PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

Diego Martins Polla de Moraes

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS

A equipe e sua estrutura em Scrum

Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar como é a estrutura da equipe em um projeto Scrum.
- Relacionar a colaboração em um projeto Scrum e a responsabilidade compartilhada entre a equipe.
- Reconhecer o papel dos especialistas em um projeto Scrum.

Introdução

Desenvolver *software* é uma tarefa complexa, e o método Scrum traz uma série de mecanismos para tornar as equipes mais produtivas e eficientes. A implantação do Scrum exige uma mudança muito forte em relação à gestão das equipes, principalmente dos altos níveis gerenciais, da gestão tradicional, do comando e controle para a auto-organização.

Desde a discussão em relação ao tamanho da equipe até a decisão da inclusão de especialistas, o Scrum proporciona um ambiente para que as equipes se tornem autogerenciáveis e multifuncionais. Apesar disso, não existem fórmulas que se apliquem a qualquer equipe. Neste capítulo, você estudará sobre o assunto e verá estratégias para lidar com as equipes Scrum.

1 Estrutura da equipe em um projeto Scrum

Há muito que se discutir em relação a como um grupo de pessoas pode se organizar para desenvolver um trabalho específico. Com desenvolvimento de *software* não é diferente, e principalmente por ser um trabalho essencialmente intelectual, os desafios são ainda maiores.

Uma equipe de desenvolvimento de *software* deve ser consistente. É um grupo de pessoas trabalhando em conjunto, onde o todo é maior que a soma das

partes. Existem alguns atributos que são geralmente encontrados em equipes de desenvolvimento de *software* eficazes (PRESSMAN; MAXIM, 2016):

- senso de propósito;
- senso de envolvimento;
- senso de confiança;
- senso de melhoria.

Na sequência serão apresentados cada um destes atributos, e veremos como o Scrum os proporciona para uma equipe (PRESSMAN; MAXIM, 2016).

Senso de propósito: está vinculado a todos os membros concordarem com um propósito específico. O Scrum facilita isto através do envolvimento efetivo da equipe na definição do objetivo da *Sprint*.

Senso de envolvimento: significa que os membros da equipe sentem que suas contribuições e suas qualidades têm importância. Dentro do Scrum isso está presente em todos os eventos nos quais o time é envolvido — responsável por definir o planejamento, acompanhar o andamento e, no final, apresentar os resultados e rever o processo.

Senso de confiança: a equipe deve confiar nas habilidades e na competência de seus pares. A ideia de possuir uma equipe multidisciplinar e auto-organizável propicia um ambiente de confiança mútua.

Senso de melhoria: a equipe deve manter revisões periódicas da sua maneira de trabalhar a engenharia do *software* e seu processo. O Scrum propicia que a cada ciclo a equipe tenha a oportunidade de praticar a melhoria contínua, na reunião de revisão da *Sprint* e de retrospectiva, e definindo estratégias para os planos de ação na reunião de planejamento.

No Scrum as equipes são auto-organizadas e decidem sozinhas a melhor forma de realizar o seu trabalho. Essas equipes são multifuncionais e têm capacidade de ser autossuficientes. Uma equipe de desenvolvimento em Scrum é responsável por entregar incrementos de produto, levando em conta a definição de "pronto" que a equipe definiu. Além disso, a equipe deve ser capaz de estimar o tamanho dos itens do *backlog* do Produto, com o intuito de selecioná-los para definir o *backlog* da *Sprint* e, consequentemente, seu objetivo (PRIKLADNICKI; WILLI; MILANI, 2014).

Segundo Schwaber e Sutherland (2017), ninguém precisa orientar como uma equipe Scrum transforma o *backlog* do produto em incrementos do produto potencialmente utilizáveis pelos usuários, ou seja, uma equipe Scrum deve ser capaz de definir sua própria estratégia para que, a partir da necessidade do usuário, consiga entregar-lhe um *software* em funcionamento.

