolver, por exemplo, desde educar em Scrum pessoas de quem o Time de Scrum depende para realizar seu trabalho, de forma que possam colaborar ao invés de gerar impedimentos, até trabalhar para modificar regulamentos que gerem desperdícios ou estruturas ineficientes que atrapalhem o trabalho do Time de Scrum.

Um ScrumMaster efetivo é persistente e perseverante, e não aceita que não consegue modificar a organização em que trabalha. Por mais difícil e lento que esse

trabalho possa parecer em algumas organizações, realizar as mudanças necessárias é uma parte essencial das atribuições do ScrumMaster.

O ScrumMaster é capaz transitar entre o Time de Scrum e a organização que o contém, construindo relacionamentos com pessoas-chave ou unidades da organização que podem afetar o desempenho do Time de Scrum. A partir do bom relacionamento, da percepção política e da habilidade em realizar perguntas, ele obtém acesso a informações-chave, e as utiliza para alinhar as necessidades e objetivos do Time de Scrum e os da organização que o contém e vice-versa (como visto anteriormente em “Liderança no time auto-organizado”).

7.2.4 Garante o uso do Scrum

O ScrumMaster é responsável por garantir que os valores, práticas e regras do Scrum estejam sendo entendidos e seguidos por todo o Time de Scrum. Ele ensina o Scrum para o Time de Desenvolvimento, para o Product Owner e para demais pessoas da organização que julgue necessário, de forma que apoiem o trabalho do Time de Scrum. O ScrumMaster fica atento a desvios na prática do Scrum e toma medidas que considere necessárias para estimular o Time a corrigir esses desvios.

É responsabilidade do ScrumMaster que todos os eventos do Scrum aconteçam, e que os horários e *timeboxes* sejam respeitados. É também de sua responsabilidade que o Time de Desenvolvimento esteja atualizando o Sprint Backlog e que esteja monitorando seu progresso em direção à Meta do Sprint. O ScrumMaster também assegura-se de que o Product Owner esteja atualizando e preparando o Product Backlog e que, para fazê-lo, esteja interagindo com os clientes do projeto e com o Time de Desenvolvimento, de forma que se chegue à Sprint Planning com os itens mais prioritários preparados para entrarem em desenvolvimento.

Um bom ScrumMaster, no entanto, não age como polícia ou fiscal dos processos, forçando os envolvidos a utilizarem o Scrum corretamente ou mesmo realizando partes do trabalho por eles para que o bom uso do Scrum aconteça. Na realidade, como descrito anteriormente, um dos importantes objetivos do ScrumMaster é tornar-se cada vez menos necessário, aumentando a autonomia do Time de Scrum. Com esse propósito, o ScrumMaster ensina e habilita o Time de Desenvolvimento e Product Owner a utilizarem o Scrum, de forma que cada vez menos dependam dele para que os valores, práticas e regras do Scrum sejam seguidos e utilizados.

O ScrumMaster poderia, por exemplo, avisar todos os dias ao Time de Desenvolvimento quando é hora de realizar a sua reunião diária. Ao agir dessa forma, no

entanto, ao invés de habilitá-lo e torná-lo independente, o ScrumMaster pode estar levando o Time de Desenvolvimento a sempre depender dele para avisá-lo da reunião, o que provavelmente acontecerá até o final do projeto. O ScrumMaster fará melhor seu trabalho se, utilizando diferentes técnicas, for capaz de mostrar e convencer o Time de Desenvolvimento da importância de realizar a reunião sempre na mesma hora e no mesmo local, de forma que crie o hábito e se auto-organize para fazê-lo.

7.3 COMO É O SCRUMMASTER?

7.3.1 Competente em soft skills

Para desempenhar bem o papel de ScrumMaster, o indivíduo deve possuir competências comportamentais e pessoais que lhe permitam realizar o seu trabalho. Essas competências são chamadas de *soft skills*.

De forma geral, um ScrumMaster tímido ou reservado encontra dificuldades para desempenhar o seu papel. A facilitação do trabalho do Time de Scrum, parte essencial do trabalho do ScrumMaster, inclui promover a detecção e visibilidade dos problemas e assegurar a boa relação entre os membros do Time de Desenvolvimento e entre esses e o Product Owner.

O ScrumMaster utiliza-se de boa comunicação e de voz ativa, mas sem comandar, para realizar esse trabalho. Ele também utiliza-se de suas habilidades de negociação e de política para estimular a resolução de conflitos e promover a tomada de decisão do Time de Scrum em direção ao consenso. Um ScrumMaster autoconfiante, corajoso e proativo atua como um agente de mudanças na organização e, utilizando-se de persistência e perseverança, promove as mudanças necessárias para que o Time de Scrum possa entregar valor para seus clientes.

O ScrumMaster utiliza-se dessas mesmas habilidades para se articular em diferentes níveis organizacionais com o propósito de realizar a remoção de impedimentos e de defender o Time de Desenvolvimento de interferências externas e de mudanças que possam afetar a Meta do Sprint.

A seguir, podemos ver exemplos de *soft skills* desejáveis em um ScrumMaster:

- habilidade de comunicação;
- saber ouvir;
- flexibilidade e adaptabilidade;

- capacidade de negociação e habilidade política;
- capacidade de resolução de problemas e pensamento crítico;
- habilidade de ensinar, por meio de *coaching* ou *mentoring*;
- habilidade de facilitação;
- ser colaborativo;
- autoconfiança;
- autoconsciência;
- paciência;
- persistência e perseverança;
- sociabilidade;
- gestão de seu tempo;
- atitude positiva.

7.3.2 Presente

O ScrumMaster presente toma ciência imediatamente de obstáculos que estejam impedindo o trabalho do Time de Desenvolvimento e pode rapidamente tomar ações para removê-los, antes que ameacem a Meta do Sprint. O ScrumMaster atento a desvios na prática do Scrum pode agir para corrigi-los, ensinando e habilitando o Time de Desenvolvimento e o Product Owner a utilizarem o Scrum corretamente. O ScrumMaster que acompanha o trabalho do Time de Scrum é capaz de detectar comportamentos disfuncionais, conflitos ou problemas de relacionamento entre os membros do Time de Desenvolvimento ou entre o Time de Desenvolvimento e o Product Owner, e atuar o mais rápido possível para ajudar a resolver esses problemas. Ele presente pode proteger o Time de Desenvolvimento de interferências externas que ameacem a Meta do Sprint, o que pode acontecer em qualquer momento. Além disso, ele atua como um facilitador nos eventos do Scrum e não há como fazê-lo sem estar neles presente.

Em suma, o ScrumMaster idealmente se faz presente sempre que o Time de Scrum necessitar dele.

Para muitos times, o ScrumMaster estar presente sempre que necessário pode significar um trabalho em tempo integral. Nesses casos, embora seja comum o uso de ScrumMasters em tempo parcial ou trabalhando em mais de um projeto ao mesmo tempo, um ScrumMaster ausente pode trazer sérios problemas para o projeto, pois ele será necessário em momentos em que não poderá estar presente. Ao invés de economizar dinheiro, a organização estará desperdiçando a oportunidade de ter alguém especializado em potencializar o trabalho do Time de Scrum.

De forma geral, o ScrumMaster em tempo integral é importante para times que trabalham juntos há pouco tempo, para times que estão começando a utilizar Scrum, para times jovens, para times inseridos em organizações hierárquicas e burocráticas e para times disfuncionais, ou seja, aqueles que enfrentam problemas graves na forma como operam.

7.3.3 Neutro

O ScrumMaster é um facilitador e, como tal, atua de forma mais neutra possível. Conforme descrito anteriormente, ele tem como objetivo aumentar a responsabilidade e capacidade do Time de Scrum de resolver seus próprios problemas e chegar a suas próprias conclusões. Com esse objetivo, ele evita interferir diretamente no conteúdo das discussões do Time de Scrum.

O ScrumMaster trabalha para o Time de Scrum como um todo, ou seja, para os membros do Time de Desenvolvimento e para o Product Owner. É um erro comum, no entanto, o ScrumMaster manter seu foco apenas no Time de Desenvolvimento, deixando assim o Product Owner desassistido em termos de facilitação. Ao agir dessa forma, o ScrumMaster assume incorretamente uma postura parcial diante do Product Owner e do resto da organização, enquanto alguém que trabalha apenas em prol do Time de Desenvolvimento.

Observam-se muitas adoções de Scrum onde o ScrumMaster trabalha parte de seu tempo como desenvolvedor, ou seja, como membro do Time de Desenvolvimento, e possui conhecimento técnico necessário para tal. Embora essa divisão de trabalho não seja proibida, a atuação dupla pode sacrificar a neutralidade do ScrumMaster e, assim, seu trabalho como facilitador. Em casos extremos, as sugestões ou decisões do ScrumMaster acabam por ter, para os membros do Time de Desenvolvimento, um peso maior do que suas próprias opiniões, que assim invariavelmente terminam por acatá-las. Esse tipo de situação pode ficar mais evidente em momentos de crise, embora não unicamente. Quando esse cenário ocorre, o ScrumMaster

está fazendo o oposto do que deveria ser seu trabalho: habilitar o Time de Desenvolvimento a aumentar sua autonomia. Além disso, principalmente devido à falta de tempo e diferenças no tipo de trabalho, observa-se que as atividades de remoção de impedimentos, facilitação e atuação como agente de mudanças organizacionais podem ser prejudicadas quando o ScrumMaster compartilha de uma atuação técnica.

No entanto, recomenda-se fortemente que o ScrumMaster nunca seja o Product Owner do projeto. Caso os papéis de ScrumMaster e Product Owner se concentrem em um único indivíduo, o conflito de interesses fica exacerbado. Não há ninguém para facilitar a relação entre o Product Owner e o Time de Desenvolvimento. Não há ninguém para defender o Time de Desenvolvimento se, disfuncionalmente, o próprio Product Owner buscar interferir em seu trabalho durante o Sprint. Nesses casos, a concentração de poder pode deixar esse indivíduo com características próximas às de um Gerente de Projetos tradicional, exercendo comando e controle sobre o Time de Desenvolvimento.

CAPÍTULO 8

Onde está o Gerente de Projetos?

Com Scrum, o trabalho de gestão do projeto é distribuído pelos seus três papéis, ou seja, pelos membros do Time de Scrum. Assim, não há um Gerente de Projetos como é tradicionalmente definido.

De forma sintética, identificamos na tabela a seguir alguns pontos-chave cujo gerenciamento cabe a diferentes papéis do Scrum:

Ponto-chave a ser gerenciado	Product Owner	Time de Desenvolvimento	ScrumMaster
Retorno sobre o investimento	X		
Necessidades/objetivos de negócios	X		
Clientes e demais partes interessadas	X		
Visão do Produto	X		
Releases	X		
Tarefas de desenvolvimento do produto		X	
Qualidade interna do produto		X	
Qualidade externa do produto	X	X	
Estimativas ou previsões		X	
Processos (funcionamento do Scrum)			X
Impedimentos no trabalho			X
Relacionamento e motivação do Time		X	X
Riscos	X	X	X
Comunicação	X	X	X

Essa divisão, na prática, significa que questões como escopo, custo, tempo, qualidade, risco e gestão do trabalho de desenvolvimento do produto não ficam sob a responsabilidade de um único papel centralizador. Ao contrário, elas são distribuídas entre o Product Owner, o Time de Desenvolvimento e o ScrumMaster.

Cada um em suas funções, eles trabalham de forma colaborativa e estão igualmente comprometidos com a satisfação dos clientes do projeto.

As funções de um gerente funcional, que é responsável por contratações, promoções, demissões, férias, planos de carreira etc., são muito específicas ao contexto de cada organização e, por essa razão, não são tratadas pelo *framework* Scrum.

Parte III

Artefatos do Scrum

O Scrum originalmente define o uso de apenas quatro artefatos: o Product Backlog, o Sprint Backlog, a Definição de Pronto e o Incremento no Produto. Adicionei à lista as metas de negócios – que incluem a Visão do Produto, a Meta de Release ou de Roadmap e a Meta do Sprint, adicionadas do Roadmap do Produto – os Gráficos de Acompanhamento do Trabalho (Release Burndown, Release Burnup e Sprint Burndown) e a Definição de Preparado, artefatos que podem ser importantes para o trabalho de um Time de Scrum.



CAPÍTULO 9

Product Backlog

9.1 O QUE É O PRODUCT BACKLOG?

O Product Backlog é uma lista de tudo o que se acredita que será desenvolvido pelo Time de Desenvolvimento no decorrer do projeto. Em cada momento, essa lista é atualizada, ordenada de acordo com a importância para os clientes do projeto e possui apenas o nível de detalhes que é possível de se ter.

O Product Backlog contém as necessidades ou objetivos de negócios dos clientes do projeto e demais partes interessadas e pode também conter melhorias a serem realizadas no produto, correções de problemas, questões técnicas, pesquisas que forem necessárias etc. Assim, tudo o que pode vir a ser desenvolvido para se alcançar a Visão do Produto é adicionado como um item do Product Backlog.

O TIME E O PRODUCT BACKLOG

O Product Backlog é a única fonte de trabalho a ser realizado no produto pelo Time de Desenvolvimento.

Esse trabalho também inclui as correções de problemas encontrados no sistema. Para o desenvolvimento de *software*, o uso de ferramentas de bug tracking, portanto, se sobreporia ao uso do Product Backlog e, assim, não é recomendado.

Os itens do Product Backlog são organizados pelo Product Owner de acordo com a ordem em que serão desenvolvidos pelo Time de Desenvolvimento, de forma a maximizar o retorno ao investimento dos clientes. Assim, os itens do topo do Product Backlog são colocados em desenvolvimento primeiro. Por essa razão, esses itens do topo são de granularidade mais fina e possuem uma descrição detalhada e, possivelmente, uma estimativa com maior precisão do tempo necessário para seu desenvolvimento. Na outra ponta, os itens da parte de baixo do Product Backlog são de granularidade mais grossa e possuem menos detalhes e estimativas mais grosseiras.

O Product Backlog está em constante evolução e, assim, nunca está terminado ou completo. Conforme o produto evolui, o Product Backlog é frequentemente modificado com a adição, subtração, reordenamento e modificação de seus itens. O produto evolui à medida que o ambiente muda e à medida que tanto clientes quanto Time de Desenvolvimento vão conhecendo e entendendo melhor esse produto que está sendo construído.

O Product Owner é o único responsável por gerenciar o Product Backlog, ou seja, ele é responsável por criar, atualizar, reordenar e dar visibilidade a ele. Outros contribuem e influenciam as opiniões do Product Owner, mas ninguém além dele pode modificar o Product Backlog. Para realizar esse trabalho, o Product Owner interage frequentemente com os clientes do projeto, com as demais partes interessadas e com o Time de Desenvolvimento.

O Scrum não prescreve nenhum formato ou padrão para o Product Backlog. O importante é que ele tenha o formato de itens em uma sequência, de forma que não haja itens com a mesma prioridade, e que seja facilmente reordenável. Existem softwares especializados que permitem essa manipulação do Product Backlog, mas uma planilha ou até mesmo notas adesivas em um quadro branco podem funcionar muito bem (veja a figura 9.1).

<input type="checkbox"/> Lorem ipsum dolor sit amet consectetur	3
<input type="checkbox"/> Adipiscing elit duis molestie nibh quis ultricies	5
<input type="checkbox"/> Adipiscing sem odio eleifend lectus ut luctus	2
<input type="checkbox"/> Augue sapien mattis	Lorem ipsum dolor sit amet consectetur
<input type="checkbox"/> Mollis elementum al	3
<input type="checkbox"/> Erat vestibulum cons	Adipiscing elit duis molestie nibh quis ultricies
<input type="checkbox"/> Eleifend id pretium t	5
<input type="checkbox"/> Iaculis lacinia neque	Adipiscing sem odio eleifend lectus ut luctus
<input type="checkbox"/> Quam imperdiet a pr	
<input type="checkbox"/> Fringilla varius ante r	Augue sapien mattis ante proin massa est
<input type="checkbox"/> Euismod tempor era	Mollis elementum aliquet eget fringilla temp
<input type="checkbox"/> A lectus arcu non feu	Erat vestibulum consequat purus in ante
	Eleifend id pretium tellus sagittis vestibulum
	Iaculis lacinia neque condimentum semper
	Quam imperdiet a proin pulvinar odio eu
	Fringilla varius ante nunc faucibus odio
	Euismod tempor erat purus at purus aliquam
	A lectus arcu non feugiat ipsum cras dapibus

PRODUCT BACKLOG

Figura 9.1: Três exemplos de formatos de Product Backlog

9.2 COMO É O PRODUCT BACKLOG?

O Product Backlog é ordenado, planejável, emergente e gradualmente detalhado.

9.2.1 Ordenado

Os itens do Product Backlog são ordenados de acordo com o grau de importância de seu desenvolvimento ou, colocando de outra forma, de acordo com o grau de importância de sua entrega pelo Time de Desenvolvimento. Esse ordenamento tem o propósito final de satisfazer os clientes do projeto ou, em outras palavras, garantir e maximizar o retorno sobre o investimento (ROI, em inglês) realizado por eles no projeto.

Na reunião de Sprint Planning, o Product Owner e o Time de Desenvolvimento selecionam um número de itens do topo do Product Backlog para serem colocados

em desenvolvimento. Como vemos na figura 9.2, os itens seguintes a esses poderão estar presentes ainda na Release atual, mas em futuros Sprints. Os itens mais abaixo no Product Backlog poderão estar presentes em Releases futuras. Quanto mais alto no Product Backlog o item estiver, mais chances de ser desenvolvido ele terá e mais cedo isso poderá acontecer. Por outro lado, quanto mais baixo no Product Backlog o item estiver, menor a sua ordem de desenvolvimento e, assim, menores são suas chances de ser realizado, já que mudanças podem fazê-lo perder o sentido ou ser significativamente modificado.

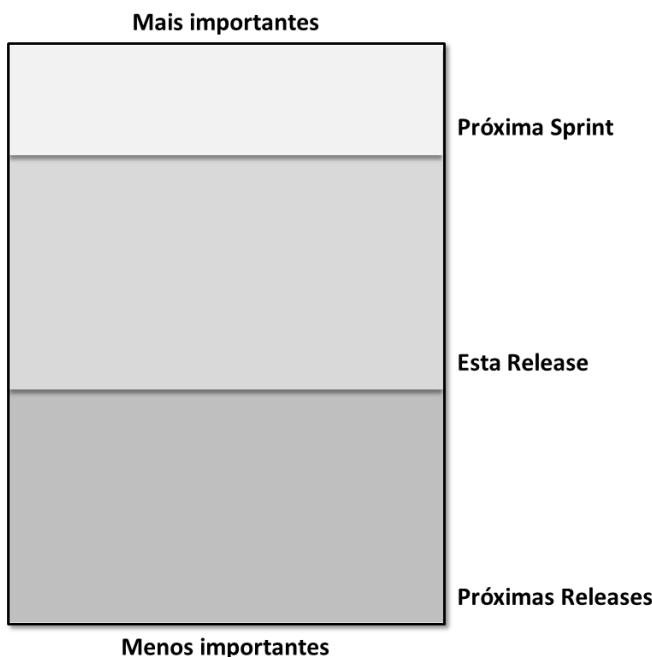


Figura 9.2: O Product Backlog é ordenado

A ordenação dos itens do Product Backlog é um trabalho contínuo, que acontece ao longo de todo o projeto. Seu objetivo é maximizar o retorno ao investimento realizado pelos clientes do projeto. Para realizar esse trabalho, o Product Owner considera o valor de negócios e o custo de desenvolvimento de cada item (ROI do item), as interdependências entre os itens, os riscos associados a cada item, a estratégia de negócios dos clientes e a estratégia de negócios da organização, entre outros fatores.

O trabalho de ordenação não é simples e ainda requer conhecimento do negócio,

das oportunidades e riscos do mercado, da concorrência, da Velocidade do Time de Desenvolvimento (veja mais adiante em “[9.2.2. Planejável](#)”) e de uma série de outros fatores. O Product Owner busca todo o conhecimento e a assistência necessários para realizar esse trabalho.

A melhor e menos complexa forma do Product Owner maximizar o ROI dos clientes do projeto é manter o foco em entregar para esses clientes as funcionalidades que eles mais necessitam em cada momento do projeto e obter *feedback* o mais cedo possível sobre essas funcionalidades para, assim, realizar as mudanças necessárias e reduzir os riscos do projeto. Ao entregar o que os clientes mais precisam, espera-se que os clientes ou usuários utilizem o Incremento do Produto entregue e, dessa forma, aumentem-se as chances de se obter o *feedback* necessário.

ROI

Retorno sobre o Investimento ou ROI (de Return On Investment, em inglês), é uma medida de desempenho de um investimento, em geral calculado pela relação entre o valor ganho ou perdido e o valor investido. Ou seja, caso se obtenha um ganho de X ao investir Y, o ROI correspondente será de X / Y.

Ao trabalharmos com Scrum, no entanto, em geral não fazemos contas matemáticas para calcular o ROI, mas sim utilizamos um conceito mais amplo e subjetivo do termo, no qual valor não necessariamente significa dinheiro, mas também ganhos institucionais e sociais, por exemplo, e o custo é baseado em estimativas realizadas pelo Time de Desenvolvimento.

O retorno sobre o investimento em projetos em que o Scrum é utilizado acontece sempre que são entregues aos clientes funcionalidades que lhe signifiquem valor de negócio.

É importante que o Product Owner possua uma estratégia para o desenvolvimento do produto focada em objetivos de negócios, que evoluí de acordo com as informações que estiverem disponíveis em cada momento. O Product Owner, dessa forma, considera o Product Backlog como um todo e em longo prazo, e não apenas seus itens individualmente.

9.2.2 Planejável

Um Product Backlog planejável permite que Time de Desenvolvimento e o Product Owner sejam capazes de, a partir das informações contidas no Product Backlog, realizar o planejamento do desenvolvimento de uma parte relevante dele – em geral o Sprint atual ou a próxima Release.

De forma geral, o Product Backlog ser planejável significa que ele permite:

- estabelecer o escopo mais provável do Sprint atual ou da próxima Release, ou seja, o que é mais provável que esteja dentro de um entregável ou de uma entrega, com uma data determinada;
- definir quando os próximos Incrementos do Produto a serem desenvolvidos poderão ser entregues para os clientes (a data da Release), dado um escopo provável;
- calcular a Velocidade do Time de Desenvolvimento, ou seja, quanto em média o Time de Desenvolvimento é capaz de entregar por Sprint;
- monitorar o progresso do trabalho em direção a uma entrega, dado o trabalho e o tempo restantes. Gráficos de Acompanhamento do Trabalho são ferramentas que podem ser utilizadas com esse propósito.

Estimativas do Product Backlog

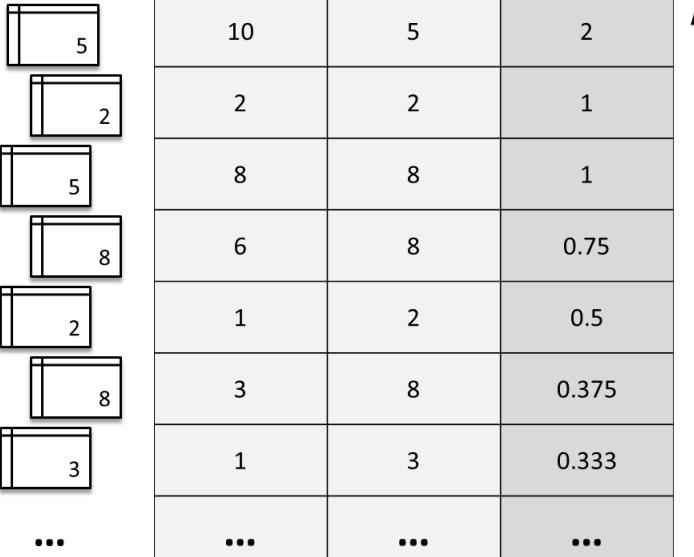
Para que serve estimar? Uma estratégia muito comum para se obter um Product Backlog planejável é aquela em que o Time de Desenvolvimento estima cada um de seus itens individualmente e então somam-se essas estimativas até o suficiente para que se possa planejar o próximo Sprint ou a próxima Release. Ou seja, estimar pode ajudar a planejar.

Entendemos por estimar o ato de julgar e formar uma opinião sobre o tempo que será necessário realizar um trabalho com acurácia e precisão adequadas. Mas, como veremos adiante, sugerimos unidades de tempo diferentes daquelas a que estamos acostumados, ou seja, dias ou horas (veja “Story Points”, a seguir).

Estimar os itens do Product Backlog individualmente ainda pode ajudar a se definir a ordem em que os itens do Product Backlog serão desenvolvidos, uma vez que sua estimativa representa o custo para se desenvolver o item. A partir dessa estimativa (custo) e de alguma medida do valor de negócios do item estabelecida junto aos clientes (valor), pode-se calcular o ROI individual do item dividindo-se seu valor

pelo seu custo. Assim, esse ROI individual pode ser utilizado como um dos critérios na ordenação do Product Backlog. Mas é importante lembrar que o ROI do Product Backlog como um todo não depende apenas dos ROI dos itens individuais.

No exemplo da figura 9.3, a ordenação foi determinada a partir do ROI individual de cada item, utilizando-se o valor de negócio (arbitrado em uma escala relativa de 1 a 10) e o custo de cada item, dado pela estimativa de seu tempo em Story Points, unidade que definiremos mais adiante.



Valor (1-10)	Custo (story points)	ROI Valor / Custo
10	5	2
2	2	1
5	8	1
8	8	0.75
2	2	0.5
8	8	0.375
3	3	0.333
...

Figura 9.3: Calculando o ROI de cada item

Por fim, uma vez que a estimativa do Product Backlog é feita por todo o Time de Desenvolvimento, o simples exercício de se reunir e discutir para realizar as estimativas leva o Time de Desenvolvimento a uma compreensão compartilhada e mais profunda do que é cada item e de como ele será desenvolvido. Ou seja, a importância não está somente nos números, mas também no caminho para se chegar a eles.

É importante ressaltar que estimar não é a única estratégia conhecida para se obter um Product Backlog planejável e, como veremos adiante, muitos líderes do

movimento Ágil hoje sequer a consideram uma boa estratégia.

Precisão e acurácia de estimativas baseadas em juízo. Ao estimarmos o tempo que levaremos para realizar um determinado trabalho no nosso dia a dia, não estamos utilizando cálculos avançados nem aplicando conceitos de probabilidade e estatística. Na realidade, construímos essa estimativa simplesmente utilizando-se de nossos conhecimentos, bom-senso, experiências passadas, contribuições de terceiros e informações disponíveis.

De forma simplificada, podemos definir uma estimativa a partir de uma faixa ou intervalo de valores com tamanho maior ou igual a zero, em torno de um **valor de referência**, dados em alguma unidade. Por exemplo, podemos estimar que uma determinada tarefa irá durar entre 3 e 5 horas, ou 4 ± 1 horas. Essa faixa de valores é chamada de **intervalo de confiança** da estimativa. O intervalo de confiança é tal que existe uma determinada probabilidade de que o valor real estará dentro dele. De forma prática, escolhemos para nossas estimativas um intervalo de confiança tal que acreditamos que há suficiente probabilidade de que o valor real estará dentro dele, de forma que essa estimativa nos seja útil.

Definimos a **precisão** de uma estimativa como a granularidade dessa estimativa. Assim, quanto menor é o intervalo de confiança utilizado para a estimativa, mais precisa ela é (veja a figura 9.4).

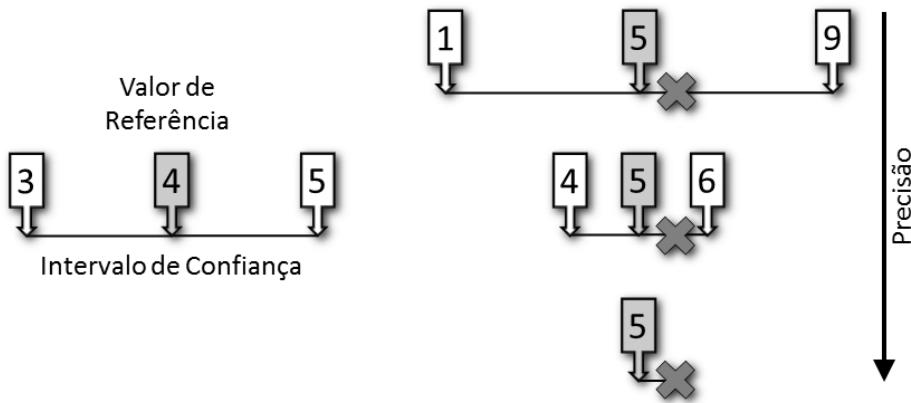


Figura 9.4: A estimativa e a precisão de uma estimativa

Por exemplo, a estimativa de que uma atividade irá durar exatamente 5 horas é mais precisa do que uma estimativa de que essa atividade irá durar entre 5 ± 1 horas (ou seja, entre 4 e 6 horas), e mais precisa ainda do que uma estimativa de que essa mesma atividade durará entre 5 ± 4 horas (ou seja, entre 1 e 9 horas).

É importante observar que também podemos aumentar a precisão da estimativa utilizando uma unidade com maior granularidade. Por exemplo, a estimativa de 5 horas, 3 minutos e 10 segundos é mais precisa do que a estimativa de que a atividade irá durar 5 horas.

Repare que a precisão independe do valor real verificado posteriormente, ou seja, a precisão não muda para qualquer dessas estimativas se a atividade, na realidade, durar 4 ou 20 horas, por exemplo.

O grau de **acurácia** de uma estimativa é definido por quão perto do valor real essa estimativa está. Assim, quanto menor a diferença entre a estimativa e o valor real verificado posteriormente, maior terá sido o seu grau de acurácia. O cálculo de um valor numérico para o grau de acurácia pode ser complicado, já que a estimativa é, na realidade, um intervalo. Aqui, porém, utilizaremos uma definição propositalmente simplificada: consideramos a estimativa acurada se ela contiver o valor real verificado posteriormente (ou seja, a estimativa se mostrou certa), e não acurada caso não o contenha (ou seja, a estimativa se mostrou errada).

Imagine, por exemplo, uma atividade para a qual se verificou posteriormente a duração de 5,5 horas. De acordo com as nossas definições, uma estimativa de que ela duraria exatamente 5 horas não é acurada, pois não contém o valor real. Uma estimativa de que ela duraria 5 ± 1 horas tem menor precisão, mas é acurada. Uma estimativa de que ela duraria 5 ± 3 horas também é acurada, mas tem uma precisão bem menor.

Em outro exemplo, se afirmarmos em nosso planejamento que uma determinada funcionalidade ficará pronta no dia 18 de novembro, essa será uma estimativa de alta precisão independente da data em que a funcionalidade de fato fique pronta. Caso essa funcionalidade fique pronta apenas no dia 25 de novembro, essa terá sido uma estimativa sem acurácia, de acordo com nossa definição de acurácia. Se tivéssemos afirmado, no entanto, que a funcionalidade estaria pronta no mês de novembro (o que na realidade significa entre os dias 1 e 30 de novembro), nossa estimativa teria sido de mais baixa precisão, porém acurada.

Ou seja, repare pelos dois exemplos anteriores que, ao reduzirmos a precisão de uma estimativa, aumentamos as chances dela ser acurada. Uma precisão cada vez menor (ou intervalo de confiança cada vez maior) é como um alvo cada maior, que

assim aumenta as chances de se acertar nele, ou seja, de se ter acurácia.

Precisão e acurácia suspeitas: Lei de Parkinson e Síndrome do Estudante. A Lei de Parkinson e a Síndrome de Estudante ilustram que um alto grau de acurácia (ou seja, de acerto) obtido em uma estimativa de alta precisão para o tempo de realização de alguma tarefa é geralmente artificial. Ou seja, ao se estimar o tempo necessário para se realizar uma determinada tarefa em x horas e, de fato, a tarefa estiver completa após exatas x horas de trabalho, há possivelmente algo de errado (Cohn, 2005).

A **Lei de Parkinson** afirma que “*a utilização de um recurso se expande até a sua disponibilidade*” ou, para nosso caso específico, “*o trabalho se expande de modo a preencher o tempo disponível para sua realização*”. Ou seja, ao se predeterminar o tempo para a realização de uma tarefa, o tempo utilizado para a sua realização naturalmente se expandirá até a duração preestabelecida, mesmo que menos tempo seja suficiente.

Para ilustrar, podemos imaginar que, quando possuímos x horas para realizar uma tarefa e julgamos que esse tempo é mais que suficiente, utilizamos uma minúcia maior ou adicionamos atributos desnecessários ao trabalho realizado, de forma a preencher o tempo que estimamos ou que nos foi dado.

A **Síndrome do Estudante** está relacionada a como o estudante realiza seus trabalhos de casa, ou seja, deixando-os para a última hora. Quando possuímos n horas para realizar uma tarefa, é um comportamento comum deixarmos sua realização para a última hora, comprometendo o prazo ou, ao menos, deixando de lado o cuidado e a qualidade necessários e só assim conseguirmos terminar a tempo.

Assim, de acordo com a Lei de Parkinson e a Síndrome do Estudante, um Gerente de Projetos tradicional que frequentemente acerta estimativas impostas para o trabalho de seus funcionários não tem razão para comemorar. O fato de existir essa estimativa, na realidade, produz artificialmente o “acerto”.

A Lei de Parkinson e a Síndrome de Estudante não possuem caráter científico, mas ilustram bem o comportamento humano.

Estimativas Ágeis. Abordagens tradicionais para projetos utilizam um alto grau de precisão em seu planejamento, o que, a uma primeira vista, pode parecer benéfico. Datas e durações exatas para o cumprimento de tarefas são marcadas em cronogramas, que vemos muitas vezes representados por meio de gráficos de Gantt. Sabemos, no entanto, que essas datas e durações estimadas serão quase que certamente erradas, pois a precisão utilizada ignora a realidade de incerteza que permeia projetos

como os de desenvolvimento de *software*. Assim, podemos afirmar que abordagens tradicionais utilizam estimativas de alta precisão, mas de baixa acurácia.

No desenvolvimento Ágil, ao utilizarmos estimativas para nosso planejamento, acreditamos que ter acurácia é mais importante do que ser preciso. Assim, em oposição aos planos com altíssima precisão utilizados em projetos tradicionais, reconhecemos que, quanto menos detalhes possuirmos acerca de um trabalho a ser realizado, menor deve ser a precisão utilizada para obtermos uma acurácia (ou seja, um acerto) razoável. E ainda entendemos que, em um ambiente de mudanças, quanto mais distantes estivermos da realização de um trabalho, menos detalhes teremos acerca desse trabalho e, assim, menor deverá ser a precisão utilizada para estimá-lo para que se possam obter estimativas acuradas.

Unidades para as estimativas. Para estimarmos o tempo para a realização de itens do Product Backlog de um projeto, podemos utilizar como unidades:

- **tempo real:** dias ou horas reais de trabalho, ou seja, uma estimativa de quanto tempo em dias ou horas a realização da atividade irá durar. É a unidade mais tradicional de estimativa;
- **tempo ideal:** dias ou horas ideias de trabalho, ou seja, quanto tempo se levaria para realizar uma atividade caso todo o foco de trabalho estivesse nessa atividade e não existissem quaisquer interrupções, como conversas, leitura de e-mails, idas ao banheiro, reuniões, cafezinho etc. A estimativa com tempo ideal também pode ser expressa em homens-dias ou homens-horas, ou seja, quantas pessoas são necessárias para completar o item de trabalho em um dia ou uma hora, sem interrupções;
- **Story Points:** unidade relativa de tempo criada pelo Time de Desenvolvimento. É a unidade mais utilizada por equipes Ágeis.

Para se estimar o tempo necessário para a realização de uma atividade, pode parecer natural ou intuitivo que utilizar dias ou horas seja a melhor opção. No entanto, a experiência nos mostra que existem alternativas mais interessantes.

De forma geral, apesar de possuírem baixa acurácia, as estimativas em dias ou horas reais carregam em si um senso de obrigação ou promessa irreal. Ao invés de serem entendidas como meras previsões, essas estimativas são tratadas como compromissos por quem espera os resultados. Por exemplo, quando alguém diz que

demorará cinco dias para executar uma determinada atividade, a expectativa criada é de que, após cinco dias, a atividade esteja pronta, e essa pessoa será questionada caso isso não ocorra. Além disso, possuir um valor em dias ou horas como referência para a realização de uma atividade pode levar a disfunções como a Lei de Parkinson e a Síndrome de Estudante.

Utilizar horas ideais ou dias ideias acentua ainda mais o problema da cobrança. Ao se dizer que uma atividade levará n dias ideais, é difícil justificar o porquê dessa atividade não estar pronta após n dias, pois a sensação que fica é de que o tempo não foi bem utilizado. Além disso, estimar em horas ou dias que não são, de fato, horas ou dias não traz nenhum benefício para o planejamento.

Para melhorar essa questão, as estimativas em horas ou dias reais podem ser explicitamente expressas com precisões menores, utilizando-se intervalos de confiança. Pode-se, por exemplo, estimar “entre quatro e seis dias” ao invés de “cinco dias”, e as chances de acertar (de ter acurácia, portanto) aumentarão. No entanto, lidar com esse tipo de estimativas no planejamento é muito trabalhoso. E mesmo assim, geralmente a acurácia obtida com precisões explicitamente menores ainda é notavelmente baixa.

A abordagem mais utilizada por times Ágeis é o uso de estimativas relativas. Entende-se que é muito mais fácil estimar realizando comparações do que fazê-lo em termos absolutos. Assim, ao invés de estimar que uma determinada atividade exigirá seis dias para ser realizada, estabelecemos que ela irá demorar, por exemplo, mais ou menos duas vezes mais tempo que uma outra determinada atividade e metade do tempo de uma terceira, essas últimas duas sendo utilizadas como referência. Contanto que as referências sejam propagadas por todas as estimativas, elas se manterão consistentes. A abordagem de estimativas relativas evita ao mesmo tempo os efeitos da Lei de Parkinson, da Síndrome do Estudante e do tratamento incorreto da estimativa como compromisso ou obrigação simplesmente ao deixar de expressar as estimativas em dias ou horas e, assim, desacoplar a execução de um item de trabalho de uma data exata de entrega (De Toledo, 2009).

Story Points

Story Point é uma unidade relativa utilizada pelo Time de Desenvolvimento para estimar o tempo necessário para se desenvolver um item de trabalho – no caso de Scrum, para transformar um item do Product Backlog em pronto, de acordo com a Definição de Pronto. Ao invés de utilizar horas ou dias, o próprio Time de Desenvolvimento cria ou escolhe uma escala e, a partir de um ou mais pontos de referência

selecionados, passa a estimar os itens de trabalho dentro dessa escala, comparando-os com esses pontos de referência e com os últimos itens estimados.

Escalas simples de um a dez ou potências de dois podem funcionar, mas as mais utilizadas por times Ágeis são a sequência de Fibonacci e a chamada “tamanhos de camisa” (*T-Shirt Sizing*), ambas explicadas em seguida.

