ADAPTWORKS TREINAMENTOS

Scrum – Guia de Referência

Al. Lorena, 638 • 4º andar
São Paulo, SP

Conteúdo

Capítulo

1

Definições e Origem

Fundamentos de Scrum

A definição de Scrum é: um framework com o qual as pessoas podem resolver problemas complexos e adaptáveis, enquanto entregam produtos de forma produtiva e criativa e com o maior valor possível. Scrum é:

- Leve
- Simples de entender
- Difícil de aplicar

Scrum consiste do Time Scrum e seus papéis, eventos e artefatos que coordenam o relacionamento e interação entre os membros.

Apesar de ter sido criado originalmente para desenvolvimento de software, Scrum pode ser usado em qualquer tipo de projeto, contanto que haja um certo grau de complexidade, já que ele se baseia na teoria de controle empírico de processos, ou empirismo.

Um processo empírico é aquele onde diversos imprevistos ocorrem – diferente de um processo definido – e tomamos alguma atitude para resolve-los. Desta forma, você melhora o processo à medida que trabalha com ele, já que as coisas não ocorrem sempre da mesma forma.

Diferente de uma linha de montagem, onde cada passo pode ser previsto e monitorado, em projetos de desenvolvimento complexo de softwares isso se torna impossível, e daí a razão de se utilizar um processo empírico.

Valores e Princípios

Em 2001, um grupo de profissionais e pensadores se reuniu para conversar e comentar sobre metodologias e práticas que vinham utilizando no gerenciamento de projetos de software. Entre eles estavam Jeff Sutherland e Ken Schwaber, os criadores do Scrum.

Este grupo, de comum acordo, criou o "The Agile Manifesto". Em cima disso, criaram 12 princípios que norteiam todos os métodos ágeis. Ainda hoje, estes princípios servem como parâmetro para testar novos métodos ágeis e "agilistas" (praticantes de métodos ágeis). E são sobre estes valores que Scrum se apoia.

Manifesto Ágil

Em uma tradução livre, o Manifesto Ágil consiste no seguinte:

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

Princípios Ágeis

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Valores de Scrum

Além de seguir o Manifesto Ágil e seus Princípios, Scrum possui cinco valores que são bastante voltados para a parte humana de Scrum. São eles:

- Foco
- Coragem
- Sinceridade
- Comprometimento
- Respeito

Os Pilares de Scrum

Scrum baseia-se em três pilares: transparência, inspeção e adaptação.

Transparência

Por se basear no empirismo, transparência é fundamental em Scrum. Por isso, todo o processo deve estar visível a todos os envolvidos na criação do produto.

Para que esta transparência seja efetiva, é necessário que todos os envolvidos estejam em sintonia quanto aos aspectos. Por exemplo, nomenclaturas e jargões devem ser compartilhados por todos; a DoD (Definition of Done, ou Definição de Pronto) deve estar definida e ser compartilhada para que as expectativas possam ser corretamente satisfeitas.

Inspeção

Como se trata de um controle empírico de processos, o processo em si deve ser inspecionado regularmente para detectar eventuais problemas. Veremos mais adiante em quais pontos e momentos as inspeções são feitas.

Adaptação

Caso a inspeção detecte algum problema no processo, ou alguma forma de melhorá-lo, adaptações devem ser feitas a ele. Estas adaptações devem ser feitas o mais rápido possível para garantir a produtividade do time e a qualidade do produto. Além disso, Scrum prevê algumas oportunidades para estas adaptações:

- Sprint Planning Meeting
- Daily Scrum
- Sprint Review
- Sprint Retrospective

É claro que para isso é necessária a transparência entre todos os que estejam comprometidos com o projeto, pois as adaptações devem ser feitas em quaisquer níveis hierárquicos em que se faça necessário.

Através da correta inspeção e adaptação, utilizamos o princípio de "Kaizen", ou melhoria contínua. Com isto, o processo é gradativamente aprimorado, sempre em busca de torna-lo o mais perfeito possível.

Capítulo

2

Responsabilidades e Papéis

Scrum possui poucos papéis, mas muito bem definidos. Além disso, as responsabilidades dentro do projeto são distribuidas de forma diferente do que em metodologias tradicionais.

Time Scrum

Os papéis que compõem um time Scrum são poucos, mas muito bem definidos. Entretanto, é importante entender que são papéis, e não cargos. Isso pode causar um pouco de confusão em empresas que tentam fazer o mapeamento de cargos existentes aos papéis de Scrum.

Times Scrum são auto-organizados e multifuncionais. Por serem auto-organizados, eles decidem qual a melhor forma de realizar seu trabalho, ao invés de serem direcionados por pessoas de fora do time. Times multifuncionais possuem capacidades que os tornam autossuficientes – que não dependem de pessoas de fora do time para completar seu trabalho. Este modelo foi desenhado para otimizar a flexibilidade, criatividade e produtividade.

Product Owner

O papel do Product Owner requer que a pessoa seja responsável por gerenciar o Product Backlog, garantir o ROI – Return on Investiment ou Retorno Sobre o Investimento – definir a visão do produto, gerenciar a entrada de novos requisitos e definir sua ordem, gerenciar o plano de releases e aceitar ou rejeitar o que for entregue ao final de cada iteração.

O Product Owner é o responsável por gerenciar o produto de forma a assegurar o valor do trabalho executado pelo Development Team. Assim, é de sua alçada transmitir ao time a visão do produto, bem como determinar o significado de cada um dos itens do backlog e as metas. E o sucesso do produto, por parte do Product Owner, está em sua capacidade de compreender as necessidades do negócio e do mercado, de forma que o Product Backlog reflita a importância de seus itens e também na efetividade em transmitir essas informações ao restante do Time Scrum.

Uma empresa pode ter um comitê para catalogar novas funcionalidades para o produto. Entretanto, como regra, deve existir somente um Product Owner para definir se uma nova funcionalidade será adicionada ao Product Backlog e quando será feita, de acordo com sua ordenação.