Uma das maiores discussões em relação à formação de uma equipe Scrum é o seu tamanho. O guia do Scrum é bem claro em relação a isso: entre 3 a 9 integrantes, sem contar o *product owner* e o Scrum Master, a não ser que eles também executem trabalho para entregar o *backlog* da *Sprint*. Este tamanho é determinado baseado no que se espera de uma equipe Scrum. Uma equipe muito pequena terá dificuldade de cobrir todas as habilidades e especialidades necessárias para transformar os itens do *backlog* do produto em *software* pronto. Por outro lado, uma equipe com muitos membros tornará o trabalho de gerenciamento complicado em virtude de (SCHWABER; SUTHERLAND, 2017):

- quantidade de pessoas nos eventos;
- senso de pertencimento diminuído;
- organização de entregas dificultada.

Em sua obra, Cohn (2011) chama atenção para o fato desses limites de tamanho não serem levados tão à risca. Suponha que você tem uma equipe Scrum com 10 pessoas. Isso significa que o limite foi desrespeitado. Na Figura 1 você pode verificar a representação de um time Scrum com 10 pessoas, um *product owner*, um Scrum Master e a execução de um evento de cada tipo, exceto a reunião diária.

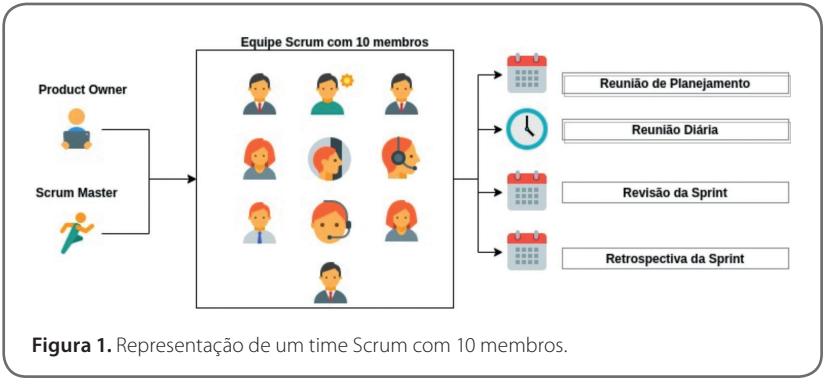

Figura 1. Representação de um time Scrum com 10 membros.

Se seguirmos a regra do guia do Scrum e da maioria dos estudiosos do assunto, teríamos que dividir esta equipe. Será que dividir em duas equipes, com cinco integrantes cada, vai gerar um resultado melhor? Na Figura 2, esta divisão é representada. Neste caso, o *product owner* e Scrum Master teriam que dividir a atenção entre as duas equipes. Além disso, todos os eventos ocorreriam duas vezes. Esta representação é para fazer você refletir que o tamanho sugerido é uma orientação. Deve-se estudar o contexto do seu projeto e definir conforme o cenário que traz mais desempenho. Isso será possível através de inspeção e adaptação, presentes no planejamento e retrospectiva constantes do time nas *Sprints*.

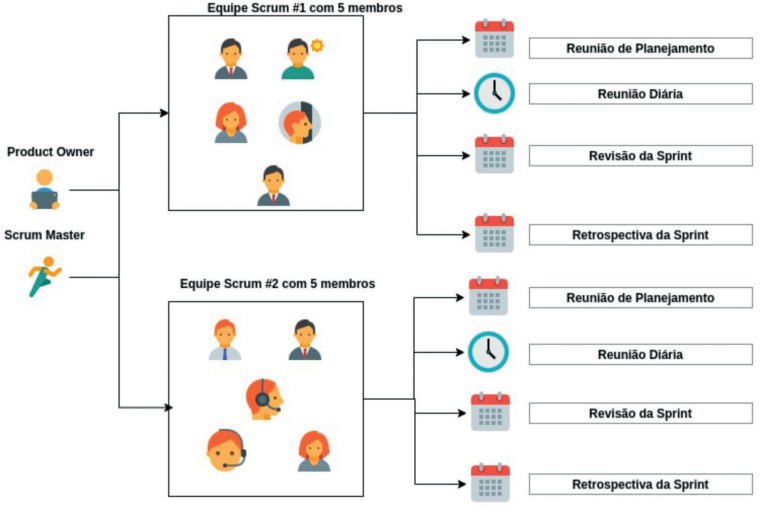

Figura 2. Representação de dois times Scrum com cinco membros cada.

No seu estudo, COHN (2011) apresenta as vantagens das equipes pequenas.