Na sequência de Fibonacci, cada número é a soma dos dois anteriores, começando com zero e um: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 e assim por diante. Para efeitos práticos, utilizamos a escala da seguinte forma:

1 2 3 5 8 13 21 34 55 89 ...

Pode-se utilizar uma versão adaptada, criada por Mike Cohn para maior simplicidade (Cohn, 2005):

1 2 3 5 8 13 20 40 100

Nessa versão, 40 e 100 simplesmente significam que um item de trabalho é grande demais. Para os tamanhos de camisa (*T-Shirt Sizing*) a escala é a seguinte:

PP P M G GG XG

P significa mais ou menos o dobro do tempo que PP, M significa mais ou menos o dobro do tempo que P e mais ou menos quatro vezes o tempo que PP, e assim por diante. Na maioria dos casos, uma escala menor e mais simples pode ser suficiente:

P M G

Outras escalas que passem a ideia de tamanho relativo podem funcionar bem, como raças de cachorro (do chihuahua ao dogue alemão), vegetais (da azeitona à melancia) ou algo mais que a criatividade do Time de Desenvolvimento o leve a criar.

Uma vez selecionada a escala, escolhem-se um ou mais itens de referência para se criar os pontos da escala. Uma vez escolhidos esse item ou itens de referência, os itens seguintes a serem estimados são comparados com ele ou eles.

Uma técnica simples para se criar os pontos da escala é a de se escolher como referência de unidade o menor item do Product Backlog naquele momento, ou seja, esse item passa a ter a estimativa de 1 story point, ou “P”, ou “chihuahua”, por exemplo, dependendo da escala escolhida. Assim, os próximos itens a serem estimados

serão comparados com essa referência. Esse item provavelmente está no alto do Product Backlog, onde estão os itens menores e mais detalhados. Caso o Time de Desenvolvimento julgue haver vários itens igualmente pequenos, ele escolherá como referência aquele mais conhecido, mais detalhado e com menos risco envolvido em seu desenvolvimento. Ao invés da unidade, muitos times preferem atribuir a esse item de referência o segundo ponto da escala (ou seja, 2 story points, por exemplo) para que no futuro haja espaço para itens menores.

Alternativamente, podem-se conseguir melhores resultados ao se escolherem dois itens de referência de tamanho médio ao invés de apenas um e pequeno. Atribuem-se a esses itens 5 ou 8 story points (ou “M”, por exemplo). De outra forma, podem-se escolher um item pequeno e um item grande, atribuindo-se a eles 2 e 13 story points (ou “P” e “G”, respectivamente). Os próximos itens a serem estimados são comparados com esses dois itens de referência.

Uma vez criada a referência ou referências, parte-se do alto do Product Backlog estimando-se o primeiro item, comparando-o com esse ou esses itens de referência. Digamos que o Time de Desenvolvimento escolheu um item como referência e atribuiu 2 story points a ele. Assim, se o Time de Desenvolvimento julga, por exemplo, que primeiro item do Product Backlog levará mais que o dobro do tempo que o item de referência, esse item é então estimado em 5 story points, ou “M” etc. Para o item seguinte, faz-se uma triangulação com os dois itens já pontuados, ou seja, compara-se com eles esse item seguinte. Assim, se o Time de Desenvolvimento julga que o desenvolvimento desse item seguinte irá demorar bem mais o do item de referência e um pouco mais que o do outro item já estimado, pode-se estimar esse item seguinte em 8, ou “G” etc., dependendo da escala escolhida.

A partir daí, o Time de Desenvolvimento segue estimando os itens a partir do alto do Product Backlog, Sprint após Sprint, sempre comparando-os com os últimos itens estimados. É importante ressaltar que o item de referência é escolhido apenas uma vez, geralmente antes do princípio do projeto, e não Sprint a Sprint.

O uso da sequência de Fibonacci como escala para as estimativas traz uma vantagem interessante. Cada número da escala carrega em si um intervalo de confiança e, assim, o tratamento da precisão e acurácia da estimativa já está embutido nessa escala. Por exemplo, ao se estimar um item do Product Backlog em 3 story points, implicitamente se está afirmado que esse item deverá ser algo entre 2 e 5 story points – um intervalo de confiança de tamanho $(5 - 2) = 3$. No entanto, ao se estimar outro item como 8 story points, implicitamente se está afirmado que esse item deverá ser algo entre 5 e 13 story points – um intervalo de confiança de tamanho $(13 - 5) = 8$ (veja

a figura 9.5).

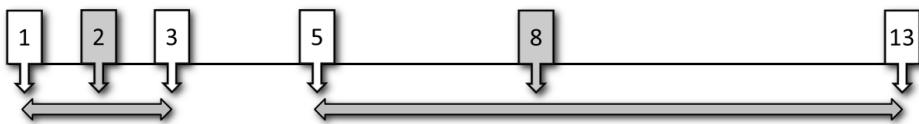


Figura 9.5: Fibonacci, precisão e acurácia

Ao utilizar a sequência de Fibonacci, podemos verificar que quanto maior for a estimativa de um item, maior será seu intervalo de confiança e, assim, menor será sua precisão. De fato, sabemos intuitivamente que, para termos mais acurácia, a precisão que podemos utilizar para estimar itens menores é maior do que para itens maiores. Como vimos anteriormente, quanto menor for a precisão de uma estimativa, maior é a possibilidade de essa estimativa ser acurada. Portanto, a sequência de Fibonacci automaticamente reduz a precisão de uma estimativa quanto maior ela for, aumentando assim sua possibilidade de acerto, ou seja, de acurácia.

Discussão: Story Points representam complexidade, risco, esforço ou tamanho?

Existe muita confusão acerca do significado dos Story Points. Eles representam a complexidade de uma funcionalidade? Talvez com um componente do risco envolvido em realizá-la, derivado da incerteza? Ou será que representam o esforço necessário para desenvolver a funcionalidade? Ou talvez o tamanho? Mas então o que exatamente significam esforço e tamanho?

Story Points foram criados pelos mesmos inventores do Extreme Programming (Jeffries et al., 2000), uma metodologia Ágil que muitas vezes anda de mãos dadas com Scrum. Seu propósito era o de burlar estimativas em unidades de tempo, como horas ou dias, pois a gerência equivocadamente entendia essas estimativas como promessas, e não como estimativas. Ou seja, era comum ouvir-se “*se você disse que iria ficar pronto em três dias, por que não está pronto três dias depois?*”. Assim, Story Points foram inicialmente criados para que se pudesse continuar realizando estimativas necessárias para o planejamento, mas se obfuscando o tempo para evitar o compromisso com uma alta precisão e, dessa forma, evitar a pressão da gerência sobre essas estimativas.

Story Points representam, nada mais, nada menos, que tempo. É claro que, ao es-

timarmos o tempo que necessitaremos para realizar determinado trabalho, levamos em consideração parâmetros como complexidade, incerteza e tamanho, e sabemos que custo e esforço são derivados do tempo. Mas Story Points representam, simplesmente, tempo. É, porém, um tempo que não é medido em horas ou dias, mas sim em uma escala relativa criada pelo próprio Time de Desenvolvimento. Assim, *n* story points nada mais significam que *n* dias ou horas vezes algum fator.

É importante ressaltar que, ao utilizar Story Points, buscamos justamente mascarar e evitar expressar a estimativa em horas e, assim, não faz sentido realizar sistematicamente qualquer tradução dos pontos em dias ou horas (como, por exemplo, “para nosso time, 3 story points significam dois dias de trabalho”) ou descobrir em qualquer momento o fator de conversão de um para outro.

Hoje, no entanto, os próprios criadores dos Story Points não mais as recomendam. Devido a seu uso exageradamente incorreto, preferem alternativamente manter os itens do Product Backlog pequenos (com cerca de dois ou três dias de tempo de desenvolvimento) e talvez contar o número de itens do Product Backlog para se realizar o planejamento, como veremos adiante.

Como fazer a estimativa?

Qualquer que seja o método utilizado para realizar as estimativas, é importante que os princípios básicos a seguir sejam respeitados:

- as estimativas dos itens do Product Backlog somente podem ser realizadas pelo Time de Desenvolvimento. Elas não devem ser realizadas pelo Product Owner, pelo ScrumMaster ou por quaisquer outras partes interessadas do projeto. Acredita-se que as melhores estimativas são aquelas feitas pelas pessoas que realizam o trabalho a ser estimado. Essa premissa, apesar de representar o senso comum, é ignorada em um grande número de projetos, nos quais frequentemente Gerentes de Projeto criam as estimativas para suas equipes;
- as estimativas são definidas por um consenso de todo o Time de Desenvolvimento. Como cada item do Product Backlog deve ser desenvolvido por todo o Time de Desenvolvimento, a estimativa de um item do Product Backlog não pode pertencer a apenas um ou a alguns de seus membros, mas sim deve ser fruto da colaboração entre todos;
- o processo de definição das estimativas não pode ser custoso nem demorado. Se esse processo não for rápido e objetivo, o tempo despendido nele será re-

vante, de forma que a própria atividade de estimar necessitará de estimativas, o que não faz sentido.

Os membros do Time de Desenvolvimento se reúnem para a realização de estimativas de itens do Product Backlog. O Product Owner, presente durante a atividade, pode esclarecer dúvidas que inevitavelmente surgirão quanto aos itens, enquanto que o ScrumMaster atua como facilitador. Nenhum deles participará da escolha dos valores para as estimativas.

Cada membro do Time de Desenvolvimento possui um conjunto de cartas com as alternativas possíveis da escala utilizada para as estimativas. Assim, se a sequência de Fibonacci adaptada for escolhida, cada um possuirá as cartas 1, 2, 3, 5, 8, 13, 20, 40, 100. Alternativamente, cada membro pode escrever as alternativas em notas adesivas ou a sua escolha diretamente em um papel em branco. Existem também aplicativos disponíveis para tablets e celulares com essa finalidade.

Alguém – o Product Owner, o ScrumMaster ou um membro do Time de Desenvolvimento – lê o próximo item do Product Backlog a ser estimado. Os membros do Time de Desenvolvimento refletem brevemente sobre esse item e esclarecem dúvidas com o Product Owner.

Cada membro do Time de Desenvolvimento então escolhe a carta com sua estimativa para o trabalho necessário para se desenvolver o item. Para a estimativa, cada um pensa no trabalho do Time de Desenvolvimento como um todo, e não apenas em seu próprio, e mentalmente compara com estimativas de itens realizadas anteriormente.

É importante que o valor escolhido não seja imediatamente discutido nem mostrado aos colegas, para não influenciar seu julgamento inicial. Ao verem prematuramente as estimativas de seus colegas, os membros do Time de Desenvolvimento que ainda não revelaram sua carta podem modificar sua escolha de forma a se aproximarem das estimativas já mostradas, sem sequer pensarem ou discutirem o porquê. É uma tendência natural de conformidade que o ser humano possui, chamada de ancoragem.

Em seguida, cada um coloca sua escolha sobre a mesa, de forma que todos possam vê-la (veja a figura 9.6). Dificilmente um consenso será atingido nessa primeira votação. As diferenças, na realidade, irão gerar conversas que levarão a uma melhor compreensão do item, o que também contribui para a identificação de problemas cedo. Mais votações são então realizadas, sempre facilitadas pelo ScrumMaster, até que um consenso seja atingido. A necessidade de um consenso faz com que todo o

Time de Desenvolvimento seja responsável pela estimativa, e não apenas um indivíduo.

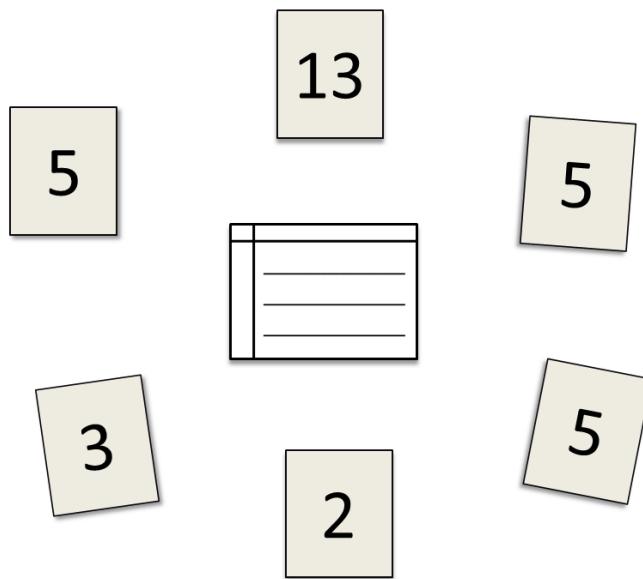


Figura 9.6: Planning Poker

Após cada votação, as pessoas que deram a estimativa mais alta e a mais baixa começam a discussão, justificando o porquê de suas escolhas. O fato de terem que justificar sua estimativa leva os membros do Time de Desenvolvimento a realmente se engajarem em buscar estimativas que lhes façam sentido.

Um membro do Time de Desenvolvimento que estimou o item em 2 story points pode dizer, por exemplo: *“esse item é fácil, já fiz algo muito parecido antes”*. O outro membro que escolheu 8 story points pode então dizer *“sim, mas você está se esquecendo que testar isso é bem complicado. Mas como você já fez algo parecido, talvez demore menos do que pensei”*. Em uma segunda rodada, eles terão estimativas mais próximas (3 e 5 story points, provavelmente), assim como o resto do Time de Desenvolvimento, e o consenso poderá até mesmo ser atingido sem uma nova votação.

Não é recomendável, no entanto, gastar tempo demais com a estimativa de cada item. O processo de estimar é idealmente rápido e objetivo, de forma que o tempo total gasto não seja relevante. Além disso, a partir de algum momento, mais con-

versas já não trarão melhores estimativas. É considerado desperdício, por exemplo, discutir profundamente em detalhes de como o item será desenvolvido.

Pode ser uma boa ideia determinar o tempo máximo para cada discussão, ou seja, estabelecer um *timebox*. Além disso, espera-se que o consenso seja atingido com não mais que duas ou três votações. No entanto, em caso de impasse, o Time de Desenvolvimento pode ter seus próprios acordos de como proceder. Por exemplo, se o consenso não é possível, pode-se determinar a escolha da estimativa mais alta, ou talvez de uma intermediária.

O uso da sequência de Fibonacci com escala de estimativas facilita significativamente o processo de se chegar a um consenso. Como a escala não é linear, quanto maior é o item, menor é o número de escolhas possíveis para sua estimativa, e assim não se desperdiça tempo com estimativas de mais baixa precisão. Por exemplo, faz muito mais sentido gastar mais tempo discutindo se um item do Product Backlog deve ter a estimativa de 2 ou 3 story points (mais alta precisão) do que se outro item deve ter 15 ou 16 story points (mais baixa precisão). Esta última discussão não é possível ao se utilizar a sequência de Fibonacci – ou o item será estimado em 13, ou em 21.

Planning Poker

O Planning Poker é a técnica mais conhecida e utilizada por times Ágeis para realizar as estimativas de itens do Product Backlog. Foi criado por James Greening, um dos signatários do Manifesto Ágil, e satisfaz todos os princípios descritos acima se utilizado corretamente (Greening, 2002). Apesar de sua popularidade, o Planning Poker não é parte integrante do Scrum e seu uso é opcional.

PLANNING POKER

A atividade de Planning Poker se resume aos seguintes passos:

- 1) Alguém - ScrumMaster, Product Owner ou membro do Time de Desenvolvimento - lê o próximo item a ser estimado. O ScrumMaster facilita a sessão;
- 2) Os membros do Time de Desenvolvimento rapidamente refletem sobre esse item e esclarecem dúvidas com o Product Owner;
- 3) Cada membro do Time de Desenvolvimento então escolhe uma carta com a sua estimativa para o trabalho necessário para o Time de Desenvolvimento desenvolver o item, sem mostrar a carta imediatamente;
- 4) Todos mostram as cartas, colocando-as sobre a mesa ao mesmo tempo. Dessa forma, evita-se a ancoragem, ou seja, que se aceite sem reflexão a influência dos outros;
- 5) Os membros do Time de Desenvolvimento que deram a estimativa mais alta e a mais baixa conversam diante dos outros e defendem suas estimativas, justificando o porquê dos valores escolhidos;
- 6) Uma ou mais rodadas de escolha de estimativas para o item são realizadas, até os membros Time de Desenvolvimento chegarem a um consenso.

Velocidade do Time de Desenvolvimento

Conhecer a Velocidade do Time de Desenvolvimento pode ajudar a tornar o Product Backlog planejável.

O Time de Desenvolvimento pode utilizar Story Points ou alguma unidade para estimar o tempo necessário para realizar um determinado trabalho. Para efeitos didáticos, chamaremos essa unidade simplesmente de “pontos”. Ela será utilizada para a estimativa de cada item do Product Backlog.

Digamos que o Time de Desenvolvimento tenha produzido, nas últimas três Sprints, itens que somados correspondem a 30, 32 e 29 pontos respectivamente. Podemos tirar a média desses valores e dizer que a Velocidade do Time de Desenvolvimento é de aproximadamente 30 pontos por Sprint (veja a figura 9.7). A Velocidade do Time de Desenvolvimento é, portanto, sua taxa média de queima de pontos por Sprint. Assim, podemos esperar que o Time de Desenvolvimento entregue algo próximo de 30 pontos no próximo ou nos próximos Sprints ou, ao menos, sabemos que essa é a nossa melhor previsão possível.

Espera-se uma Velocidade relativamente constante ao longo do projeto, dado que a composição do Time de Desenvolvimento seja mantida e o mesmo ocorra com o tamanho dos Sprints. O primeiro Sprint se inicia, é claro, sem nenhuma referência de Velocidade, mas a partir do segundo ou terceiro já faz sentido se utilizar os Sprints anteriores para seu cálculo, ainda que o valor não tenha se estabilizado. No entanto, é considerada uma boa prática utilizar-se apenas os valores dos três últimos Sprints para fazer a média ao invés de todos os valores históricos, para assim produzir-se um valor mais realista e atualizado para a Velocidade do Time de Desenvolvimento.

Esse valor pode servir como referência no planejamento do próximo Sprint. Por exemplo, durante a reunião de Sprint Planning seguinte, o Time de Desenvolvimento pode não aceitar para o Sprint Backlog um número de itens cujas estimativas somem muito mais do que 30 pontos, e se for necessário negociará com o Product Owner para reduzir o número de itens e se aproximar desse valor. O Product Owner, por sua vez, poderá contestar a escolha do Time de Desenvolvimento, para o Sprint Backlog, de um número de itens cujas estimativas somem muito menos do que esses 30 pontos e, caso necessário, negociará com o Time de Desenvolvimento um aumento no número de itens para se aproximar desse valor.

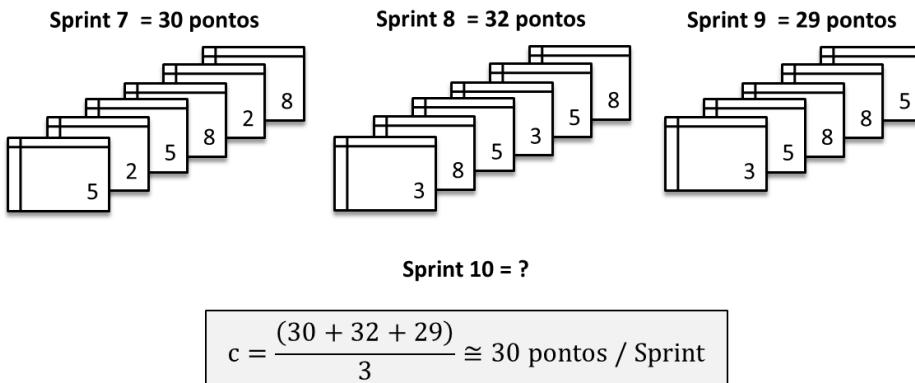


Figura 9.7: Medir a Velocidade do Time de Desenvolvimento pode aumentar a previsibilidade

Na realidade, existe uma grande discussão sobre se uma variabilidade natural da Velocidade do Time do Desenvolvimento não impediria sua utilização no planejamento em curto prazo, ou seja, para o próximo Sprint por exemplo. Mas a Velocidade também pode ser útil para o planejamento da próxima Release, mesmo que seja necessária uma extração de baixa precisão. Tendo-se estimativas para os itens do Product Backlog e sabendo-se da Velocidade do Time de Desenvolvimento, pode-se planejar qual meta de negócios (Meta da Release) poderá ser atingida dado um prazo ou, dada uma meta de negócios, quando ela poderá ser atingida. Essa técnica é geralmente utilizada na reunião de Release Planning (veja “[22.1. O que é a Release Planning?](#)”).

Estimativas são desperdício

Existem alternativas importantes à abordagem de se estimar os itens do Product Backlog individualmente. Por exemplo, ao se manter os itens do alto do Product Backlog pequenos ou, ao menos, com tamanhos razoavelmente próximos e bem distribuídos, pode-se determinar a Velocidade do Time de Desenvolvimento apenas fazendo-se a média da quantidade de itens entregues nos últimos Sprints, e utilizar esse valor para planejar o próximo Sprint. Assim, um Time de Desenvolvimento que tenha entregue 8, 8 e 9 itens nos últimos três Sprints tem a Velocidade de 8 itens por Sprint (aproximadamente). Essa forma simplificada de se trabalhar é suficiente para muitos contextos, nos quais estimar além desses limites pode ser considerado

desperdício.

Outros times não realizam qualquer tipo de estimativa, seja de itens ou de uma parte relevante do Product Backlog. Eles consideram que qualquer estimativa é desperdício. Apenas com seu conhecimento e experiência, são capazes de prever que objetivo entregarão naquele Sprint ou naquela Release, trabalhando sempre do item mais importante para o item menos importante.

9.2.3 Emergente

O Product Backlog é uma lista dinâmica, que reflete o ambiente de mudança a partir do refinamento gradual dos detalhes do produto que está sendo desenvolvido.

À medida que o Time de Desenvolvimento trabalha desde o alto do Product Backlog, transformando-o em produto funcionando, o Product Owner adiciona, remove, modifica, refina e reordena seus itens. Esse é um trabalho contínuo, que ocorre desde o início até o final do projeto.

O Product Backlog nunca está completo. Ele evolui em um processo contínuo de descoberta, refletido na inspeção e adaptação do produto realizada a partir do *feedback* dos clientes e demais partes interessadas sobre os Incrementos do Produto a eles mostrados ou entregues.

9.2.4 Gradualmente detalhado

O Product Backlog é progressivamente refinado ao longo de todo o projeto, de forma que seus itens possuam um detalhamento adequado. Itens do alto do Product Backlog possuem granularidade mais fina e, assim, representam mais detalhes. Os próximos itens a serem desenvolvidos devem possuir os detalhes de negócios suficientes e necessários para que possam ser colocados em desenvolvimento. Itens abaixo no Product Backlog possuem granularidade gradativamente mais grossa e, assim, representam menos detalhes, de forma que itens da parte de baixo, em geral, não possuem detalhe algum. Mais detalhes do que o necessário e suficiente é, geralmente, desperdício.

À medida que um item ganha importância, ele sobe no Product Backlog, pode evoluir em itens menores e os detalhes de negócios que serão necessários para seu desenvolvimento são esclarecidos e elaborados.

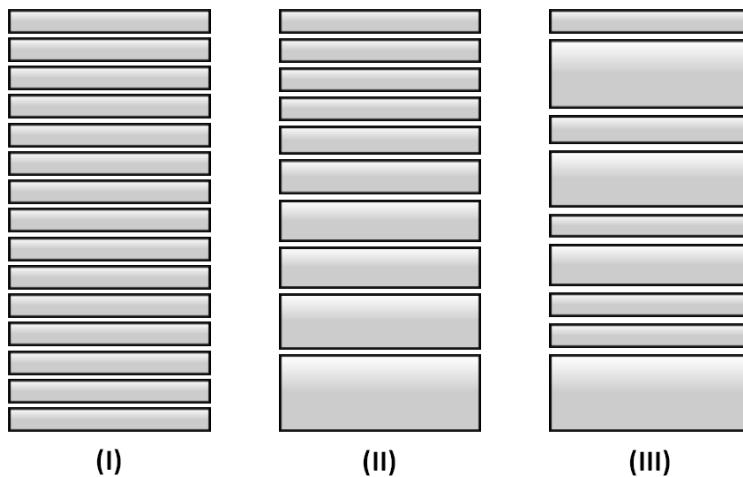


Figura 9.8: O Product Backlog é gradualmente detalhado

Na figura 9.8, apresentamos três possíveis formatos para o Product Backlog. No Product Backlog (I), todos os itens têm granularidade fina e muitos detalhes. Mais detalhes do que o suficiente e necessário constituirão desperdício, já que, pela natureza emergente do Product Backlog, os detalhes mudarão. No Product Backlog (III), ao menos um item de seu alto não possui detalhes suficientes para ser colocado em desenvolvimento, o que significa um alto risco para o trabalho do Time de Desenvolvimento. Além desse item, diversos outros mais abaixo possuem mais detalhes do que o necessário, o que novamente constitui desperdício, já que há uma grande chance de que os detalhes mudem até o momento do seu desenvolvimento. O Product Backlog (II) é o que consideramos o ideal, pois é gradualmente detalhado, ou seja, possui detalhes suficientes e necessários, dependentes da ordem do item.

9.3 USER STORY

9.3.1 O que é a User Story?

User Story ou “história de usuário” é uma descrição concisa de uma necessidade do usuário do produto (ou seja, de um “requisito”) sob o ponto de vista desse usuário. A User Story busca descrever essa necessidade de uma forma simples e leve.

Um dos princípios por trás das User Stories é a de que o produto poderia ser

integralmente representado por meio das necessidades de seus usuários (Jeffries et al., 2000). O produto desenvolvido com Scrum é descrito por meio de itens do Product Backlog e, assim, de acordo com esse princípio, cada um desses itens deveria ser representado no formato de User Stories. Ou seja, uma User Story representa um e apenas um item do Product Backlog.

É importante destacar que as User Stories não fazem parte do *framework* Scrum e, assim, seu uso é opcional.

Veja um exemplo de User Story na figura 9.9.

	<p><i>Eu, enquanto Comprador de Livros, quero encontrar um livro de que sei o título para poder comprá-lo</i></p>

Figura 9.9: Exemplo de User Story, representando uma necessidade do usuário

A User Story é apenas uma promessa de uma conversa, um lembrete de que mais detalhes serão necessários, e não deve ser considerada suficiente para a realização do trabalho. Ela é o começo, mas somente será útil para o desenvolvimento do produto se for seguida por uma série de conversas entre as pessoas de negócios (em geral, o Product Owner) e os membros do Time de Desenvolvimento. Essas conversas visam definir e capturar os detalhes de negócios necessários para o desenvolvimento da funcionalidade que atenderá a essa necessidade do usuário.

Os detalhes de negócios podem ser documentados de diferentes formas e anexados à User Story. Acredita-se, no entanto, que apenas Critérios e Testes de Aceitação da User Story já representem documentação suficiente, uma vez que devem cobrir todos os aspectos de negócios do requisito.

Na prática, questões técnicas (como de refatoração de código ou população de

um banco de dados, por exemplo), de pesquisa ou de correção de problemas dificilmente podem ser descritas sob a perspectiva do usuário. Embora pertençam ao Product Backlog, não devem, assim, ser representadas por User Stories. As questões técnicas são, no entanto, de difícil entendimento pelo Product Owner e consequentemente ele enfrentará dificuldades em ordená-las. Dessa forma, recomenda-se que, sempre que possível, elas sejam embutidas como parte das User Stories para as quais são necessárias.

9.3.2 Como é a User Story?

A User Story possui três aspectos críticos, chamados de “os três C’s”: o Cartão, as Conversas e a Confirmação. Veremos em seguida o que cada um desses aspectos significa (Cohn, 2004).

Cartão

O Cartão é a descrição da necessidade do usuário, ou seja, a própria User Story. Essa descrição é concisa, e assim suficiente para apenas identificar qual é e de que se trata essa necessidade. O nome “Cartão” é dado porque frequentemente as User Stories são escritas em cartões de índice ou fichas.

Os padrões mais utilizados para se escrever o Cartão da User Story estabelece três parâmetros da necessidade do usuário: “QUEM”, “O QUÊ” e “POR QUÊ”.

“QUEM” define quem é o usuário que tem a necessidade. Pode ser representado por um tipo de usuário do produto (como “Comprador de Livros” ou “Administrador do Sistema”, por exemplo), por uma persona ou até mesmo por um usuário específico.

“QUEM” ajuda a criar no leitor da User Story uma imagem mental desse usuário. User Stories que começam “Eu, enquanto um usuário...” ou “Eu, enquanto um cliente...”, portanto, não atingem esse objetivo e devem ser evitadas.

Personas são frequentemente utilizadas com esse propósito. Uma persona é um usuário imaginário que representa um grupo distinto de usuários do produto, que tem necessidades e comportamentos específicos. A persona possui um nome, que pode ser inventado ou ser emprestado de alguma personalidade conhecida, e uma descrição. Pode também possuir uma foto. Na figura 9.10, descrevemos “Sara Sensível”, uma persona de um *site* de relacionamentos.

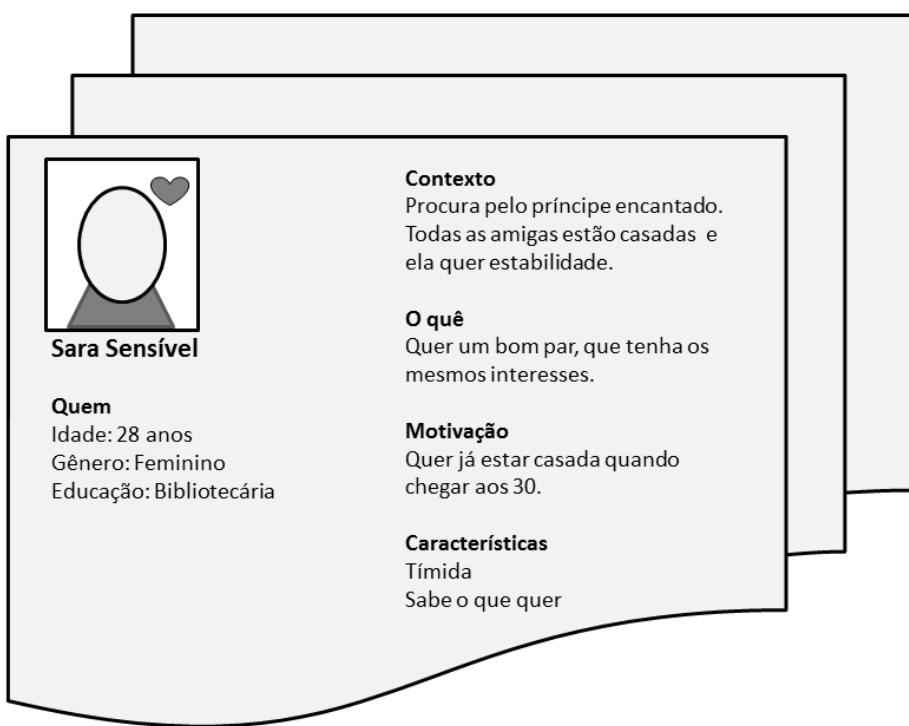


Figura 9.10: Exemplo de persona

Uma User Story para essa persona poderia ser algo assim: “Para encontrar alguém com interesses parecidos com os meus, enquanto Sara Sensível, eu quero comparar os interesses de diferentes candidatos”.

“**O QUÊ**” define qual é a necessidade do usuário. Tradicionalmente, os requisitos do produto são representados apenas por essa parte.

“**POR QUÊ**” define qual o benefício do usuário ao ter a funcionalidade desenvolvida para atender a essa necessidade. Em outras palavras, qual o valor direto obtido pelo usuário.

PONTO DE VISTA DO USUÁRIO

O ponto de vista do usuário é o de quem tem um problema ou uma solução? Um problema. Vejamos, a seguir, uma User Story:

“Eu, enquanto Comprador de Livros, quero buscar livros por nome para escolher o que vou comprar.”

Essa é a forma mais comum de se escreverem User Stories e é perfeitamente aceitável. No entanto, buscar livros por nome é um problema ou uma solução? Uma solução. Então essa User Story não está realmente expressa sob a perspectiva do usuário. Qual seria o problema do usuário então? Se a reescrevermos da seguinte forma:

“Eu, enquanto Comprador de Livros, quero encontrar um livro cujo nome sei para escolher comprá-lo.”

O problema do usuário nesse caso é encontrar um livro cujo nome conhece, e uma das possíveis soluções é buscar livros por nome. Outra seria mostrar simplesmente uma lista de todos os livros para que ele possa encontrar o que precisa. Não que uma lista seja melhor que uma busca, mas essa alternativa poderia, por exemplo, ser mais adequada para uma versão inicial do sistema devido a seu custo mais baixo.

Expressar a User Story a partir do problema e não de uma solução particular abre espaço para discussões sobre as possíveis soluções para o mesmo problema. Assim, pode fazer com que soluções mais adequadas sejam encontradas.

Na figura 9.11, vemos o padrão mais comum para escrita de User Stories:

	<i>Eu, enquanto <QUEM>, quero <O QUÊ> para <POR QUÊ></i>

Figura 9.11: Padrão de escrita de uma User Story

Esse formato, embora largamente utilizado, não é obrigatório. Podemos ver na figura 9.12 uma outra forma de dizer o mesmo, mas com ênfase no benefício do usuário.

	<i>Para <POR QUÊ>, enquanto <QUEM>, eu quero <O QUÊ></i>

Figura 9.12: Outro padrão de escrita de uma User Story

Outra forma simples de se escrever a User Story é: “Um <QUEM> pode <O QUÊ> para <POR QUÊ>”. Por exemplo, se imaginarmos como produto um aplicativo móvel para viagens, uma de suas User Stories poderia ser: “*Um Viajante de Turismo quer mostrar sua viagem para seus amigos para lhes causar inveja*”.

Ao escrever uma User Story em algum desses formatos, o Product Owner realiza

uma profunda reflexão para estabelecer quem é o usuário da funcionalidade a ser desenvolvida e qual o benefício que esse usuário obtém, o que lhe permite manter os itens do Product Backlog alinhados à Visão do Produto. Ao mesmo tempo, a User Story ajuda o Time de Desenvolvimento a manter seu foco no usuário e em qual benefício é esperado que ele obtenha.

Conversas

São conversas sobre a User Story, por onde a solução de negócios e os detalhes dessa solução são discutidos, negociados, definidos e então documentados na forma de Critérios e Testes de Aceitação. Elas geram uma compreensão compartilhada sobre o que é necessário para que a funcionalidade gere valor de negócio e retorno ao investimento. As conversas são imprescindíveis para o trabalho do Time de Desenvolvimento, uma vez que o Cartão não possui detalhes que permitam que o item seja desenvolvido.

Em geral, participam dessas conversas o Product Owner, membros do Time de Desenvolvimento e outras pessoas de negócios envolvidas, caso haja, mas qualquer pessoa que possa contribuir com detalhes pode participar.

Tomemos como exemplo a User Story do exemplo anterior, “*Um Viajante de Turismo quer mostrar sua viagem para seus amigos para lhes causar inveja*”. O Time de Desenvolvimento e Product Owner, por meio de conversas, podem concluir que permitir o compartilhamento de fotos e vídeos na rede social do momento diretamente do aplicativo é a solução de negócios desejada, e a partir daí escrevem os Critérios de Aceitação, que serão transformados em Testes de Aceitação automatizados no decorrer do Sprint.

As conversas, que podem ocorrer em qualquer momento, são geralmente parte das sessões de Refinamento do Product Backlog (veja “[23. Refinamento do Product Backlog](#)”).

Confirmação

São critérios e testes deles derivados que documentam os detalhes da User Story, definindo seus limites. Ou seja, são regras que estabelecem como a funcionalidade deve se comportar uma vez implementada. Esses critérios são chamados de Critérios de Aceitação e os testes, de Testes de Aceitação.

Enquanto todas as User Stories devem satisfazer à mesma Definição de Pronto, cada User Story possui seu próprio conjunto de Critérios e Testes de Aceitação.

Critérios de Aceitação são expressos por enunciados pequenos e de fácil entendimento. São utilizados para determinar quando a funcionalidade produzida pelo Time de Desenvolvimento está completa e, assim, nada mais deve ser adicionado a ela.

Os Critérios de Aceitação ajudam o Product Owner a determinar para o Time de Desenvolvimento o que ele precisa para que essa funcionalidade propicie valor. A partir desses critérios, o Time de Desenvolvimento gera os Testes de Aceitação.

Os Critérios de Aceitação são negociados e definidos antes do desenvolvimento da funcionalidade, geralmente em sessões de Refinamento do Product Backlog.

Como exemplo, vamos imaginar que a seguinte User Story é forte candidata a ser desenvolvida no próximo Sprint:

Eu, enquanto Comprador de Livros, quero utilizar meu cartão de crédito no pagamento dos livros escolhidos, para ter praticidade e segurança no pagamento.

Assim, em uma das sessões de Refinamento do Product Backlog, os seguintes Critérios de Aceitação podem ser definidos para a User Story, entre outros:

Critério #1: somente podemos aceitar cartões de crédito com bandeiras com que temos convênio.

Critério #2: somente podemos aceitar cartões de crédito com data de expiração no futuro.

Testes de Aceitação são criados a partir de aplicações de exemplos aos Critérios de Aceitação, onde se verificam se dadas entradas estão produzindo os resultados esperados. Eles servem para verificar se a funcionalidade está se comportando conforme esperado, sob um ponto de vista de negócios.

Ao se trabalhar com Testes de Aceitação automatizados e executados frequentemente, reduzem-se os riscos de que mudanças no produto, muito comuns em projetos Ágeis, gerem erros que passem desapercebidos. Existem inúmeras ferramentas para fazer essa automatização, e diversas permitem sua escrita em linguagem humana.

A seguir, vemos a aplicação de exemplos de dados aos Critérios de Aceitação definidos anteriormente. Para efeitos didáticos, os Testes de Aceitação estão descritos aqui de forma textual. Para a User Story e os Critérios do exemplo anterior:

Critério #1: somente podemos aceitar cartões de crédito com bandeiras com que temos convênio.

Testes de aceitação:

- Comprador de Livros utiliza cartão de crédito Visa
 - Aceitou = correto.
 - Recusou = errado, deve ser corrigido!
- Comprador de Livros utiliza cartão de crédito MasterCard
 - Aceitou = correto.
 - Recusou = errado, deve ser corrigido!
- Comprador de Livros utiliza cartão de crédito Amex
 - Recusou = correto.
 - Aceitou = errado, deve ser corrigido!

Critério #2: somente podemos aceitar cartões de crédito com data de expiração no futuro.

Testes de aceitação:

- Comprador de Livros utiliza cartão de crédito com expiração em 01/01/2020
 - Aceitou = correto.
 - Recusou = errado, deve ser corrigido!
- Comprador de Livros utilizou cartão de crédito com expiração em 01/01/2000
 - Recusou = correto.
 - Aceitou = errado, deve ser corrigido!

Nos exemplos acima, pode-se ver claramente que as bandeiras de cartão aceitas são Visa e MasterCard, enquanto que Amex não.