Somente o Product Owner possui autoridade para cancelar uma Sprint. Não é o tipo de coisa que deva acontecer com frequência, pois pode afetar negativamente o restante do Time Scrum. Mas, por exemplo, isso pode ser feito caso o Time Scrum chegue à conclusão de que a meta da iteração não faça mais sentido, devido a alguma mudança nas condições do mercado.

Development Team

Em Scrum, todo o desenvolvimento é feito pelo Development Team. Para isto, é necessário que ele seja composto de pessoas de diferentes perfis profissionais – arquitetos, analistas, designers, desenvolvedores, testadores, etc. Em suma, um Development Team deve ser autossuficiente para alcanças as metas das Sprints. Um time composto desta forma consegue ter diferentes opiniões e pontos de vista, além da experiência diversificada, o que permite uma maior criatividade ao transformar um item de negócio em um incremento do produto.

Um Development Team é responsável por desenvolver incrementos potencialmente entregáveis do produto segundo a Definition of Done ao final de cada iteração. Também são responsáveis pela estimativa quanto ao tamanho dos itens do Product Backlog e por acordarem com a meta da iteração.

Um Development Team deve ser auto-organizado e autogerenciado e ninguém (nem o ScrumMaster) diz a eles como transformar o Product Backlog em incrementos de funcionalidades potencialmente entregáveis. A auto-organização se deve ao fato de que a interação entre os membros do time apresentarem comportamentos imprevisíveis. Desta forma, a adaptação entre os membros acaba sendo fruto do convívio, e não algo determinado por pessoas estranhas ao time.

Ser autogerenciado significa que o Development Team possui autoridade sobre o trabalho que fazem, assim como também são responsáveis por ele. Por estarem trabalhando diariamente com as tarefas da iteração, o Development Team é responsável pelo microgerenciamento – andamento das tarefas, qualidade, prazo, etc. Além disso, o Development Team, como um todo, é responsável pelo sucesso ou fracasso da iteração – ou seja, independe se os membros, individualmente, tenham terminado as tarefas das quais eram responsáveis. O resultado é sempre do time inteiro.

Os Development Teams devem ser pequenos o suficiente de forma a se manter ágil e produtivo, e grande o suficiente de forma que a coordenação dos membros não cause problemas. Com menos de três pessoas, o time passa a ter menos interação e pode encontrar problemas em relação ao conhecimento necessário durante a execução da Sprint. Com mais de nove pessoas, há muita complexidade para o gerenciamento através de um processo empírico.

Um time auto-organizado geralmente passa por quatro estágios: a) **forming**, que é o início da formação do time, onde ninguém ainda se conhece; b) **storming**, que é quando os membros do time passam a se conhecer e ainda não estão habituados ao comportamento uns dos outros;

c) **norming**, que é quando os membros começam a aparar as arestas, criando algumas normas ou regras implícitas; e d) **performing**, que é quando o time passa a funcionar de forma sincronizada, melhorando bastante o desempenho do time como um todo. É importante entender que para um time chegar até o quarto estágio é preciso tempo e paciência, e isto requer bastante trabalho do ScrumMaster (a seguir) atuando como um facilitador.

ScrumMaster

Em diversos dicionários, ao procurar a palavra "mestre" (master), você encontra as seguintes definições:

- Pessoa que ensina ou orienta, professor, orientador
- Pessoa que sabe muito, sábio

O ScrumMaster é a pessoa que mais conhece Scrum dentre todos os papéis. E seu papel não é fácil: como o maior conhecedor de Scrum, ele é o responsável por orientar o Product Owner na criação e ordenação do Product Backlog; é o responsável por garantir que as regras de Scrum estejam sendo cumpridas e seus valores seguidos; é o responsável por ser o facilitador nos eventos de Scrum; também ajuda a remover impedimentos que o time enfrente. E isso sem fazer uso de qualquer autoridade.

A relação entre o ScrumMaster e os outros membros do Time Scrum é o de um líder na questão do processo – embora seja fundamental para as práticas de Scrum do time, ele não exerce papel ativo no desenvolvimento do processo de engenharia do time. Ao invés disso, ele utiliza técnicas de facilitação e coaching para que os membros do time consigam visualizar os problemas e encontrem a melhor solução. Ele atua de forma a fazer com que o time se desafie constantemente. Além disso, ele trabalha com os membros do time para que cada um consiga ser o mais eficiente possível, sempre de acordo com os valores e princípios de Agile e de Scrum.

Durante os eventos, o ScrumMaster tem a responsabilidade de fazer com que a reunião flua de forma adequada, utilizando técnicas de facilitação, embora não seja o responsável por conduzi-las. Como os eventos têm duração fixa, é de extrema importância que situações onde hajam divergências sejam rapidamente resolvidas.

Para o restante da organização, o ScrumMaster servirá como interface e especialista em Scrum. No início da adoção de Scrum, o ScrumMaster se comportará como agente de mudanças para que o restante da organização ajude a suportar Scrum. Por isso, para o restante da empresa, o

ScrumMaster servirá como referência de Scrum para o planejamento de uma eventual adoção em outros times.

Assim, é importante notar que o papel de ScrumMaster é fundamental na adoção de Scrum. E a pessoa a assumir isto deve estar ciente de que seu papel, embora importante, poderá não receber o mesmo destaque de outros papéis – uma típica relação de líder/servidor.

Capítulo

3

Eventos e Artefatos

Eventos de Scrum

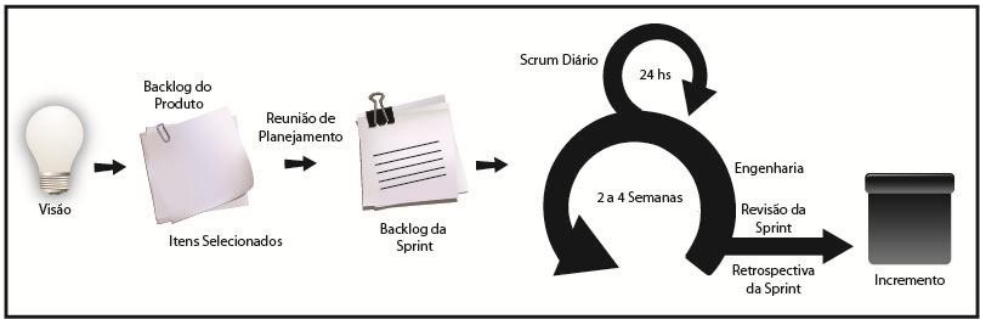

Figura 1 – Fluxo de Scrum

Scrum possui alguns eventos de duração fixa (time-boxed) realizados em intervalos regulares. Cada um destes eventos são oportunidades para inspeção e adaptação.