Diminui a ociosidade social: existe uma tendência de as pessoas diminuírem o esforço quando acreditam que outras pessoas assumirão as responsabilidades. Há um senso de responsabilidade pessoal mais apurado em uma equipe pequena, pois os membros estão mais próximos, compartilham mais informações e estão mais ligados ao resultado.

Interações construtivas: equipes com mais de 12 pessoas têm dificuldades para criar relacionamento de confiança, responsabilidade mútua e coesão.

Menos tempo gasto com gestão: planejar os eventos e coordenar a comunicação com uma equipe menor é muito mais rápido.

Todos têm oportunidade de participar: em equipes maiores, há uma participação menor de cada membro em discussões em grupo. São muitas pessoas para falar e, por vezes, existe uma disparidade entre espaços de fala e protagonismo entre os membros da equipe.

Satisfação individual é maior: existe uma tendência dos membros das equipes menores de se sentirem mais responsáveis, pois suas contribuições são mais visíveis e significativas.

Tendência de multifuncionalidade: em uma equipe menor, não há muito espaço para alguém que só faça uma função, ou seja, tenha uma única especialidade. Todos tendem a executar mais funções com o objetivo de transformar itens de *backlog* em *software* pronto.

Para formar uma equipe Scrum, deve-se considerar alguns fatores (COHN, 2011):

- incluir todas as áreas de atuação necessárias para que o time seja realmente multifuncional;
- equilibrar os níveis de habilidade técnica para que os membros mais experientes façam atividades de acordo com a habilidade e não se sintam entediados em trabalhar em atividades mais básicas, estas executadas pelos mais iniciantes;
- equilibrar os níveis de conhecimento sobre o negócio para que os times tenham acesso a este conhecimento e garantir que ele seja disseminado na empresa;
- buscar a diversidade, pois equipes heterogêneas têm uma tendência de levar mais tempo para tomar decisões, mas costumam ter uma visão de diversas perspectivas, e estas decisões consideram isso.

Fique atento

As equipes de desenvolvimento de *software* precisam lidar constantemente com urgências. A pergunta é: como lidar com o atendimento de uma urgência que só poderá ser planejada num período específico, já que o Scrum prescreve ciclos de *Sprint* planejados entre 7 a 30 dias?

Quando a quantidade de urgências é muito grande, tornando impossível a conciliação com as atividades da *Sprint*, a sugestão é que se crie uma equipe paralela para tratar as urgências, o que Kniberg (2007) chama de "equipes de bombeiros". Uma equipe de bombeiros blindará o time Scrum de uma série de intervenções, como correções de *bugs* e melhorias urgentes, e podem atuar na prevenção de novas falhas.

Se for necessário criar uma equipe de bombeiros, lembre-se de dividir membros experientes entre a equipe Scrum e a equipe de bombeiros, e de criar uma rotina de rodízio entre as equipes. Assim, cada uma enxerga a realidade do projeto como um todo.

2 Time Scrum: auto-organizável e colaborativo

O manifesto ágil deixa bem claro a importância que a equipe tem para o desenvolvimento de *software*. Como você pode verificar a seguir, entre os 12 princípios ágeis, cinco falam diretamente sobre a equipe (BECK *et al*., c2001):

- pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto;
- construa projetos em torno de indivíduos motivados. Dê-lhes o ambiente e o suporte necessários e confie neles para fazer o trabalho;
- o método mais eficiente e eficaz de transmitir informações para uma equipe de desenvolvimento é através de conversa face a face;
- as melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis;
- em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz, e então refina e ajusta seu comportamento de acordo com esta percepção.

O Scrum, através dos seus papéis e eventos, proporciona às suas equipes um cenário adequado para que estes princípios possam ser vivenciados na prática.

Em sua obra, Cohn (2011) ressalta um ponto muito relevante em relação à equipe Scrum: não existe trabalho individual, existe o trabalho da equipe.

Portanto o sucesso ou fracasso nunca é atribuído de forma individual, e sim reconhecido pela equipe como um todo.

Você pode pensar, por exemplo, que um código fonte limpo, bem escrito e com qualidade deve ser responsabilidade única e exclusiva dos programadores. Mas e se um testador identificar alguma funcionalidade que tem uma alta incidência de problemas e montar algum tipo de cenário de testes para ajudar os programadores a testar o código de uma forma mais apurada? É este tipo de abordagem que o Scrum proporciona. Não interessa exatamente o papel mais específico que cada um exerce, mas o que cada um pode fazer para que a equipe em conjunto entregue o resultado esperado no prazo e com qualidade (COHN, 2011).