Para alguns Critérios de Aceitação, como por exemplo “*o botão deve ser azul*”, não é possível gerar Testes de Aceitação e, assim, não podem ser automatizados. Por meio de testes exploratórios, os membros do Time de Desenvolvimento realizam a verificação desses critérios.

Os Critérios e Testes de Aceitação cobrem cada aspecto de negócios da User Story e, assim, geralmente são suficientes para documentá-las.

9.3.3 Criando boas User Stories

Bill Wake, autor do livro “*Extreme Programming Explored*”, descreveu em seu blog quais seriam as características de boas User Stories. Ele formou o acrônimo INVEST (“investir”, em inglês) com a primeira letra de cada uma dessas características, afirmando que devemos “investir em boas User Stories” (Wake, 2003).

De acordo com Wake, uma User Story deve ser:

- **independente:** User Stories devem ser o mais independentes ou desacopladas possível umas das outras, ou seja, User Stories com grande número de dependências em outras User Stories devem ser evitadas. Essa independência visa ser viável alterar livremente a ordem que serão desenvolvidas e, ao fazê-lo, não ser necessário alterar suas estimativas.

Cabe adicionar que uma User Story se traduzirá sempre em uma funcionalidade de ponta a ponta, que representa valor para o usuário. Além disso, deve ser possível entender uma User Story sem ser necessário ler quaisquer outras;

- **negociável e negociada:** seus detalhes serão discutidos, negociados e definidos entre as pessoas de negócios (com o Product Owner, entre elas) e o Time de Desenvolvimento. São as “Conversas”, dos três C’s;
- **valiosa:** devem representar valor de negócio para os clientes do projeto. Ao dividir-se uma User Story, o resultado dessa divisão deve ser User Stories menores que também representem funcionalidades de ponta a ponta, e não partes de trabalho técnico;
- **estimável:** o Time de Desenvolvimento deve possuir detalhes suficientes – tanto técnicos quanto de negócios – para estimar o trabalho de se transformar a User Story em parte do produto, de forma que o Product Owner possa ordená-la apropriadamente.

Assim, conversas suficientes devem ser realizadas entre as pessoas de negócios e os membros do Time de Desenvolvimento para se obter os detalhes de negócios da User Story a ser desenvolvida. Os membros do Time de Desenvolvimento devem discutir possíveis soluções e executar pequenos experimentos, caso necessário, para reduzir os riscos técnicos da User Story. É claro que o nível de detalhes necessário depende de quão próxima a User Story está de ser colocada em desenvolvimento.

Estimativas fornecem o custo de desenvolvimento de uma User Story. Mesmo caso não se utilizem estimativas, os detalhes são necessários para dividir as User Stories de forma que fiquem pequenas o suficiente para serem desenvolvidas;

- **pequena** (“*small*”, em inglês) ou dimensionada apropriadamente: apenas User Stories pequenas e com um bom nível de detalhes podem ser colocadas em desenvolvimento. Quanto mais para baixo no Product Backlog a User Story estiver, maior ela será, a princípio.

User Stories pequenas próximas a seu desenvolvimento têm maior precisão em suas estimativas, permitem um melhor ordenamento do Product Backlog e representam um menor risco ao possibilitarem que mais itens façam parte de um Sprint;

- **testável** – deve ser possível verificar e confirmar que a User Story está pronta, ou seja, que foi transformada em parte do produto e está funcionando conforme esperado. A verificação é realizada por meio dos Critérios e Testes de Aceitação. É a “Confirmação”, dos três C’s.

9.3.4 Épicos e Temas

O Product Backlog é ordenado, de forma que seus itens são tão menores e mais detalhados quanto mais acima estiverem nessa lista. Os itens da parte superior do Product Backlog são pequenos e possuem detalhes suficientes que os permitem serem aceitos no próximo Sprint para desenvolvimento. Os itens mais abaixo são cada vez maiores e menos detalhados.

Épicos são User Stories que representam itens grandes demais ou sem detalhes suficientes para serem desenvolvidos. Enquanto épicos, essas User Stories somente necessitarão de mais detalhes quando ganharem prioridade.

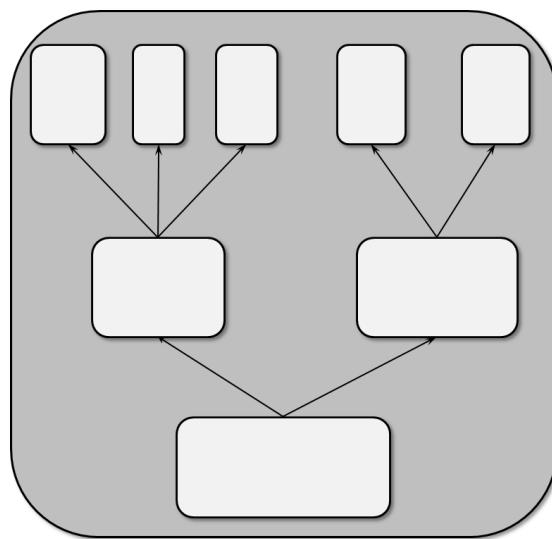


Figura 9.13: Épicos evoluem para User Stories menores

À medida que ganha prioridade, o épico ou parte dele evolui para User Stories menores (veja a figura 9.13).

Temas são coleções ou conjuntos de User Stories que estão relacionadas e, assim, podem ser agrupadas. Razões para agrupar User Stories em temas incluem, entre outras:

- juntas definem uma meta de negócios específica;
- são provenientes de um mesmo épico;
- pertencem a um Time de Desenvolvimento em um ambiente com múltiplos times trabalhando a partir de um mesmo Product Backlog.

O agrupamento de User Stories em temas relacionados a metas de negócios pode facilitar o trabalho de ordenação do Product Backlog em planejamentos mais longos, mantendo-se o foco em quais metas são mais importantes de serem atingidas primeiro ao invés de manter-se o foco em itens individuais.

Cada tema geralmente possui um curto título que o identifica.

CAPÍTULO 10

Sprint Backlog

10.1 O QUE É O SPRINT BACKLOG?

O Sprint Backlog é uma lista de itens selecionados do alto do Product Backlog para o desenvolvimento do Incremento do Produto no Sprint (o quê), adicionada de um plano de como esse trabalho será realizado (como).

10.1.1 O quê?

A lista de itens é escolhida pelo Time de Desenvolvimento e negociada com o Product Owner na reunião de Sprint Planning, a partir do alto do Product Backlog, de acordo com a ordenação realizada pelo Product Owner. Ela representa uma previsão de que itens os membros do Time de Desenvolvimento acreditam que conseguirão entregar naquele Sprint. Para construí-la, o Time de Desenvolvimento coopera com o Product Owner para obter detalhes suficientes e necessários para ser capaz de escolher uma quantidade de itens, desde o alto do Product Backlog, que acredite que preencha sua capacidade de trabalho em um Sprint. Com frequência, Times de

Desenvolvimento utilizam sua Velocidade como parâmetro para essa escolha (veja “Velocidade do Time de Desenvolvimento” em “[9.2.2. Planejável](#)”).

Um Product Backlog bem ordenado terá em seu topo itens que, juntos, levam a se alcançar a necessidade de negócios mais importante para os clientes naquele momento (veja “[9.2.1. Ordenado](#)”). Essa necessidade de negócios a ser alcançada, após ajustada de acordo com a capacidade de produção do Time de Desenvolvimento, se transformará na Meta do Sprint. Assim, os itens para o Sprint Backlog serão desenvolvidos, do mais importante ao menos importante, de forma a se atingir a Meta do Sprint (veja “[13.2. Meta de Sprint](#)”).

10.1.2 Como?

O plano de como os itens do Sprint Backlog serão desenvolvidos é geralmente expresso por um conjunto de tarefas correspondente a cada item, além da indicação do andamento de cada tarefa, ou seja, se ela ainda não foi iniciada, se está em andamento ou se já está terminada. Estimativas do tempo de desenvolvimento de cada tarefa podem ser adicionadas para que o Time de Desenvolvimento possa acompanhar seu progresso em direção ao final do Sprint com o uso de Gráficos de Sprint Burndown, mas diversos times simplesmente contam o número de tarefas restantes com esse propósito (veja “[14.3. Gráfico de Sprint Burndown](#)”).

Essas tarefas são estabelecidas na própria reunião de Sprint Planning pelo Time de Desenvolvimento. Elas representam os pequenos passos, em geral técnicos, necessários para que o item esteja pronto, de acordo com a Definição de Pronto (veja “[11. Definição de Pronto](#)”). Assim, quando todas as tarefas de um item estão terminadas, o item está pronto. O formato de tarefas, embora largamente utilizado, não é prescrito pelo Scrum.

10.2 COMO É O SPRINT BACKLOG?

Um Quadro de Tarefas real, como o da figura [10.1](#), é considerado uma boa representação de Sprint Backlog. O quadro é bem visível, na própria sala de trabalho do Time de Desenvolvimento, e notas adesivas são, em geral, utilizadas para representar os itens e suas tarefas correspondentes. Colunas separam os itens das tarefas, e classificam as tarefas de acordo com seu status: “a fazer”, “fazendo” e “feito” (ou quaisquer termos semelhantes). Esse tipo de quadro é classificado como um “irradiador de informação”, ou seja, a informação chega aos olhos de quem é importante chegar sem haver um esforço significativo para buscá-la. É muito mais eficiente, por exemplo,

do que um programa de computador, que necessita de ser acessado para mostrar a informação.

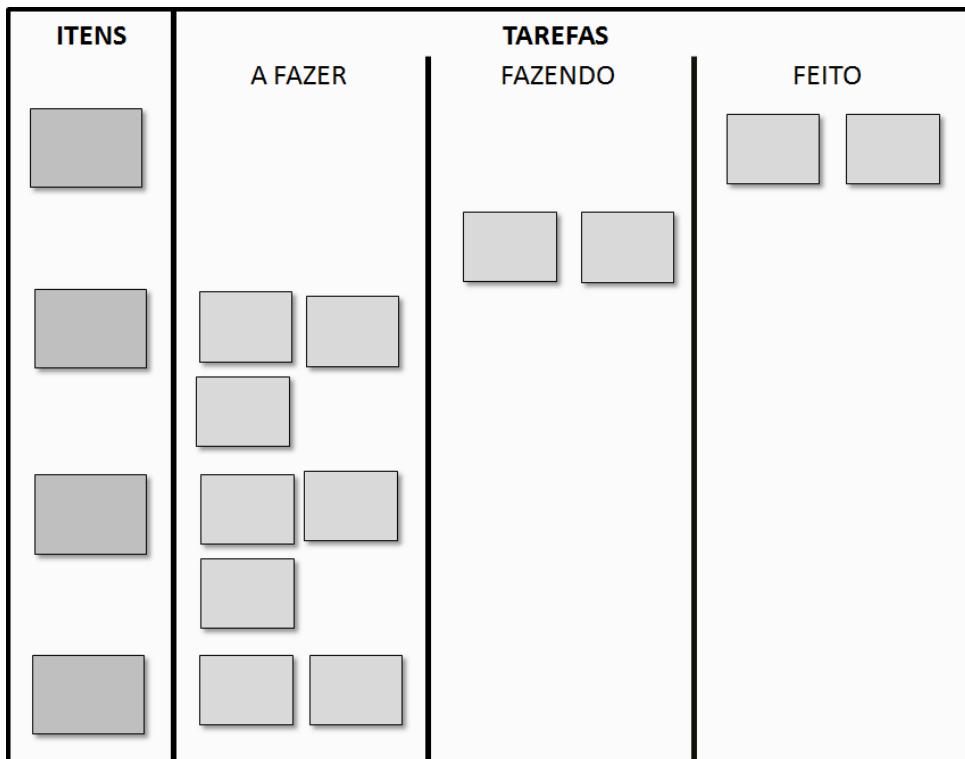


Figura 10.1: Sprint Backlog é mais bem representado em um quadro real

O Sprint Backlog pertence ao Time de Desenvolvimento, que é responsável por seu uso e manutenção. Ele deve refletir o momento atual e, para isso, o Time de Desenvolvimento o atualiza sempre que houver qualquer mudança, mesmo que no andamento de uma tarefa, por exemplo. O Sprint Backlog não deve ser utilizado por outras pessoas como instrumento de controle – nem mesmo pelo Product Owner. É um artefato que buscar maximizar a visibilidade do trabalho, de forma que o próprio Time de Desenvolvimento possa acompanhar seu progresso em busca da Meta do Sprint.

Durante o Sprint, os membros do Time de Desenvolvimento trabalham a partir do item mais importante e em direção ao menos importante, visando atingir a Meta do Sprint. Embora em geral trabalhem em tarefas diferentes, eles colaboram

buscando tornar prontos – de acordo com a Definição de Pronto – os itens mais importantes primeiro. A alternativa indesejável seria cada membro ou conjunto de membros trabalhar em itens diferentes, de acordo com áreas de conhecimento ou algum outro critério de divisão (veja a figura 10.2). Além de não estimular o trabalho em equipe e o compartilhamento de conhecimento, essa abordagem carrega o risco de se chegar ao final do Sprint sem os itens mais importantes prontos e, portanto, sem se atingir a Meta do Sprint.

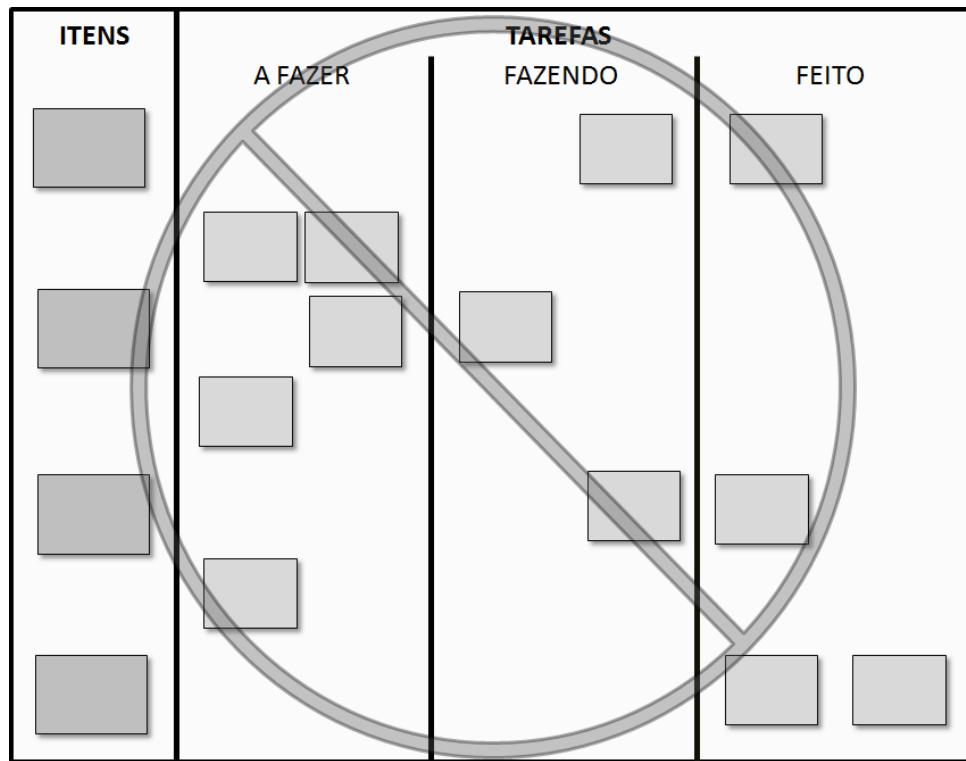


Figura 10.2: Divisão indesejável de trabalho

O Time de Desenvolvimento utiliza o melhor de seu conhecimento no momento do planejamento para criar o Sprint Backlog. Espera-se, no entanto, que esse plano evolua e seja atualizado no decorrer do Sprint, de forma a refletir o trabalho restante. Ou seja, alterações no plano como, por exemplo, a adição, reestimativa ou remoção de tarefas são aceitáveis. No entanto, uma quantidade de mudanças que dificulte ou até mesmo torne impossível de se atingir a Meta do Sprint indicam, em geral, um

planejamento para o Sprint mal executado.

CAPÍTULO 11

Definição de Pronto

11.1 O QUE É A DEFINIÇÃO DE PRONTO?

A Definição de Pronto é um acordo formal entre Product Owner e Time de Desenvolvimento sobre o que é necessário para se considerar que um trabalho realizado no Sprint está “pronto”. São, portanto, critérios definidos por ambos para garantir a transparência, por meio da compreensão compartilhada do que significa quando o Time de Desenvolvimento afirma que qualquer item ou o Incremento do Produto está “pronto”.

A mesma Definição de Pronto se aplica a cada item do Sprint Backlog e ao Incremento do Produto como um todo, que é o conjunto dos itens desenvolvidos na Sprint. Ela guia o Time de Desenvolvimento durante o trabalho no Sprint e serve ao Product Owner para que, ao final do Sprint, ele possa verificar se o que foi produzido está de fato pronto.

PRONTO?

Um desenvolvedor de *software* entrega uma funcionalidade que considera pronta. Ao começar a experimentar o que lhe foi entregue, o usuário dessa funcionalidade encontra um erro. Ele então pergunta ao desenvolvedor:

- Você testou isso?

Ao que o desenvolvedor responde:

- Não...

- Mas você não disse que está pronto?

- Sim, está pronto. Só falta testar. – afirma o desenvolvedor.

- Se falta testar, não está pronto!

Situações como essa são comuns em projetos onde não há uma compreensão conjunta do que “estar pronto” significa.

Uma Definição de Pronto ideal estabelece que o resultado do trabalho de um Time de Desenvolvimento em um Sprint seja entregável. Ou seja, idealmente nenhum desenvolvimento, teste, integração ou quaisquer tarefas adicionais devem ser necessárias para que o Incremento do Produto produzido no Sprint possa ser entregue aos clientes do projeto. Entregar ou não ao final do Sprint, no entanto, é uma decisão de negócios e, mesmo que já seja possível, cabe ao Product Owner decidir quando o fazer.

A Definição de Pronto é criada antes do início do desenvolvimento do produto. Ela pode ser modificada e evoluir ao longo do desenvolvimento do produto, à medida que novas necessidades são identificadas ou que se aproxime, de fato, o “pronto” do “entregável”. No entanto, diferentemente dos Testes e Critérios de Aceitação, a Definição de Pronto não é específica para um item ou para um grupo de itens. Ela é a mesma para todos os itens e Incrementos do Produto gerados.

Uma boa Definição de Pronto exige, preferencialmente de forma explícita, que cada item passe nos Testes de Aceitação – e, portanto, nos Critérios de Aceitação – acordados especificamente para o item. Uma boa Definição de Pronto também exige que o Incremento do Produto tenha qualidade suficiente para ser entregue.

Para a construção de uma Definição de Pronto efetiva é importante que se considerem as restrições organizacionais que afetam o trabalho do Time de Scrum, sejam restrições de negócio, de processo, tecnológicas ou culturais. Devido a essas restrições, muitos Times de Scrum iniciam o projeto com uma Definição de Pronto

distante da ideal: após o término do Sprint, algum trabalho adicional ainda será necessário para tornar o resultado do trabalho entregável. Esse trabalho é geralmente realizado apenas imediatamente antes da Release. Exemplos incluem certos tipos de testes (no caso de *software*, são comuns testes de estabilidade, segurança e integração), integrações com outros produtos, alguns tipos de documentação e validações, como auditorias externas.

A existência desse trabalho adicional é uma disfunção e traz riscos consideráveis ao projeto, já que diversos problemas serão descobertos tarde. Na medida do possível, o Time de Scrum trabalha ao longo do projeto para reduzir a disfunção, transferindo progressivamente esse trabalho para dentro dos Sprints e assim, tornando sua Definição de Pronto cada vez mais estrita.

Em casos muito particulares – em se tratando, por exemplo, de projetos de alto risco em que são necessários testes adicionais e auditorias externas – o trabalho adicional antes da Release existirá durante todo o projeto.

11.2 COMO É A DEFINIÇÃO DE PRONTO?

A Definição de Pronto varia de time para time, de projeto para projeto. O Time de Desenvolvimento possui, entre seus membros, todas as habilidades e conhecimentos necessários para entregar, ao final do Sprint, um Incremento do Produto pronto de acordo com a Definição de Pronto. Assim, é fácil entender como a Definição de Pronto e a formação do Time de Desenvolvimento são intrinsecamente relacionadas.

A figura 11.1 mostra uma Definição de Pronto possível para um sistema de *software*. Como no exemplo, a Definição de Pronto geralmente se parece com uma lista de atividades. Essa lista é definida de acordo com as necessidades do produto – das quais qualidade sempre faz parte – e em conformidade com as convenções e padrões organizacionais. Essas atividades incluem tudo o que deve ser realizado para se construir o produto, como por exemplo diferentes tipos de testes que se considerem necessários, documentação que faça parte do produto, quaisquer integrações que devam ser realizadas etc.

Definição de Pronto

Somente consideraremos o Incremento do Produto “pronto” se estiver:

- Codificado
- Passando nos testes unitários
- Passando nos testes de aceitação
- Integrado ao sistema XPTO
- Com o manual do usuário atualizado

Figura 11.1: Um exemplo de Definição de Pronto

As atividades da Definição de Pronto, portanto, são consideradas quando o Time de Desenvolvimento seleciona quanto trabalho prevê que conseguirá produzir no Sprint. Também são consideradas quando o Time de Desenvolvimento estabelece o plano para o Sprint, quebrando os itens do Sprint Backlog em tarefas, por exemplo.

A Definição de Pronto é criada, compreendida e compartilhada por todos os membros do Time de Scrum. Mantê-la visível para eles, portanto, é essencial.

CAPÍTULO 12

Incremento do Produto

Em cada Sprint do projeto, o Time de Desenvolvimento trabalha nos itens selecionados para o Sprint Backlog, do mais importante para o menos importante, visando atingir a Meta do Sprint. O Incremento do Produto é o resultado desse trabalho, ou seja, é a soma de todos os itens completos no Sprint.

É um incremento de funcionalidades do produto prontas, de acordo com a Definição de Pronto (veja “[11. Definição de Pronto](#)”), que o Time de Desenvolvimento produz em toda e cada Sprint do projeto. Esse incremento é composto por novas funcionalidades e por melhorias no que foi produzido anteriormente.

Espera-se que o Incremento do Produto seja entregável, de forma que o Product Owner possa decidir por fazer uma Release para os clientes do projeto ao final do Sprint. O Product Owner pode, no entanto, optar por acumular alguns Incrementos do Produto para só então fazer uma Release. Um Incremento nem sempre representa valor suficiente para ser utilizado por seus usuários. Ele pode ser incompleto, portanto, e nesse caso não há sentido em entregá-lo ao final do Sprint em que foi produzido. Ainda que represente valor suficiente, diferentes razões podem justificar

que não seja entregue ao final de cada Sprint: os clientes podem não conseguir absorver mudanças tão frequentemente, ou mesmo questões políticas, burocráticas ou técnicas podem impedir o Product Owner de realizar a Release imediatamente.

O Time de Desenvolvimento e o Product Owner demonstram o Incremento do Produto para os clientes do projeto e pessoas relevantes ao final de cada Sprint, na reunião de Sprint Review. O principal objetivo dessa reunião é obter *feedback* dessas pessoas sobre o trabalho realizado e dar a elas um senso do progresso do projeto. Para tornar esse *feedback* possível, o Incremento do Produto deve representar valor visível para eles, ou seja, funcionalidades que podem ser experimentadas e utilizadas.

CAPÍTULO 13

Metas de negócios

13.1 O QUE SÃO METAS DE NEGÓCIOS?

Seja no contexto de um Sprint, de uma Release ou do projeto como um todo, o Time de Desenvolvimento trabalha do item de maior importância para o item de menor importância, em busca de um objetivo ou meta de negócios clara e alcançável. Como os itens mais importantes para se alcançar essa meta estabelecida são desenvolvidos primeiro, ela pode ser satisfeita mesmo que, apesar de todo o esforço do Time de Desenvolvimento, nem todos os itens previstos inicialmente tenham sido desenvolvidos, pois apenas restarão itens de menor importância.

A meta ou objetivo no Scrum não é um valor financeiro, uma data ou uma determinada quantidade de trabalho. É, na verdade, uma necessidade ou objetivo de negócios a ser alcançado como resultado de um trabalho.

As metas ou objetivos de negócios funcionam ao mesmo tempo como guias e motivadores. Elas são definidas pelo Product Owner e ajustadas com o Time de Desenvolvimento. Juntos, eles as tornam factíveis, de forma que o Time de Desenvolvimento possa se comprometer a atingi-las e trabalhar em direção a elas.

As metas ou objetivos geralmente utilizados no Scrum são:

- Meta do Sprint (contexto de um Sprint): é parte integrante do Scrum e é obrigatoriamente estabelecida na reunião de Sprint Planning de cada Sprint do projeto. Cada Meta de Sprint significa um passo em direção à Meta da Release ou de Roadmap, e cada funcionalidade desenvolvida pelo Time de Desenvolvimento significa um passo em direção à Meta do Sprint;
- Meta da Release ou de Roadmap (contexto de uma Release ou de um marco do Roadmap): a Meta da Release, caso utilizada, existe para cada Release do projeto e é estabelecida antes do início da Release (possivelmente durante a reunião de Release Planning). Caso o Time de Scrum realize Releases com alta frequência – a cada Sprint ou em entregas contínuas, por exemplo – as Metas de Release podem ser substituídas por metas de mais alto nível, que aqui chamamos de Metas de Roadmap. Elas são expressas em um Roadmap do Produto. Cada Meta de Release ou de Roadmap significa um passo em direção à Visão do Produto;
- Visão do Produto (contexto do produto): é o objetivo maior a ser alcançado por meio do desenvolvimento do produto. Ela é estabelecida antes de se iniciarem os Sprints e existe ao longo de todo o desenvolvimento.

A Visão do Produto não é parte do *framework* Scrum. Mas, caso utilizada, provê contexto para o trabalho do Time de Scrum e alinhamento entre os membros do Time de Scrum, os clientes do projeto e as demais partes interessadas. Da mesma forma, as Metas de Release ou de Roadmap não são obrigatórias nem são parte do Scrum, mas facilitam a ligação entre as Metas dos Sprints e a Visão do Produto.

O Roadmap do Produto é um plano em alto nível que pode ser utilizado para se ter uma visão de como se dará a evolução do produto no futuro. Esse plano é expresso em uma linha do tempo com datas e objetivos a serem alcançados. Cada um desses objetivos pode ser expresso como uma Meta de Release ou de Roadmap.

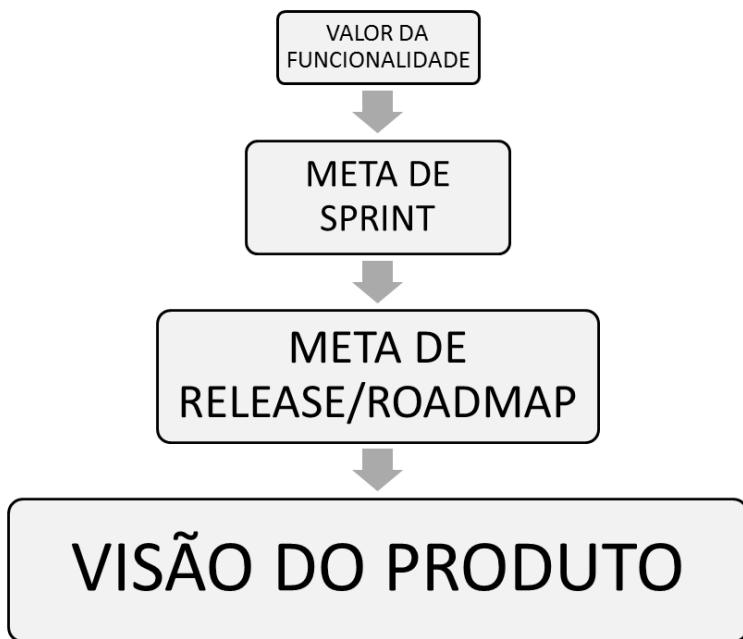


Figura 13.1: Do valor da funcionalidade à Visão do Produto

O conjunto de Metas de Release ou de Roadmap e cada Meta de Sprint, estabelecida no início de cada Sprint, representam a estratégia de negócios do Product Owner para alcançar a Visão do Produto, por meio do trabalho do Time de Desenvolvimento. Assim, cada Meta de Release ou de Roadmap definida é um passo em direção à Visão do Produto, e cada Meta de Sprint acordada entre Product Owner e Time de Desenvolvimento é um passo em direção à Meta da Release ou de Roadmap. O valor de negócio de cada funcionalidade produzida durante o Sprint pelo Time de Desenvolvimento é um passo em direção à Meta do Sprint e, assim, é um pequeno passo em direção à Visão do Produto (veja a figura 13.1).

Essa estratégia de negócios definida pelo Product Owner é, assim como o próprio desenvolvimento do produto, iterativamente definida, detalhada, reavaliada e modificada de acordo com o *feedback* obtido sobre o produto e mudanças em seu ambiente, por exemplo.

13.2 META DE SPRINT

13.2.1 O que é a Meta de Sprint?

Em cada Sprint, o Time de Desenvolvimento produz valor visível para os clientes do projeto. Assim, eles podem prover, na reunião de Sprint Review, um *feedback* sobre o Incremento do Produto gerado. A Meta do Sprint se traduz nesse valor a ser produzido.

A Meta do Sprint é estabelecida e acordada entre Product Owner e Time de Desenvolvimento durante a reunião de Sprint Planning de cada Sprint. O Product Owner geralmente chega à reunião trazendo definida a necessidade de negócios dos clientes mais importante naquele momento. Colabora-se então para ajustá-la de acordo com a capacidade de produção do Time de Desenvolvimento, de onde se estabelece a Meta do Sprint.

Essa Meta guia, dá propósito, motiva e mantém o foco do trabalho do Time de Desenvolvimento no Sprint. Ela determina qual objetivo de negócios é imprescindível que seja atendido pelo Incremento do Produto que estará pronto ao final do Sprint. A Meta do Sprint, dessa forma, estimula e reforça a auto-organização, já que o Time de Desenvolvimento define como a atingir e se torna responsável por fazê-lo acontecer.

A Meta do Sprint é um passo em direção à Meta da Release ou de Roadmap, e assim é também um passo em direção à Visão do Produto. A Visão do Produto é o objetivo maior a ser alcançado no projeto, é a direção para a qual o projeto está caminhando. Ela, no entanto, é grande e vaga e, portanto, é difícil de estabelecer, apenas com o trabalho sobre os itens do Product Backlog, se a direção para a qual o projeto está caminhado é a correta.

A Meta do Sprint é um alvo muito menor e mais tangível tanto para o Time de Desenvolvimento quanto para o Product Owner. Ela provê uma conexão entre o valor de negócio dos itens do Product Backlog e a Meta da Release ou de Roadmap, que por sua vez a conecta com a Visão do Produto.

Durante o Sprint, os itens do Sprint Backlog serão transformados em um Incremento do Produto, visando alcançar a Meta do Sprint. Esses itens serão desenvolvidos do mais importante para se alcançar a Meta ao menos importante. O desenvolvimento de todos os itens do Sprint Backlog não é, portanto, o objetivo final do Time de Desenvolvimento no Sprint, mas sim a Meta do Sprint.

A Meta do Sprint oferece alguma flexibilidade para o trabalho do Time de Desenvolvimento, que se faz necessária devido à natural imprecisão no planejamento do

Sprint. Assim, embora o Time de Desenvolvimento faça o seu melhor para concluir todo o trabalho planejado, muitas vezes ele não será capaz de fazê-lo. No entanto, o que o Time de Desenvolvimento produzir deve obrigatoriamente atingir a Meta do Sprint. Caso sobrem itens não prontos ao término do tempo de desenvolvimento do Sprint, estes serão sempre os menos importantes, o que aumenta as chances da Meta ter sido atingida. Ao final do Sprint, o Product Owner irá avaliar se o trabalho realizado pelo Time de Desenvolvimento foi suficiente para atingir a Meta do Sprint ou não.

Se os itens selecionados para o Sprint Backlog tiverem pouca ou nenhuma relação uns com os outros, será muito difícil se estabelecer uma Meta coerente para o Sprint. Essa desconexão entre os itens escolhidos para o Sprint pode sinalizar, na realidade, que não há uma estratégia clara de produto sendo utilizada, ou seja, não se está caminhando em direção a uma Visão de Produto. Esse problema geralmente ocorre quando o Product Owner, por diversas razões, não está realizando seu trabalho como deve. Além disso, o trabalho sem um propósito claro, realizado a partir de uma sequência de partes pouco relacionadas, é geralmente desmotivador.

A Meta de um Sprint, uma vez definida na reunião de Sprint Planning, não pode mais ser alterada. Para garantir a integridade do Sprint, o ScrumMaster tem o papel de assegurar que não seja feita durante o Sprint nenhuma mudança que possa afetar a Meta do Sprint. No entanto, caso a sua Meta perca o sentido, o Sprint pode ser imediatamente cancelado (veja “[16.3. O Sprint pode ser cancelado?](#)”).

13.2.2 Como é a Meta de Sprint?

Pode-se dizer que a Meta funciona como um tema para o Sprint, que permeia o trabalho do Time de Desenvolvimento no decorrer do Sprint. Ela não é um item selecionado do Product Backlog. Ela também não é a reescrita dos itens do Sprint Backlog por extenso, como “desenvolver X, Y e Z” ou “desenvolver esses n itens”. É, na realidade, uma e apenas uma necessidade de negócios razoavelmente ampla a ser alcançada a partir do desenvolvimento dos itens selecionados para o Sprint Backlog, do mais importante para o menos importante. Assim, a Meta do Sprint é alcançada por meio do conjunto de valores de negócio dos itens produzidos no Sprint.

Um formato que propomos neste livro para as Metas de Sprint é:

“[Por meio deste Sprint,] possibilitaremos <QUEM> a <O QUÊ>”.

Onde <QUEM> pode ser uma categoria de usuários ou um cliente ou parte interessada específica etc. e <O QUÊ> define qual é o objetivo ou necessidade de negócios a ser atendida. Podemos ver exemplos abaixo para produtos de *software*:

“Possibilitaremos o Comprador a se registrar no sistema”.

“Possibilitaremos ao Administrador um login mais seguro”.

No primeiro exemplo acima, o item mais importante do Sprint Backlog tratará da funcionalidade de cadastro mais essencial (nome e contato, por exemplo), enquanto que itens mais abaixo tratarão de questões relacionadas com gradativa menor importância para essa Meta, como outros campos não essenciais ou a inserção de uma foto do usuário, por exemplo. Entre os itens, pode também haver, por exemplo, ajustes de Sprints anteriores, como correções de problemas ordenadas de acordo com sua importância relativa aos outros itens, e até mesmo itens pouco ou nada relacionados, esses em geral com baixa importância.

Alguns Times de Desenvolvimento se colocam na posição de <QUEM> ao realizarem um Sprint inteiro dedicado ao aprendizado. Por exemplo: *“Por meio deste Sprint, queremos aprender como realizar uma conexão remota ao sistema XXX”*. Essa, no entanto, não é considerada uma boa prática, já que todo Sprint deve ter como objetivo produzir valor para os clientes, embora itens de aprendizado possam fazer parte do Sprint.

13.3 META DE RELEASE/ROADMAP

13.3.1 O que é a Meta de Release/Roadmap?

A Meta da Release é um objetivo ou necessidade de negócios de alto nível a ser alcançada por meio do trabalho do Time de Desenvolvimento para uma entrega (Release). Ela representa um passo em direção à Visão do Produto.

A Meta da Release geralmente existe quando se trabalha com uma Release planejada. Ela forma, juntamente com a data da Release e um conjunto de itens selecionados, o Plano da Release (veja “[22.1. O que é a Release Planning?](#)”).

A Meta de Roadmap é um objetivo ou necessidade de negócios de alto nível em direção à Visão do Produto e é similar à Meta da Release. Sua diferença é que ela representa um marco no Roadmap do Produto (uma data, aproximada ou exata) e não é necessariamente alcançada no momento de uma entrega.

A Meta de Roadmap é útil quando não faz sentido se ter uma Meta de Release. Por exemplo, quando se realizam Releases em cada Sprint, caso em que a Meta da Release e a Meta do Sprint seriam a mesma. Outro exemplo ocorre quando se trabalha entrega contínua, onde cada Meta de Release corresponderia ao valor de negócios do item entregue. Nesses casos, a granularidade da Meta da Release seria muito fina e, assim, distante demais da Visão do Produto (veja “[21.2. Como é a Release?](#)”).

A Meta de Release também pode representar um marco no Roadmap do Produto. Nesses casos, cada um desses marcos corresponde a uma entrega. Veja no exemplo da figura [13.2](#) um Roadmap de um produto de *software* para vendas pela Internet.



Figura 13.2: Roadmap do Produto com Metas de Release

Assim como a Visão do Produto, a Meta da Release ou de Roadmap fornece contexto, orientação, motivação e inspiração para todo o trabalho de desenvolvimento do produto até o momento da Release ou da chegada do marco do Roadmap.

13.3.2 Como é a Meta de Release/Roadmap?

Não existem formatos prescritos para as Metas de Release ou de Roadmap. Essas Metas são necessidades de negócio a serem atendidas em uma dada janela de tempo, e essas necessidades podem ser descritas a partir de seus usuários ou de seus clientes. Um formato que sugerimos neste livro para as Metas de Release ou de Roadmap, similar ao da Meta do Sprint, é:

“Por meio da Release/Até <QUANDO>, possibilitaremos <QUEM> a <O QUÊ>.”

Onde <QUANDO> se refere a um identificador de Release (número ou nome, por exemplo) ou à data da Release ou do marco do Roadmap, <QUEM> pode ser

uma categoria de usuários ou um cliente ou parte interessada específica etc. e <O QUÊ> define qual é a necessidade de negócios a ser atendida. Podem-se ver dois exemplos abaixo para produtos de *software* distintos:

“Por meio da Release #3, possibilitaremos o Comprador a ter uma experiência de compra integrada com redes sociais”.

“Até o final do primeiro semestre, possibilitaremos o cliente X a oferecer seus produtos para os usuários do sistema”.

13.4 ROADMAP DO PRODUTO

13.4.1 O que é o Roadmap do Produto?

O Roadmap do Produto é um plano em alto nível de como o produto irá evoluir ao longo do tempo até um momento futuro determinado, que pode ser o final previsto para o projeto ou algum momento futuro relevante. Assim, ele é útil apenas quando é possível se prever, em alto nível, como o produto irá evoluir no futuro, mesmo que apenas em termos de objetivos a serem alcançados.

Em um projeto com Scrum, o Product Owner é o responsável por criar, manter e comunicar o Roadmap do Produto. O Product Owner geralmente cria o Roadmap do Produto em uma sessão de trabalho em que podem estar presentes diferentes partes interessadas e Time de Desenvolvimento.

O Roadmap do Produto facilita o diálogo entre o Time de Scrum, os clientes e as demais partes interessadas sobre a evolução do produto, indicando a todos o que se pretende alcançar, em alto nível, ao longo do projeto. Ele é também útil na estratégia da organização, uma vez que ajuda a coordenar o desenvolvimento e entregas de produtos relacionados, além de outras atividades necessárias.