Sprint

Todo o desenvolvimento em Scrum é feito de forma iterativa e incremental – ciclos completos de desenvolvimento de duração fixa onde ao final temos incrementos potencialmente entregáveis do produto. Estas iterações são chamadas de Sprints.

Cada Sprint tem duração de até um mês, o que permite feedbacks constantes do Product Owner quanto ao produto sendo desenvolvido. Da mesma forma, o Product Owner tem a possibilidade de analisar o que foi produzido e reorganizar o Product Backlog caso seja necessário.

Uma Sprint consiste da Sprint Planning Meeting, Daily Scrum, o trabalho de desenvolvimento, a Sprint Review e a Sprint Retrospective. Ou seja, uma Sprint é como um container para todos os outro eventos.

Durante a execução da Sprint não devem ser feitas quaisquer alterações que afetem a Meta da Sprint. Além disso, o escopo pode ser renegociado entre o Development Team e o Product Owner conforme o conhecimento evolua durante o desenvolvimento, novas tecnologias sejam criadas, etc.

Sprint Planning Meeting

Ao início de cada uma das Sprints, o Time Scrum se reúne para planejar o que será feito naquela Sprint. Esta reunião chama-se Sprint Planning Meeting (Reunião de Planejamento da Sprint) e tem duração fixa (time-boxed) de até oito horas para uma Sprint de um mês. Para Sprints menores, o tempo é reduzido de forma proporcional.

A Sprint Planning Meeting é dividida em duas partes, cada uma com duração fixa correspondente à metade da duração total. Cada parte responde às seguintes perguntas:

- O que será entregue no Incremento resultante nesta Sprint?
- Como faremos para entregar o Incremento nesta Sprint?

Sprint Planning Meeting – 1ª Parte

Na primeira parte da Sprint Planning Meeting, o Development Team faz uma previsão das funcionalidades que serão desenvolvidas durante a Sprint. O Product Owner apresenta os itens do topo do Product Backlog ao Development Team e o Time Scrum inteiro colabora na definição dos itens. A quantidade de itens selecionados para a Sprint, entretanto, é prerrogativa do Development Team – somente o Development Team é capaz de avaliar o que consegue realizar durante a Sprint.

Para isso, o Development Team leva em consideração os Incrementos mais recentes, a capacidade projetada do Development Team para a Sprint e a performance de Sprints passadas.

Uma vez selecionados os itens, o Time Scrum define uma Meta da Sprint. Esta Meta da Sprint serve como um guia para o Development Team sobre o que estará sendo desenvolvido durante a Sprint.

Sprint Planning Meeting – 2ª Parte

Na segunda parte da Reunião de Planejamento, o Development Team decide como transformará os itens selecionados em um Incremento durante a Sprint.

O Development Team, sendo autogerenciado, se coordena para decompor os itens em unidades de um dia ou menos, o suficiente para os primeiros dias da Sprint.

Os itens selecionados mais o plano para o desenvolvimento do Incremento dá origem ao Sprint Backlog.

Com o Sprint Backlog criado, o Time Scrum define a Meta da Sprint.

Daily Scrum

Diariamente, o Development Team se reúne para uma reunião de no máximo 15 minutos, onde cada membro por vez responde para os outros membros:

- O que fiz desde a última Daily Scrum?
- O que pretendo fazer até a próxima Daily Scrum?
- Existe algo me impedindo de concluir alguma tarefa?

Este é o formato geral de uma Daily Scrum. Como o Development Team trabalha de forma colaborativa, a Daily Scrum permite que os membros comuniquem e sincronizem seu trabalho. E por serem autogerenciados, a reunião não é para fazer qualquer tipo de relatório, nem para o ScrumMaster, nem para o Product Owner, nem para ninguém – somente para eles mesmos.

Caso haja algum impedimento, o Development Team se auto-organiza para resolve-lo. No caso do impedimentos que o próprio Development Team não possa resolver, estes são passados ao ScrumMaster para que ele resolva.

Daily Scrums fazem parte do ciclo de inspeção e adaptação de Scrum. Através da análise diária do andamento da Sprint, o Development Team pode corrigir algum problema no processo imediatamente.

Sprint Review

Ao final da Sprint, temos uma outra cerimônia chamada Sprint Review. Nesta cerimônia podem participar quem quer que esteja interessado no produto.

Embora essa cerimônia também seja utilizada para demonstrar as novas funcionalidades feitas durante a Sprint, seu principal motivo é o de inspecionar o que o Development Team produziu e colher opiniões e impressões dos presentes para, caso seja necessário, adaptar o plano para a Sprint seguinte. Então, o foco desta cerimônia é aprimorar o produto.

Nesta cerimônia, o Product Owner valida ou não a Sprint, de acordo com a meta que tenha sido acordada com o Development Team durante a Sprint Planning Meeting. O Development Team discute sobre a Sprint, o que correu bem e os problemas enfrentados e as soluções encontradas e após a demonstração, respondem a quaisquer perguntas dos presentes.

Sprint Retrospective

O último evento de uma Sprint é a Sprint Retrospective e ocorre imediatamente após a Sprint Review. Participam desta reunião todos os membros do Time Scrum, e o foco é o aprimoramento do processo – a interação entre os membros do time, as práticas e ferramentas utilizadas, o que funcionou e o que precisa ser melhorado na próxima Sprint. Claro, identificar problemas não é suficiente. Assim, é importante também identificar medidas a serem tomadas para a melhoria do processo para as próximas Sprint.