Para reforçar este conceito, Cohn (2011) afirma que a única forma de criar um ambiente em que a responsabilidade é compartilhada é acabar com a ideia de que é preciso culpar alguém. Afirmar isso não significa que não haja um responsável, e sim que, em uma equipe de sucesso, os membros fazem a sua parte e vão até mesmo além do papel que se vinculam a fim de garantir que a equipe alcance seus objetivos.

Uma equipe Scrum é auto-organizável. Mas será que um time Scrum é auto-organizável já no primeiro dia que começa a executar o método? Quando você poderá ter certeza de que seu time é auto-organizável e a empresa para a qual atua o reconhece assim?

Ao conhecerem o Scrum, diretores e gerentes de empresas costumam fazer as perguntas a seguir (COHN, 2011).

- Se eles determinam como trabalhar e quanto tempo levam, irão reduzir a velocidade e entregar menos, afinal não tem ninguém cobrando?
- Se a responsabilidade é da equipe, ninguém se responsabilizará? Todos repassarão a responsabilidade para outro?
- Se eles se autogerenciam, para que eu vou servir?

A resposta para estas perguntas vem da maturidade na utilização do método. É natural que a adaptação ao método Scrum cause estas dúvidas a todos, não só à alta gerência. Mas a verdade é que numa equipe de desenvolvimento de *software* não há méritos ou culpas individuais, o trabalho é executado em conjunto e o time deve ser responsabilizado como um todo. O apoio das gerências funcionais para utilização do Scrum, seguindo suas regras e investimento na melhoria contínua, dará ao time cada vez mais autonomia.

No Scrum, o sentimento de pertencimento e de responsabilidade dos membros da equipe começa na reunião de planejamento. Naquele momento o time será responsável por determinar quanto tempo e como realizará o trabalho para

concluir um item de *backlog*. Veja como isso faz toda a diferença: ao invés de um gerente determinar à equipe como ela deve se organizar e quanto tempo levará para executar uma atividade, a própria equipe faz isso. A tendência é que as pessoas se comprometam muito mais com algo que elas mesmo planejaram. Se alguém determinar o modo de realizar uma tarefa e o tempo que você deve levar para executá-la, se você não conseguir seguir aquele plano, ou atrasar a entrega, é bem provável que simplesmente culpe quem o delegou. Agora, se você mesmo determinou o modo de trabalho e o tempo necessário, se não conseguir cumprir, irá procurar os motivos por que isso ocorreu e uma forma de evitar que aconteça novamente (COHN, 2011).

Na reunião de retrospectiva, conforme afirmam Schwaber e Sutherland (2017), a equipe irá identificar as práticas que devem ser continuadas e estimuladas, o que deve ser revisto e o que não deve ser repetido. É o momento para que a própria equipe corrija seu modo de trabalhar. Dessa forma o time pratica a melhoria contínua, aplicando na próxima *Sprint* os planos de ação definidos.

Saiba mais

A realização dos eventos do Scrum pela equipe é o ponto crucial para que o trabalho seja executado e o processo, aprimorado. A retrospectiva é um evento muito importante, pois traz a oportunidade de a equipe se autoavaliar e propor formas de melhorar sua *performance*. Por ser um evento que consiste em uma reunião com toda equipe, é importante que ocorra de forma célere e produtiva. Leia o artigo "5 passos para melhorar as dinâmicas para retrospectivas Scrum", de Lauren Moon, para conhecer formas de otimizar sua reunião de retrospectiva.

3 Especialistas ou generalistas em uma equipe Scrum?

Schwaber e Sutherland (2017) dizem que o time Scrum é autossuficiente e multifuncional. Isso por acaso significa que todos membros serão generalistas e deverão saber fazer tudo? Veremos a seguir.

Desenvolvimento de *software* é uma atividade muita complexa, que envolve muitas habilidades, formações e conhecimento sobre tecnologias específicas, e exigir que todos os membros da equipe tenham esses requisitos é inviável. Torna o custo da equipe altíssimo.