13.4.2 Como é o Roadmap do Produto?

O Roadmap do Produto é uma linha do tempo que contém marcos, que são datas exatas ou aproximadas no futuro e objetivos do produto a serem alcançados até cada uma delas (veja a figura 13.3).

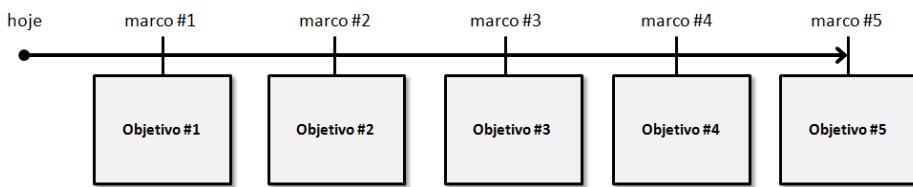


Figura 13.3: Roadmap do Produto

O Roadmap do Produto em um projeto que utiliza Scrum mostra o caminho em direção à Visão do Produto. Assim, sugerimos que se indique em cada um de seus marcos a meta de negócios que se quer alcançar, que é uma ideia da Meta de Release ou de Roadmap esperada. Podem-se também adicionar a cada marco objetivos parciais, sejam técnicos ou de negócios. Embora não seja recomendável, é também aceitável haver mais de uma Meta de Release ou de Roadmap por marco, em geral direcionadas a diferentes classes de usuários, clientes ou demais partes interessadas.

13.5 VISÃO DE PRODUTO

13.5.1 O que é a Visão do Produto?

A Visão do Produto é um objetivo ou necessidade de negócios de alto nível que fornece contexto, orientação, motivação e inspiração para o trabalho de desenvolvimento do produto durante todo o projeto. É o objetivo maior a ser alcançado pelo Time de Scrum, que se traduz na satisfação dos clientes (Pichler, 2010).

A Visão do Produto provê um alinhamento entre todos os envolvidos no projeto, desde os clientes e demais partes interessadas até o Time de Scrum, sobre o que deve ser alcançado. Para que esse entendimento comum seja possível, é necessária uma Visão do Produto de fácil compreensão e, assim, é desejável que seja concisa e clara, contendo apenas o necessário e suficiente.

A Visão do Produto é estabelecida antes que o desenvolvimento do produto se inicie e, de forma geral, permanece estável durante todo o projeto. Ela é criada, gerenciada e compartilhada pelo Product Owner, que garante que o Product Backlog esteja sempre alinhado com ela. No entanto, o Time de Desenvolvimento, os clientes e quaisquer outras pessoas relevantes interessadas podem estar diretamente envolvidos no refinamento dessa Visão.

13.5.2 Como é a Visão do Produto?

Geoffrey Moore, no seu livro “*Crossing the Chasm*” (Moore, 2001), apresenta um modelo interessante para a Visão do Produto, o chamado “Teste do Elevador”. A ideia é que seja possível explicar o que é o produto durante a subida de um elevador, ou seja, em um tempo bastante curto. Adaptado por Jim Highsmith, esse modelo tem o seguinte formato:

“**Para** (cliente-alvo),
que (problema ou oportunidade),
o (nome do produto) **é um** (categoria do produto)
que (benefício-chave, razão convincente para utilizar).
Ao contrário de (alternativa primária competitadora),
nosso produto (diferenciação primária).”

Um exemplo de aplicação para um *site* de relacionamentos pode ser visto em seguida:

“**Para** usuários de Internet solteiros em busca de um relacionamento sério,
que estão insatisfeitos com encontros mal sucedidos e a oferta de numerosas opções pouco criteriosas,
o LoveFinder **é um** *site* de relacionamentos amorosos
que os ajuda a encontrar sua alma gêmea.
Ao contrário de *sites* de encontros populares,
nosso produto oferece pretendentes compatíveis com as preferências do usuário, apresentando detalhes suficientes que incluem fotos.”

Outro exemplo, agora para um aplicativo móvel de viagens, pode ser visto abaixo:

“**Para** turistas usuários de smartphone,
que desejam aproveitar melhor seus locais de destino,
o MyTrip **é um** aplicativo móvel de viagens
que sugere roteiros diários flexíveis de acordo com seu perfil.
Ao contrário de guias de viagens com roteiros predefinidos,
nosso produto elabora trajetos personalizados e adaptáveis.”

O “Teste do Elevador” é um enunciado de Visão do Produto efetivo ao informar quem são os clientes-alvo do produto, qual o problema desses clientes ou oportunidade de mercado que será suprida, um nome e categorização que fornecem um posicionamento do produto no mercado, uma razão convincente para utilizar o produto e a diferenciação do produto diante das alternativas existentes.

CAPÍTULO 14

Gráficos de Acompanhamento do Trabalho

Os Gráficos de Acompanhamento do Trabalho são ferramentas para a visualização do progresso do cumprimento de uma quantidade mensurável estimada de trabalho em um determinado tempo. Eles servem para criar visibilidade de uma forma simples, ajudando a rapidamente identificar problemas e, assim, reduzir os riscos do projeto.

Os Gráficos de Burndown de Trabalho são os mais utilizados por Times de Scrum com esse propósito. A ideia de “*burndown*” (queima) é que o trabalho previsto é “queimado” em direção a um determinado momento no tempo, formando um gráfico de inclinação negativa, ou seja, para baixo.

Os Gráficos de Burndown de Trabalho envolvem, portanto, alguma unidade de trabalho restante no eixo y e alguma unidade de tempo no eixo x (veja a figura 14.1). Unidades utilizadas para trabalho restante incluem número de tarefas restantes, quantidade de horas das tarefas restantes, número de itens restantes, e quanti-

dade de story points dos itens restantes. Unidades comuns para tempo incluem dias, semanas e Sprints, entre outros.

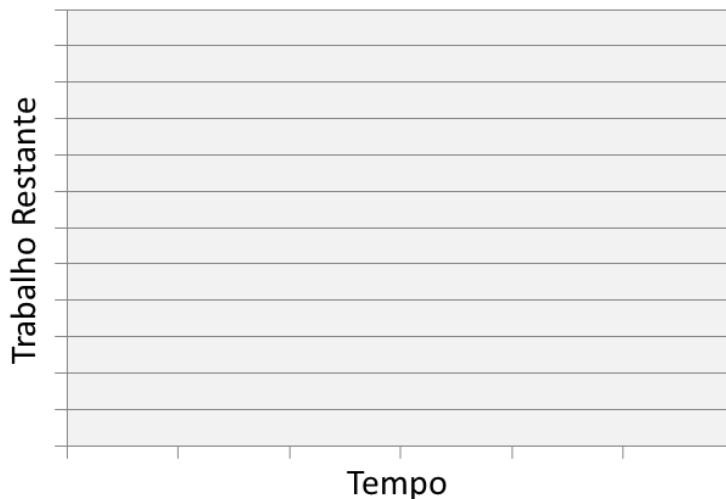


Figura 14.1: Gráfico de Burndown de Trabalho: Trabalho Restante x Tempo

Os Gráficos de Release Burndown e de Sprint Burndown são os dois tipos de Gráficos de Burndown de Trabalho mais utilizados por Times de Scrum.

Os Gráficos de Burnup de Trabalho, diferentemente dos de Burndown, mostram a quantidade de trabalho já realizado no tempo e o trabalho total a se realizar. O Gráfico de Burnup de trabalho mais utilizado por Times de Scrum é o Gráfico de Release Burnup.

14.1 GRÁFICO DE RELEASE BURNDOWN

14.1.1 O que é o Gráfico de Release Burndown?

O Release Burndown é um gráfico mantido pelo Product Owner e utilizado tanto pelo Product Owner quanto pelo Time de Desenvolvimento para monitorar o progresso no desenvolvimento em direção a uma entrega (Release). O Gráfico de Release Burndown não é parte do *framework* Scrum, mas pode ser útil quando se utiliza um Plano de Release, geralmente estabelecido em uma reunião de Release Planning.

Ao avaliar o gráfico, o Product Owner pode decidir mudanças de rumo, como mudanças no escopo ou na Meta da Release, adiamento da Release e até mesmo seu cancelamento.

O Gráfico de release Burndown mostra trabalho restante previsto para uma Release (eixo y), correspondente aos itens do Product Backlog selecionados para essa Release (a soma de seus story points, por exemplo), ao final de cada Sprint dessa Release (eixo x). O Gráfico de Release Burndown é uma maneira rápida e prática de se visualizar o andamento da Release.

A figura 14.2 mostra um exemplo de Release Burndown antes do início do último Sprint de uma Release. Nesse exemplo, a quantidade inicial de trabalho é de 235 story points, a serem queimados em seis Sprints.

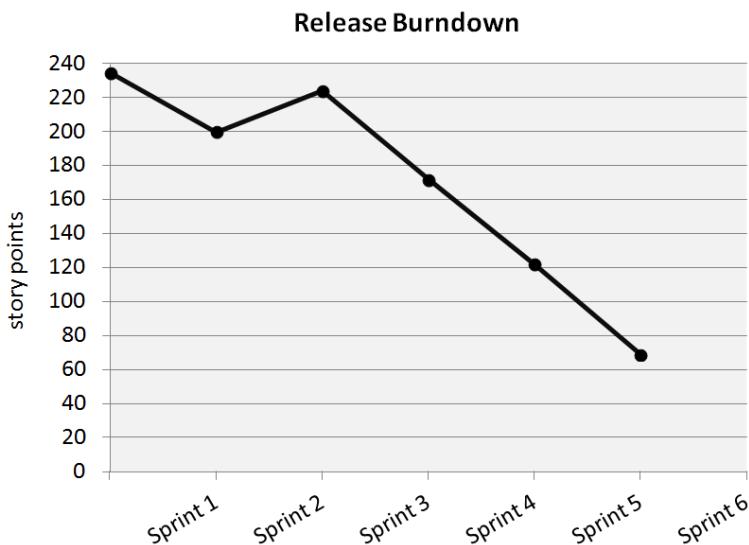


Figura 14.2: Um exemplo de Release Burndown ao final do penúltimo Sprint da Release

Nos exemplos desta seção, utilizaremos Story Points como unidade de “trabalho restante previsto”. Os Story Points são atribuídos pelo Time de Desenvolvimento aos itens do Product Backlog selecionados para a Release. Há, no entanto, diversas outras unidades que podem ser utilizadas (veja “[9.2.2 Planejável](#)”, no capítulo sobre o Product Backlog).

Para “tempo”, utilizaremos os Sprints previstos para a Release. Assim, cada ponto do gráfico mostrará a soma dos story points restantes estimados dos itens do Product Backlog selecionados para a Release, ao final de cada Sprint dela.

No exemplo da figura 14.2, podemos verificar pelo gráfico que, nos Sprints 3, 4 e 5, o Time de Desenvolvimento “queimou” 52, 50 e 53 story points respectivamente, atingindo quase uma constância. Definimos que, antes do início do sexto e último Sprint da Release atual, a Velocidade do Time de Desenvolvimento (ou seja, a taxa esperada de “queima” de story points por Sprint) é a média inteira dos pontos entre-gues nesses três últimos Sprints, ou seja, 52 pontos por Sprint.

O Gráfico de Release Burndown é criado na reunião de Release Planning ou logo antes do primeiro Sprint e é atualizado ao final de cada Sprint com o valor do trabalho previsto restante naquele momento.

14.1.2 Como é o Gráfico de Release Burndown?

O exemplo da figura 14.3 mostra um Gráfico de Release Burndown em story points que acaba de ser atualizado logo após o final do terceiro Sprint (de um total de seis previstos) da Release.

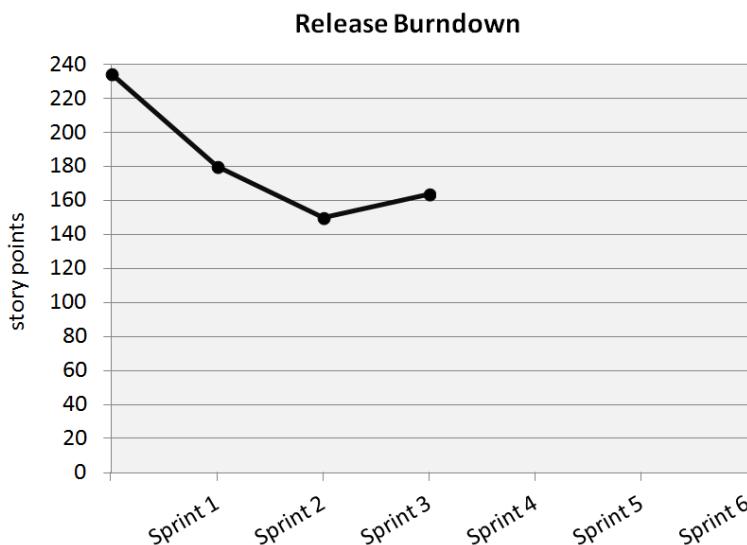


Figura 14.3: Release Burndown com inclinação para cima no final do terceiro Sprint

Observe que a inclinação súbita para cima, de 150 para 164 story points, serve como sinal de alerta, indicando que mais story points entraram do que foram “queimados” na Release durante Sprint 3. Essa subida pode acontecer quando há uma combinação de um ou mais dos seguintes fatores: mudanças no escopo da Release com a inclusão de novas funcionalidades, mudanças nas estimativas dos itens restantes do Product Backlog e parte significativa do Sprint não entregue.

Uma forma prática de se verificar o andamento da Release é traçando-se uma linha reta iniciando três Sprints antes do Sprint que acaba de se encerrar, que no exemplo é o ponto inicial do gráfico em ([Início], 235), e o ponto atual, que é ([final do Sprint 3], 164), e prolongando essa linha até encontrar o eixo x. Esse ponto no eixo x é o momento em que se projeta que o Time de Desenvolvimento terá “queimado” todos os story points planejados, mantendo-se a Velocidade atual. Pode-se ver pela projeção da figura 14.4 que, nesse momento, as maiores chances são que o trabalho não estará terminado ao final do Sprint 6. Esse cenário de possível atraso provavelmente levará Time de Desenvolvimento e Product Owner e, possivelmente, Product Owner e clientes, a negociarem sobre o futuro da Release.

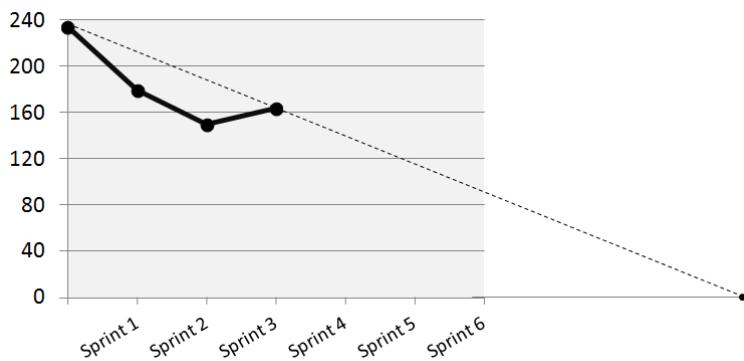


Figura 14.4: Verificando o andamento da Release

É importante destacar que não se espera que o Time de Desenvolvimento complete todos os itens definidos inicialmente para a Release. O Time de Desenvolvimento trabalha a partir dos mais importantes em direção aos menos importantes previstos para a Release. Esses itens vão evoluindo para itens menores e mais detalhados à medida que o trabalho segue e eles ganham importância, e partes de menor importância são movidas para baixo no Product Backlog. Novos itens também

surgirão. Dessa forma, espera-se que os que sobrarem ao final da Release sejam os menos importantes, assim aumentando as chances de que a Meta da Release tenha sido atingida.

14.2 GRÁFICO DE RELEASE BURNUP

14.2.1 O que é o Gráfico de Release Burnup?

O Release Burnup, assim como o Release Burndown, é um gráfico mantido pelo Product Owner e utilizado tanto pelo Product Owner quanto pelo Time de Desenvolvimento para monitorar o progresso no desenvolvimento em direção a uma entrega (Release). O Gráfico de Release Burnup também não é parte do *framework* Scrum, mas pode ser útil quando se utiliza um Plano de Release, geralmente estabelecido em uma reunião de Release Planning.

O Gráfico de release Burnup mostra duas linhas. Uma das linhas representa o trabalho total previsto para a Release, correspondente aos itens escolhidos para o Plano de Release. A outra linha representa o trabalho já realizado na Release (apenas itens prontos de acordo com a Definição de Pronto). Assim, à medida que o Time de Desenvolvimento trabalha nos Sprints dessa Release, a quantidade de trabalho realizado sempre aumenta, aproximando-se do trabalho total previsto. Esse, no entanto, pode variar, quando novos itens são introduzidos, quando itens previstos são divididos em itens menores e quando estimativas são refeitas.

Dessa forma, o Gráfico de Release Burnup mostra mais informações que o de Burndown: ele mostra a variação no total de trabalho previsto para a Release.

A figura 14.5 mostra um exemplo de Release Burnup ao final do quinto Sprint da Release. Repare que, nesse exemplo, a quantidade total de trabalho, representada pela linha pontilhada, sobe ao longo da Release, o que fica visível com o gráfico.

Nos exemplos desta seção, utilizaremos Story Points como unidade de “trabalho realizado” e “trabalho total”, que são atribuídos pelo Time de Desenvolvimento aos itens do Product Backlog selecionados para a Release. No entanto, outras unidades podem ser utilizadas.

Para “tempo”, utilizaremos os Sprints previstos para a Release. Assim, cada ponto do gráfico mostrará a soma dos story points dos itens já desenvolvidos do Product Backlog nessa Release, ao final de cada Sprint dessa Release.

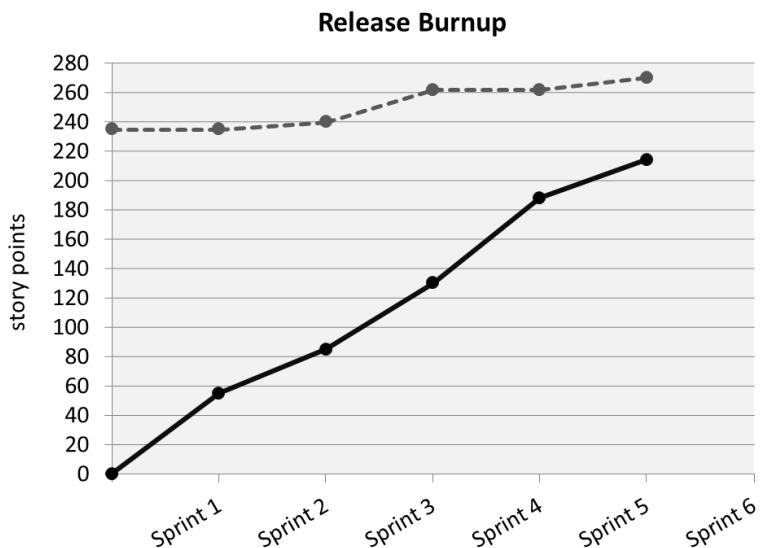


Figura 14.5: Um exemplo de Release Burnup no final do penúltimo Sprint da Release

Outra forma comum de se definir o trabalho já realizado e o trabalho total é utilizando-se o valor de negócio de cada item. Esse valor de negócio é estabelecido pelo Product Owner ou diretamente pelos clientes do projeto. É uma unidade relativa, ou seja, obtém-se o valor de negócio de um item comparando-o com o valor de negócio de outros. Nesse caso, o trabalho total ao final de cada Sprint é representado no Release Burnup pelo valor de negócios total esperado naquele momento, e o trabalho já realizado é representado pelo valor de negócios já gerado até aquele momento.

O Gráfico de Release Burnup é criado na reunião de Release Planning ou logo antes do primeiro Sprint e é atualizado ao final de cada Sprint.

14.2.2 Como é o Gráfico de Release Burnup?

Para efeitos didáticos, o exemplo da figura 14.6 mostra um Gráfico de Release Burndown e um Gráfico de Release Burnup juntos. Os gráficos mostram o trabalho em story points, e foram atualizados logo após o final do quarto Sprint (de um total de cinco) da Release. O gráfico pontilhado é o total de trabalho previsto, o gráfico na descendente é o Gráfico de Burndown e o gráfico na ascendente é o Gráfico de Burnup.

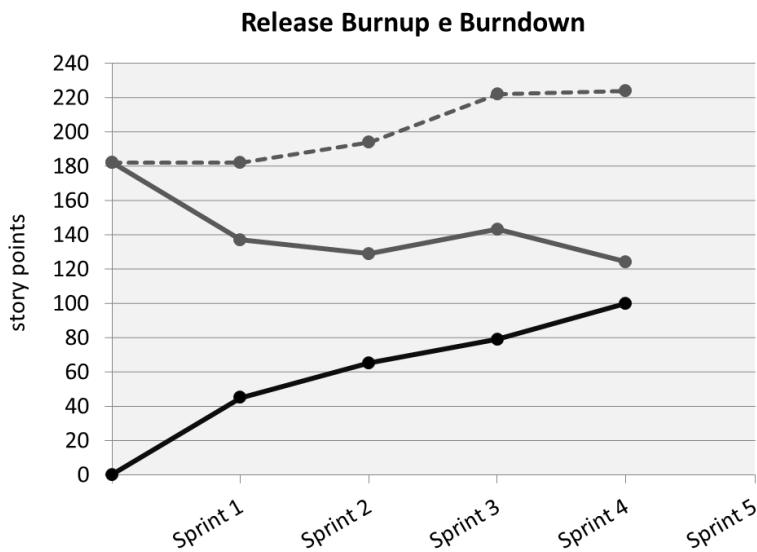


Figura 14.6: Release Burnup e Burndown juntos

À medida que o trabalho é realizado Sprint a Sprint, o Gráfico de Burnup acumula mais pontos e sobe, aproximando-se da linha que representa o trabalho total. Quanto mais próximo dessa linha o Gráfico de Burnup estiver ao final da Release, maiores são as chances de se ter alcançado a Meta da Release.

Observe que a inclinação súbita para cima do Gráfico de Burndown durante o Sprint 3 é melhor explicada pelo Gráfico de Burnup, onde se pode ver que a quantidade de trabalho total cresceu mais do que o trabalho realizado naquele Sprint. Esse é um sinal de alerta importante para o Product Owner, já que pode significar uma ameaça à Meta da Release.

Novamente é importante destacar que não se espera que o Time de Desenvolvimento complete todos os itens definidos inicialmente para a Release, mas sim que atinja sua meta, trabalhando a partir dos itens mais importantes.

14.3 GRÁFICO DE SPRINT BURNDOWN

14.3.1 O que é o Gráfico de Sprint Burndown?

O Sprint Burndown é um gráfico mantido e utilizado pelo Time de Desenvolvimento para monitorar seu progresso no desenvolvimento em direção ao final de um Sprint.

Ele mostra o trabalho restante estimado para um Sprint (eixo y) em cada dia de trabalho para o desenvolvimento do produto no Sprint (eixo x). Embora não seja parte integrante do *framework* Scrum, o Gráfico de Sprint Burndown é a maneira mais rápida e prática de se visualizar o andamento do Sprint. A figura 14.7 mostra um exemplo de Sprint Burndown ao final de um Sprint bem sucedido.

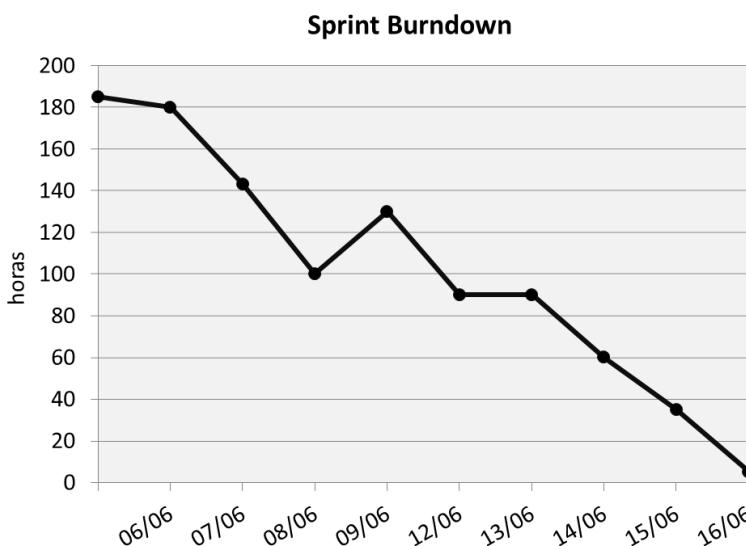


Figura 14.7: Gráfico de Sprint Burndown ao final de um Sprint bem sucedido

A estratégia mais comum para a definição de um plano para o Sprint é a de se quebrar os itens selecionados para o Sprint Backlog em tarefas. Nesse caso, o Time de Desenvolvimento pode estimar as tarefas individualmente e utilizar essas estimativas para traçar o Gráfico de Sprint Burndown.

Se o Time de Desenvolvimento estima as tarefa em horas de trabalho, então cada ponto do gráfico mostrará a soma das horas estimadas de todas as tarefas restantes (tanto aquelas não iniciadas quanto as que já estão sendo desenvolvidas) que fazem parte do Sprint Backlog versus o tempo, representado por cada dia útil de desenvolvimento do Sprint.

Caso o Time de Desenvolvimento utilize uma unidade não numérica, como por exemplo os “tamanhos de camisas” (*T-Shirt Sizing*), pode-se estabelecer uma proporção entre os valores da escala (no exemplo, 1G = 2M = 4P) e somar os valores

das tarefas restantes em termos da menor unidade da escala (no caso, P) para traçar os pontos do gráfico. Outra forma é simplesmente designar um valor para cada unidade da escala (como $P = 1$, $M = 2$ e $G = 4$).

Se o Time de Desenvolvimento não estima as suas tarefas, pode-se somar o número de tarefas restantes em cada dia e marcar no gráfico. Outra alternativa é utilizar-se a estimativa de cada item restante do Sprint Backlog (em Story Points, por exemplo). Nesse caso, geralmente não se consideram pontos parciais. Assim, somente se consideram os pontos de um item como “queimados” quando o item está pronto, de acordo com a Definição de Pronto, o que gera saltos no gráfico. Caso o Time de Desenvolvimento utilize uma estratégia diferente e não divida os itens do Sprint Backlog em tarefas, é importante que ele busque outra forma de contabilizar o trabalho restante em cada momento para ser capaz de monitorar seu progresso em direção ao final do Sprint.

O Gráfico de Sprint Burndown é criado ao final da reunião de Sprint Planning ou antes da primeira reunião de Daily Scrum e é atualizado diariamente com o valor do trabalho restante naquele momento. Alguns Times de Desenvolvimento preferem atualizá-lo logo antes de cada reunião de Daily Scrum, enquanto que outros preferem fazê-lo no início ou no final do dia. O último ponto do gráfico é marcado quando o desenvolvimento se encerra no Sprint.

14.3.2 Como é o Gráfico de Sprint Burndown?

No exemplo da figura 14.8, o Time de Desenvolvimento utiliza horas de trabalho para estimar suas tarefas. Vamos imaginar que o Gráfico de Sprint Burndown é atualizado logo antes do início da reunião de Daily Scrum, e que essa reunião ocorre no meio da manhã de cada dia de desenvolvimento (de um total de nove). A reunião de Sprint Planning foi realizada na metade do primeiro dia e o primeiro ponto (sobre o eixo y) reflete a soma do total de horas de todas as tarefas definidas na reunião. O segundo ponto do gráfico (dia 06/06) reflete a quantidade de horas estimadas para as tarefas restantes (ainda não desenvolvidas ou em andamento) logo antes da primeira reunião de Daily Scrum do Sprint. O gráfico acaba de ser atualizado no quinto dia do Sprint e a reunião de Daily Scrum está para começar. Observe que o último ponto do gráfico (16/06) será atualizado no último dia do Sprint, logo antes da reunião de Sprint Review.

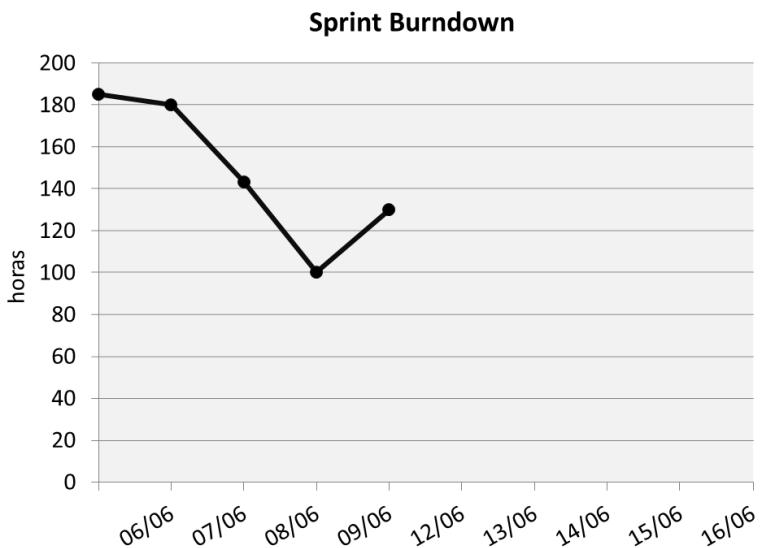


Figura 14.8: Gráfico de Sprint Burndown com inclinação para cima

A inclinação súbita para cima no dia 09/06, de 100 para 135 horas de trabalho, serve como sinal de alerta, podendo indicar que ocorreu algum problema entre a reunião de Daily Scrum dos dias 08/06 e 09/06. A reestimativa de tarefas já existentes ou a introdução de novas tarefas são perfeitamente normais e esperadas, mas aqui alguma dessas questões pode ter ocorrido em excesso, ter somado-se a algum impedimento ao trabalho do Time de Desenvolvimento ou as duas coisas ao mesmo tempo.

Nesse outro exemplo da figura 14.9, vemos pelo Gráfico de Sprint Burndown que o Time de Desenvolvimento trabalha em Sprints de quatro semanas e utiliza o número de tarefas como unidade de trabalho restante. Vemos, no exemplo, que a quantidade de tarefas restantes se manteve a mesma por três dias seguidos, formando uma linha reta.

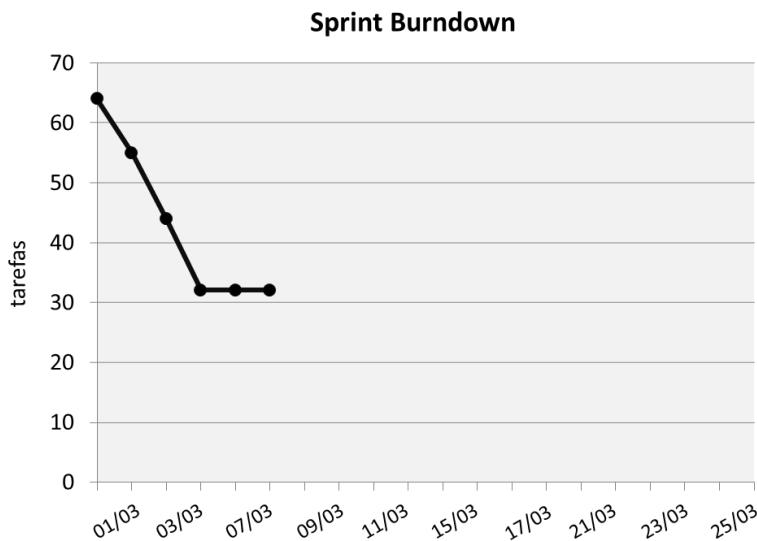


Figura 14.9: Gráfico de Sprint Burndown com linha reta

Da mesma forma que no caso anterior, esse comportamento também serve como sinal de alerta, já que o Time de Desenvolvimento está cada vez mais distante de “queimar” o trabalho estimado restante.

É importante destacar que, mesmo que o Time de Desenvolvimento chegue ao final de um Sprint com tarefas sobrando, ainda assim poderá ter cumprido seu compromisso. Embora dê o seu melhor para completar o trabalho, o compromisso do Time de Desenvolvimento é o de cumprir a Meta do Sprint e, mesmo que nem todos os itens selecionados tenham sido completados, ainda assim essa Meta poderá ter sido atingida. Assim, não é incomum se chegar ao final de um Sprint sem que o Sprint Burndown chegue ao zero do eixo y e mesmo assim pode-se considerar o Sprint um sucesso.

Nos dois exemplos da figura 14.10, vemos que, ao final do Sprint, não foi “queimado” todo o trabalho estimado.

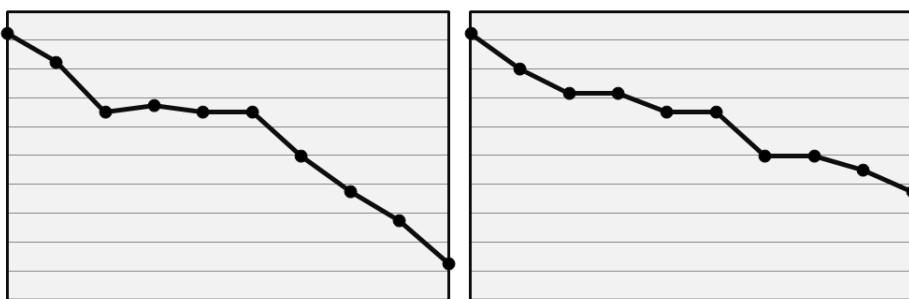


Figura 14.10: Dois exemplos de gráficos que não chegam a zero trabalho restante

São, no entanto, dois cenários com resultados bastante distintos. No primeiro caso, o Time de Desenvolvimento provavelmente entregou valor suficiente para atingir a Meta do Sprint, o que será determinado pelo Product Owner ao final do Sprint. Mas, no segundo caso, a Meta do Sprint dificilmente terá sido alcançada, já que muito deixou de ser feito.

O Gráfico de Sprint Burndown é criado pelo Time de Desenvolvimento, para o Time de Desenvolvimento. Ele serve para sinalizar para o Time de Desenvolvimento o quanto perto ou o quanto longe ele está de cumprir as tarefas planejadas para o Sprint, e não deve – de forma alguma – servir como instrumento de cobrança para o Product Owner, o ScrumMaster ou qualquer parte interessada do projeto exercer pressão sobre o Time de Desenvolvimento.

14.3.3 Linha ideal

A linha ideal serve como guia visual para o comportamento do Gráfico de Sprint Burndown ao longo do Sprint. A linha ideal é opcional e, caso utilizada, é traçada no momento que o Gráfico de Sprint Burndown é criado. É uma linha diagonal que liga a quantidade inicial de trabalho restante sobre o eixo y ao último dia do Sprint sobre o eixo x, ou seja, ela liga (*[primeiro dia do Sprint], [trabalho total]*) a (*[último dia do Sprint], [zero trabalho restante]*). A figura 14.11 mostra a linha ideal.

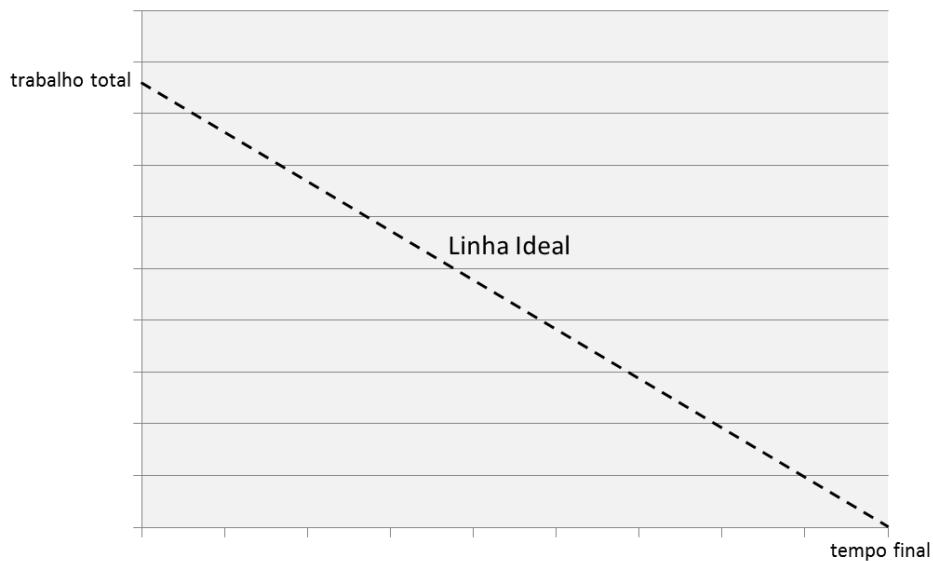


Figura 14.11: Linha ideal

É natural, ao longo do Sprint, o gráfico oscilar em torno da linha ideal. Um exemplo de oscilação saudável pode ser visto na figura 14.12. Repare que o gráfico oscila em torno da linha ideal sem, no entanto, se distanciar muito dela.

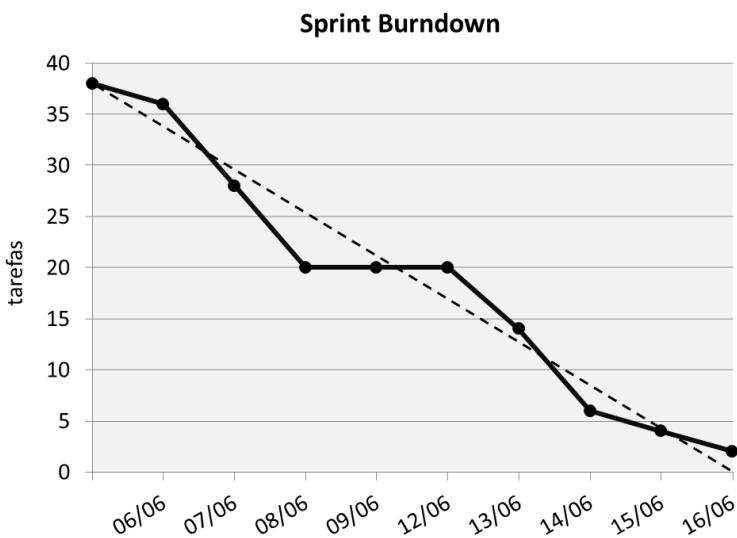


Figura 14.12: Oscilação natural do gráfico em torno da linha ideal

No entanto, se o ponto do gráfico referente ao momento atual estiver muito acima da linha, o Time de Desenvolvimento ficará em alerta. Quanto mais próximo do final do Sprint isso acontecer, em mais risco a Meta do Sprint estará. No exemplo da figura 14.13, a situação parece crítica a três dias do final do Sprint.

Por diversas razões, a linha ideal é muito criticada por vários autores e praticantes de Scrum, e muitos preferem não a utilizar. Alguns acreditam que essa linha somente serve de instrumento de pressão sobre o Time de Desenvolvimento e a chamam de “linha da opressão”. Outros acreditam que, ao buscar adequar-se ao comportamento da linha ideal, o Time de Desenvolvimento acaba por não ser honesto em suas estimativas. Outra questão é que, ao traçar-se a linha ideal, está se considerando uma irreal precisão na definição e nas estimativas iniciais das tarefas. Um percentual significativo de tarefas surge no decorrer do Sprint e diversas tarefas são reestimadas, fazendo com que o ponto inicial da linha ideal (e, portanto, a própria linha) não tenha significado real.