Como o objetivo é a melhoria do processo, os membros do Time não devem encarar alguma crítica como algo pessoal.

Release Planning Meeting

Ao final de uma Sprint, o Development Team entrega um Incremento do produto, que é algo *potencialmente entregável*. Entretanto, apesar de poder entrar imediatamente em produção, existem situações em que isso é postergado para que seja feito somente na *Release* do produto.

Um Release é a representação de uma entrega de produto em produção. Ele é comumente planejado através de um cerimônia chamada Release Planning Meeting. Releases são planejados principalmente em ambientes onde estas entregas precisam ter uma data fixa, seja por obrigações contratuais, necessidades de mercado, eventos (conferências, *product lauch*, etc.) ou – principalmente em ambientes largos – para seguir um roadmap de releases corporativos.

Um Release Planning Meeting permite que o Time Scrum consiga enxergar além de uma Sprint, mesmo sem muita acuracidade, e assim planejar entregas em produção para os clientes. Mesmo assim, é importante ter em mente que o período de tempo entre as Releases deve ser o mais curto possível. Isso porque Releases curtos permitem feedbacks mais rápidos e frequentes, o que evita o desenvolvimento de funcionalidades desnecessárias ou com um comportamento diferente do esperado pelo cliente.

Todo o Time Scrum participa do Release Planning Meeting. Apesar de inicialmente dar a impressão de uma reunião mais voltada a negócios, isso permite que o time inteiro consiga enxergar o produto da mesma forma, canalizando os esforços na mesma direção. Através desta reunião, o time todo tem conhecimento e concordam com o valor de cada um dos itens para aquele Release, os riscos e as dependências. Assim, o Time Scrum consegue ter a visão do todo.

Apesar de não haver uma regra que determine a duração de um Release Planning Meeting, e nem a sua frequência, através de nossa experiência, a duração é em média de dois dias – estender a reunião por muito mais tempo não aumenta a acuracidade do planejamento, e nem é este o objetivo. Quanto a frequência, é recomendado que o período de um Release não ultrapasse seis meses. Entretanto, o time deve atualizar o planejamento de acordo com o resultado e estimativa das Sprints a cada Sprint Review. E caso haja a necessidade, replanejar os Releases antes do previsto.

Exemplo de uma Release Planning Meeting

1. **Abertura**: ScrumMaster revisita o propósito da reunião, sua estrutura, agenda, etc.

2. **Visão do Produto e Roadmap**: Product Owner revisa a Visão do Produto com o propósito de trazer todos novamente para o foco e alinhar expectativas.
3. **Status atual do projeto**: Product Owner apresenta gráficos e Status Reports onde possam ser visualizados resultado das últimas Sprints e momento atual do projeto.
4. **Meta/tema da Release**: Product Owner propõe a meta para o Release. Normalmente esta meta é expressa em objetivo de negócio alinhado à data de entrega.
5. **Estimativa de velocidade**: Time estima sua velocidade para esta Release baseando-se no resultado de Sprints anteriores.
6. **Agenda da Release e número de Sprints**: ScrumMaster facilita um trabalho colaborativo entre Development Team e Product Owner para enxergar Milestones, definir tamanho de Sprints, etc.
7. **Estimativa de itens do Product Backlog**: Development Team estima itens do Product Backlog caso estes não estejam estimados. Só deverão ser estimados uma quantidade de itens suficiente para planejar o Release e, talvez, para deixar uma "sobra".
8. **Mapear itens nas Sprints do Release**: seguindo a priorização do Product Owner, o Development Team irá mapear quais itens "cabem" em quais Sprints. Como o Release Planning será revisto ao fim de cada Sprint, não é aconselhado tentar encaixar item a item em cada Sprint. Se você está planejando, por exemplo, um Release de cinco Sprints, mapeie itens para as duas primeiras Sprints apenas, e deixe o restante selecionado sem posicionar na Sprint três, quatro ou cinco.
9. **Riscos, dependência e preocupações**: de forma colaborativa Development Team, ScrumMaster e Product Owner trabalham com práticas para identificar riscos, dependências, impedimentos arquiteturais, desafios, etc. Caso necessário será elaborado um plano de riscos, impediments backlog e/ou um simples plano de ação.
10. **Comprometimento**: ScrumMaster provoca Development Team e Product Owner para que haja um comprometimento com esta meta. É importante todos estarem confiantes e entusiasmados com a meta.
11. **Plano de comunicação e logística da Sprint**: principalmente em ambientes largos, a comunicação do Release (e consequentemente do projeto) perante a organização não pode falhar. Se necessário montar plano de ação para este trabalho.
12. **Montar gráfico(s)**: montar Release Burndown e/ou Parking Lot e/ou qualquer gráfico que ajude a comunicar o andamento do Release.

13. **Retrospectiva**: Por fim, como de costume em qualquer cerimônia de Scrum, considero uma boa prática ser feita uma Retrospectiva para avaliar pontos positivos e de melhoria desta reunião.

Novamente, esta agenda serve somente como um exemplo. Caso sinta necessidade, você pode incluir outras práticas ou atividades, ou utilizar técnicas diferentes. E também pode remover algo caso ache desnecessário.

Artefatos de Scrum

Scrum possui alguns artefatos que dão uma visão do andamento do projeto e das Sprints.

Os artefatos de Scrum são:

- Product Backlog
- Sprint Backlog
- Product Increment
- Definition of Done
- Burndown Charts

Embora utilizados ao longo do projeto, estes artefatos possuem momentos de criação diferentes.

Product Backlog

O Product Backlog é uma lista ordenada criada pelo Time Scrum, mas onde somente o Product Owner pode inserir, remover ou reordenar os itens.

O formato mais utilizado para estes itens é o de User Stories, que deverão ser ordenados de acordo com o critério do Product Owner – geralmente itens mais importantes ficam no topo, e serão implementados antes. Em geral, os itens de maior importância são os itens que se têm maior conhecimento, e por isso possuem um detalhamento maior. Itens que precisem de maior refinamento geralmente têm uma importância menor e ficam mais abaixo no Product Backlog.