Em uma equipe, sempre existirão pessoas especializadas em determinadas atividades, tecnologias ou áreas de negócio específicas. O convite que o Scrum faz é que elas não se isolem. Em equipes tradicionais, a tendência é que um especialista sempre seja demandado para executar as atividades dentro da sua especialidade: ele executará rápido, sem muitas perguntas, e com qualidade, considerando toda sua experiência. O Scrum sugere que as equipes criem um ambiente de compartilhamento, para que este especialista aos poucos repasse esse conhecimento a outros membros da equipe. Assim, se em uma determinada *Sprint* houver uma demanda alta daquele tipo de atividade, outros membros podem ser envolvidos e permitir que a equipe faça aquela entrega, sem sobrecarregar o especialista.

Um dos maiores exemplos disso, e uma das grandes dúvidas em relação às equipes Scrum, é a inclusão ou não de especialistas em qualidade de *software* na equipe. Se a definição de "pronto" do time diz que o incremento é um *software* funcionando, como garantir isso sem uma pessoa especializada em qualidade?

Na Figura 3, você pode visualizar a representação de uma Equipe Scrum sem especialista em qualidade de *software*. Esta equipe só tem desenvolvedores de *software*, programadores. Eles não têm nenhuma habilidade específica de testes, mas se comprometeram com a qualidade do *software*. Eles executam testes sempre que acabam uma funcionalidade, mas às vezes não têm tempo para isso, pois a maior parte do tempo deles é utilizada com o que mais gostam e saber fazer: desenvolver código. O *software* é liberado com muitos *bugs*, os testes deles são muito viciados, sem uma estrutura, um pensamento como usuário.

Figura 3. Equipe Scrum sem um especialista em qualidade.

Já na Figura 4, é retratada uma Equipe Scrum com especialista em qualidade de *software*. Esta equipe tem um desenvolvedor a menos que a anterior — que foi substituído pelo especialista em qualidade de *software*. A pergunta que você pode fazer é: enquanto os desenvolvedores não entregaram nenhuma funcionalidade testável, o que o especialista em qualidade faz? A resposta é simples: ele planeja como executar os ensaios de qualidade, e pode inclusive se especializar em testes automatizados e apoiar a equipe em decidir sobre detalhes de regras e experiência do usuário. Ele também deve ser responsável por disseminar este conhecimento com os outros membros do time, que continuam responsáveis pela qualidade.

Figura 4. Equipe Scrum com um especialista em qualidade.

Saiba mais

Leia o artigo "Garantia de qualidade no Scrum: muito além dos testes", de Priyanka Hasija, em que é apresentado de que forma a garantia de qualidade pode levar um time Scrum a ótimos resultados.

Referências

BECK, K. *et al*. *Manifesto ágil*: manifesto para o desenvolvimento ágil de software. c2001. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html. Acesso em: 3 jun. 2020.

COHN, M. *Desenvolvimento de software com Scrum*: aplicando métodos ágeis com sucesso. Porto Alegre: Bookman 2011.

KNIBERG, H. *Scrum e XP direto das Trincheiras*: como nós fazemos Scrum. [*S. l.*]: InfoQ, 2007.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software*: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. (Org.). *Métodos ágeis para desenvolvimento de software*. Porto Alegre: Bookman, 2014.

SCHWABER, K.; SUTHERLAND, J. *Um guia definitivo para o Scrum*: as regras do jogo. [*S. l.*: *s. n.*], 2017. Disponível em: https://www.Scrumguides.org/docs/Scrumguide/v2017/2017-Scrum-Guide-Portuguese-Brazilian.pdf. Acesso em: 3 jun. 2020.

Leituras recomendadas

HASIJA, P. *Garantia de qualidade no Scrum*: muito além dos testes. muito além dos testes. 2012. Disponível em: https://www.infoq.com/br/articles/experience-qa-Scrum/. Acesso em: 3 jun. 2020.

MOON, L. *5 passos para melhorar as dinâmicas para retrospectivas Scrum*. 2015. Disponível em: https://blog.trello.com/br/dinamicas-retrospectivas-Scrum. Acesso em: 3 jun. 2020.

Fique atento

Os *links* para *sites* da *web* fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declaram não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:

SAGAH
SOLUÇÕES EDUCACIONAIS INTEGRADAS