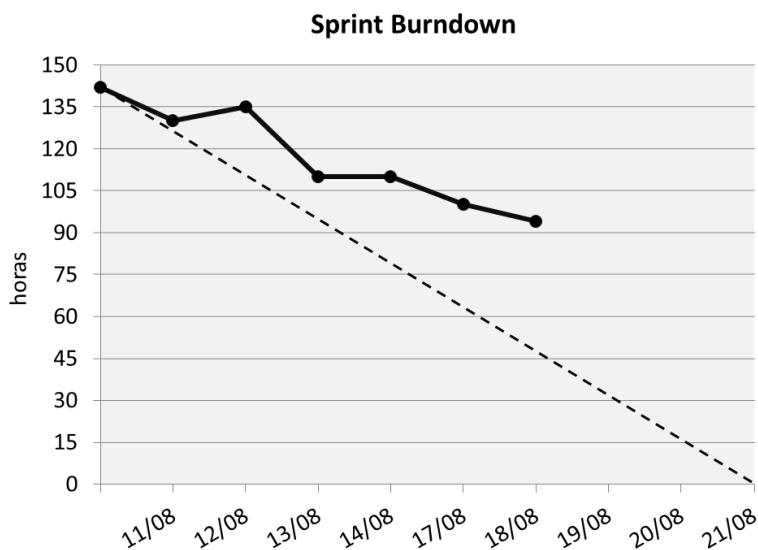


Figura 14.13: Nesse momento, a distância do ponto atual à linha ideal indica uma situação crítica

A linha ideal, no entanto, pode cumprir seu propósito se compreendida apenas como um guia visual para o comportamento do Gráfico de Sprint Burndown.

A linha ideal também pode ser utilizada para o Gráfico de Release Burndown, ligando a quantidade inicial de trabalho previsto sobre o eixo y ao final do último Sprint da Release sobre o eixo x, ou seja, ela liga (*[início do primeiro Sprint da Release], [trabalho total]*) a (*[final do último Sprint da Release], [zero trabalho restante]*).

CAPÍTULO 15

Definição de Preparado

15.1 O QUE É A DEFINIÇÃO DE PREPARADO?

O principal resultado esperado das sessões de Refinamento do Product Backlog, realizadas pelo Product Owner em conjunto com o Time de Desenvolvimento, é que um número suficiente de itens a serem considerados na reunião de Sprint Planning seguinte esteja preparado para ser desenvolvido. Esse trabalho de Refinamento do Product Backlog é realizado, ao longo do Sprint, em sessões agendadas ou esse pode ser um trabalho contínuo (veja “[23. Refinamento do Product Backlog](#)”).

A preparação é importante para diminuir os riscos de um Sprint mal planejado, já que de outra forma detalhes demais são deixados para ser discutidos na reunião de Sprint Planning. Esse trabalho excessivo torna a reunião longa, cansativa e ineficiente, levando-a muitas vezes ao fracasso e colocando em risco todo o trabalho do Sprint.

A Definição de Preparado é um artefato utilizado para garantir que os itens a serem considerados na reunião de Sprint Planning estejam preparados segundo um

critério bem definido. É um acordo formal entre Product Owner e Time de Desenvolvimento sobre o estado em que um item do Product Backlog deve estar para estar qualificado para discussão na reunião de Sprint Planning. Caso um dos itens de alta prioridade não chegue à reunião de Sprint Planning preparado de acordo com a Definição de Preparado, o item será rejeitado pelo Time de Desenvolvimento.

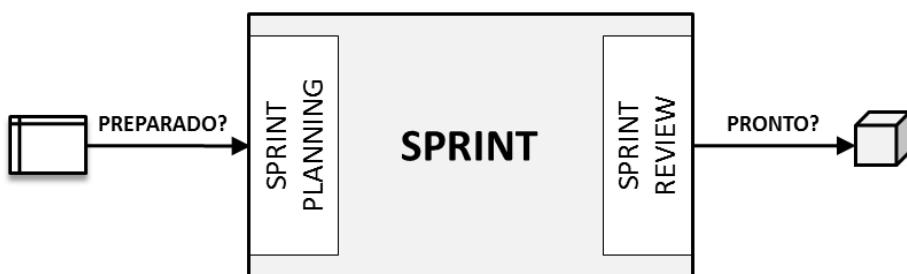


Figura 15.1: Fluxo do item: Definição de Preparado e Definição de Pronto

Pode-se observar na figura 15.1 o fluxo de um item do Product Backlog através de um Sprint. Caso o item do Product Backlog esteja preparado de acordo com a Definição de Preparado (trabalho que é feito nas sessões de Refinamento do Product Backlog), ele pode ser aceito para discussão na reunião de Sprint Planning, e consequentemente pode fazer parte do Sprint. Se, ao final do Sprint, o item estiver pronto de acordo com a Definição de Pronto, ele é aceito na reunião de Sprint Review e sai do Sprint como parte do Incremento do Produto gerado no Sprint.

Discussão: Sprint Planning ineficiente

O Product Owner inicia o Sprint Planning apresentando os itens mais importantes do Product Backlog, um a um. O Time de Desenvolvimento, que ainda não conhecia esses itens, faz perguntas de todo o tipo, que são prontamente respondidas pelo Product Owner.

Os Critérios de Aceitação de cada item ainda não foram discutidos e, para alguns dos itens, sequer foram criados. O Product Owner não teve tempo de prepará-los previamente. Como é a primeira vez que tem acesso a esses itens, o Time de Desenvolvimento também ainda não os estimou.

Time de Desenvolvimento e Product Owner então consideram item a item para discutir e preparar os Critérios de Aceitação. É um trabalho detalhado, que gera

muitas ideias e discussão e, assim, mais um tempo considerável é necessário para esclarecimentos. Em seguida à definição dos Critérios de Aceitação de um item, o Time de Desenvolvimento realiza a atividade de Planning Poker para estimá-lo, de onde surgem novas dúvidas que, novamente, são prontamente esclarecidas pelo Product Owner.

O Time de Desenvolvimento logo identifica que um dos itens mais importantes é muito grande, e que provavelmente deve ser quebrado em itens menores. Product Owner e Time de Desenvolvimento então colaboram para dividi-lo, e decidem que uma de suas partes não tem importância para esse Sprint e deve voltar para o Product Backlog. Não demoram a identificar outro item a ser quebrado, e novamente realizam esse trabalho.

Para um dos itens mais importantes, o Time de Desenvolvimento entende que o Product Owner não possui detalhes suficientes para que seja colocado em desenvolvimento. O Product Owner procura negociar e até mesmo convencê-los a aceitar para o Sprint esse item importante, mas o Time de Desenvolvimento recusa, pois considera muito arriscado que um item com pouquíssimos detalhes faça parte de seu Sprint Backlog. Essa negociação eleva o nível de estresse da reunião e o Scrum-Master, enquanto facilitador, faz o seu melhor para que o conflito seja resolvido.

Após algumas horas de reunião, mais próximo do final do dia, Time de Desenvolvimento e Product Owner estão exaustos. Tantos detalhes foram discutidos e tantas negociações ocorreram que ambos sentem que o resto da reunião não será mais produtivo. Os itens que restam são discutidos mais rapidamente e, pelo cansaço, o Time de Desenvolvimento está mais propenso a aceitá-los com uma compreensão menor dos detalhes. Uma Meta para o Sprint é rapidamente negociada, e o Time de Desenvolvimento parte para a segunda parte da reunião, onde quebrará os itens em tarefas.

A exaustão toma conta do Time de Desenvolvimento, e o que seus membros mais querem é chegar ao fim do dia. As tarefas são geradas sem o cuidado e reflexão adequados, e o Sprint Backlog resultante não representa um bom plano para o Sprint.

Como consequência, o Sprint é bastante conturbado. Muitos esclarecimentos são necessários, e uma quantidade grande de novas tarefas são diariamente adicionadas ao Sprint Backlog. Ao final, o Time de Desenvolvimento não atinge a Meta do Sprint.

O que levou o Time de Desenvolvimento ao fracasso nesse Sprint? Será que uma preparação não o teria ajudado?

15.2 COMO É A DEFINIÇÃO DE PREPARADO?

A Definição de Preparado, assim como a Definição de Pronto, tem geralmente o formato de uma lista de critérios ou condições (veja a figura 15.2).

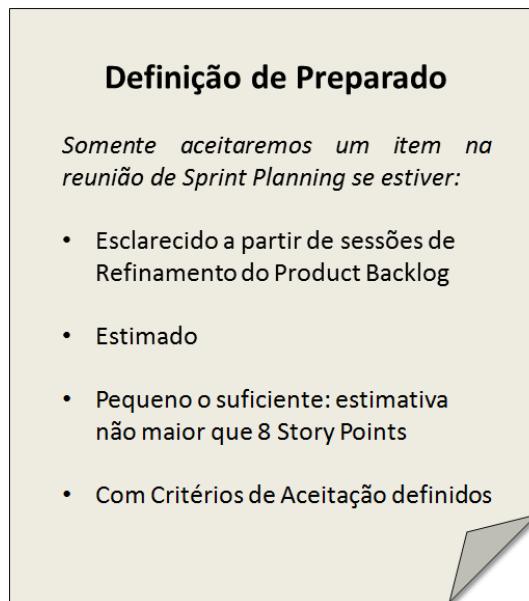


Figura 15.2: Exemplo de Definição de Preparado

Um exemplo comum de Definição de Preparado para um item do Product Backlog pode conter os seguintes tópicos:

- o item deve possuir Critérios de Aceitação, discutidos, compreendidos e acordados entre Product Owner e Time de Desenvolvimento;
- o item deve ter sido estimado pelo Time de Desenvolvimento;
- o item deve ser pequeno o suficiente, de acordo com algum critério estabelecido pelo Time de Desenvolvimento (um valor máximo na sua estimativa, por exemplo).

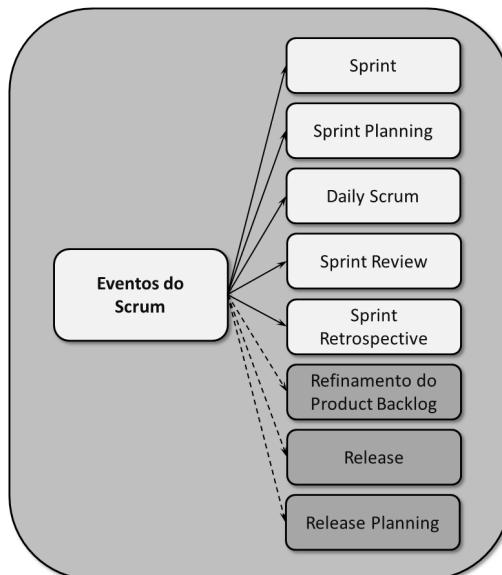
Assim como a Definição de Pronto, a Definição de Preparado é criada, compreendida e compartilhada por todos os membros do Time de Scrum e, assim, deve se

manter visível para todos eles.

Parte IV

Eventos do Scrum

Os eventos do Scrum são o próprio ciclo de desenvolvimento, chamado de Sprint, e as reuniões ou cerimônias realizadas durante o ciclo, que são: Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective. Adicionamos a essas as sessões de Refinamento do Product Backlog e as Releases, acompanhadas ou não de reuniões de Release Planning.



Os eventos do Scrum possuem uma duração definida, chamada de *timebox*. O *timebox* pode definir o tempo máximo em que um evento deve ocorrer, como nas reuniões, ou o tempo exato, como no Sprint. O objetivo dos *timeboxes* é limitar o tempo em que um objetivo deve ser alcançado, de forma que não se gera desperdício com trabalhos intermináveis. Os *timeboxes* ajudam a criar um ritmo ou uma regularidade no trabalho do Time de Desenvolvimento.

CAPÍTULO 16

Sprint

- Objetivo: atingir a Meta do Sprint;
- Quando: durante todo o desenvolvimento do produto, um atrás do outro;
- Duração: fixa de uma a quatro semanas;
- Participantes obrigatórios: Time de Desenvolvimento, Product Owner e ScrumMaster;
- Saídas esperadas: um Incremento do Produto pronto, de acordo com a Definição de Pronto, que atinja a Meta do Sprint.

16.1 O QUE É O SPRINT?

O Sprint é o ciclo de desenvolvimento, onde o Incremento do Produto pronto é gerado pelo Time de Desenvolvimento a partir dos itens mais importantes do Product Backlog.

Além do trabalho de desenvolvimento propriamente dito, ocorrem durante o Sprint a reunião de Sprint Planning, as reuniões de Daily Scrum, a reunião de Sprint Review, a reunião de Sprint Retrospective e quaisquer outras atividades ou reuniões realizadas com a participação do Time de Desenvolvimento. Todos esses eventos estão dentro do *timebox* do Sprint.

Até o seu final, o projeto com Scrum funciona inteiro dentro de Sprints, que acontecem um atrás do outro, sem paradas ou intervalos. Assim, não se para um Sprint após o seu final ou o interrompe por alguns dias para resolver questões relativas ao projeto. Tudo o que acontece em um projeto com Scrum é trazido para dentro do Sprint.

O Sprint possui um objetivo ou uma necessidade de negócios bem definida, negociada e acordada entre o Time de Desenvolvimento e o Product Owner durante a reunião de Sprint Planning. É a Meta do Sprint, que guia o trabalho do Time de Desenvolvimento, determinando qual objetivo de negócios é imprescindível que seja atendido pelo que estará pronto ao final do Sprint (veja “[13.2. Meta de Sprint](#)”). Essa Meta não pode ser modificada no decorrer do Sprint, mas ele pode ser cancelado caso sua Meta perca o sentido (veja “[16.3. O Sprint pode ser cancelado?](#)”).

Para alcançar a Meta do Sprint, o Time de Desenvolvimento desenvolve os itens do Sprint Backlog ao longo do Sprint, do mais importante ao menos importante.

SPRINT = MINIPROJETO

Cada Sprint pode ser entendido como um miniprojeto, no qual se pretende que determinados itens sejam desenvolvidos visando-se alcançar um objetivo de negócios bem definido – a Meta do Sprint. Para efeitos de compreensão, podemos dizer que essa Meta funciona como uma (mini) Visão do Produto a ser gerado nesse miniprojeto.

Os itens a serem desenvolvidos no Sprint são os mais importantes naquele momento e, portanto, são selecionados do alto do Product Backlog. Na prática, no entanto, esse trabalho de seleção não é necessariamente uma transposição direta de itens desde o alto do Product Backlog para o Sprint Backlog do Sprint. A partir do alto do Product Backlog, os itens serão negociados e selecionados de forma a fazerem sentido juntos, de forma que definam e sejam definidos pela Meta do Sprint.

16.2 COMO É O SPRINT?

16.2.1 Duração

Os Sprints são *timeboxes* e têm duração fixa e constante, o que significa que eles não devem durar nem mais, nem menos que o estabelecido e que essa duração não varia de Sprint para Sprint. A ideia por trás desse conceito é a de criar ritmo e disciplina no trabalho do Time de Desenvolvimento, além de regularidade na obtenção de *feedback*, cobrança e estímulo do Product Owner e dos clientes e demais partes interessadas. A duração fixa do Sprint também viabiliza o cálculo da Velocidade do Time de Desenvolvimento (veja 9.2.2 “Velocidade do Time de Desenvolvimento”), facilitando o seu planejamento.

Quando Scrum foi criado, a literatura determinava que os Sprints tivessem obrigatoriamente um mês de duração. A prática do Scrum em projetos com diferentes características fez com que oficialmente se aceitassem Sprints de três, de duas e, mais recentemente, de uma semana. A duração dos Sprints, portanto, é fixada em algum valor entre uma e quatro semanas.

Para se definir a duração dos Sprints, é comum o uso de semanas (dias corridos) ou de dias úteis. A medida em semanas cria uma maior regularidade, o que ajuda a dar ritmo no trabalho do Time de Desenvolvimento e pode, por exemplo, facilitar a agenda do Product Owner e favorecer a presença de determinadas pessoas nas reuniões de Sprint Review, como clientes e outras partes interessadas. Adicionalmente, pode ser psicologicamente interessante iniciar o Sprint em uma segunda-feira e terminá-la em uma sexta-feira. Mas nem sempre isso é possível. Por exemplo, quando o Product Owner atua em dois projetos, os Sprints de um de seus projetos podem começar nas segundas (terminando nas sextas) e as do outros projetos, nas quartas (terminando nas terças), para que ele possa estar presente em todos os eventos que lhe cabem. Já a medida da duração do Sprint em dias úteis facilita a medição da Velocidade do Time de Desenvolvimento e o seu uso para o planejamento, já que o número de dias de trabalho do Sprint não varia com o advento de feriados.

Pode ser necessário modificar a duração do Sprint dentro do mesmo projeto conforme as condições do projeto e do ambiente mudam. Essa necessidade é geralmente detectada pelo Time de Scrum em uma reunião de Sprint Retrospective e deve raramente ocorrer. A nova duração escolhida para o Sprint deve também fixa e constante.

Apresentamos aqui alguns fatores que podem influenciar a escolha da duração do Sprint de um projeto.

Frequência das mudanças

A frequência com que mudam as necessidades de negócios e suas prioridades é um fator importante na escolha da duração do Sprint. Sprints menores aumentam a capacidade do Time de Desenvolvimento para responder a essas mudanças e podem ser utilizados quando a natureza do projeto ou dos clientes tornam as necessidades de negócios mais voláteis. Sprints maiores podem ser utilizados em projetos com necessidades de negócios mais estáveis, em que o Product Owner encontrará menos problemas em planejar para um tempo mais longo.

Em qualquer um dos casos, o planejamento deve ser feito para um período curto o suficiente para que sejam mínimas as chances da Meta do Sprint perder o sentido nesse período.

Frequência de feedback

Quanto menor for o Sprint, mais frequente será o *feedback* dos clientes e demais partes interessadas sobre o que foi produzido. Quanto mais frequente for esse *feedback*, maiores serão as chances de se efetuarem correções de curso no trabalho realizado, o que diminui os riscos de desperdício do trabalho do Time de Desenvolvimento. Outro benefício é que os clientes e demais partes interessadas também verão os progressos do projeto mais frequentemente.

Da mesma forma, quanto menor for o Sprint, mais frequentes serão as retrospectivas do Time de Scrum, possibilitando-o experimentar e aprender mais rapidamente com soluções que possam trazer melhorias a seus processos. Além desse benefício, a sensação de sucesso e de trabalho cumprido que vem com o desfecho de um Sprint bem sucedido pode ser um forte motivador para o trabalho do Time de Desenvolvimento. Assim, quanto mais frequentemente isso ocorrer, melhor.

Sobrecarga

A prática mostra que, mesmo para Sprints curtos como os de uma semana, a reunião de Sprint Planning pode ocupar boa parte de um dia de trabalho para alguns Times de Scrum. O mesmo pode ocorrer com as reuniões de Sprint Review e de Sprint Retrospective juntas. Nesses casos, o custo de se iniciar e de se encerrar o Sprint pode ser alto demais para que valha a pena se realizar um Sprint tão curto.

Ritmo

Sprints mais curtos podem ajudar a criar um ritmo para o trabalho do Time

de Desenvolvimento e a mais facilmente se manter esse ritmo. O ritmo, por sua vez, ajuda a se manter o foco do trabalho, aumentando o compromisso do Time de Desenvolvimento com a Meta de cada Sprint.

No entanto, Sprints de uma semana podem acabar ditando um ritmo frenético e estressante demais para alguns Times de Desenvolvimento, que podem assim preferir Sprints com duração maior.

Tamanho do projeto

Para projetos curtos, é mais apropriado se estabelecer Sprints curtos, pois dessa forma o número de oportunidades de *feedback* dos clientes e demais partes interessadas é maior. Para um projeto com um ou dois meses de duração, por exemplo, Sprints de uma ou duas semanas são mais adequados.

Valor de negócio

A duração do Sprint escolhida é tal que o Time de Desenvolvimento pode produzir um Incremento do Produto que signifique valor de negócio visível para os clientes, que lhes será demonstrado durante a reunião de Sprint Review. Dependendo da natureza do projeto, um Sprint muito curto pode não ser suficiente para que seja possível se produzir valor visível suficiente.

Iniciantes

Para Times de Scrum que estão em seus primeiros passos na utilização de Scrum, é comum a escolha de Sprints mais curtos – de uma semana, por exemplo – para aumentar a frequência de *feedback*. Esse *feedback* frequente, tanto externo (nas reuniões de Sprint Review) quanto interno (nas reuniões de Sprint Retrospective), permite ao time iniciante evoluir mais rapidamente ao identificar onde e como melhorar.

Foco

Em Sprints muito longos, como os de quatro semanas, o foco do Time de Desenvolvimento na Meta do Sprint tende, com o passar do tempo, a se enfraquecer. Sprints mais curtos favorecem a manutenção do foco na Meta do Sprint.

16.2.2 Incrementos entregáveis

Em cada Sprint, é gerado pelo Time de Desenvolvimento um Incremento do Produto pronto, de acordo com a Definição de Pronto. Idealmente, esse Incremento está sempre apto a ser entregue para os clientes, ou seja, é entregável. Essa entrega ou Release somente é feita quando o Product Owner julga que há valor suficiente para que seja utilizado, o que em muitos casos ocorre apenas ao somarem-se Incrementos do Produto de mais de um Sprint (veja “[12. Incremento do Produto](#)”).

Em muitos casos, no entanto, é necessário algum trabalho adicional antes que um conjunto de Incrementos do Produto possa, de fato, ser entregue, o que aumenta os riscos do projeto ao postergar a detecção de problemas (veja “[11. Definição de Pronto](#)”).

16.3 O SPRINT PODE SER CANCELADO?

Caso julgue que a Meta do Sprint perdeu o seu sentido, o Product Owner pode decidir pelo cancelamento do Sprint. Se houver, por exemplo, uma mudança na legislação ou regulamentação, uma ação da concorrência ou qualquer mudança drástica de prioridades que torne a Meta do Sprint obsoleta, não faz sentido prosseguir com o Sprint até o seu final.

Assim, se a Meta perder seu sentido, o Product Owner informará ao Time de Desenvolvimento sobre o cancelamento do Sprint. Os itens já prontos (de acordo com a Definição de Pronto) serão revistos, antecipando-se assim a reunião de Sprint Review. Uma reunião de Sprint Retrospective também será, em seguida, realizada.

Em seguida, idealmente, um novo Sprint será iniciado. No entanto, como é uma situação de exceção, é razoável que Time de Desenvolvimento e Product Owner decidam como proceder, já que podem preferir não modificar o calendário já previsto de Sprints. Ao invés de iniciar um novo Sprint com o tamanho regular, é comum optarem por iniciar um Sprint maior que contenha os dias que sobraram ou por realizar um Sprint curto com esses dias restantes, de forma a manter as datas de início e término dos próximos Sprints.

No entanto, escolhe-se a duração fixa dos Sprints curta o suficiente para que sejam mínimas as chances da Meta perder o sentido nesse período, de forma que cancelamentos de Sprint sejam raros. É importante que seja dessa forma, pois cancelamentos de Sprint são traumáticos para todo o Time de Scrum e devem ser evitados, na medida do possível.

Alguns autores e praticantes de Scrum defendem que, no momento em que o Time de Desenvolvimento detecta que não conseguirá atingir a Meta do Sprint, o Product Owner seja chamado para se negociar o cancelamento do Sprint ou uma redução da Meta. Esse, na realidade, é um comportamento que dá margem a transformar uma disfunção – não se atingir a Meta – em uma parte normal do trabalho. Ou seja, ao invés de aprender com o erro e passar a se comprometer com Metas de tamanho mais adequado à sua capacidade de trabalho, o Time de Desenvolvimento pode, como consequência, sentir-se confortável com o cancelamento de Sprints ou reduções da Meta o suficiente para ameaçar seu compromisso com as Metas dos Sprints seguintes.

CAPÍTULO 17

Sprint Planning

- Objetivo: planejar o ciclo de desenvolvimento (Sprint) que se inicia;
- Quando: no primeiro dia do Sprint, iniciando o mesmo;
- Duração: máxima proporcional a 8 horas para Sprints de 1 mês (máximo);
- Participantes obrigatórios: Product Owner, Time de Desenvolvimento e ScrumMaster;
- Saídas esperadas: Meta do Sprint e Sprint Backlog.

17.1 O QUE É A SPRINT PLANNING?

O planejamento de um ciclo de desenvolvimento, ou seja, de um Sprint, é realizado na reunião de Sprint Planning. Ela se inicia logo no primeiro momento do primeiro dia do Sprint.

A reunião de Sprint Planning conta com a participação obrigatória do Product Owner e dos membros do Time de Desenvolvimento. Eles colaboram e negociam o que será desenvolvido e qual a Meta desse Sprint. Os membros do Time de Desenvolvimento estabelecem então como o que foi negociado será desenvolvido. Essa lista de itens selecionados do alto do Product Backlog para desenvolvimento, somada ao plano de como esse trabalho será realizado, é chamada de Sprint Backlog.

Todos os membros do Time de Desenvolvimento participam da reunião com igual poder de opinião e decisão, quaisquer que sejam suas áreas de conhecimento ou de atuação.

Essa reunião também conta com a participação obrigatória do ScrumMaster, que atua como um facilitador.

17.1.1 Duração

A reunião de Sprint Planning não deve durar mais que um dia de trabalho. Recomenda-se que ela seja um *timebox* de não mais que 5% da duração do Sprint, ou no máximo oito horas para Sprints de um mês.

Ao se conjugar a prática de se realizarem sessões de Refinamento do Product Backlog com Product Owner e Time de Desenvolvimento, aliada ao uso da Definição de Preparado, pode-se reduzir significativamente a duração do Sprint. No entanto, quando essas boas práticas não são utilizadas, as reuniões de Sprint Planning tendem a ficar mais longas e menos produtivas (veja “[23. Refinamento do Product Backlog](#)”).

Qualquer que seja a duração escolhida, ela deve ser previamente estabelecida e, como qualquer *timebox*, rigorosamente respeitada.

17.1.2 Saídas

Meta do Sprint. Time de Desenvolvimento e Product Owner negociam e estabelecer uma meta de negócios para o Sprint corrente, que determina qual necessidade de negócios (em geral, dos usuários ou clientes) é imprescindível que seja atendida nesse Sprint. É a chamada Meta do Sprint (veja “[13.2. Meta de Sprint](#)”).

Uma boa Meta de Sprint pode ser, por exemplo, “*permitir que a loja virtual disponibilize novos produtos*”, ao passo que “*desenvolver os itens A, B, C e D do Product Backlog*” ou “*fazer A, B, C e D*” não são metas aceitáveis.

A Meta do Sprint é, de fato, definida pelo Product Owner. Mas a negociação é necessária, porque é o Time de Desenvolvimento que sabe quanto trabalho é possível

ser realizado em um Sprint. Assim, o Time de Desenvolvimento ajuda o Product Owner a estabelecer uma meta realista e se compromete a atingi-la.

Em geral, a Meta do Sprint é estabelecida apenas após os itens terem sido selecionados para o Sprint Backlog. É comum, no entanto, que o Product Owner traga para a reunião uma boa ideia da Meta que esperava ver alcançada ao final do Sprint, para que então seja negociada com o Time de Desenvolvimento. Afinal, a Meta do Sprint é parte da estratégia do produto desenvolvida pelo Product Owner para se alcançar uma Visão de Produto (veja “[13. Metas de negócios](#)”).

Uma vez estabelecida, a Meta do Sprint não pode ser modificada durante o Sprint em hipótese alguma. No entanto, caso ela perca o sentido em termos de negócios, o Sprint pode ser cancelado pelo Product Owner (veja “[16.3. O Sprint pode ser cancelado?](#)”).

Sprint Backlog. O Time de Desenvolvimento e o Product Owner colaboram na reunião de Sprint Planning para estabelecer o que será desenvolvido no Sprint corrente. São itens selecionados do alto do Product Backlog – portanto, os mais importantes naquele momento – para se gerar o Incremento do Produto de forma a se alcançar a Meta do Sprint.

A partir desses itens selecionados, o Time de Desenvolvimento também cria um plano mais detalhado de como o trabalho será realizado. Esse plano é geralmente expresso a partir da quebra de cada item em tarefas e, possivelmente, de estimativas para essas tarefas. O conjunto de itens selecionados para o Sprint, adicionados do plano de como serão desenvolvidos é chamado de Sprint Backlog (veja “[10. Sprint Backlog](#)”). Na figura [17.1](#), podemos ver um exemplo de Sprint Backlog ao final de uma reunião de Sprint Planning.

O Time de Desenvolvimento monitora seu progresso, verificando o trabalho restante em cada momento do Sprint. Uma forma comum de se viabilizar esse acompanhamento é a prática de se estimar as tarefas do Sprint Backlog em alguma unidade de tempo ou tamanho, utilizar essas estimativas para determinar o trabalho restante e visualizá-lo por um Gráfico de Sprint Burndown (veja “[14.3. Gráfico de Sprint Burndown](#)”). “Horas” e “tamanhos de camisa” (ou *T-Shirt Sizing*) são unidades muito utilizadas para essas estimativas. No entanto, uma prática comum consiste em apenas contar o número de tarefas restantes sem estimá-las, o que também pode fornecer informação suficiente.

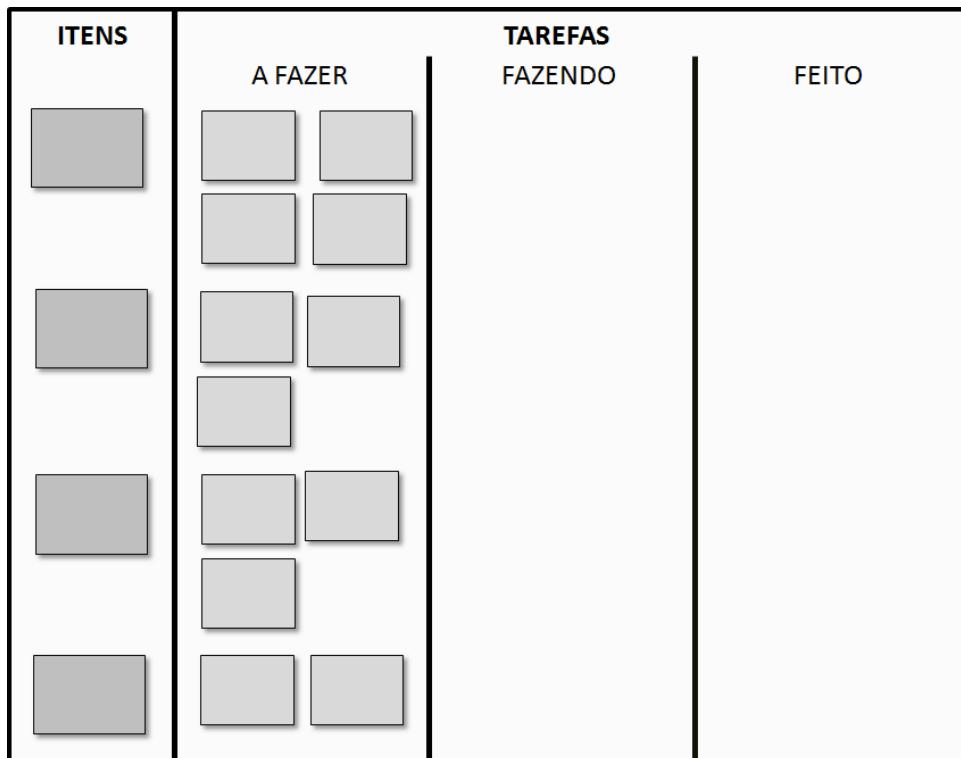


Figura 17.1: Sprint Backlog recém-criado na reunião de Sprint Planning

O Time de Desenvolvimento trabalha na reunião de Sprint Planning para criar o melhor plano possível com todo o conhecimento de que dispõe naquele momento. Ainda assim, provavelmente o plano do Sprint Backlog evoluirá durante o Sprint, à medida que o Time de Desenvolvimento adquirir maior conhecimento ao executá-lo.

17.1.3 Preparação

Itens preparados previamente à reunião de Sprint Planning podem aumentar as chances de sucesso do Sprint, ou seja, de o Time de Desenvolvimento atingir a Meta do Sprint definida. É importante, portanto, que Product Owner e Time de Desenvolvimento definam o que é necessário para um item estar preparado para entrar em desenvolvimento, e que assim estabeleçam uma Definição de Preparado. “Preparado” pode significar, por exemplo, que o item já tenha sido discutido pelo Product

Owner e Time de Desenvolvimento, que já possua Critérios de Aceitação definidos, que já esteja estimado, que seja pequeno o suficiente e que possua detalhes suficientes e necessários.

Uma vez que se utilize uma Definição de Preparado, é responsabilidade do Product Owner garantir que um número suficiente de itens esteja preparado para a reunião de Sprint Planning. Esse trabalho de preparação é realizado em colaboração do Time de Desenvolvimento, geralmente durante sessões de Refinamento do Product Backlog durante o Sprint anterior (veja “[23. Refinamento do Product Backlog](#)”).

17.2 COMO É A SPRINT PLANNING?

No formato tradicional da reunião de Sprint Planning, ela é dividida em duas partes. Na primeira, chamada de “Sprint Planning 1”, Product Owner e Time de Desenvolvimento se reúnem para estabelecer o que será desenvolvido no Sprint corrente. Na segunda parte, chamada de “Sprint Planning 2”, o Time de Desenvolvimento planeja como será desenvolvido o que foi estabelecido na primeira parte da reunião.

Existem algumas variações para o formato tradicional da reunião de Sprint Planning. No Planejamento Intercalado de Sprint, o Time de Desenvolvimento quebra em tarefas item a item, do alto para baixo do Product Backlog, até que julgue que já selecionou itens suficientes para o Sprint Backlog, e então negocia a Meta do Sprint com o Product Owner. No Planejamento Just-In-Time, o Time de Desenvolvimento apenas negocia e seleciona os itens para o Sprint Backlog, deixando para quebrá-los em tarefas apenas no momento do desenvolvimento de cada um deles durante o Sprint. A seguir, descrevemos esses diferentes formatos utilizados para a reunião de Sprint Planning.

17.2.1 Sprint Planning 1 e Sprint Planning 2

O quê?

Na primeira parte da reunião de Sprint Planning, o Product Owner e o Time de Desenvolvimento colaboram, com a facilitação do ScrumMaster, para definir o que será desenvolvido no Sprint corrente. Eles escolhem, a partir do alto do Product Backlog, quais itens farão parte do Sprint Backlog e definem a Meta do Sprint.

A Sprint Planning 1 deve durar no máximo a metade do tempo previsto para toda a reunião de Sprint Planning. Recomenda-se não mais que quatro horas para Sprints de um mês e proporcionalmente menos para Sprints menores.

O Product Owner apresenta ao Time de Desenvolvimento os itens mais importantes do Product Backlog. Ele percorre cada item, desde o topo do Product Backlog, e responde a perguntas do Time de Desenvolvimento sobre o seu conteúdo, propósito e importância, tanto individualmente quanto o relacionando aos outros itens já descritos.

Sessões de Refinamento do Product Backlog realizadas no Sprint anterior tornam a reunião de Sprint Planning mais objetiva e eficiente, já que os itens chegam preparados para discussão. Ainda assim, podem acontecer durante a reunião o detalhamento e a evolução de um ou mais itens em itens menores, a retirada ou inserção de algum item e reestimativas de itens que se façam necessárias, caso se utilizem estimativas.

É importante o Product Owner não tentar impor mais trabalho do que o Time de Desenvolvimento acredita que é capaz de realizar, seja forçando a inserção de mais itens no Sprint Backlog ou impondo a redução de estimativas. É também igualmente importante que o Product Owner entenda que é o Time de Desenvolvimento quem melhor pode dizer quanto ele é capaz de produzir.

Essa colaboração se encerra quando Product Owner e Time de Desenvolvimento concordam que itens além dos que já foram apresentados estarão acima de quanto o Time de Desenvolvimento acredita ser capaz de produzir. Essa decisão é tomada a partir de sua experiência com Sprints anteriores, que pode ser apoiada no cálculo da Velocidade do Time de Desenvolvimento.

Os itens selecionados fazem agora parte do Sprint Backlog do Sprint corrente. Essa lista de itens não gera um compromisso, mas trata-se apenas de uma previsão do Time de Desenvolvimento sobre o quanto acredita que será possível desenvolver durante o Sprint.

Time de Desenvolvimento e Product Owner então negociam e estabelecem a Meta do Sprint, que determina qual objetivo ou necessidade de negócios deve ser atendida por meio do desenvolvimento dos itens selecionados.

Os resultados obrigatórios da Sprint Planning 1 são os itens escolhidos para o Sprint Backlog e a Meta do Sprint. A reunião se encerra no momento em que o Time de Desenvolvimento, diante do Product Owner, se compromete com a Meta acordada, tornando-se assim responsável por atingi-la.

Como?

A segunda parte da reunião de Sprint Planning também dura no máximo metade do tempo previsto para toda a reunião. Durante esse tempo, o Time de Desenvolvi-

mento planeja como será feito o desenvolvimento dos itens escolhidos para o Sprint Backlog.

Embora não haja um formato prescrito pelo Scrum, esse plano é geralmente expresso por tarefas a serem realizadas pelo Time de Desenvolvimento durante o Sprint. Dessa forma, os membros do Time de Desenvolvimento trabalham na Sprint Planning 2 percorrendo item a item entre os escolhidos para o Sprint Backlog e quebrando cada um em um conjunto de tarefas correspondentes. Essas tarefas representam os pequenos passos que serão seguidos para que o item esteja pronto, de acordo com a Definição de Pronto, o que inclui estar de acordo com seus Critérios de Aceitação. Para criá-las, o Time de Desenvolvimento reflete sobre quais serão os passos necessários para transformar o item em uma parte funcionando do produto. Assim, os membros do Time de Desenvolvimento provavelmente observarão atentamente tanto os Critérios de Aceitação dos itens quanto a Definição de Pronto ao quebrar cada item em tarefas.

As tarefas são em geral pequenas, representando no máximo algumas horas de trabalho. Tarefas maiores que um dia de trabalho são de difícil acompanhamento e devem ser evitadas. Caso elas existam, a visibilidade que se ganharia durante a reunião de Daily Scrum seria prejudicada, já que um membro do Time de Desenvolvimento poderia, por vários dias seguidos, informar ao resto do time que ainda está trabalhando na mesma tarefa.

Uma vez que todos os itens selecionados para o Sprint estejam quebrados em tarefas, estimativas para o tempo de desenvolvimento de cada tarefa podem ser adicionadas. Estimativas de tarefas possuem o único propósito de servirem ao Time de Desenvolvimento, de forma que seja possível acompanhar o progresso de seu trabalho em direção ao final do Sprint por meio de um Gráfico de Sprint Burndown.