Além dos itens de negócios (ou funcionais), o Product Backlog também contém itens não-funcionais (tempo de resposta), arquiteturais (ser desenvolvido com determinada tecnologia) e de infraestrutura (estar disponível somente dentro de uma intranet). Também pode conter itens que representem riscos a serem removidos.

Durante o andamento do projeto, algumas funcionalidades podem acabar perdendo a importância – não importando sob quais circunstâncias. Isso é normal na maioria dos projetos, uma vez que é impossível saber, desde o início, os detalhes de tudo o que queremos no produto. Assim, algumas funcionalidades podem acabar até mesmo desaparecendo. Da mesma forma, novas funcionalidades também podem ser adicionadas, de acordo com a necessidade.

Um detalhe importante quanto aos itens do Product Backlog é quando o item está considerado como pronto para ser trabalhado, ou *readiness*: para que um item possa ser incluído em uma Sprint, ele deve ser pequeno o

suficiente para que caiba em uma única Sprint e deve deixar claro quanto a expectativa do Product Owner (geralmente através de um critério de aceite – acceptance criteria).

Sprint Backlog

O Sprint Backlog é o conjunto de itens selecionados para serem implementados durante a Sprint mais o plano para transforma-los em um Incremento. Assim, ao final de cada Sprint Planning Meeting, um novo Sprint Backlog é criado. Normalmente, o plano é composto das tarefas técnicas necessárias para transformar o item em um incremento do produto.

Um dos formatos mais utilizados de Sprint Backlog é o de Task Board:

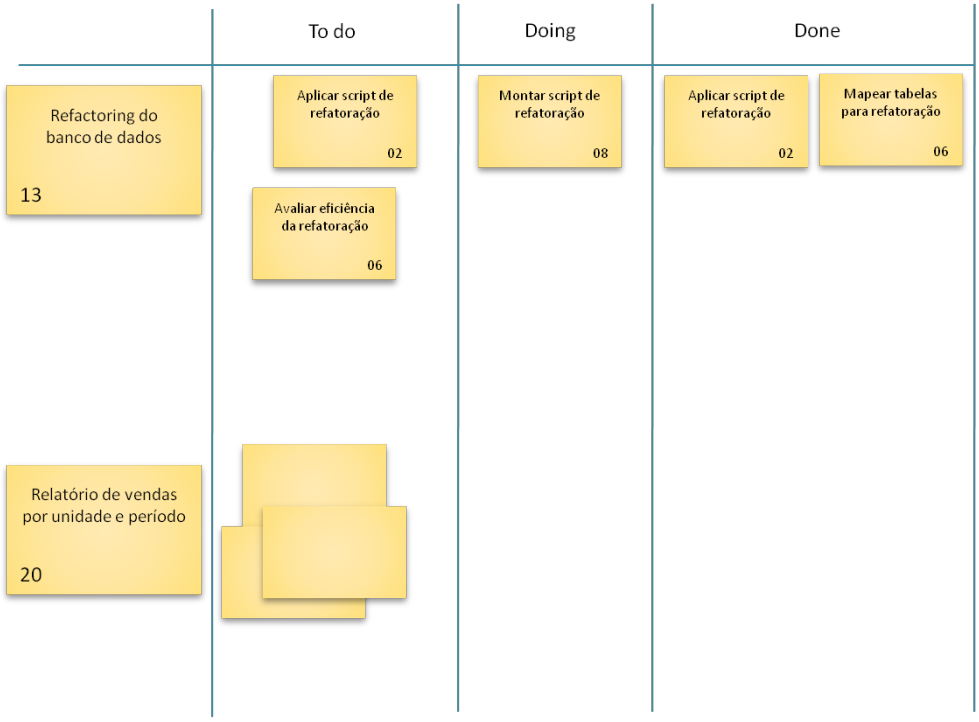

Figura 2 – Exemplo de Task Board

No Task Board, à esquerda temos os itens vindos do Product Backlog, enquanto que nas colunas **To do**, **Doing** e **Done**, temos as tarefas a serem feitas, em andamento e prontas, respectivamente. Opcionalmente, as tarefas podem apresentar o nome da pessoa que esteja trabalhando nela.

O objetivo do Sprint Backlog é tornar visível o trabalho necessário para que o Development Team atinja a meta da Sprint. Para isto, os membros do Development Team (e somente eles) podem adicionar novas tarefas caso descubram, no decorrer da Sprint, que mais trabalho seja necessário. Da mesma forma, também podem remover tarefas caso estas se mostrem

desnecessárias. Mas é importante que ele seja atualizado pelo menos uma vez por dia.

Product Increment

Ao final de cada Sprint, o Development Team entrega um incremento do produto, resultado do que foi produzido durante a Sprint. Este é um dos conceitos principais de Scrum e vai de encontro com a sua natureza empírica, já que permite que o Product Owner perceba o valor do investimento e também vislumbre outras possibilidades.

Para o Development Team é importante entender que o Incremento deve ser algo *potencialmente entregável* – o cliente pode optar por colocar imediatamente em produção. Com isso, o Development Team deve produzir código que tenha qualidade, com o mínimo de *bugs* possível (existem práticas de engenharia ágil que ajudam a conseguir isso), evitando os famosos "quebra-galhos", "depois eu arrumo isso", etc.

Definition of Done (DoD)

Scrum não define quaisquer práticas de engenharia de software para o Development Team. Entretanto, ela recomenda que se utilize práticas ágeis para este fim, que permitam ao Development Team garantir a qualidade do que se esteja produzindo durante a Sprint.

Assim, mediante as práticas adotadas pelo Time, tem-se a definição do que é uma funcionalidade "pronta" – código testado unitariamente, código testado mediante teste de aceitação, código revisto e assim por diante. No mínimo, o Definition of Done deve conter "potencialmente entregável".

Uma funcionalidade somente é considerada "pronta" se tiver passado por todas as etapas definidas pelo Development Team. E todo o Time Scrum deve entender o que significa "pronto". Uma funcionalidade que não esteja pronta ao final da Sprint deve retornar ao Product Backlog para que seja incluída em uma próxima Sprint.