Diferentes unidades podem ser utilizadas para essas estimativas. A maioria dos times utiliza simplesmente “horas”. O principal problema com estimativas em “horas” é que o tempo estimado em horas por um desenvolvedor provavelmente seria diferente do estimado por outro para uma mesma tarefa. Por essa razão, o Time de Desenvolvimento é forçado a estabelecer no planejamento – e, portanto, prematuramente – quem irá desenvolver cada tarefa. Essa definição deveria ser realizada ao longo do Sprint e conforme necessário, já que cada tarefa pertence ao Time de Desenvolvimento como um todo, e não a um ou outro membro com conhecimentos específicos.

Já com o uso de uma escala mais simples como “tamanhos de camisa” (*T-Shirt Sizing*), ou seja, Pequeno, Médio e Grande (P, M, G), esse problema é minimizado

devido à menor precisão. Pode-se ver na figura 17.2 um exemplo de um trecho do Quadro de Tarefas com as tarefas estimadas em Tamanhos de Camisas.

Alternativamente, preferimos quebrar cada item em tarefas pequenas e utilizar a contagem do número de tarefas restantes em cada dia para traçar o Gráfico de Sprint Burndown, sem a necessidade de estimativas.

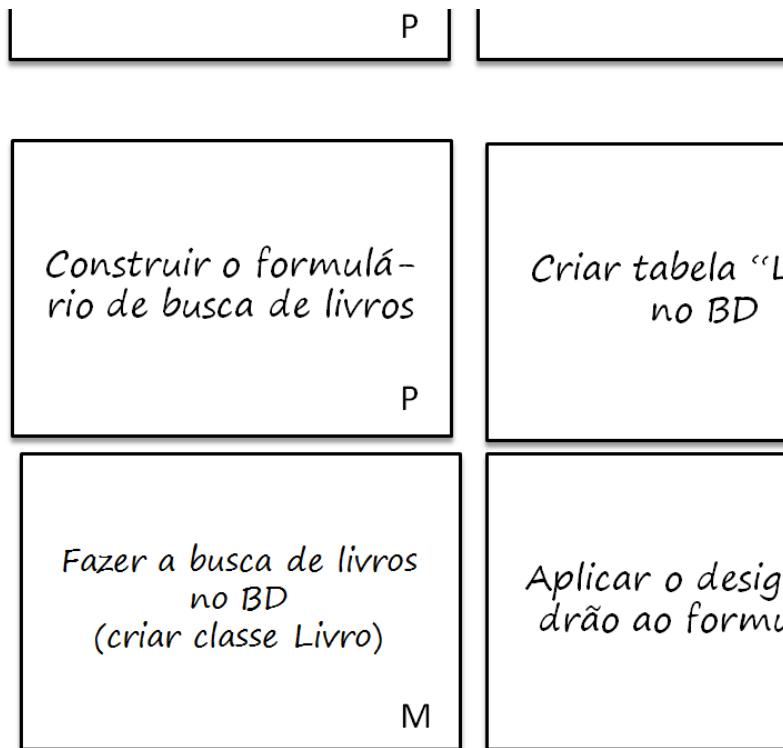


Figura 17.2: Trecho do Quadro de Tarefas com estimativas em Tamanhos de Camisas

É importante destacar que todos os membros do Time de Desenvolvimento participam, com igual poder de opinião e decisão, da quebra dos itens em tarefas e das estimativas, se utilizadas.

O trabalho de definição e estimativa das tarefas não deve ser exaustivo nem completo. O Time de Desenvolvimento o faz da melhor forma possível, com as informações e conhecimento que tem sobre os itens no momento da reunião. Inevitavelmente, à medida que o Time de Desenvolvimento avança no Sprint e entende melhor o trabalho que está realizando, novas tarefas surgirão para os itens do Sprint Backlog,

outras não mais serão necessárias e desaparecerão e estimativas, caso presentes, serão modificadas.

A participação do Product Owner na Sprint Planning 2 não é obrigatória. No entanto, é altamente recomendado que, no mínimo, ele fique acessível e disponível para o Time de Desenvolvimento. O Product Owner pode ser requisitado pois, durante a reunião, dúvidas sobre os itens e sobre a Meta do Sprint invariavelmente surgirão e elas devem ser sanadas o mais rapidamente possível.

Ao final da Sprint Planning 2, o Sprint Backlog inicial estará pronto, contendo os itens escolhidos na Sprint Planning 1 e o plano de como esses itens serão desenvolvidos. Pode ser interessante nesse momento que o Time de Desenvolvimento já crie o Gráfico de Sprint Burndown.

17.2.2 Planejamento Intercalado de Sprint

Também chamado de Planejamento Baseado em Compromisso, o Planejamento Intercalado é uma alternativa a se considerar ao formado oficial da reunião de Sprint Planning, pois traz algumas vantagens concretas.

Em um Planejamento Intercalado, Product Owner e Time de Desenvolvimento trabalham juntos por toda a reunião, facilitados pelo ScrumMaster. Observe que, ao contrário do que ocorre no formato tradicional da reunião de Sprint Planning, o Product Owner deve obrigatoriamente estar presente durante toda a reunião. Métricas como Velocidade e estimativas não são utilizadas, apenas a experiência, o conhecimento e um acordo entre os envolvidos.

As partes “o quê” e “como” do Sprint Backlog são definidas de forma intercalada ao longo da reunião, ou seja, o que será desenvolvido e como será desenvolvido não estão mais separados.

Seleciona-se o item mais importante do Product Backlog, ou seja, aquele no topo, que é então lido em voz alta. O Product Owner tira dúvidas sobre o item até que o Time de Desenvolvimento esteja confortável com sua própria compreensão do item. Esse é geralmente um trabalho rápido se o Product Backlog estiver previamente refinado (veja “[23. Refinamento do Product Backlog](#)”).

O Time de Desenvolvimento então realiza as discussões técnicas necessárias e estabelece um plano de como o item será desenvolvido, o que geralmente é expresso por tarefas a serem realizadas durante o Sprint. O item e seu plano são adicionados ao Sprint Backlog.

Segue-se portanto para o item seguinte, que também é lido, discutido e quebrado

em tarefas (ou em alguma outra forma de plano). Ao fazê-lo, o Time de Desenvolvimento tem uma maior compreensão sobre o desenvolvimento do item, e pode melhor decidir se o adiciona ao Sprint Backlog ou não. Essa decisão é basicamente baseada no que o Time de Desenvolvimento pode compreender do item e em experiências passadas. Caso decida adicioná-lo, segue-se para o item seguinte, até que o Time de Desenvolvimento julgue que já selecionou itens suficientes para o Sprint.

Em seguida, Product Owner e Time de Desenvolvimento, tendo em vista os itens selecionados, irão negociar e estabelecer a Meta do Sprint.

PLANEJAMENTO INTERCALADO DE SPRINT: PASSOS

- 1) o item mais importante (mais ao alto) do Product Backlog ainda não selecionado é lido e dúvidas do Time de Desenvolvimento são tiradas pelo Product Owner;
- 2) o Time de Desenvolvimento gera um plano de como o item será desenvolvido, geralmente quebrando-o em tarefas;
- 3) a partir das tarefas, o Time de Desenvolvimento decide se acredita que será possível desenvolver o item durante o Sprint;
- 4) em caso positivo, o item é selecionado para o Sprint Backlog e volta-se ao passo 1. Em caso negativo, o item não é selecionado;
- 5) Product Owner e Time de Desenvolvimento estabelecem a Meta do Sprint.

Há diversas vantagens nessa abordagem. As principais são:

- o resultado é, a princípio, a melhor previsão possível que o Time de Desenvolvimento pode dar naquele momento, já que a produzirá sobre um plano mais detalhado;
- a reunião é mais dinâmica, já que a participação de todos é mais ativa todo o tempo e as atividades são mais variadas, já que se intercalam partes diferentes

do planejamento. O dinamismo leva a um maior engajamento dos participantes;

- há um maior foco dos membros do Time de Desenvolvimento. Ao se iniciar a discussão sobre o que é o item, alguns participantes já começarão a pensar em como desenvolvê-lo. Ao não criarem esse plano imediatamente, poderão ter maior dificuldade em manter seu foco ao seguirem na discussão dos próximos itens, como muitas vezes ocorre ao se utilizar o formato original da reunião;
- obtém-se uma melhor compreensão conjunta sobre cada item, já que Time de Desenvolvimento e Product Owner trabalham juntos do início ao fim da reunião;
- a colaboração mais próxima entre Product Owner e Time de Desenvolvimento aumenta o espírito de equipe e a responsabilidade conjunta sobre o Sprint.

O Planejamento Intercalado pode ser útil mesmo quando se deseja utilizar Velocidade do Time de Desenvolvimento como preditora em futuras reuniões de Sprint Planning. Ao se iniciar o projeto, o Time de Desenvolvimento não tem nenhum número para sua Velocidade, já que nenhum Sprint foi ainda completado. Nessa primeira reunião de Sprint Planning, o Time de Desenvolvimento não tem parâmetros para determinar quanto trabalho irá aceitar para o Sprint Backlog. Mesmo antes do segundo ou terceiro Sprint, o valor que pode se pode obter para Velocidade (a partir da média de quanto foi entregue até então) ainda pode ser muito distante da realidade.

É necessário que os itens do alto do Product Backlog estejam estimados ou pequenos o suficiente para que se possa medir a Velocidade após alguns Sprints (veja “[9.2.2. Planejável](#)”, no capítulo sobre o Product Backlog). Pode-se então realizar o Planejamento Intercalado nos primeiros Sprints e utilizar a quantidade de trabalho entregue em cada um deles para se calcular a Velocidade. A partir do quarto Sprint, por exemplo, ou quando a Velocidade se estabilizar, pode-se passar a utilizar o formato de planejamento oficial, em duas partes.

17.2.3 Planejamento Just-In-Time de Sprint

A máxima do Planejamento *Just-In-Time* é a redução do desperdício. Esse formato considera que o melhor momento para se realizar o planejamento de como os itens

serão desenvolvidos é imediatamente antes do início do desenvolvimento de cada item, ou seja, o último momento possível.

Se o planejamento de todos os itens for realizado na reunião de Sprint Planning (ou seja, em geral a sua quebra em tarefas), o Time de Desenvolvimento pode não ter dados suficientes para criar um bom plano. Mudanças significativas no plano terão que ser feitas no decorrer do Sprint, o que caracterizará o desperdício.

Na reunião de Sprint Planning de um Planejamento *Just-In-Time*, Time de Desenvolvimento e Product Owner apenas selecionam os itens do Product Backlog para o Sprint Backlog e criam a Meta do Sprint (“o quê?”). Sua execução é similar ao Sprint Planning 1, descrito na seção anterior. Assim, não é definido nessa reunião o plano de como será realizado o desenvolvimento dos itens escolhidos para o Sprint Backlog (“como?”).

Após a reunião de Sprint Planning, planeja-se sobre o primeiro e mais importante item do Product Backlog, em geral quebrando-o em tarefas a serem realizadas pelo Time de Desenvolvimento. O item é desenvolvido, e quando estiver pronto, de acordo com a Definição de Pronto, o Time de Desenvolvimento seguirá para planejar sobre o item seguinte, em geral quebrando-o em tarefas, o desenvolverá e seguirá assim pelos outros itens do Sprint Backlog.

Times com maior maturidade podem obter mais sucesso com o Planejamento *Just-In-Time*. No entanto, embora colabore significativamente para a redução do desperdício, essa modalidade de planejamento pode aumentar os riscos de surpresas perigosas no Sprint, uma vez que detalhes importantes surgirão apenas no seu decorrer. Quando, ao contrário, o planejamento é realizado no início do Sprint, problemas podem ser identificados cedo e então tratados imediatamente.

CAPÍTULO 18

Daily Scrum

- Objetivo: planejar o próximo dia de desenvolvimento;
- Quando: em cada dia de desenvolvimento do Sprint;
- Duração: máxima de 15 minutos;
- Participantes obrigatórios: Time de Desenvolvimento;
- Saídas esperadas: plano informal para o próximo dia de trabalho.

18.1 O QUE É A DAILY SCRUM?

A Daily Scrum é uma reunião curta realizada diariamente pelo Time de Desenvolvimento. Essa reunião é um *timebox* de quinze minutos e acontece preferencialmente no mesmo local e à mesma hora.

A reunião de Daily Scrum facilita a auto-organização do Time de Desenvolvimento e, assim, sua realização é de grande importância. Ela tem os propósitos de

proporcionar, entre os membros do Time de Desenvolvimento, visibilidade ao trabalho realizado e a realizar, promover a comunicação sobre esse trabalho, dar visibilidade a quais obstáculos atrapalham ou atrapalharam o desenvolvimento desde a última reunião de Daily Scrum e servir de oportunidade para decisões rápidas com relação ao progresso do Sprint. Assim, essa reunião produz um plano informal para o próximo dia de trabalho do Time de Desenvolvimento, ou seja, para até a próxima reunião de Daily Scrum.

18.2 COMO É A DAILY SCRUM?

Existe um padrão utilizado por Times de Desenvolvimento para endereçar essas questões na reunião de Daily Scrum. Cada membro se dirige a seus colegas e responde a três perguntas:

- O que eu fiz desde a última reunião de Daily Scrum?
- O que eu pretendo fazer até a próxima reunião de Daily Scrum?
- Quais obstáculos/impedimentos estiveram/estão em meu caminho, impedindo a realização do trabalho?

Para estimular a reflexão sobre o trabalho até o próximo dia e diminuir seus riscos, pode-se adicionar à última questão quais obstáculos ou impedimentos o membro do time espera encontrar até a próxima Daily Scrum.

O ScrumMaster pode participar, caso seja necessário, como um facilitador na reunião de Daily Scrum. Ele pode ajudar o Time de Desenvolvimento a manter o foco e a utilizar apenas os quinze minutos previstos, o que pode ser particularmente importante para times novos no uso de Scrum ou com pouca maturidade. Sua presença na reunião, no entanto, não é obrigatória. O Product Owner, em geral, não participa da reunião de Daily Scrum, pois ele não tem poder para interferir no andamento do trabalho no Sprint e, assim, não necessita dessa visibilidade.

A reunião de Daily Scrum é também conhecida como Daily Meeting (reunião diária) ou Stand-up Meeting (reunião em pé). Essa última designação é utilizada no Extreme Programming e indica uma prática potencialmente útil: a realização da reunião com todos os participantes de pé, com o objetivo de não permitir que eles sintam-se confortáveis o suficiente para quebrar o *timebox*. Cabe destacar, no entanto, que não há obrigatoriedade no Scrum de se realizar essa reunião em pé.

18.3 O QUE NÃO DEVE ACONTECER NA DAILY SCRUM?

18.3.1 Informe de impedimentos ao ScrumMaster

Informar impedimentos ao ScrumMaster não é um dos objetivos da reunião de Daily Scrum. Informa-se o impedimento ao ScrumMaster assim que ele é identificado, de forma que possa ser tratado o mais rapidamente possível. O objetivo de se informar, durante a reunião, quais impedimentos surgiram desde a última reunião de Daily Scrum é apenas o de dar visibilidade a todos os membros do Time de Desenvolvimento sobre o que atrapalhou ou ainda atrapalha o trabalho.

Ao se deparar com um impedimento, é um erro bastante comum o membro do Time de Desenvolvimento aguardar até a reunião de Daily Scrum para solicitar a ajuda do ScrumMaster. Esse comportamento acaba por atrasar a sua resolução, que pode atrapalhar por mais tempo do que o necessário o trabalho do Time de Desenvolvimento e ameaçar que se consiga alcançar a Meta do Sprint.

Outra questão é que, ao informar um impedimento para o ScrumMaster, o membro do Time de Desenvolvimento fornece a ele o máximo de detalhes possível, de forma a ajudá-lo na remoção do mesmo. Essa é uma atividade que consome tempo, o que acabaria por atrapalhar o andamento da reunião caso realizada no seu decorrer e poderia comprometer seu *timebox*.

18.3.2 Reunião de trabalho

A reunião de Daily Scrum não deve ser transformada em reunião de trabalho. Os membros do Time de Desenvolvimento se limitam a sucintamente informar seus colegas sobre seu trabalho realizado e definir o trabalho a realizar. Ao se dar visibilidade sobre o trabalho, é comum surgirem assuntos relacionados, como discussões técnicas ou dúvidas de negócios, e é natural o Time de Desenvolvimento querer aproveitar para discuti-los. Esse tipo de desvio, no entanto, deve ser evitado para que o foco da reunião seja mantido. Contudo, é normal nessas situações uma reunião de trabalho ser agendada para acontecer logo após a reunião diária, contando inclusive, quando necessário, com a presença do Product Owner.

18.3.3 Prestação de contas a outros

A reunião de Daily Scrum não deve servir como instrumento de cobrança externa sobre o Time de Desenvolvimento. Nesta reunião, cada membro do Time de Desenvolvimento não está prestando contas do seu trabalho para o ScrumMaster, para o

Product Owner ou para qualquer parte interessada do projeto, mas sim apenas para os outros membros do Time de Desenvolvimento.

Infelizmente, não é incomum se ver o ScrumMaster repassando a lista de atividades e perguntando a cada membro do Time de Desenvolvimento o que cada um fez, o que cada um pretende fazer e quais são os impedimentos. O ScrumMaster, na verdade, não é alguém a quem o Time de Desenvolvimento responde por suas tarefas e a responsabilidade sobre a reunião de Daily Scrum é somente do Time de Desenvolvimento. O ScrumMaster pode estar presente na reunião de Daily Scrum apenas como facilitador.

18.3.4 Falta de atenção

Todos os membros do Time de Desenvolvimento devem manter o foco e prestar total atenção ao que seus colegas estão relatando durante a reunião de Daily Scrum. Nenhum outro trabalho ou conversa paralela devem acontecer durante a reunião. Para que se alcance esse nível de concentração, recomenda-se que conversas paralelas, telefones e computadores abertos sejam evitados durante a reunião.

CAPÍTULO 19

Sprint Review

- Objetivo: obter *feedback* sobre o Incremento do Produto desenvolvido no Sprint (inspeção e adaptação do produto);
- Quando: no último dia de cada Sprint, antes da reunião de Sprint Retrospective;
- Duração: máxima proporcional a 4 horas para Sprints de 1 mês;
- Participantes obrigatórios: clientes do projeto, Time de Desenvolvimento, Product Owner e ScrumMaster. Podem estar presentes usuários e quaisquer outras partes interessadas que possam prover *feedback*;
- Saídas esperadas: entradas para o Product Owner adicionar ao Product Backlog ou modificá-lo, visibilidade sobre o produto para clientes e demais partes interessadas.

19.1 O QUE É A SPRINT REVIEW?

Na reunião de Sprint Review (ou revisão do Sprint), Time de Desenvolvimento e Product Owner trabalham em conjunto, com a facilitação do ScrumMaster, para demonstrar o trabalho pronto, produzido durante o Sprint, a clientes do projeto e demais partes interessadas. Seu principal objetivo é a obtenção de *feedback* dessas pessoas sobre o Incremento do Produto produzido, o que o Product Owner utilizará como matéria-prima para modificar o Product Backlog para Sprints futuros. É, portanto, uma reunião de inspeção e adaptação do produto.

As presenças do Time de Desenvolvimento, do Product Owner e do ScrumMaster são obrigatórias nessa reunião. As demais pessoas convidadas podem ser clientes, usuários, gerentes e outros para os quais as funcionalidades desenvolvidas no Sprint são relevantes e cujo *feedback* é considerado importante.

A reunião de Sprint Review acontece no último dia do Sprint, antes da reunião de Sprint Retrospective, e tem a duração máxima de quatro horas para Sprints de um mês ou proporcionalmente menos para Sprints mais curtas.

19.2 COMO É A SPRINT REVIEW?

19.2.1 Preparação

Não é necessário haver grandes preparações para essa reunião. A demonstração realizada para os clientes e demais partes interessadas é informal e deve manter seu foco inteiramente no produto funcionando, ou seja, no Incremento do Produto gerado.

É interessante, no entanto, haver ao menos uma pequena sessão no final do trabalho do Sprint, na qual Time de Desenvolvimento e Product Owner alinharão o entendimento sobre o que ocorreu durante o Sprint e definirão o que será apresentado e quem fará essa demonstração.

19.2.2 Gestão de expectativas

É saudável que Product Owner e Time de Desenvolvimento se preocupem com a gestão das expectativas dos clientes e demais partes interessadas que estarão presentes.

Ao entenderem claramente qual é a Meta do Sprint e qual é a capacidade de trabalho do Time de Desenvolvimento, por exemplo, essas pessoas irão compreender melhor o que o Time de Desenvolvimento foi capaz de produzir no Sprint.

Assim, como parte da preparação para o encontro, pode ser interessante o Product Owner deixar seus objetivos claros ao realizar o convite para as pessoas que pretende que participem da reunião. Da mesma forma, antes do início da demonstração, Time de Desenvolvimento e Product Owner podem informá-los qual era a meta de negócios a ser atingida nesse Sprint (veja “[13.2. Meta de Sprint](#)”).

19.2.3 Quem participa

Além do ScrumMaster e Product Owner, todos os membros do Time de Desenvolvimento estão presentes na reunião de Sprint Review. Afinal, eles têm igual responsabilidade sobre os resultados de seu trabalho.

A presença de pessoas relevantes na reunião cria a oportunidade para valiosos *feedbacks* sobre o produto. O *feedback* reduz os riscos do projeto ao conformar os rumos dos próximos Incrementos do Produto de acordo com os objetivos e necessidades dos clientes e demais partes interessadas, os quais eles podem vislumbrar com mais clareza ao verem (e, talvez, experimentarem) o Incremento do Produto pronto e funcionando. Por essa razão, pode ser ainda mais interessante haver usuários reais entre os presentes.

Em alguns casos, infelizmente, os únicos participantes da reunião de Sprint Review são o Time de Desenvolvimento, que faz a demonstração, e o Product Owner, que fornece o *feedback* sobre o que foi produzido. Nesse cenário, confiança em excesso poderá estar sendo depositada na representatividade do Product Owner, pois ele será o único a ver o Incremento do Produto pronto e a ser capaz de dar *feedback* sobre o mesmo. Assim, as pessoas para as quais o produto desse Sprint é relevante somente poderão fornecer algum tipo de *feedback* após a Release, o que aumenta consideravelmente os riscos do projeto.

O ScrumMaster possui o importante papel de facilitar essa reunião, mas ele não participa diretamente na apresentação do que foi feito e nem dá suas opiniões sobre o Incremento do Produto gerado. Esse simplesmente não é o seu papel.

19.2.4 Demonstração

A demonstração é realizada a partir de uma colaboração entre Time de Desenvolvimento e Product Owner. Esse comportamento reflete uma proximidade e colaboração saudáveis acontecendo no trabalho de ambos. Em alguns casos, no entanto, apenas o Product Owner apresenta para os presentes o que foi feito durante o Sprint, e os membros do Time de Desenvolvimento dão o apoio necessário. Em outros, apenas

o Time de Desenvolvimento ou alguns de seus membros realizam a demonstração, enquanto que o Product Owner pode se colocar junto aos outros presentes.

Em geral, os itens desenvolvidos durante o Sprint são apresentados e demonstrados um a um, do mais importante ao menos importante. Clientes e demais presentes trabalham colaborativamente com o Time de Desenvolvimento e com o Product Owner, fazendo perguntas e obtendo respostas sobre o que lhes está sendo demonstrado, e apresentando suas ideias sobre o que esperam do produto nos próximos Sprints.

É uma excelente prática convidar os presentes a experimentarem diretamente o produto, o que os estimulará a fornecerem *feedback* e permitirá que ele seja mais profundo e preciso. No entanto, a Sprint Review não é uma reunião para testes do produto e, assim, não deve ser utilizada para substituir práticas de testes que devam ser realizadas ao longo do Sprint.

É igualmente interessante perguntar aos presentes o que esperam ver pronto nas próximas reuniões de Sprint Review, ou seja, estimulá-los a elaborar sobre quais são as próximas necessidades de negócios mais importantes a serem atendidas.

19.2.5 O que é demonstrado

O Incremento do Produto funcionando, ou seja, o valor gerado para os clientes do projeto, é o foco da demonstração. Evitam-se *slides*, documentos que não fazem parte do produto, explicações excessivas sobre planos ou sobre o processo de desenvolvimento, intenções não realizadas, desculpas e justificativas. Da mesma forma, questões técnicas somente serão discutidas se forem de interesse das pessoas presentes. Caso contrário, há o risco de que considerem a reunião uma perda de tempo, o que dificultará o seu comparecimento nas próximas.

Demonstram-se apenas os itens do Sprint Backlog que estejam prontos, de acordo com a Definição de Pronto.

19.2.6 Resultados

Em geral, o Product Owner verifica na reunião se cada item planejado e demonstrado está de fato pronto, de acordo com a Definição de Pronto. No entanto, para garantir um alinhamento, essa é uma atividade que pode ser realizada em uma sessão no último dia de desenvolvimento.

A partir do que foi e do que não foi gerado no Sprint, o Product Owner estabelece se o Time de Desenvolvimento conseguiu ou não atingir a Meta do Sprint. É

importante observar que, para atingir essa Meta, o Time de Desenvolvimento não necessariamente deverá ter completado todos os itens planejados. Salvo tenha havido algum impedimento prevenindo um item importante de ser desenvolvido, geralmente os itens não prontos ao final do Sprint são os de mais baixa importância para se atingir a Meta do Sprint. Na figura 19.1, pode-se ver um Quadro de Tarefas representando o Sprint Backlog de um Sprint possivelmente bem-sucedido, apesar de um dos itens não ter sido completado.

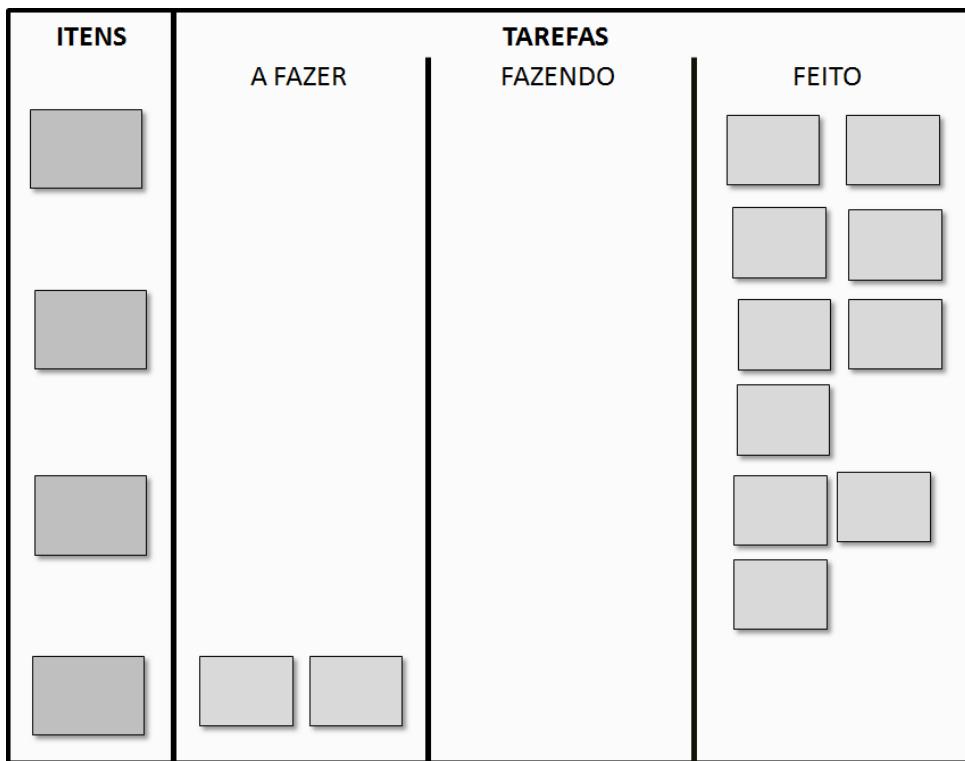


Figura 19.1: Quadro de Tarefas (Sprint Backlog) ao final de um Sprint

O *feedback* obtido a partir da interação entre Time de Desenvolvimento, Product Owner e demais presentes na reunião de Sprint Review é utilizado pelo Product Owner como matéria-prima para adicionar, remover ou modificar itens do Product Backlog. É dessa forma que o produto é construído incrementalmente para melhor atender às necessidades dos seus clientes.

Caso haja, na reunião de Sprint Review, itens do Sprint não prontos de acordo

com a Definição de Pronto, eles podem retornar ao Product Backlog e reaparecerem no próximo ou em algum dos próximos Sprints. Pode-se também decidir que sejam eliminados, se assim fizer sentido a partir do *feedback* obtido ou de outras questões de negócio. É papel do Product Owner e apenas dele decidir o destino desses itens.

CAPÍTULO 20

Sprint Retrospective

- Objetivo: melhoria incremental contínua - inspeção e adaptação dos processos de trabalho do Time de Scrum;
- Quando: no último dia de cada Sprint, após a reunião de Sprint Review;
- Duração: máxima proporcional a 3 horas para Sprints de 1 mês;
- Participantes obrigatórios: Time de Desenvolvimento, Product Owner e ScrumMaster;
- Saídas esperadas: planos de ação para melhorias nos processos de trabalho do Time de Desenvolvimento para o próximo Sprint.

20.1 O QUE É A SPRINT RETROSPECTIVE?

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

-- Princípio Ágil

20.1.1 Lições aprendidas em times tradicionais

Os acertos e erros em projetos que utilizam métodos tradicionais são geralmente levantados e discutidos apenas uma vez ao final do projeto, na esperança de que essas lições aprendidas possam ser úteis em trabalhos futuros.

A reunião de *postmortem*, por exemplo, busca cumprir esse papel, realizando uma verdadeira autópsia no projeto sem, no entanto, ter contribuído em nada para a sua saúde enquanto o projeto ainda estava vivo. Além de não ser útil para o próprio projeto, apenas uma pequena parte dessas lições pode ser aproveitada, se tanto, pois novos projetos envolvem novas condições, pessoas e desafios.

20.1.2 Melhoria incremental contínua em times Ágeis

Times que utilizam Scrum estão sempre buscando melhores formas de fazer seu trabalho, aprendendo com seus acertos e erros ao longo de todo o projeto. O objetivo da reunião de Sprint Retrospective (ou retrospectiva do Sprint) é estimular o Time de Scrum a realizar essa prática, que é chamada de melhoria incremental contínua. O princípio Ágil citado no princípio deste capítulo reflete exatamente isso: o Time de Scrum, ou seja, Time de Desenvolvimento, Product Owner e ScrumMaster se reúnem periodicamente com os objetivos de identificar pontos de melhoria no seu trabalho (inspeção) e de traçar planos de ação para executar essas melhorias (adaptação).

Por buscar a geração de melhorias já no Sprint seguinte do mesmo projeto, a prática da retrospectiva é muito mais objetiva e eficiente do que a prática de se avaliar as lições aprendidas somente ao final do projeto.

20.1.3 A retrospectiva do Sprint

Na reunião de Sprint Retrospective, o Time de Scrum inspeciona o Sprint que está se encerrando quanto a seus processos de trabalho, dinâmicas, comportamentos, práticas, ferramentas utilizadas e ambiente, e planeja as melhorias necessárias. Facilitados pelo ScrumMaster, Time de Desenvolvimento e Product Owner identificam o que foi bem, e que por essa razão pode ser mantido no próximo Sprint, e o que se pode melhorar, buscando as causas raízes dos problemas enfrentados no período e traçando planos de ação com formas práticas para se realizarem as melhorias.

A reunião é uma colaboração e nunca deve se configurar como trocas de acusações ou discussões improdutivas.

20.2 COMO É A SPRINT RETROSPECTIVE?

20.2.1 Regularidade, frequência e duração

Visando garantir a melhoria incremental contínua, o Time de Scrum realiza a reunião de Sprint Retrospective no último dia de cada Sprint, logo após a reunião de Sprint Review. A obrigatoriedade dessa prática visa assegurar regularidade e frequência na realização dessas melhorias, dando oportunidades ao Time de Scrum de promover mudanças para corrigir problemas antes que esses afetem negativamente os resultados de Sprints futuros.

Em projetos com prazos curtos demais e ritmo de trabalho consequentemente acelerado, é comum o Time de Desenvolvimento sofrer pressões internas e externas para suprimir quaisquer esforços que não façam parte do trabalho de desenvolvimento propriamente dito. A reunião de Sprint Retrospective, nesse cenário, é uma das primeiras a ser repetidamente cancelada e até mesmo definitivamente descartada. Sem a reunião, o Time de Scrum dificilmente realiza a inspeção e a adaptação. Assim, sequer consegue avaliar o porquê de não ser capaz de realizar as tarefas em um ritmo sustentável, fechando-se em um círculo vicioso.

Como guardião das práticas do Scrum, é papel do ScrumMaster estimular o Time de Scrum a realizar as retrospectivas com a regularidade exigida pelo *framework*.

A reunião de Sprint Retrospective deve durar no máximo três horas para Sprints de um mês, ou proporcionalmente menos para Sprints menores.

20.2.2 Participação do ScrumMaster

Facilitação

O ScrumMaster atua como um facilitador na reunião de Sprint Retrospective, trabalhando junto ao Time de Scrum para minimizar as dificuldades de comunicação e de colaboração, ingredientes necessários para a geração de ideias. A partir das questões levantadas nessa reunião, o ScrumMaster incentiva o Time de Scrum a encontrar soluções para trabalhar com qualidade e tornar-se cada vez mais efetivo, buscando sempre manter um ritmo sustentável de trabalho.

A variação na forma é, em geral, positiva para o resultado das reuniões de Sprint Retrospective. Assim, nesse trabalho de facilitação, o ScrumMaster pode introduzir diferentes técnicas e práticas, propondo de tempos em tempos sua experimentação

ao Time de Scrum, visando tornar as retrospectivas mais efetivas. Para estimular a variação de práticas, pode também ser interessante ocasionalmente permutarem-se os ScrumMasters entre diferentes Times de Scrum nas reuniões de Sprint Retrospective, quando há vários Times de Scrum trabalhando próximos.

O ScrumMaster estimula todos os membros do Time de Desenvolvimento e o Product Owner a participarem das atividades, prestando especial atenção àqueles que se mantêm calados e àqueles que fazem exatamente o oposto, deixando pouco espaço para os outros se manifestarem. O ScrumMaster também cuida da agenda da reunião, assegurando-se que nela aconteçam todas as atividades necessárias para se chegar a resultados e garantido que o seu *timebox* seja cumprido.

Neutralidade

O ScrumMaster, no entanto, não participa diretamente na realização das atividades ou interfere com suas opiniões nas ideias e no processo decisório do Time de Scrum. Para que as melhorias sejam corretamente identificadas e colocadas em prática, é importante que os membros do Time de Desenvolvimento e o Product Owner sintam-se responsáveis pelo processo de geração de ideias, sentindo-se, assim, igualmente responsáveis por seus resultados.

Impedimentos

O ScrumMaster também exerce na reunião o importante papel de não deixar passarem despercebidos pontos que exigirão sua atuação direta. Impedimentos dificultam consideravelmente ou obstruem o trabalho do Time de Desenvolvimento, e sua remoção é responsabilidade do ScrumMaster. Embora sejam identificados pelo Time de Desenvolvimento no seu trabalho diário e imediatamente comunicados ao ScrumMaster, não é incomum impedimentos emergirem na reunião de Sprint Retrospective na forma de pontos a serem melhorados. A responsabilidade pela resolução desses impedimentos detectados na reunião também recai sobre o ScrumMaster.

20.2.3 Participação do Product Owner

O Product Owner é membro do Time de Scrum e, como tal, colabora com as melhorias incrementais contínuas e participa das reuniões de Sprint Retrospective. No entanto, alguns comportamentos disfuncionais podem prejudicar a sua participação nessas reuniões. Por essa razão, não é incomum encontrarmos recomendações equivocadas de que o Product Owner não deve participar delas. Enquanto facili-

tador, todavia, é papel do ScrumMaster trabalhar para garantir uma participação positiva e efetiva de todos os membros do Time de Scrum nas reuniões de Sprint Retrospective.

Na presença de um Product Owner não colaborativo ou que se coloque em um nível hierarquicamente superior, os membros do Time de Desenvolvimento podem se sentir intimidados caso questões a serem levantadas toquem em pontos sensíveis ou estejam diretamente relacionadas com o trabalho do Product Owner e, assim, essas questões podem acabar não se revelando durante a reunião. Dessa forma, a reunião de Sprint Retrospective torna-se ineficiente.

A intimidação pode também acontecer na direção contrária. Os membros do Time de Desenvolvimento podem entender que devem aproveitar a presença do Product Owner e exercer um foco excessivo em problemas relacionadas ao trabalho dele. O Product Owner, nesse caso, passa a ser o centro das atenções durante a reunião, ou pior, um alvo de reclamações e acusações. Esse comportamento é comum quando o Product Owner é ausente durante o Sprint, o que, de fato, configura-se como um problema a ser resolvido.

A ocorrência de algum desses casos depende de que tipo de relação Product Owner e Time de Desenvolvimento cultivam. De forma geral, quanto mais próximos Product Owner e Time de Desenvolvimento trabalharem e quanto mais interagirem no seu dia a dia, mais a participação do Product Owner na reunião será vista como natural e positiva.

20.2.4 Dinâmica básica de uma retrospectiva

Descrevemos em seguida um exemplo de como pode se parecer a dinâmica básica de uma reunião de Sprint Retrospective. É bom destacar que existem diversas formas de se fazer essa reunião, mas objetivo final é sempre o de se realizarem melhorias contínuas a partir de planos de ação exequíveis.



Figura 20.1: Quadro básico de retrospectiva

- 1) Os membros do Time de Desenvolvimento, Product Owner e o ScrumMaster se reúnem em uma sala privada para realizar a reunião;
- 2) Time de Desenvolvimento e Product Owner repassam rapidamente os pontos levantados na reunião de retrospectiva anterior. O objetivo é verificar se e como as ações definidas para realizar as melhorias foram executadas, e que pontos importantes ainda restaram para serem tratados em Sprints futuros;
- 3) Time de Desenvolvimento e Product Owner utilizam alguns minutos para lembrar os fatos mais marcantes do Sprint que está se encerrando. Para viabilizar essa atividade, o ScrumMaster pode sugerir o uso de diferentes práticas;
- 4) Mais alguns minutos são então reservados para que os membros do Time de Desenvolvimento e o Product Owner reflitam e escrevam (em geral, em notas adesivas) o que consideram que foi bem no Sprint que está se encerrando e o que consideram que pode ser melhorado para futuros Sprints;
- 5) Em seguida, os membros do Time de Desenvolvimento e o Product Owner colam suas notas adesivas em um quadro com as colunas “o que foi bem” e “o que pode

“melhorar” (um exemplo desse quadro pode ser visto na figura 20.1);

- 6) O ScrumMaster ou algum dos outros participantes agrupa itens parecidos ou relacionados. Os membros do Time de Desenvolvimento e o Product Owner ordenam esses grupos de itens de acordo com sua importância;
- 7) Time de Desenvolvimento e Product Owner, em seguida, discutem em torno dos grupos de itens levantados. Se houver itens demais para serem discutidos dentro do *timebox* da reunião e para serem tratados no próximo Sprint, os participantes devem concentrar-se apenas nos itens mais importantes, em geral não mais que dois ou três. Para definir quais são os itens mais importantes, pode-se utilizar como critério o número de participantes que apontaram o item ou alguma forma de votação;
- 8) Os membros do Time de Desenvolvimento e o Product Owner repassam rapidamente cada item da coluna “o que foi bem”, comprometendo-se a manter ou repetir em Sprints futuros esses pontos identificados como positivos, caso faça sentido. As pessoas refletem sobre como tornar isso possível;
- 9) Com relação aos pontos a melhorar, Time de Desenvolvimento e o Product Owner buscam identificar as causas raízes dos problemas e, a partir daí, as ações necessárias (planos de ação) para colocar as melhorias correspondentes em prática, de forma a tornar seu trabalho mais eficiente já nos próximos Sprints. Planos de ação são mais efetivos quando indicam “o quê”, “quem” e “quando”, ou seja, qual é a questão a ser tratada, quem ficará responsável por ela e em que prazo ela será tratada;
- 10) Após repassar todos os pontos, traçar planos de ação adequados e registrá-los de alguma forma (em um papel à parte, por exemplo, ou até mesmo em notas adesivas coladas sobre os itens da coluna “o que pode melhorar”), os presentes dão a reunião como encerrada.

20.2.5 Algumas variações na retrospectiva

O estabelecimento de uma rotina repetitiva pode ser uma receita de fracasso em médio prazo para retrospectivas de um Time de Scrum. A escolha de técnicas apropriadas para cada situação e a simples variação nas práticas utilizadas pode tornar a reunião mais dinâmica, o que mantém os membros do Time de Scrum envolvidos

e interessados. O ScrumMaster, enquanto facilitador, geralmente escolhe e sugere como se dará a reunião.

O livro “*Agile Retrospectives: Making Good Teams Great*”, de Esther Derby e Diana Larsen, oferece, além de um *framework* para a execução das retrospectivas, diversas práticas que podem ser utilizadas (Derby & Larsen, 2006). Mostramos aqui algumas das práticas mais populares entre times Ágeis:

- **coluna “a fazer”:** adiciona-se mais uma coluna ao quadro da retrospectiva, onde serão colocadas notas adesivas com as ações de melhorias planejadas. As melhorias podem estar alinhadas aos pontos da coluna “O que pode melhorar” correspondentes;
- **coluna “delta”:** a coluna “o que pode melhorar” é trocada pela coluna “delta”, que simboliza mudança. Assim, o foco se desloca do que pode melhorar para a melhoria em si, ou seja, as notas adesivas coladas nessa coluna já descrevem as mudanças a serem realizadas;
- **começar-parar-tentar:** ao invés do quadro tradicional, divide-se o quadro nas colunas “Começar a fazer”, “Parar de fazer” e “A tentar”. Os membros do Time de Desenvolvimento colam na primeira coluna notas adesivas com ações que consideram que devem começar a fazer, na segunda coluna ações que realizaram no último Sprint, mas que consideram que devem parar de fazer e, na terceira, ações a se tentar no próximo Sprint;
- **classificação dos pontos:** existem diversas possíveis classificações para os pontos positivos e os pontos a melhorar que podem ser úteis para o Time de Desenvolvimento, entre as quais destacamos:
 - classificar os pontos a melhorar entre “Time” e “ScrumMaster” (ou “Organização”), o que significa, respectivamente, pontos sobre os quais o próprio Time de Desenvolvimento deve agir e pontos a respeito dos quais somente o ScrumMaster pode fazer algo, pois são questões organizacionais;
 - classificar os pontos a melhorar entre “Ação” e “Gradual”, ou seja, entre pontos para os quais é possível se traçar planos de ação mais precisos ou objetivos e pontos para os quais as mudanças são mais subjetivas e graduais, como mudanças comportamentais;

- classificar tanto os pontos positivos quanto os pontos a melhorar entre “Causado pelo Scrum”, quando a questão foi causada pelo uso do Scrum (repare que aqui geralmente são pontos positivos), e “Visível pelo Scrum”, nos casos em que foi o Scrum que tornou a questão visível;
- **linha do tempo:** o Time de Desenvolvimento traça uma linha do tempo com os acontecimentos mais importantes do Sprint e, a partir deles, identifica os pontos positivos e os pontos a melhorar. Observe que o próprio Gráfico de Sprint Burndown pode ser utilizado como linha do tempo, com a vantagem que seus pontos de inflexão para cima e linhas retas podem indicar impedimentos ou problemas que ocorreram durante o Sprint. Nesse caso, pode ser interessante que os membros do Time de Desenvolvimento colam notas adesivas sobre pontos relevantes no próprio Gráfico de Sprint Burndown, indicando esses acontecimentos (veja a figura 20.2).

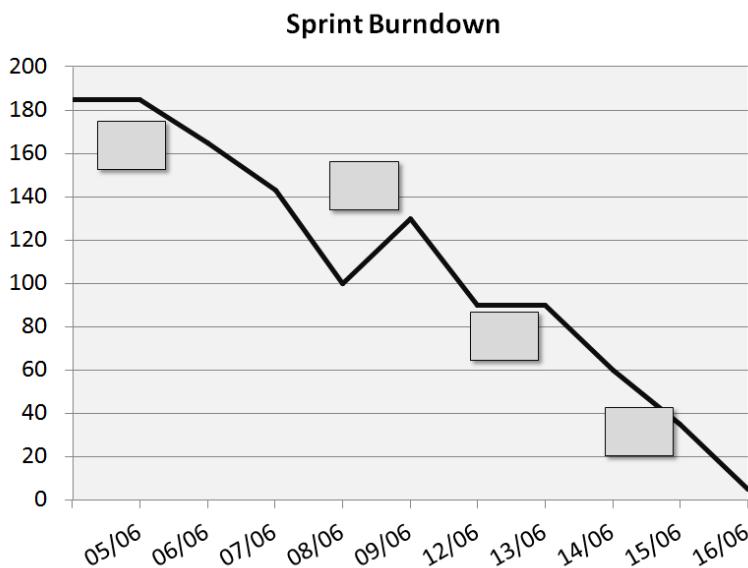


Figura 20.2: Sprint Burndown utilizado como linha do tempo para a retrospectiva

- **foco único:** caso tenha havido algum problema importante suficiente que tenha causado grande impacto no Sprint e que o Time de Desenvolvimento acre-

dite que poderá se repetir, a busca de soluções para esse problema pode ser o tema único (ou, ao menos, central) da reunião.

20.3 O QUE NÃO DEVE ACONTECER NA SPRINT RETROSPECTIVE?

A reunião de Sprint Retrospective não deve ser utilizada para se identificarem ações de melhoria no produto, trabalho que acontece na reunião de Sprint Review.

O principal foco da reunião de Sprint Retrospective é a identificação do que precisa ser melhorado nos processos de trabalho do Time de Scrum. Por essa razão, é necessário que falhas ocorridas durante o Sprint que se encerra sejam identificadas e exploradas com franqueza. Assim, embora não se devam apontar erros individuais nessa reunião, o trabalho de inspeção envolve a capacidade dos participantes de expor, em algum grau, suas próprias limitações e pontos fracos e, principalmente, as do Time de Scrum como um todo. Diante dessa necessidade, o Time de Scrum enfrenta importantes desafios que interferem em sua capacidade de executar uma reunião de Sprint Retrospective eficaz.

20.3.1 Busca de culpados

Norm Kerth, em seu livro de 2001 sobre retrospectivas de projetos, definiu o que chamou de Principal Diretiva das Retrospectivas (Kerth, 2001).

PRINCIPAL DIRETIVA DAS RETROSPECTIVAS

“Independente do que descobrirmos, entendemos e verdadeiramente acreditamos que todos fizeram o melhor trabalho que puderam, dados o que sabiam no momento, suas competências e habilidades, os recursos disponíveis, e a situação que se apresentou.”

Os objetivos de reuniões de Sprint Retrospective são sempre construtivos. O foco dos participantes da reunião deve se manter no desejo objetivo de melhoria, em aprender-se com os erros cometidos.

Buscar culpados para os problemas que aconteceram durante o Sprint tem efeitos extremamente negativos sobre o andamento da reunião e sobre seus resultados. Em

hipótese alguma devem-se fazer acusações, apontar nomes ou citar erros individuais, relacionando-os com problemas que ocorreram no Sprint.

O ScrumMaster, como facilitador, trabalha para que a busca de culpados não aconteça, de forma que os membros do Time de Desenvolvimento e Product Owner se sintam mais seguros para falar livremente sobre os problemas.

20.3.2 Insegurança e medo de exposição

Algum nível de exposição dos presentes é necessário para uma participação efetiva na reunião de Sprint Retrospective. No entanto, falar sobre o que não funcionou bem em seu próprio trabalho e no trabalho de outros que estão presentes não é uma tarefa fácil para a maioria das pessoas. Muitos não estão acostumados com o nível de exposição necessário e se sentem desconfortáveis com a possibilidade de haver qualquer tipo de enfrentamento. Essa dificuldade é ainda mais evidente em indivíduos habituados até então com o trabalho no estilo comando e controle.

Os membros do Time de Scrum poderão se sentir ainda mais intimidados caso haja algum cliente ou parte interessada importante do projeto presente na reunião, que esteja direta ou indiretamente relacionado a causas de problemas ocorridos durante o Sprint. Nesses casos, os membros do Time de Scrum não se sentem em um ambiente seguro para falar de problemas, e assim não será surpreendente se até mesmo problemas bastante visíveis não forem mencionados na reunião. Por essa razão, somente estão presentes na reunião pessoas diretamente convidadas pelo Time de Scrum.

Os membros do Time de Desenvolvimento, no entanto, devem se sentir confortáveis para discutir problemas com o Product Owner. Caso não haja um ambiente de confiança que permita essa transparência, uma disfunção está configurada e cabe ao ScrumMaster, enquanto facilitador, trabalhar para ajudar os envolvidos a resolvê-la.

De forma geral, em uma retrospectiva saudável os membros do Time de Scrum conseguem levantar os pontos a melhorar sem inibição e sem medo de sofrer qualquer retaliação, pois sabem que todos os presentes compreendem que identificar possíveis melhorias é positivo para toda a organização. Essas questões, no entanto, variam naturalmente de Time de Scrum para Time de Scrum, de acordo com as relações estabelecidas entre os envolvidos no projeto.

20.3.3 Conflitos da diversidade

A diversidade de personalidades presentes em um Time de Scrum e, portanto, em uma reunião de Sprint Retrospective, favorece a manifestação de uma variedade de conflitos durante a reunião. As pessoas naturalmente têm diferentes opiniões, pontos de vista e interesses e, quanto mais heterogêneo for o grupo, mais essas diferenças se acentuam.

Um ambiente hostil ou simplesmente não amigável pode levar os membros do Time de Desenvolvimento ou Product Owner a adotarem posturas defensivas e a se fecharem, deixando consciente ou inconscientemente de trazer à tona pontos que ameacem a sua imagem ou a de colegas mais próximos perante o resto do Time de Scrum.

O ScrumMaster atua dentro e fora das retrospectivas, buscando dinâmicas que aumentem o entrosamento dos membros do Time de Scrum e que os estimulem a construir objetivos comuns, levando-os a funcionar como um verdadeiro time.

20.3.4 Pensamento grupal

Uma armadilha relativamente comum em grupos muito coesos é a tendência de seus membros criarem pressões intensas sobre si mesmos para obter e manter o consenso ou a conformidade. Essas pressões internas anulam a motivação do grupo em buscar, por meio do pensamento crítico, caminhos alternativos de ação e, assim, o grupo acaba por tomar decisões disfuncionais. Esse processo nocivo é chamado de pensamento grupal, ou *groupthink* (Janis, 1982). Um Time de Scrum ou Time de Desenvolvimento que incorre em pensamento grupal pode apresentar alguns dos seguintes sintomas:

- acredita ser invulnerável a falhas, apresentando um otimismo excessivo e assumindo riscos desnecessários;
- trata membros que questionam ou discordam da opinião da maioria como desleais e exerce pressões para que se adequem às opiniões do resto do time;
- acredita possuir uma moralidade inerente, independente das ações de seus membros, o que os leva a ignorar consequências morais e éticas de suas decisões;

- cria estereótipos de pessoas externas que podem representar uma ameaça a seu trabalho, caracterizando-as como más demais para que seja possível negociar com elas ou tão fracas e tolas que não representam uma ameaça a ser considerada;
- acredita existir uma unanimidade em torno da visão da maioria de seus membros, de forma que o silêncio de uma minoria diante de discussões ou argumentações é tratado como concordância;
- induz seus membros a se autocensurarem por temerem a desaprovação dos outros membros ou por minimizarem a relevância de suas questões diante do aparente consenso do time;
- induz o surgimento de membros que protegem o time de informações externas que possam ameaçar os valores já estabelecidos;
- cria explicações coerentes e aceitáveis de forma a ser capaz de ignorar qualquer *feedback* negativo que, caso fosse levado em consideração, levaria seus membros a reconsiderarem suas concepções ou suposições.

Times auto-organizados possuem uma tendência natural a serem muito coesos e, assim, são particularmente vulneráveis ao pensamento grupal (Manz & Neck, 1997). A reunião de Sprint Retrospective é, provavelmente, o momento em que esses fatores podem se tornar mais visíveis e por essa razão é quando o ScrumMaster pode atuar com maior intensidade para detectar, criar visibilidade e estimular o time a resolver o problema.

CAPÍTULO 21

Release

- Objetivo: entregar os Incrementos do Produto gerados para serem utilizados;
- Quando: frequentemente, quando já se produziu valor suficiente para ser utilizado;
- Participantes obrigatórios: Time de Desenvolvimento e Product Owner;
- Saídas esperadas: produto utilizável, em funcionamento.

21.1 O QUE É A RELEASE?

Release é a entrega de um ou mais Incrementos do Produto prontos, gerados pelo Time de Desenvolvimento em um ou mais Sprints sucessivos, para que sejam utilizados. Em conjunto e somados ao que já foi entregue anteriormente, esses Incrementos do Produto formam um produto que possui valor suficiente para ser utilizado.

Projetos com Scrum realizam Releases frequentes, com intervalos máximos de alguns poucos Sprints entre elas.

A realização de Releases ao longo do projeto tem três objetivos principais: obter *feedback* frequentemente, prover retorno ao investimento dos clientes e dar um senso de progresso a eles.

- **obter *feedback*:** uma vez realizada a Release, o Product Owner busca, principalmente junto a usuários do produto, impressões e opiniões sobre o que foi recebido e utilizado, e o que se espera da próxima Release. A partir desse *feedback*, mudanças serão realizadas no Product Backlog e, assim, o produto é construído incrementalmente;
- **prover retorno ao investimento dos clientes:** busca-se realizar Releases frequentemente para usuários finais do produto, pois em cada entrega é gerado um retorno ao investimento realizado pelos clientes do projeto. No entanto, dependendo de características do produto e de sua estratégia de negócios, pode fazer sentido entregar para quem de fato irá utilizar o produto apenas ao final do projeto ou, ao menos, com pouca frequência. Produtos de prateleira, por exemplo, em geral não são oferecidos ao público antes do final de seu projeto;
- **dar um senso de progresso do projeto:** ao receber uma Release, os clientes do projeto e demais partes interessadas têm visibilidade do estado atual do projeto, isto é, o que já está pronto e o que vislumbram que ainda há a ser feito em direção à Visão do Produto.

A estratégia de Releases do projeto é de inteira responsabilidade do Product Owner, que deve defini-la e, sempre que necessário, modificá-la. Essa estratégia estabelece, por exemplo, quais dos objetivos entre os descritos acima cada Release busca alcançar, o que inclui a frequência das Releases e quem irá recebê-las. A estratégia de Releases também garante que as Releases estejam alinhadas com a estratégia de negócios do produto e que sejam tecnicamente viáveis.

21.2 COMO É A RELEASE?

Consideram-se dois pontos essenciais em uma estratégia de Release: quando ou com que frequência serão realizadas e quem irá recebê-las. Assim, classificamos as Releases de um projeto quanto à frequência e quanto a quem as recebe.

21.2.1 Quanto à frequência

Com relação a quando ou com que frequência deverão ser realizadas, as Releases podem ser classificadas em Release por valor, Release por Sprint, Release por item e Release por plano.

Release por valor

Dependendo da natureza do negócio, a Release pode ser realizada quando o Product Owner julgar que os Incrementos do Produto já gerados pelo Time de Desenvolvimento nos Sprints representam valor de negócio suficiente para que valha a pena entregá-los para usuários a utilizarem. Dessa forma, o Time de Desenvolvimento segue gerando Incrementos do Produto prontos Sprint a Sprint até que o Product Owner julgue que o que foi produzido é suficiente para se realizar uma Release.

Release por Sprint

Em outros casos, pode fazer sentido realizar uma Release ao final de cada Sprint. Nesse cenário, a reunião de Sprint Planning já é suficiente para que os clientes e demais partes interessadas enxerguem, por meio do Product Owner, o que será recebido e quando. Assim, o Time de Desenvolvimento trabalha para que o resultado final do Sprint seja imediatamente entregue, dependendo apenas da aprovação do Product Owner ao final do Sprint. Realizar a Release em cada Sprint é extremamente Ágil, pois antecipa o *feedback* dos usuários sobre o que foi produzido e maximiza as chances de melhorias no produto.

Release por item

Em um cenário ainda mais Ágil, os itens desenvolvidos pelo Time de Desenvolvimento são entregues durante o próprio Sprint, como parte de sua Definição de Pronto utilizada, o que pode envolver uma aprovação imediata do Product Owner. Esse tipo de abordagem muitas vezes não é viável e funciona em casos muito específicos, como por exemplo no desenvolvimento de sistemas disponíveis na Internet.

Release por plano

Pode-se, para cada Release, criar um plano de alto nível do que será desenvolvido. Pode-se realizar, nesse caso, uma reunião de Release Planning para cada Release, antes de seu início. Nessa reunião, é estabelecido o Plano da Release, que contém uma

data aproximada para a Release, uma Meta a ser atingida e um conjunto de itens selecionados a partir do alto do Product Backlog. O progresso em direção à data da Release e, portanto, ao cumprimento da Meta da Release deve ser inspecionado em cada Sprint, e as ferramentas mais utilizadas com esse propósito são o Gráfico de Release Burndown e o Gráfico de Release Burnup (veja “[14 Gráficos de Acompanhamento do Trabalho](#)”).

É comum medir o tamanho da Release pelo número de Sprints que acontecerão até a entrega. É igualmente comum nesses casos chamar-se também de Release todo o período de trabalho realizado desde o início do primeiro Sprint planejado para a Release até a entrega propriamente dita. Dizemos, por exemplo, que “*essa Release durará cinco Sprints*” ou que “*estamos trabalhando na quinta Release*”.

21.2.2 Quanto a quem a recebe

Uma Release deve idealmente ser realizada para os usuários finais, de forma a se obter *feedback* sobre os Incrementos do Produto gerados e proporcionar retorno ao investimento para os clientes do projeto. Quando esse cenário não é possível, a Release pode ser feita para um grupo de usuários intermediários ou de usuários selecionados, e assim, no mínimo, garante-se um *feedback* para reduzir os riscos do projeto.

Em um mesmo projeto pode aparecer, em momentos distintos, qualquer um dos três cenários definidos em seguida, mas uma Release para os usuários finais obrigatoriamente ocorrerá no final do projeto.

Release para os usuários finais

O Incremento ou Incrementos do Produto são disponibilizados para os usuários finais do produto. Tendo as funcionalidades que mais necessitam em suas mãos, esses usuários as utilizarão, gerando retorno ao investimento feito pelos clientes do projeto.

Sempre que é possível de ser adotado, esse cenário representa o menor risco, pois Releases frequentes para usuários reais os permitem dar *feedback* a partir do uso real do produto.

Release para usuários intermediários

Quando não é possível realizar a Release para os usuários finais do produto, um grupo de pessoas pode ser escolhido para representá-los. Esses usuários intermediários são geralmente selecionados ou na própria organização que gera o produto,

ou nos clientes. Podem ser especialistas no negócio do produto ou em seus usuários finais, especialistas em usabilidade ou quaisquer pessoas capacitadas a utilizar o produto e a verificar o que é necessário para atrair os usuários e satisfazer suas necessidades. Eles receberão essa Release interna e serão responsáveis por prover *feedback* sobre o que lhes foi entregue.

Esse cenário é comum nos produtos em que os usuários reais são anônimos e não há sentido ou não é interessante disponibilizar um produto parcial. Exemplos incluem produtos de prateleira e determinados *softwares* de jogos.

Release para usuários selecionados

Em produtos com um grande número de usuários, um grupo menor e representativo desses usuários reais pode ser escolhido para receber a Release. Esse grupo utilizará cada conjunto de Incrementos do Produto entregue com a consciência de que se trata de um produto parcial e será responsável por prover *feedback* sobre ele.

Em muitos casos, o benefício e estímulo para que continuem usando o produto e provendo *feedback* é a possibilidade de experimentar o produto antes de todos ou a possibilidade de utilizá-lo de graça, por exemplo.

Os usuários selecionados com esse propósito são geralmente chamados de *beta-testers*, *beta-users* ou grupos de foco.

CAPÍTULO 22

Release Planning

- Objetivo: planejamento da próxima Release;
- Quando: antes do início do trabalho para a Release (em geral, ao final do último Sprint da Release anterior ou antes do primeiro Sprint do projeto);
- Duração: não há duração estabelecida, mas é importante se definir um *time-box*;
- Participantes obrigatórios: Product Owner, Time de Desenvolvimento e ScrumMaster;
- Saídas esperadas: Plano da Release.

22.1 O QUE É A RELEASE PLANNING?

Em projetos que utilizam Releases por plano como parte de sua estratégia de Releases, podem-se realizar as reuniões de Release Planning. Product Owner e Time de

Desenvolvimento, facilitados pelo ScrumMaster, encontram-se para criar o Plano da Release, que indica qual o objetivo a ser alcançado pela Release e quando será alcançado.

22.2 PLANO DA RELEASE

O Plano da Release contém:

- a Meta da Release, que é um objetivo de negócios a ser alcançado por meio da entrega. A Meta da Release é um passo em direção à Visão do Produto;
- a data exata ou aproximada em que a entrega será realizada;
- um conjunto de itens selecionados do Product Backlog para a Release. Visa-
ndo se alcançar a Meta da Release, esses itens serão desenvolvidos do mais
importante (granularidade mais fina) ao menos importante (granularidade
mais grossa), até se chegar à data da Release.

A abordagem de uso de um Plano de Release e de reuniões de Release Planning, conforme descritas neste capítulo, somente é viável se o Time de Desenvolvimento atribuir estimativas para os itens do Product Backlog.

22.3 COMO É A RELEASE PLANNING?

22.3.1 Agendamento e duração

Como o Time de Desenvolvimento trabalha continuamente dentro de Sprints, um atrás do outro, não existe um intervalo entre dois Sprints que possa ser utilizado para realização da reunião de Release Planning. Assim, a reunião de planejamento para uma Release é geralmente realizada durante o último Sprint da Release anterior, preferencialmente perto do seu final. O uso desse tempo reduz um pouco o quanto o Time de Desenvolvimento será capaz de produzir nesse último Sprint, e assim pode ser levado em consideração em seu planejamento.

Caso se trate da primeira Release do projeto, em geral a reunião de Release Planning é realizada antes do início dos Sprints, ou seja, antes do início do desenvolvimento do produto. No entanto, como nesse momento se conhece pouco sobre a capacidade de produção do Time de Desenvolvimento, é também comum esperarem-se dois ou três Sprints para se adquirir mais parâmetros, e então realizar o planejamento da primeira Release do projeto.

Alternativamente, alguns times preferem realizar a reunião de Release Planning imediatamente antes da reunião de Sprint Planning do primeiro Sprint da Release a ser planejada.

Não há duração oficial estabelecida para a reunião de Release Planning. Recomenda-se, no entanto, que se estabeleça um tempo máximo de duração para ela (*timebox*) e que esse tempo não ultrapasse um dia.

22.3.2 A reunião

Podemos identificar dois diferentes cenários para uma reunião de Release Planning. No primeiro cenário, é dada uma data para a Release, mas não se sabe qual a Meta a ser alcançada. É o caso em que é necessário fazer uma entrega até uma determinada data. No segundo cenário, é dada uma necessidade de negócios (que será a própria Meta da Release) e pergunta-se quando será possível alcançá-la.

Em ambos cenários, itens suficientes do Product Backlog devem estar estimados para que seja possível realizar o planejamento.

Dada a data da Release. A partir da data da Release, dados um Product Backlog estimado, o tamanho constante do Sprint e a Velocidade do Time de Desenvolvimento, é possível se determinar o resto do plano: um conjunto de itens selecionados do Product Backlog para a Release e a Meta da Release. Vamos explicar com um exemplo.

Digamos que o Product Owner estabeleça, a partir de questões de negócios, a necessidade de uma entrega a ser realizada daqui a dois meses. Digamos também que o tamanho do Sprint utilizado pelo Time de Desenvolvimento seja de duas semanas. Assim, podemos calcular quantos Sprints serão executados até a data dessa Release:

$$\text{n. Sprints} = (\text{Tempo Total} \div \text{Tam. do Sprint}) = (8 \text{ semanas} \div 2 \text{ semanas}) = 4 \text{ Sprints}$$

Sabemos, portanto, que quatro Sprints serão executados até a data dessa Release. Também podemos estimar quantos pontos o Time de Desenvolvimento entregará nessa Release:

$$\text{n. pontos} = (\text{Velocidade} \times \text{n. Sprints}) = (30 \text{ pontos/Sprint} \times 4 \text{ Sprints}) = 120 \text{ pontos}$$

Para determinar o escopo provável da Release, parte-se do topo do Product Backlog e se desce, somando-se as estimativas de cada item dadas pelo Time de Desenvolvimento até se chegar em algo próximo (um pouco mais ou um pouco menos) que esses 120 pontos.

É importante, no entanto, entender que essa extração traz uma previsão de baixa precisão, e assim gera um planejamento que deve ser revisto Sprint a Sprint.

Dada a Meta da Release. Em outro caso, digamos que, ao invés de fornecer uma data, o Product Owner deseja saber em quanto tempo uma determinada necessidade de negócios poderá ser atendida. Essa necessidade de negócios é a própria Meta da Release.

Primeiramente, o Product Owner deve garantir que o Product Backlog esteja ordenado e com os itens adequados para atender a essa necessidade. O Product Owner partirá então do topo do Product Backlog e descerá, somando as estimativas de cada item dadas pelo Time de Desenvolvimento, até considerar que já percorreu os itens necessários para atender a essa necessidade de negócios. Assim, digamos que a soma dessas estimativas seja 184 pontos. Se o Time de Desenvolvimento produz, em média, 30 pontos por Sprint (Velocidade), poderemos então estimar em quantos Sprints ele será capaz de queimar esses pontos:

$$\text{n. Sprints} = (\text{n. pontos} \div \text{Velocidade}) = (184 \text{ pontos} \div 30 \text{ pontos/Sprint}) = \text{aprox. 6 Sprints}$$

Podemos prever, portanto, que o Time de Desenvolvimento será capaz de queimar esses pontos em seis Sprints (parte inteira), ou seja, daqui a três meses. É sempre importante lembrar que essa estimativa é de baixa precisão e deve ser revista em cada Sprint.

22.3.3 Granularidade dos itens

Como resultado da reunião de Release Planning, um conjunto de itens terá sido selecionado a partir do alto do Product Backlog. Na realidade, esses itens não são separados do Product Backlog, de forma que não existe um Release Backlog. Faz-se apenas algum tipo de marcação nos itens que pertencem à Release planejada, como uma etiqueta.

Como são parte do Product Backlog, esses itens prováveis para a Release têm diferentes níveis de granularidade. Os itens que estão no alto do Product Backlog e que assim, de acordo com o plano, serão desenvolvidos nos primeiros Sprints da Release, devem ser menores e representar mais detalhes. Possuem, portanto, possuem granularidade mais fina. Os itens que supostamente serão desenvolvidos nos Sprints seguintes da Release, portanto mais abaixo no Product Backlog, serão maiores e mais vagos, com granularidade mais grossa.

A medida que o trabalho de desenvolvimento avança Sprint a Sprint em direção à Release, os itens do alto vão sendo retirados do Product Backlog para desenvolvimento. Assim, os itens seguintes que chegam ao topo do Product Backlog serão gradativamente detalhados, refinando-se a sua granularidade.

22.3.4 Escopo da Release

Embora persiga-se uma meta de negócios clara, o escopo da Release é aberto. Assim, novos itens irão surgir durante a Release e outros irão desaparecer. Itens existentes irão evoluir, ganharão mais detalhes, serão divididos em itens menores e serão reordenados. Os itens menos importantes, resultados dessa evolução, serão relegados a partes mais baixas do Product Backlog. Assim, ao se atingir a data prevista para a Release, os itens que restarão na lista de itens prevista inicialmente para a Release serão os de menor importância e, assim, há uma maior chance da Meta da Release ter sido alcançada.

É importante destacar que a reunião de Release Planning de forma alguma substitui as reuniões de Sprint Planning. O escopo de um Sprint futuro da Release está em aberto até que se realize a sua reunião de Sprint Planning, ou seja, conjunto de itens que entrarão em cada Sprint Backlog será definido na sua reunião de Sprint Planning respectiva, e não na reunião de Release Planning.

CAPÍTULO 23

Refinamento do Product Backlog

- Objetivo: refinamento e preparação do Product Backlog;
- Quando: pelo Product Owner, sempre que necessário. Durante o Sprint, é um trabalho contínuo, eventual ou realizado em sessões agendadas entre Product Owner e Time de Desenvolvimento;
- Duração: não há duração estabelecida, mas em geral o Time de Desenvolvimento não utiliza no total mais do que 10% do seu tempo produtivo em cada Sprint;
- Participantes obrigatórios: Product Owner e Time de Desenvolvimento;
- Saídas esperadas: o Product Backlog ordenado, planejável, emergente e gradualmente detalhado. Espera-se obter uma quantidade suficiente de itens preparados para o próximo Sprint.

23.1 O QUE É O REFINAMENTO DO PRODUCT BACKLOG?

O Product Backlog é progressivamente atualizado e detalhado ao longo de todo o projeto. Esse trabalho é chamado de Refinamento do Product Backlog, e visa garantir que ele seja:

- ordenado, para maximizar o retorno sobre o investimento dos clientes do projeto;
- planejável, de forma que o Time de Desenvolvimento e o Product Owner sejam capazes de planejar o desenvolvimento do produto;
- emergente, refletindo o dinamismo do ambiente de mudanças no qual o projeto está imerso;
- gradualmente detalhado, de forma que seus itens possuam um detalhamento adequado e que, assim, um número suficiente de itens esteja preparado para o Sprint seguinte.

O trabalho de Refinamento do Product Backlog inclui a adição de novos itens, a remoção de itens que não farão mais parte do produto, o desmembramento de itens maiores em itens menores, a junção de itens menores em um item maior, o detalhamento de itens, seu reordenamento e, possivelmente, a criação de algum tipo de estimativa.

23.2 COMO É O REFINAMENTO DO PRODUCT BACKLOG?

23.2.1 Participantes

Pela natureza emergente do Product Backlog, o Refinamento do Product Backlog é um trabalho contínuo, realizado pelo Product Owner ao longo de todo projeto. O Product Owner tem a prerrogativa de alterar o Product Backlog sempre que se fizer necessário, adicionando, removendo, dividindo e detalhando seus itens.

No entanto, o Product Owner não realiza todo o Refinamento do Product Backlog sozinho. Embora o Product Owner deva prepará-los da melhor forma possível, o Time de Desenvolvimento é quem utilizará os itens do Product Backlog para realizar seu trabalho no Sprint seguinte, transformando-os em um Incremento do Produto pronto. Assim, enquanto usuário ou “consumidor” desses itens, o Time de

Desenvolvimento interage com o Product Owner durante o Sprint para que, juntos, preparem um número de itens suficiente para serem trabalhados no Sprint seguinte.

É importante destacar, no entanto, que não é uma boa prática o Product Owner chegar a uma sessão de Refinamento do Product Backlog realizada em conjunto com o Time de Desenvolvimento sem trazer itens com algum nível de preparação. Ele em geral o faz utilizando o máximo de informações de que puder dispor. De outra forma, a sessão pode tornar-se cansativa e ineficiente.

Ao final do refinamento, os itens mais importantes do Product Backlog possuem os detalhes necessários para serem colocados em desenvolvimento. Prepara-se durante o Sprint um número de itens que se julgue suficiente para estar no próximo Sprint, mas os itens somente serão de fato escolhidos na reunião de Sprint Planning.

Frequentemente, a criação de estimativas é parte do Refinamento do Product Backlog. Esse trabalho de estimar é realizado unicamente pelo Time de Desenvolvimento. É importante, no entanto, que o Product Owner esteja presente para esclarecer dúvidas que invariavelmente irão surgir.

Alguns Times de Desenvolvimento preferem realizar as sessões de Refinamento do Product Backlog com apenas alguns de seus membros, que em seguida transmitem os resultados para o restante do time que seguiu com o trabalho do Sprint durante a sessão. Para outros, a participação de todos é valorizada, pois pode levar a um melhor compartilhamento das informações e a uma sensação maior de propriedade sobre os resultados das sessões.

23.2.2 Quando ocorre

Cada Time de Scrum possui seu próprio processo para a colaboração entre Product Owner e Time de Desenvolvimento nas atividades de Refinamento do Product Backlog. Para alguns, esse é um trabalho contínuo, que ocorre durante todo o Sprint. Para outros, é um trabalho eventual, muitas vezes solicitado pelo Product Owner, que traz novos itens a serem preparados. Outros Times de Scrum preferem agendar sessões de trabalho em dias e horários específicos, também durante o Sprint. Essas podem ser curtas sessões diárias (o que somente é possível com uma alta disponibilidade do Product Owner), sessões semanais ou até mesmo uma ou duas sessões próximas ao fim do Sprint.

Quando itens são preparados cedo demais, mudanças muito frequentes podem diminuir suas chances de pertencerem ao trabalho a ser realizado no Sprint seguinte, levando à geração de desperdício. Nesses casos, uma preparação mais próxima ao

final do Sprint pode ser mais produtiva. Mas, se forem preparados em apenas uma sessão ao final do Sprint, pode não haver tempo suficiente para reflexões sobre decisões de seus detalhes que podem ser necessárias.

Adicionalmente, uma sessão de Refinamento do Product Backlog pode ser realizada juntamente com ou imediatamente antes de uma reunião de Release Planning exatamente para viabilizar o trabalho a ser realizado nessa reunião, preparando-se itens com estimativas, por exemplo.

Seja qual for o formato escolhido, essa colaboração não deve ocupar muito tempo do trabalho do Time de Desenvolvimento. Usualmente, recomenda-se que não se utilize mais que 10% do tempo produtivo do Time de Desenvolvimento no Sprint com essas atividades.

Parte V

Final

CAPÍTULO 24

Além do Scrum

* por Marcos Garrido e Rodrigo de Toledo

24.1 INTRODUÇÃO

Até aqui, este livro mostrou como o Scrum pode ajudar a se obter sucesso em projetos. No entanto, aprender Scrum é apenas o primeiro passo. O objetivo deste capítulo é guiar o leitor para seus próximos passos na absorção de conhecimentos necessários para promover a constante evolução do profissional, de forma que seja possível obter níveis cada vez maiores de sucesso no uso de práticas Ágeis.

Agrupamos aqui esse conjunto de conhecimentos em quatro grandes dimensões, descritas a seguir:

- **técnica:** a dimensão técnica trata dos conhecimentos técnicos e práticas de engenharia, utilizadas para construir produtos com qualidade e agilidade, eliminando-se desperdícios e gargalos técnicos;

- **time:** essa dimensão é formada pelas questões que envolvem a formação e evolução de times Ágeis, a partir da adoção de práticas que melhorem os processos de trabalho, facilitando a comunicação e a relação entre as pessoas e o desempenho do time;
- **organizacional:** essa dimensão aborda os desafios do relacionamento dos times Ágeis com o resto da organização, visando a criação de um ambiente cooperativo, com espaço para inovação e crescimento a partir do incentivo ao trabalho de times auto-organizados;
- **negócios:** a dimensão de negócios compreende os conhecimentos e técnicas necessários para transformar as demandas dos clientes em incrementos de produto que agreguem valor de negócio.

A figura 24.1 descreve a relação entre as funções normalmente encontradas em empresas de tecnologia e as dimensões de conhecimento necessárias para a evolução do profissional que pretende se aventurar pela Agilidade. A mesma figura descreve também a relação entre as dimensões de conhecimento e os papéis do Scrum, de forma que seja possível identificar as habilidades e conhecimentos necessários para que cada papel possa evoluir nos caminhos da Agilidade.

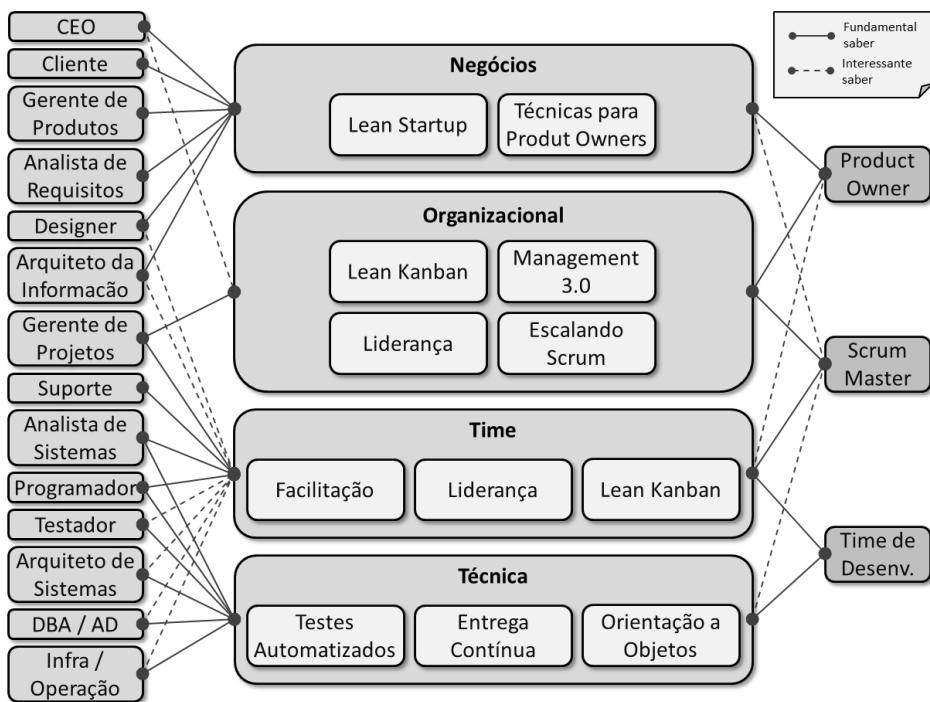


Figura 24.1: Dimensões de conhecimento

A seguir, vamos descrever algumas das habilidades e técnicas encontradas nas quatro dimensões de conhecimento.