Conforme o Time Scrum amadureça, é esperado que esta Definition of Done se expanda para acomodar mais critérios visando melhoria na qualidade.

Burndown Charts

Em Scrum, o Development Team é autogerenciado, e para ajudar neste autogerenciamento, o Development Team pode fazer uso do Sprint Burndown Chart.

O Sprint Burndown Chart torna visível a evolução diária do trabalho do Development Team, pois mostra a comparação entre o trabalho estimado

inicialmente no Sprint Planning Meeting com a quantidade restante estimada de trabalho. Normalmente, as unidades utilizadas são de esforço (em horas) planejado pelo tempo decorrido.

A figura a seguir mostra um exemplo de um Sprint Burndown Chart:

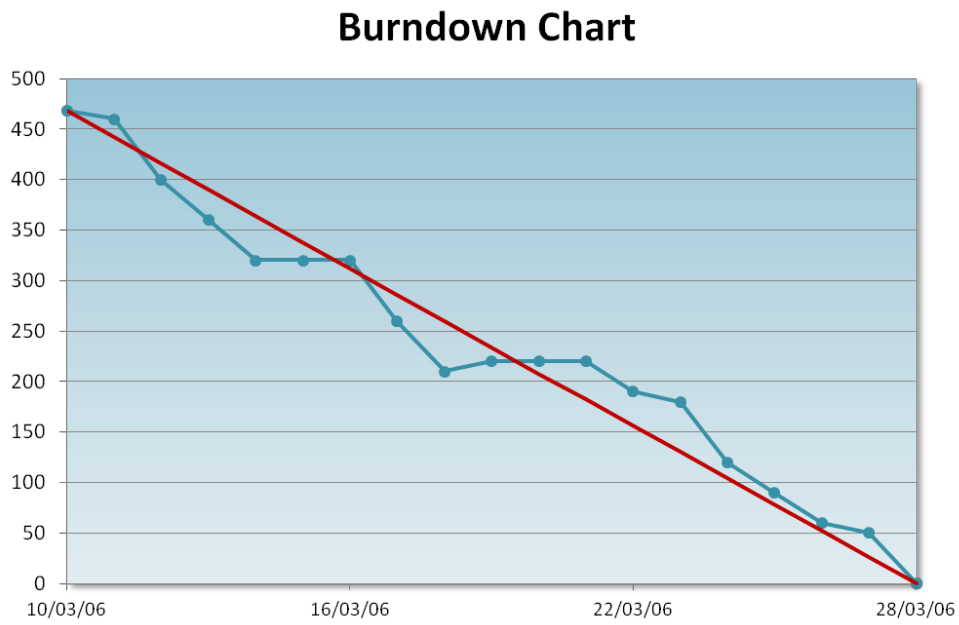

Figura 3 – Sprint Burndown Chart

Além de ajudar o time a se autogerenciar, um Sprint Burndown Chart também permite que o restante da organização também possa acompanhar o andamento da Sprint.

Enquanto o Sprint Burndown Chart permite que o Development Team faça o autogerenciamento da Sprint, o Product Owner pode fazer uso de um outro gráfico para gerenciar o Release: o Release Burndown Chart.

Através do Release Burndown Chart, o Product Owner tem uma indicação do término da Release de acordo com os dados e estimativas feitas pelo Development Team, ou a quantidade do Product Backlog que será completada.

Um exemplo de Release Burndown Chart:

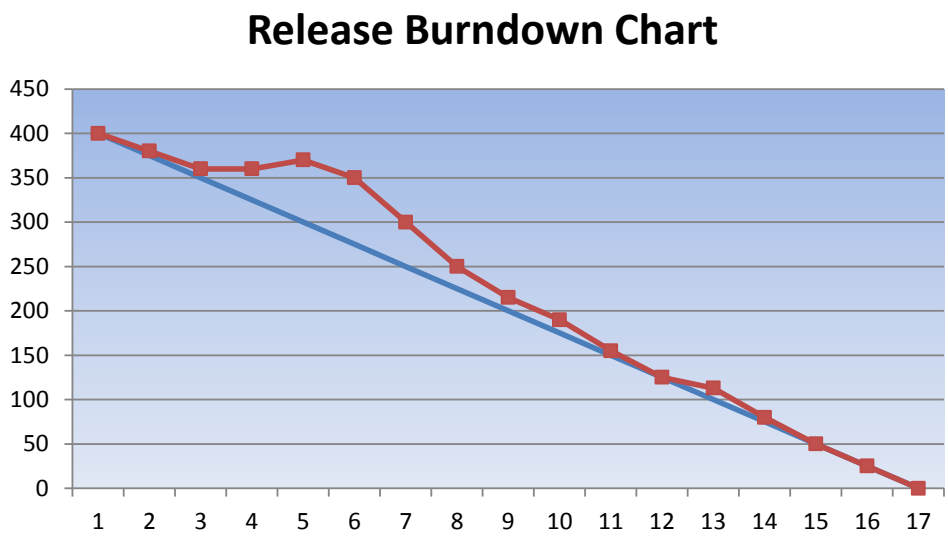

Figura 4 – Release Burndown Chart

As unidades utilizadas para o Release Burndown Chart são de story points por Sprints.

Apêndice

Auto-Organização

Quando falamos em auto-organização, muitos ainda entendem como "desorganização" ou "indisciplina". Entretanto, a realidade é muito diferente.

Times auto-organizados recebem bastante autonomia, e consequentemente bastante responsabilidade. Alguns exemplos onde a auto-organização é necessária:

- Na estimativa dos itens do Product Backlog
- Na definição dos itens que farão parte do Sprint Backlog
- No acordo com a meta da Sprint
- No Daily Scrum
- No planejamento técnico de como os itens serão transformados em Incrementos do produto
- Na atribuição de tarefas

É importante notar que para que o time se auto-organize, é necessário que os mesmos trabalhem em busca de uma meta compartilhada. Caso contrário, independemente do nível de maturidade do time, será difícil a auto-organização.