24.2 FACILITAÇÃO

O ScrumMaster trabalha como um facilitador para ajudar o Product Owner e Time de Desenvolvimento a serem mais eficientes na realização do seu trabalho. Ele utiliza-se de suas habilidades de facilitação no dia a dia de trabalho do Time de Scrum e em suas reuniões.

No capítulo 7, há uma descrição detalhada do papel de facilitador que é exercido pelo ScrumMaster (veja “[7.1. Quem é o ScrumMaster?](#)”).

Apesar da importância do tema, poucos ScrumMasters parecem investir seu tempo na aquisição e desenvolvimento de suas habilidades em relação às técnicas de facilitação, de forma que seu trabalho não fique apenas focado em reforçar o cor-

reto uso de práticas Ágeis.

As técnicas de facilitação fornecem ferramentas para que o ScrumMaster seja um especialista em:

- identificar e lidar com comportamentos disfuncionais;
- trabalhar para que todos os envolvidos entendam seus próprios objetivos e quais resultados lhes são esperados;
- preparar corretamente o ambiente de trabalho e em que ocorrerão as reuniões;
- atuar proativamente quando os envolvidos perdem seu foco;
- trabalhar para que o grupo chegue a decisões informadas a partir do consenso, maximizando a participação de todos os envolvidos.

Para saber mais sobre técnicas de facilitação, acesse o *site* do livro em <http://livrodescrum.com.br>.

24.3 MANAGEMENT 3.0

Desde que Adam Smith, em seu livro *"A Riqueza das Nações"* de 1776, discutiu a importância da especialização do trabalho, passando pela publicação dos princípios da administração científica por Frederic Taylor em 1911 e chegando até os dias de hoje, as organizações são desenhadas da mesma maneira. O modelo vigente segue uma orientação *top-down*, na qual as decisões são tomadas na cúpula e então irradiadas para os níveis inferiores da organização. Mesmo organizações mais abertas, que já adotaram métodos Ágeis, em sua maioria foram desenhadas seguindo essa lógica *top-down*. Dessa forma, os benefícios no uso dos métodos Ágeis acabam ficando restritos ao dia a dia dos times. Ao mesmo tempo, gestores que lidam com times Ágeis no cotidiano, muitas vezes se veem limitados pelas técnicas tradicionais de gestão.

Management 3.0 é um conjunto de teorias e técnicas, divididas em seis visões de gestão, que visa fornecer a gestores e líderes os conhecimentos necessários para levar a Agilidade para as camadas superiores de gestão. Altos níveis de eficiência podem ser alcançados por meio da formação de times Ágeis empoderados, inovadores, motivados e trabalhando em estruturas que permitam a comunicação eficiente em todas as camadas da organização.

As seis visões do Management 3.0 são descritas a seguir:

- **energizar as pessoas:** como as pessoas são a parte mais importante das organizações, seus gerentes devem fazer o possível para mantê-las ativas, criativas e motivadas. Para isso, faz-se necessário entender a diferença entre motivação intrínseca e extrínseca e conhecer técnicas para identificar o que motiva as pessoas à sua volta;
- **empoderar times:** auto-organização é um valor importante para times Ágeis. Gerentes devem entender como times auto-organizados funcionam, quais são os desafios envolvidos e como construir relações de confiança mútua que levem a times auto-organizados de sucesso;
- **alinhar restrições:** restrições são importantes. Gestores devem entender a diferença entre restrições e regras, de forma que os times possam trabalhar de maneira auto-organizada, seguindo em direção aos objetivos propostos pelos gestores. Além disso, quando não há uma meta ou propósito claro, o resultado pode não ser o esperado. Também é importante que os gestores aprendam quando gerenciar e quando liderar, sabendo utilizar critérios diferentes para criar metas úteis;
- **desenvolver competências:** times só conseguem atingir suas metas se seus membros forem capazes o suficiente. Embora times auto-organizados possam lidar com suas lacunas técnicas, gerentes devem contribuir para o desenvolvimento das competências necessárias dos membros do time;
- **crescer a estrutura organizacional:** o desenho da estrutura organizacional impacta de forma significativa em como a organização funciona. Vários times operam no contexto de uma organização complexa, e por isso é importante considerar uma estrutura que privilegie a comunicação. Gestores precisam entender a diferença entre times funcionais e multifuncionais, e conhecer técnicas para desenhar áreas que funcionam e se comunicam com eficácia;
- **melhorar continuamente:** é um dos maiores desafios nas organizações. Pessoas, times e organizações precisam melhorar continuamente. Na prática, isso significa que gestores e líderes devem agir como agentes da mudança, tentando mudar os complexos sistemas sociais à sua volta.

Para saber mais sobre Management 3.0, acesse o *site* do livro em <http://livrodescrum.com.br>.

24.4 LEAN KANBAN

Os métodos Ágeis trouxeram uma grande mudança nos rumos da computação. O Scrum, seu maior ícone, tem liderado essa nova forma de pensar, quebrando diversos paradigmas das décadas passadas. O método Kanban surgiu depois dessa grande onda, em 2007, explicitando ainda mais essas quebras de paradigmas. Conceitos como fluxo contínuo, foco, visibilidade e melhoria contínua são reforçados.

A **visibilidade** é sempre o primeiro passo: colocar na parede em forma de cartões o que está sendo efetivamente feito pelo time. Há então dois movimentos nessa visibilidade. Um é interno ao time, pois muitas vezes apenas três colunas não representam as etapas de trabalho de um time. É comum, portanto, surgirem outras colunas para etapas, como “em teste”, “integração”, “especificação”, entre outras, além de colunas para representar as filas que ocorrem entre essas fases. Um outro movimento da visibilidade é para fora do time, pois a tendência é que a gestão visual extrapole as paredes do time e comece o cobrir todo o fluxo, “da concepção ao retorno financeiro” (*“from concept to cash”*). Atividades que são realizadas antes e depois da implementação do time passam a ser também mapeadas, como por exemplo: concepção, análise, homologação e subida à produção.

Limitar a quantidade de trabalho (*limited WIP - work in progress*) talvez seja a maior contribuição do Kanban para a Agilidade de um time ou de uma empresa. A afirmação de que quanto menos trabalho fazemos ao mesmo tempo, maior é a nossa eficiência, pode parecer um pouco contraintuitiva, mas como demonstra a Lei de Little, essa restrição é a chave para diminuir tempo de entrega. O time é mais focado e o limite faz com que não entre em zonas de conforto, como começar algo menos importante ou não correr atrás de um impedimento.

Finalmente, o **fluxo**, é uma consequência das ações anteriores (visibilidade e quantidade de trabalho limitado). Com objetivo de manter o fluxo contínuo de entrega, diversos mecanismos emergem. Um dos possíveis efeitos é o de que o Sprint deixe de existir e a entrega passe a ser item a item. Mas observe que é uma opção avançada e, portanto, muitos times que adotam Kanban escolhem permanecer com iterações de *timebox* fixo.

Além desses pontos, há muito o que se aprender e alguns assuntos importantes são tratados somente em treinamentos avançados de Kanban. Por exemplo, como abordar problemas usando um pensamento sistêmico, melhoria contínua baseada nos cinco porquês e relatórios A3, criação de *design* de sistemas de trabalho, modelos econômicos e métricas. Tais assuntos avançados podem se tornar verdadeiras alavancas de melhoria contínua dos times e suas empresas.

Para saber mais sobre Lean Kanban, acesse o *site* do livro em <http://livrodescrum.com.br>.

24.5 TÉCNICAS PARA PRODUCT OWNERS

A função do Product Owner é a chave do sucesso de qualquer projeto com Scrum. O Product Owner determina a Visão do Produto e direciona o Time de Desenvolvimento, fazendo sua conexão com clientes e negócios. Times de Scrum podem avançar extremamente rápido e criar produtos de alta qualidade técnica. Investe-se muito tempo e esforço em tornar os Times de Desenvolvimento cada vez mais produtivos. No entanto, nada é mais improutivo do que criar um produto ou serviço que não atende às necessidades dos clientes, ou seja, que não gera retorno sobre o investimento (ROI).

As três atividades mais importantes de um bom Product Owner são: fatiar, priorizar e descartar.

- **fatiar:** épicos carregam muita incerteza. Fatiar ajuda a reduzir essa incerteza, uma vez que se reduz a quantidade de detalhes necessários para um mesmo item. Além disso, aumenta-se a precisão das estimativas e permite-se um encaixe mais fácil nos Sprints. O Product Owner tem à sua disposição um conjunto de critérios para fatiar os itens de Backlog: desempenho, passos de um fluxo de trabalho, valor de negócio e complexidade, por exemplo. Após fatiar os itens, pode-se utilizar a técnica de descarte para eliminar complexidade desnecessária, conforme explicado mais à frente;
- **priorizar:** uma priorização bem feita permite que os itens mais importantes para o cliente possam ser entregues primeiro. Ao entregar os itens mais importantes mais cedo, aprende-se mais sobre o produto, garante-se o ROI e o cliente fica mais satisfeito. No entanto, priorizar não é uma tarefa fácil. É preciso entender profundamente sobre o mercado e quais são as reais necessidades do cliente. É uma arte. Mas, como toda arte, existem técnicas apuradas que suportam todo o processo. Há inúmeras maneiras para se calcular e maximizar o ROI do Product Backlog e, como cada projeto possui características específicas, não existe uma resposta única que atenda a todos os casos;
- **descartar:** assim como o Product Owner deve priorizar os itens de Backlog para descobrir o que há de mais importante para ser feito, descobrir quais

itens devem ser descartados é fundamental para manter o foco na entrega de valor. Ao fatiar épicos, por exemplo, o Product Owner pode buscar a simplificação, descartando itens que entregam pouco ou nenhum valor e consequentemente não contribuem para a satisfação do cliente. Pode-se utilizar também o conceito de MVP (*Minimum Viable Product* ou produto mínimo viável) para ajudar a eliminar detalhes dos itens.

Treinamentos devem formar o Product Owner, ensinando técnicas como: construção da Visão do Produto e do Roadmap do Produto, trabalho com épicos e personas, escrita de User Stories efetivas, fatiamento de User Stories, atribuição de valor de negócios, criação de critérios de aceitação, priorização, Lean Startup e MVP.

Para saber mais sobre técnicas para Product Owners, acesse o *site* do livro em <http://livrodescrum.com.br>.

24.6 LEAN STARTUP

Cenários em que o cliente é desconhecido (quando se constrói um produto novo para se lançar no mercado) são complexos exatamente porque a ausência do cliente torna o processo de descoberta do produto muito mais difícil de ser executado. No entanto, mesmo quando ele é conhecido (e muitas vezes o próprio contratante do projeto), o processo de descoberta do produto é crucial para o sucesso do projeto. Product Owners tem à disposição um conjunto de técnicas de descoberta de produto que permitem entender as necessidades do cliente. A fase de descoberta é, na verdade, um grande processo de aprendizado, pois é por meio da tentativa e erro que se descobre qual é o produto ideal a ser construído. Quanto mais cedo for feita, melhor, pois é possível entregar valor para o cliente desde o início do projeto. Quando é feita tarde demais, corre-se o risco de jogar fora todo o esforço empregado até então. Uma das formas mais discutidas ultimamente para entender o que o cliente quer é o Lean Startup.

Lean Startup é uma abordagem para a idealização de novos produtos e serviços com base na geração de aprendizado sobre o mercado, o produto e o cliente a partir da validação de hipóteses.

Hipóteses são crenças que o responsável pela concepção do produto tem, baseado em sua experiência pessoal e observação, mas que precisam ser validadas, ou seja, precisam ser transformadas em fatos.

A validação das hipóteses é potencializada a partir do lançamento de um MVP (*Minimum Viable Product* ou produto mínimo viável). O MVP representa a versão

mais simples possível que pode ser desenvolvida com o objetivo de gerar aprendizado sobre o usuário, de forma que as primeiras crenças sobre o produto possam ser validadas.

Embora tenha sido pensado para atender às necessidades de validação de *startups*, o Lean Startup é aderente ao trabalho do Product Owner, que pode passar a trabalhar com hipóteses em vez de requisitos, de forma que o mesmo se certifique de que o seu Product Backlog reflete o problema que o cliente realmente precisa resolver.

Para saber mais sobre Lean Startup, acesse o *site* do livro em <http://livrodescrum.com.br>.

24.7 ESCALANDO SCRUM

Scrum foi concebido para times pequenos, ou seja, que possuem entre 3 e 9 membros. Assim, para empresas de médio e grande porte que possuem times grandes, pode ser necessário adotar técnicas adicionais para se utilizar o *framework*.

Um mecanismo utilizado para ajudar a contornar essa limitação é conhecido como Scrum of Scrums (SofS), e é utilizado quando há múltiplos times trabalhando em um mesmo projeto. SofS é uma reunião adicional à reunião de Daily Scrum, onde representantes dos times se encontram e cada um informa o que seu time realizou desde a última reunião de SofS, o que seu time pretende realizar até a próxima e quais impedimentos enfrentaram ou estão enfrentando. Os presentes mantêm o foco dessa interação nas interdependências, ou seja, no que o trabalho de um time afetou ou afetará o trabalho de outros times. Os times envolvidos devem decidir em conjunto qual a frequência e duração das reuniões, em geral diárias com quinze minutos de duração ou semanais, com uma duração mais longa. O representante de cada time que comparece à reunião de SofS deve ser escolhido por seu respectivo time e é sempre um de seus membros, não o ScrumMaster ou o Product Owner.

Já o contexto em que múltiplos times trabalham em múltiplos projetos é um pouco diferente. Esse cenário, apesar de ser bem comum em empresas, não apresenta uma nomenclatura formal na comunidade Ágil. Por essa razão criamos e adotamos o termo Scrum and Scrum (SandS) para a descrição do mesmo (Souza et al., 2012).

Tanto SofS como SandS demandam mecanismos específicos para controle de prioridade e dependência (ou independência) dos times. No SofS é comum encontrar uma hierarquia de Product Owners responsável pela manutenção do Product Backlog, que normalmente é compartilhado pelos times. No SandS, times trabalham em

projetos independentes e não há coordenação alguma entre os seus Product Owners. Porém, por se tratarem de times que trabalham na mesma empresa ou gerência, sempre há dependências e impedimentos comuns que devem ser trabalhados. Assim, fazem-se necessários o entendimento, a classificação e a visualização dessas dependências e impedimentos, de forma que possam ser tratados em conjunto pelos times envolvidos.

Para saber mais sobre como escalar Scrum, acesse o *site* do livro em <http://livrodescrum.com.br>.

24.8 TESTES AUTOMATIZADOS E ENTREGA CONTÍNUA

Uma evolução que devemos sempre perseguir reside na qualidade. Devido à sua grande complexidade, garantir a qualidade de *software* só é possível por meio de testes. A automação desses testes exponencia o benefício de um *software* testado, pois em apenas alguns minutos de centenas a milhares de testes podem ser rodados. Isso permite que sejam rodados frequentemente num processo que chamamos de Integração Contínua. Para sistemas de *software*, existem várias camadas de testes: testes unitários, testes integrados, testes funcionais, testes de interface e testes de aceitação, por exemplo. Há também uma série de técnicas e práticas também nessa área, incluindo TDD (*Test Driven Development*) e ATDD (*Acceptance Test Driven Development*).

O desenvolvimento de *software* tem evoluído também na busca da redução do tempo de *feedback*. Os produtos desenvolvidos hoje em dia passaram a focar nessa questão, por exemplo, evoluindo das entregas anuais (*waterfall*) para entregas trimestrais (RUP), entregas mensais/semanais (Scrum/XP), até as entregas contínuas (Kanban/Lean Startup).

O valor de reduzir o tempo de *feedback* na construção de um produto é inegável:

- antecipação do valor;
- correção de desvios;
- adaptação a mudanças.

Quando se trata do trabalho do time, uma série de evoluções também ocorreram. Podemos classificar essas evoluções em níveis de maturidade e automação:

Nível 1 – Integração Contínua

- **objetivo:** responder o mais rápido possível à pergunta “estamos construindo da maneira correta?”;
- **nível de automação:** simples;
- **como alcançar:** montar um ambiente de Integração Contínua que valide que o *software* foi construído da maneira correta, propiciando segurança para o time. Esse ambiente utiliza-se de práticas como testes automatizados, além de práticas de codificação e versionamento de código.

Nível 2 – Implantação Contínua

- **objetivo:** responder o mais rápido possível à pergunta “estamos construindo o *software* de acordo com o que o cliente definiu?”. O *software* pode parecer estar correto segundo a Integração Contínua, mas não necessariamente está de acordo com o que o cliente (ou Product Owner) definiu;
- **nível de automação:** médio;
- **como alcançar:** automação do processo de implantação (*deploy*) e gerenciamento de dependências, práticas de *review* contínuo e montagem de ambiente intermediário que simule o ambiente de produção para que se possa validar a nova versão antes mesmo da liberação para produção.

Nível 3 – Entrega Contínua

- **objetivo:** responder o mais rápido possível à pergunta “estamos construindo o *software* que o usuário precisa?”. O *software* pode estar ok segundo a Integração Contínua e ter recebido o aceite do Product Owner ou cliente. Mas o que realmente agrega valor é entregar o *software* para o usuário final. É fundamental diminuir o tempo que levamos para entregar para os reais consumidores;
- **nível de automação:** extremo;
- **como alcançar:** automação do ciclo de Release, práticas DevOps, definição de métricas, montagem de ambiente de produção automatizado e preparado para escalabilidade e resolução automatizada de problemas.

Para evoluir e atingir a maturidade da Entrega Contínua, há uma sequência de conhecimentos acumulados: testes automatizados, TDD, versionamento, integração contínua, definição de ambientes de testes e homologação. O perfil técnico deve buscar continuamente absorver esses conhecimentos.

Para saber mais sobre testes automatizados e entrega contínua, acesse o *site* do livro em <http://livrodescrum.com.br>.

24.9 CONCLUSÃO

De acordo com o *shu-ha-ri*, conceito provindo das artes marciais japonesas, existem três níveis ou etapas para o completo desenvolvimento de uma habilidade, conforme descritos a seguir (veja a figura 24.2):

- ***shu***: nessa fase, o estudante é um aprendiz, e seu objetivo é compreender os fundamentos para futuramente dominá-los, sem promover modificações ou customizações, praticando o que aprendeu com alto grau de fidelidade;
- ***ha***: uma vez que dominou os fundamentos e adquiriu experiência, o aprendiz possui maturidade suficiente para compreender o que está por trás das técnicas que aprendeu, e a partir daí, pode promover melhorias. No uso do Scrum, este é o momento em que o profissional passa a adicionar técnicas e práticas avançadas ao *framework*, de forma a buscar ferramentas que lhe permitam ser cada vez mais eficiente no seu papel;
- ***ri***: o terceiro e último estágio é conhecido como o estágio da transcendência ou libertação. Nessa fase, o profissional é como um faixa preta de judô, que domina as técnicas que utiliza e seus fundamentos, de modo que consegue agir naturalmente ao colocar em prática o que aprendeu. A partir do domínio dos fundamentos, o profissional passa a produzir conhecimento original, contribuindo para a comunidade com novas técnicas, práticas e teorias.



Figura 24.2: *Shu ha ri*: etapas para o completo desenvolvimento de uma habilidade

Ensinar Scrum e ajudar o leitor a praticá-lo bem é o principal objetivo deste livro. No entanto, um livro pode apenas levar o leitor ao *shu* e, no máximo, indicar caminhos possíveis que o ajudem a seguir para o *ha*. As habilidades e técnicas mostradas neste capítulo complementam esses caminhos já mostrados ao longo do livro, sugerindo alguns dos passos seguintes que um verdadeiro praticante de Scrum pode seguir para obter sucesso em seus projetos a partir do uso de práticas Ágeis.

Assim, para chegar ao *ha*, pratique o Scrum. Experimente novas técnicas. Participe de treinamentos. E, se esbarrar em obstáculos muito altos, busque a ajuda de um Agile Coach profissional. Não tenha medo de errar. Ao contrário, aprenda com seus erros. Persiga a melhoria contínua. E, afinal, chegue ao *ri*, onde usar Scrum ou outra técnica reconhecida não mais terá importância.

CAPÍTULO 25

Apêndice - Glossário

Em ordem alfabética.

Clientes do projeto: pessoas, grupos ou organizações que solicitam o projeto ou apenas o patrocinam e recebem o retorno ao investimento uma vez que o que é produzido lhes é entregue.

Critérios de Aceitação: critérios que documentam os detalhes da User Story, expressos por enunciados pequenos e de fácil entendimento, que são utilizados para determinar quando a funcionalidade produzida pelo Time de Desenvolvimento está completa e, assim, nada mais deve ser adicionado a ela.

Daily Scrum: reunião curta, realizada diariamente pelo Time de Desenvolvimento, com o propósito de planejar o próximo dia de trabalho, proporcionando visibilidade ao trabalho realizado e a realizar, promovendo a comunicação sobre esse trabalho, dando visibilidade a quais obstáculos atrapalharam o desenvolvimento e servindo de oportunidade para decisões rápidas com relação ao progresso do Sprint.

Definição de Preparado: acordo formal entre Product Owner e Time de Desenvolvimento sobre o estado em que um item do Product Backlog deve estar para

estar qualificado para discussão na reunião de Sprint Planning, que visa garantir que o item chegue à reunião preparado segundo um critério bem definido.

Definição de Pronto: acordo formal entre Product Owner e Time de Desenvolvimento sobre o que é necessário para se considerar que um item ou o Incremento do Produto produzido no Sprint está “pronto”.

Gráfico de Release Burndown: gráfico mantido pelo Product Owner e utilizado tanto pelo Product Owner quanto pelo Time de Desenvolvimento para monitorar o progresso no desenvolvimento em direção a uma entrega (Release), através da “queima” do trabalho previsto para a Release.

Gráfico de Release Burnup: gráfico mantido pelo Product Owner e utilizado tanto pelo Product Owner quanto pelo Time de Desenvolvimento para monitorar o progresso no desenvolvimento em direção a uma entrega (Release), através do acúmulo de trabalho realizado em direção ao trabalho previsto para a Release.

Gráfico de Sprint Burndown: gráfico mantido e utilizado pelo Time de Desenvolvimento para monitorar seu progresso no desenvolvimento em direção ao final de um Sprint, através da “queima” do trabalho previsto para o Sprint.

Impedimento: obstáculo ou barreira que dificulta significativamente ou impede que o trabalho do Time de Desenvolvimento seja realizado, bloqueando um ou mais itens do Sprint Backlog de forma a ameaçar a Meta do Sprint.

Incremento do Produto: incremento de funcionalidades do produto prontas, de acordo com a Definição de Pronto, produzida pelo Time de Desenvolvimento em cada Sprint a partir dos itens do Product Backlog.

Meta de Release: objetivo ou necessidade de negócios de alto nível a ser alcançada por meio do trabalho do Time de Desenvolvimento para uma Release.

Meta de Roadmap: objetivo ou necessidade de negócios de alto nível a ser alcançada por meio do trabalho do Time de Desenvolvimento até uma determinada data ou marco no Roadmap do Produto.

Meta de Sprint: objetivo ou necessidade de negócios bem definida, obrigatoriamente estabelecida e acordada entre Product Owner e Time de Desenvolvimento durante a reunião de Sprint Planning, que deve ser alcançada pelo Time de Desenvolvimento em seu trabalho no Sprint.

Plano da Release: plano que indica qual o objetivo a ser alcançado pela Release e quando será alcançado, contendo assim a Meta da Release, a data exata ou aproximada em que a entrega será realizada e um conjunto de itens selecionados do Product Backlog para a Release.

Product Backlog: lista ordenada, planejável, emergente e gradualmente detalhada, criada e mantida pelo Product Owner, que evolui ao longo de todo o projeto e contém o que se acredita que será desenvolvido pelo Time de Desenvolvimento para se alcançar a Visão do Produto, como necessidades ou objetivos de negócios dos clientes, melhorias a serem realizadas no produto, correções de problemas, questões técnicas, pesquisas que forem necessárias etc.

Product Owner: pessoa responsável por definir o produto a ser desenvolvido, de forma a garantir e maximizar, a partir do trabalho do Time de Desenvolvimento, o retorno sobre o investimento do projeto.

Quadro de Tarefas: quadro frequentemente utilizado para representar o Sprint Backlog, dividido em colunas que contêm os itens selecionados para o Sprint e as suas tarefas correspondentes. As tarefas são distribuídas entre as colunas “A Fazer”, “Fazendo” e “Feito” (ou termos similares), de acordo com seu andamento no Sprint. Os itens e as tarefas são muitas vezes escritos em notas adesivas. Alguns softwares simulam o Quadro de Tarefas, representando-o virtualmente.

Refinamento do Product Backlog: trabalho de criação, detalhamento e modificação de itens do Product Backlog realizado pelo Product Owner de forma contínua ao longo de todo o projeto e também realizado em sessões ao longo do Sprint em conjunto pelo Product Owner e Time de Desenvolvimento, com o propósito de preparar itens para o Sprint seguinte.

Release: entrega de um ou mais Incrementos do Produto prontos gerados pelo Time de Desenvolvimento em um ou mais Sprints sucessivos, que em conjunto possuem valor suficiente para serem utilizados.

Release Planning: reunião na qual se planeja a próxima Release, a partir da criação do Plano da Release, que contém a Meta da Release, uma data e um conjunto de itens selecionados do Product Backlog para a Release.

Roadmap do Produto: plano em alto nível de como o produto irá evoluir ao longo do tempo até um momento futuro determinado, expresso por uma linha do tempo com marcos, que são datas no futuro e objetivos do produto a serem alcançados.

ScrumMaster: pessoa que facilita e potencializa o trabalho do Time de Scrum, e nesse trabalho ensina Scrum para o Time de Scrum, remove impedimentos que previnem o Time de Scrum de realizar seu trabalho e promove as mudanças organizacionais necessárias.

Sprint: ciclo de desenvolvimento de duração fixa, em que o Incremento do Produto pronto é gerado pelo Time de Desenvolvimento a partir dos itens mais impor-

tantes do Product Backlog, e que vai desde a reunião de planejamento até as reuniões de encerramento.

Sprint Backlog: lista de itens selecionados na reunião de Sprint Planning do alto do Product Backlog para o desenvolvimento do Incremento do Produto no Sprint (o quê), adicionada de um plano de como esse trabalho será realizado (como), geralmente expresso por um conjunto de tarefas correspondente a cada item.

Sprint Planning: reunião realizada no primeiro momento do primeiro dia do Sprint, na qual Product Owner e Time de Desenvolvimento planejam o que será desenvolvido no Sprint corrente e Time de Desenvolvimento planeja como o que foi selecionado será desenvolvido.

Sprint Retrospective: última reunião realizada no Sprint, em que o Time de Scrum inspeciona o Sprint que está se encerrando quanto a seus processos de trabalho, dinâmicas, comportamentos e ambiente, e planeja as melhorias necessárias a serem executadas no próximo Sprint.

Sprint Review: reunião realizada no último dia do Sprint em que o Time de Scrum demonstra o Incremento do Produto gerado no Sprint para obter *feedback* dos clientes e demais partes interessadas e modificar o que será produzido em seguida de acordo.

Partes interessadas do projeto: pessoas, grupos ou organizações que contribuem de alguma forma para o projeto ou simplesmente são informados do seu progresso. Além de clientes e usuários, esse grupo pode incluir patrocinadores do projeto, executivos e gerentes da organização etc.

Story Point: unidade relativa criada pelo Time de Desenvolvimento para estimar o tempo necessário para desenvolver um item de trabalho.

Timebox: é uma alocação máxima ou fixa de tempo dentro da qual uma atividade deve ocorrer. Uma vez alcançado o tempo máximo definido, a atividade deve ser terminada imediatamente.

Time de Desenvolvimento: grupo multidisciplinar e auto-organizado de pessoas, responsável por realizar o trabalho de desenvolvimento do produto propriamente dito.

Time de Scrum: time formado pelo Time de Desenvolvimento, Product Owner e ScrumMaster.

User Story: descrição concisa, simples e leve de uma necessidade do usuário do produto sob o ponto de vista desse usuário, que funciona como um convite para uma conversa entre as pessoas de negócios e desenvolvedores para definirem os detalhes do que deve ser desenvolvido.

Usuário do produto: pessoa que recebe e utiliza o produto incrementalmente gerado no decorrer do projeto.

Velocidade do Time de Desenvolvimento: é a média da quantidade de trabalho produzido pelo do Time de Desenvolvimento nos últimos Sprints, medida pela taxa média de queima de Story Points por Sprint ou de itens por Sprint.

Visão do Produto: objetivo ou necessidade de negócios de alto nível que fornece contexto, orientação, motivação e inspiração para todo o trabalho de desenvolvimento do produto, alinhando o entendimento todos os envolvidos no projeto sobre o que deve ser alcançado.

CAPÍTULO 26

Apêndice - Bibliografia

Em ordem alfabética.

ARMONY, R. S. 196 f. Fatores críticos para a prática de valores Ágeis em equipes de tecnologia da informação. Dissertação (Mestrado em Administração de Empresas) – Instituto de Administração e Gerência, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, RJ, 2010.

COCKBURN, A. Agile software development: the cooperative game. 2. ed. Reading, MA, Estados Unidos: Addison-Wesley, 2007.

COHN, M. User Stories applied: for Agile software development. Reading, MA, Estados Unidos: Addison-Wesley, 2004.

COHN, M. Agile estimating and planning. Englewood Cliffs, NJ, Estados Unidos: Prentice Hall PTR, 2005.

CUMMINGS, T. G. Self-regulating work groups: A socio-technical synthesis. The Academy of Management Review, v. 3, n. 3, p. 625-634, jul. 1978.

DEGRACE, P.; STAHL L. H., Wicked problems, righteous solutions: a Catalogue of Modern Software Engineering Paradigms. Upper Saddle River, NJ, Estados

Unidos: Yourdon Press, 1990.

DERBY, E.; LARSEN, D. Agile retrospectives: making good teams great. Raleigh, North Carolina, and Dallas, TX, Estados Unidos: Pragmatic Bookshelf, 2006.

DRUSKAT, V. U.; WHEELER, J. V. Managing from the boundary: the effective leadership of self-managing work teams. *Academy of Management Journal*, Mississippi State, MS, Estados Unidos, v. 46, n. 4, p. 435-457, 2003.

GREENING, J. Planning Poker or how to avoid analysis paralysis while release planning. 2002. Disponível em <<http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf>> . Acesso em 15 dez. 2012.

GROSS, J. M.; MCINNIS, K. R. Kanban made simple: Demystifying and Applying Toyota's Legendary Manufacturing Process. Nova Iorque, NY, Estados Unidos: AMACOM, 2003.

HACKMAN, J. R.; OLDHAM, G.; JANSON, R.; PURDY, K. A new strategy for job enrichment. *California Management Review*, Berkeley, CA, Estados Unidos, v. 17, n. 4, p. 55-71, 1975.

HIGHSMITH, J. Agile software development ecosystems. Boston, MA, Estados Unidos: Addison-Wesley, 2002.

HIGHSMITH, J. Agile project management: Creating Innovative Products. Reading, MA, Estados Unidos: Addison-Wesley, 2004.

HOLLAND, J. H. Studying complex adaptive systems. *Journal of Systems Science and Complexity*, v. 19, p. 1-8, 2006.

JANIS, I. L. Groupthink: psychological studies of policy decisions and fiascoes. 2. ed. Boston, MA, Estados Unidos: Houghton Mifflin, 1982.

JEFFRIES, R., ANDERSON, A., HENDRICKSON, C. Extreme Programming installed. Reading, MA, Estados Unidos: Addison-Wesley, 2000.

KERTH, N. Project retrospectives: a handbook for team reviews. NY, Estados Unidos: Dorset House, 2001.

LARMAN, C. Agile and iterative development: a manager's guide. Reading, MA, Estados Unidos: Addison-Wesley, 2003.

LIKER, J. K. The Toyota way: 14 management principles from the world's greatest manufacturer. McGraw-Hill Professional Publishing: Blacklist, OH, Estados Unidos, 2003. Arquivo CHM.

LOCKE, E. A. Motivation through conscious goal setting. *Applied & Preventive Psychology*, Amsterdã, Holanda, v. 5, p. 117-124, 1996.

LOCKE, E. A.; LATHAM, G. P. New directions in goal-setting theory. Current Directions in Psychological Science, Nova Iorque, NY, Estados Unidos, v. 15, n. 5, p. 265-268, out. 2006.

MANZ, C. C.; NECK, C. P. Teamthink: beyond the groupthink syndrome in self-managing work teams. Team Performance Management, v. 3, n. 1, p. 18-31, 1997.

MANZ, C. C.; SIMS, H. P., Jr. Leading workers to lead themselves: the external leadership of self- managing work teams. Administrative Science Quarterly, v. 32, n. 1, p. 106-129, mar. 1987.

MOORE, G. A. Crossing the chasm: marketing and selling high-tech products to mainstream customers. Nova Iorque, NY, Estados Unidos: PerfectBound, 2001. Arquivo PDF.

MOTTA, P. R. Gestão contemporânea: a ciência e a arte de ser dirigente. Rio de Janeiro: Record, 1991.

OGUNNAIKE, B. A.; RAY, W. H. Process dynamics, modeling and control. NY, Estados Unidos: Oxford University Press, 1994.

OSONO, E., SHIMIZU, N., TAKEUCHI, H. Relatório Toyota: contradições responsáveis pelo sucesso da maior montadora do mundo. Tradução de Carlos Szlak. 1. ed. São Paulo: Ediouro, 2008.

PICHLER, R. Agile product management with Scrum: creating products that customers love. Boston, MA, Estados Unidos: Addison-Wesley, 2010.

PIMENTEL, M. Tenha nojo dos impedimentos. InfoQ Brasil, dez. 2009. Disponível em <<http://www.infoq.com.br/articles/tenha-nojo-impedimentos>> . Acesso em 17 set. 2011.

ROYCE, W. Managing the Development of Large Software Systems: Concepts and Techniques. In: Proceedings of IEEE WESCON. Piscataway, NJ, Estados Unidos: IEEE Press, ago. 1970, p. 1-9.

SCHWABER, K. Agile project management with Scrum. Redmond, WA, Estados Unidos: Microsoft Press, 2004.

SCHWABER, K.; BEEDLE, M. Agile software development with Scrum. Upper Saddle River, NJ, Estados Unidos: Prentice Hall, 2002.

SCHWARZ, R. The skilled facilitator: a comprehensive resource for consultants, facilitators, managers, trainers and coaches. 2 ed. San Francisco, CA, Estados Unidos: Jossey-Bass, 2002. Kindle Edition.

SCRUM ALLIANCE: Core Scrum. Agile Atlas, dez. 2012. Disponível em <<http://agileatlas.org/atlas/scrum>> . Acesso em 25 abr. 2013.

SNOWDEN, D. J.; BOONE, M. E. A leader's *framework* for decision making. Harvard Business Review, Boston, MA, Estados Unidos, v. 85, n. 11, p. 68-76, nov. 2007.

SOUZA, D.; DE TOLEDO, R.; OLIVEIRA, J. Estudo sobre dependências e suas consequências no cenário Scrum and Scrum (SandS). 3rd Brazilian Workshop on Agile Methods (WBMA'2012), set. 2012. Disponível em <<http://www.ime.usp.br/\char126kon/wbma2012.pdf>> . Acesso em 25 abr. 2013.

STACEY, R. Complexity and creativity in organizations. San Francisco, CA, Estados Unidos: Berrett-Koehler, 1996.

SUTHERLAND, J. Agile development: lessons learned from the first Scrum. Cutter Agile Project Management Advisory Service: Executive Update, v. 5, n. 20, p. 1-4., 2004.

TAKEUCHI, H.; NONAKA, I. The new new product development game. Harvard Business Review, Boston, MA, Estados Unidos, v. 64, n. 1, p. 137-146, jan./fev. 1986.

THE STANDISH GROUP. The CHAOS Report. West Yarmouth, MA: The Standish Group International Inc. 2002.

DE TOLEDO, R. Por que usar “story points”? Blog Visão Ágil, jan. 2009. Disponível em <<http://visaoagil.files.wordpress.com/2009/01/storypoints.pdf>> . Acesso em 17 set. 2011.

VERSIONONE. State of Agile survey – “the state of Agile development” – 6th annual”. 2011. Disponível em <http://www.versionone.com/state_of_agile_development_survey/11/> . Acesso em 08 jan. 2013.

WAKE, W. INVEST in good stories, and SMART tasks. XP123, aug. 2003. Disponível em <<http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>> . Acesso em 14 dez. 2012.

WILKINSON, M. The Secrets of Facilitation: The S.M.A.R.T. Guide to Getting Results With Groups. San Francisco, CA, Estados Unidos: Jossey-Bass, 2004. Kindle Edition.

WOMACK, J. P.; JONES, D. T. A mentalidade enxuta nas empresas: elimine o desperdício e crie riqueza. Tradução de Ana Beatriz Rodrigues e Priscilla Martins Celeste. 4. ed. Rio de Janeiro: Campus, 1998.

WOMACK, J. P.; JONES, D. T.; ROOS, D. A máquina que mudou o mundo. Tradução de Ivo Korytowski. 17. ed. Rio de Janeiro: Campus, 1992.



Casa do Código
Livros para o programador

**Uma editora de livros técnicos
feita por desenvolvedores
para desenvolvedores.**



Inscreva-se em nossa newsletter e
receba novidades e lançamentos

www.casadocodigo.com.br/newsletter



Curta nossa fanpage no Facebook

www.facebook.com/casadocodigo

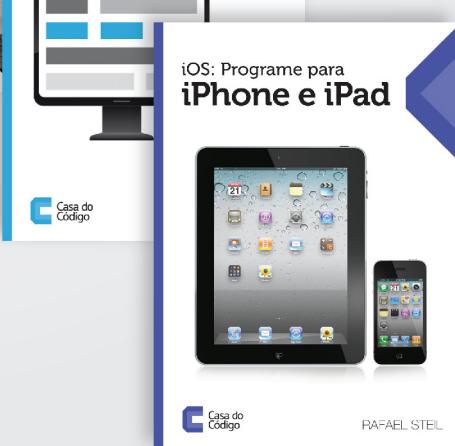
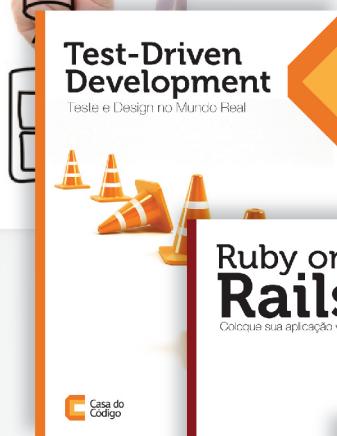


Caelum:
Cursos de TI presenciais e online

www.caelum.com.br



Dê seu feedback sobre o livro. Escreva para contato@casadocodigo.com.br



Já conhece
os nossos
títulos?

E muito mais em:
www.casadocodigo.com.br



Casa do Código
Livros para o programador

RAFAEL STEL