Facilitação

Segundo Peter Pfeiffer no livro Facilitação de Projetos, “um facilitador é um catalisador para as diversas idéias que se apresentam em um grupo”.

- Assim, um facilitador procura sempre:
- Otimizar processos de aprendizagem
- Evitar que o grupo tome decisões erradas
- Detectar e reduzir conflitos para aumentar a produtividade
- Evitar que um grupo seja dominado por um indivíduo que se impõe
- Motivar as pessoas mais tímidas e proporcionar uma participação mais ativa
- Fazer com que a equipe entenda, aceite, apoie e tome decisões

Embora existam técnicas bastante eficazes que podem auxiliar o facilitador em diversos eventos, ele deve estar sempre atento para detectar possíveis dificuldades no dia a dia do Time Scrum.

Leitura Recomendada

FACILITAÇÃO DE PROJETOS, de Peter Pfeiffer. Brasport.

FACILITATION 201: AN INTENSIVE INTRO TO TECHNIQUES, Bonner Foundation (http://www.bonner.org/pdffiles/modules/BonCurFacilitation201.pdf)

Visão

Apesar de não ser oficialmente uma cerimônia Scrum, a Reunião de Visão do produto permite ao Product Owner expôr ao Development Team e ao Scrum Master a visão do produto que norteará o projeto. Nesta reunião, o Product Owner pode coletar sugestões e idéias para compor o Product Backlog.

Esta reunião é importante para que todos dentro do Time Scrum saibam o direcionamento do projeto. Através desta visão, o Development Team sabe durante todo o ciclo de vida do projeto se estão ou não se desviando da visão original.

Criar uma Declaração de Visão também ajuda a definir a finalidade do produto. Em essência, técnicas como "Elevator Statement" nos ajudam a descrever o produto em poucas sentenças. Com isto, temos uma clareza maior sobre o que é o produto e para que ele serve.

Além disso, técnicas como: *Product Vision Box*, *Product Roadmap*, *Project Datasheet* e outras, são extremamente úteis para que todos estejam mais certos dos objetivos do produto antes do início das Sprints. Apesar de parecer desnecessário, isto resulta em um grupo extremamente alinhado com a visão resultante.

Esta reunião também é uma excelente oportunidade para que o Product Owner e o Development Team decidam quanto a alguns aspectos importantes sobre o produto, o projeto e o processo.

Leitura Recomendada

INNOVATION GAMES: CREATING BREAKTHROUGH PRODUCTS THROUGH COLLABORATIVE PLAY, de Luke Hohmann. Addison-Wesley Professional.

GAMESTORMING: A PLAYBOOK FOR INNOVATORS, RULEBREAKERS, AND CHANGEMAKERS, de Dave Gray, Sunni Brown e James Macanufo. O'Reilly Media.

AGILE PRODUCT MANAGEMENT WITH SCRUM: CREATING PRODUCTS THAT CUSTOMERS LOVE, de Roman Pichler. Addison-Wesley Professional.

User Stories

Ao descrever os itens de negócios para o Product Backlog, o Product Owner pode optar por diversos formatos, tais como requisitos, Use Cases, features (FDD), User Stories, entre outros, sendo este último um dos mais utilizados. Isso porque um User Story descreve uma funcionalidade que tenha valor ao cliente, mas em termos de negócios – e não de uma forma técnica. Assim, um User Story é composta de três aspectos:

- Uma descrição escrita da história como um lembrete
- Conversas sobre a história que servem para detalha-la
- Testes que conduzem e documentam aspectos importantes da história e que podem determinar quando ela está completa

Como normalmente User Stories são escritos em cartões de notas, convencionou-se a utilização dos 3Cs para User Stories:

- Cartão
- Conversação
- Confirmação

O modelo mais utilizado para a descrição de uma história em uma User Story é:

Eu, como um [papel], gostaria de [funcionalidade] para [motivo].

Neste modelo, papel representa qualquer tipo de usuário que esteja utilizando o produto. Funcionalidade é a funcionalidade desejada, e motivo é o que determina o propósito desta funcionalidade. Desta forma, o Product Owner é capaz de justificar a necessidade de uma determinada funcionalidade e certificar que ela esteja alinhada à visão; já os membros do Development Team serão capazes de se lembrar para qual papel/perfil/persona estão desenvolvendo aquela funcionalidade.

Ao se criar um User Story, o Product Owner deve sempre ter em mente que ela atenda aos critérios INVEST:

Independent	**idealmente, pode ser implementado em qualquer ordem**
Negotiable	**pode ser negociável**
Valuable	**possui valor ao cliente**
Estimatable	**é capaz de ser estimado e ordenado**
Small	**pequeno e com descrição curta**

Testable **podem-se criar testes para ele**

É claro, dificilmente encontraremos dois User Stories idênticos. E por isso, Times de Desenvolvimento definem o "tamanho" de um User Story de acordo com alguma escala. Embora existam diversas escalas que possam ser utilizadas, a mais adotada é a de Story Points.

Leitura Recomendada

USER STORIES APPLIED: FOR AGILE SOFTWARE DEVELOPMENT, de Mike Cohn. Addison-Wesley Professional

Estimativa

Quando utilizamos User Stories, é comum utilizarmos uma outra unidade de medida, ao invés do usual “tempo” utilizado frequantemente em metodologias tradicionais. No caso de User Stories, utilizamos “Story Points”.

Isso porque utilizar “horas ou dias” como unidade de medida estima não somente o esforço necessário para a funcionalidade, mas também a velocidade individual dos membros que trabalhem nela.

Story Points é uma unidade de medida relativa que leva em consideração o esforço necessário para realizar uma determinada funcionalidade. Se uma funcionalidade requerer o dobro de esforço para ser implementada, ela receberá aproximadamente o dobro de Story Points.

Para estimar a quantidade de Story Points de um User Story, o Development Team o compara com outros já estimados. Caso não haja ainda nada estimado no Product Backlog, o Development Team localiza o User Story com o menor esforço para o desenvolvimento, e o utiliza como base para comparações.

Uma das melhores formas de se estimar Story Points é utilizando o Planning Poker. Isso porque ele combina a opinião de experts, analogia e desagregação em uma abordagem divertida na estimação de itens ou User Stories e elimina a influência que um membro do Development Team possa exercer sobre outros.

Planning Poker

No início do Planning Poker, cada membro do time recebe um conjunto de cartas. Cada carta exibe um dos valores válidos para a estimativa (0, 1, 2, 3, 5, 8, 13, 20, 40 e 100, por exemplo). Em geral, os valores seguem uma escala baseada na sequência Fibonacci – outra sequência pode ser escolhida, porém Fibonacci é a mais utilizada.

Para cada User Story a ser estimado, o Product Owner lê a descrição e esclarece quaisquer dúvidas que os membros do time tenham. Entretanto, é importante lembrar que em determinado ponto, qualquer discussão adicional não busca uma precisão maior.

Após todas as questões serem respondidas, cada membro seleciona uma carta representando sua estimativa, mas sem mostrar aos outros. As cartas não são mostradas até o momento em que todos, simultaneamente, exibem seus valores, de forma que todos vejam os valores selecionados simultaneamente.

Neste ponto, é normal que as estimativas sejam significantemente diferentes. E na realidade é um bom sinal. Quando as estimativas diferem, os membros com o maior e menor valores expõem os motivos que os levaram a escolher aqueles valores. Depois das explanações e discussões, todos recolhem suas cartas e estimam novamente, da mesma forma.

Por que Planning Poker funciona?

Planning Poker funciona porque utiliza a opinião de diversos experts na estimativa. Como estes experts compõem um time multidisciplinar (composto de indivíduos de diversas disciplinas em um projeto de software), eles são os mais indicados para estimar as histórias do que qualquer outra pessoa.

Além disso, os diálogos e justificativas permitem uma maior acuracidade das estimativas, especialmente nos itens com maior incerteza. E isto é de extrema importância em um projeto que tenha um certo nível de complexidade.

Finalmente, estudos mostram que a média de estimativas individuais levam a melhores resultados, uma vez que promovem discussões em grupo. Estas discussões em grupo são a base do Planning Poker, e elas conduzem a um consenso entre os indivíduos participantes.

Sequência de Fibonacci

A sequência de Fibonacci é a escala mais utilizada na estimativa de User Stories utilizando-se Planning Poker. Isso se deve ao fato da sequência Fibonacci ser uma função quadrática, ao invés de um função linear, o que produz uma curva assim:

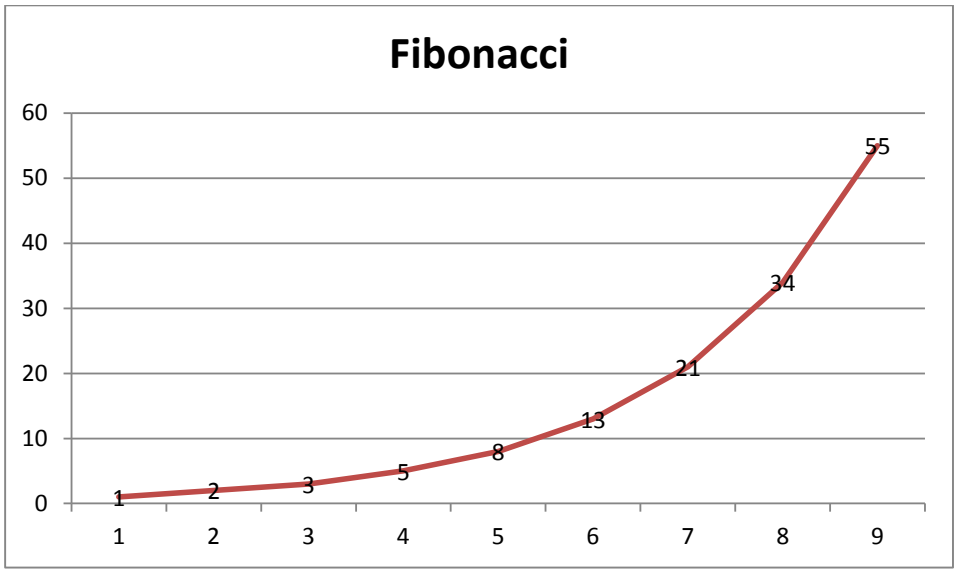

Gráfico 1 – Gráfico da sequência de Fibonacci

Embora existam outras funções quadráticas, a sequência de Fibonacci produz as diferenças muito rapidamente, criando intervalos logo no início da sequência.

Quando estimamos o esforço para o desenvolvimento de um User Story, estes intervalos refletem a imprecisão de acordo com o tamanho da história – uma história de tamanho 8 não necessariamente significa que ela tem este tamanho, mas sim, que seu tamanho está entre 5 e 13.

Velocidade

Velocidade representa a capacidade de um Development Team em uma Sprint, e é sempre medida utilizando a mesma unidade aplicada na estimativa dos itens.

Esta velocidade ajudará Product Owners a fazer, se necessário, o Release Planning e Development Teams a fazer o planejamento de Sprints.

Leitura Recomendada
AGILE ESTIMATING AND PLANNING, de Mike Cohn. Prentice Hall.

Ferramentas Online

Rally
http://www.rallydev.com/

VersionOne
http://www.versionone.com/

Mingle
http://www.thoughtworks-studios.com/mingle-agile-project-management

ScrumWorks
http://www.open.collab.net/products/scrumworks/

iceScrum
http://www.icescrum.org/en/

ScrumHalf
http://www.scrumhalf.com.br/login.jsf

Bibliografia

Cohn, Mike. 2005. *Agile Estimating and Planning.* s.l. : Prentice Hall, 2005.

—. 2004. *User Stories Applied: For Agile Software Development.* s.l. : Addison-Wesley Professional, 2004.

Pfeiffer, Peter. 2006. *Facilitação de Projetos.* Rio de Janeiro : Brasport, 2006.

Schwaber, Ken e Sutherland, Jeff. 2011. *The Scrum Guide.* 2011.