Projeto ágil

Fundamentos da agilidade representados pelos valores e princípios do Manifesto Ágil assinado em 2001, Conceitos e características de dois métodos ágeis muito utilizados mundialmente, Extreme Programming (XP) e Scrum.

Prof. Wagner Rodrigues Ribeiro

Propósito

Compreender a essência do conceito de agilidade, que possui em seu cerne a cultura de frequente entrega de valor, com qualidade e foco em pessoas, seja o cliente ou quem produz o produto. Compreender as principais características dos métodos ágeis XP e Scrum.

Objetivos

- Reconhecer valores e princípios do Manifesto Ágil.
- Identificar as principais características do método Extreme Programming.
- Reconhecer as principais características do framework Scrum.

Introdução

O tema apresenta inicialmente o conceito da agilidade baseada nos valores e princípios do Manifesto Ágil, com o objetivo de ressaltar que, antes de pensarmos em métodos, temos que entender a cultura de entrega de valor no desenvolvimento de produtos, com ciclos curtos de feedback, para validação frequente da expectativa do usuário. Após este entendimento, apresentaremos as principais características dos métodos ágeis Extreme Programming e Scrum.

Manifesto Ágil

Assista ao vídeo a seguir para saber mais sobre valores e princípios do Manifesto Ágil.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

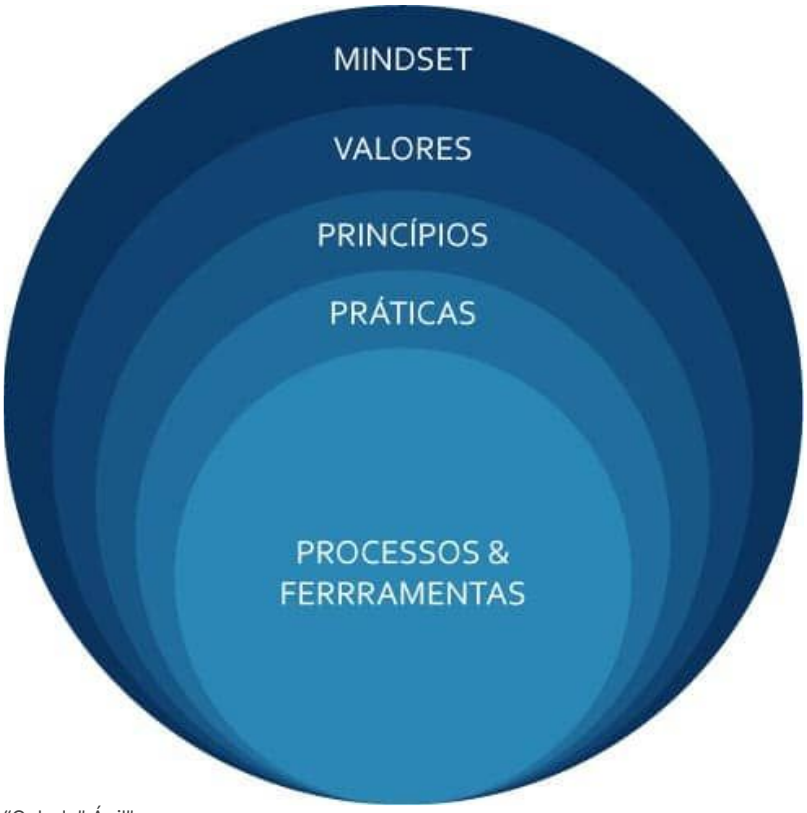
O Manifesto Ágil configura um marco no contexto do desenvolvimento de software e consequentemente na gestão de projetos para esta indústria. O documento foi assinado em um encontro nos dias 12 e 13 de fevereiro de 2001 em Snowbird, Utah, EUA, pelos seguintes expoentes desse nicho: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Bob Martin, Stephen Mellor, Jeff Sutherland, Ken Schwaber e Dave Thomas.

Os signatários já atuavam com alguns métodos como: XP (Extreme Programming), DSDM (Dynamic System Development Model), ASD (Adaptive Software Development), FDD (Feature-Driven Development), Crystal e Scrum.

> Esses métodos compartilhavam conceitos que foram incorporados ao manifesto por meio de suas ideias como **valores** e **princípios**.

A figura apresenta o que ficou popularizada como “cebola ágil”. As camadas simbolizam a relevância na transformação ágil em uma organização. As camadas internas são mais fáceis de mostrar, mas não representam criação de cultura ágil. Os processos e ferramentas podem ser impostos por uma gestão autoritária, por exemplo. Já as camadas como valores e princípios são fundamentais para a criação do mindset, ou seja, da forma de pensar ágil.

Os valores e princípios do manifesto foram criados na ocasião basicamente para atender às dores conhecidas no desenvolvimento de software, mas com a propagação dos métodos ágeis, o mercado foi percebendo os benefícios da agilidade em diversas áreas, como: RH, jurídico, comercial, inovação, entre outras.



“Cebola” Ágil”.

Valores

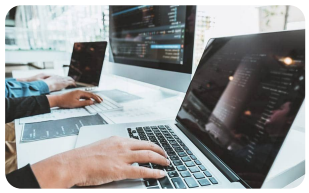
Segundo o Agile Manifesto (2001), estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

Valor 1 - Indivíduos e interações mais que processos e ferramentas

Com esse valor, o manifesto apresenta a preocupação em colocarmos mais foco nas pessoas e nas relações entre elas, do que ficarmos apoiados em processos e ferramentas. A ideia é fornecermos condições para que os times executem suas tarefas e não perdermos tempo em engessar o trabalho com processos. Outro ponto é a supervalorização das ferramentas. Ferramentas, como o próprio nome diz, devem servir de apoio.

Valor 2 - Software em funcionamento mais que documentação abrangente

A má interpretação deste valor foi responsável pelo mito que projetos ágeis não possuem documentação. Mas o que o valor quer dizer é que mais do que documentação abrangente devemos priorizar produto na mão do cliente, mas isso não quer dizer que não se documenta em projetos ágeis. Se a entrega esperada pelo cliente contempla documentação, esta deverá ser adicionada ao conjunto de critérios de aceitação.



Valor 3 - Colaboração com o cliente mais que negociação de contratos

É óbvio que a relação entre empresas deve ser baseada em contratos, mas se não houver colaboração com o cliente, com preocupação em entender suas dores e expectativas, e com ciclos curtos de feedback para validação de que as entregas atendem às necessidades, o contrato não será suficiente para segurar a relação.

Valor 4 - Responder a mudanças mais que seguir um plano

A má interpretação deste valor também foi responsável pelo mito de que a agilidade não possui planejamento. Na verdade, o que este valor apresenta é que, como atuamos muitas vezes em contextos complexos, com muitas incertezas, devemos, mais do que seguir um plano, experimentar, validar e adaptar.

Princípios

Os princípios são preceitos básicos para a condução de práticas de agilidade ou utilização de métodos ágeis. Para que um método possa ser chamado de ágil, este não pode infringir nenhum dos doze princípios abaixo:

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado

Com ciclos curtos de desenvolvimento, conseguimos ter inspeção e adaptação, que possibilitam validar frequentemente se o que está sendo entregue atende à expectativa de valor pelo cliente.

2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando à vantagem competitiva para o cliente

A mudança faz parte do contexto de complexidade do mundo em que vivemos atualmente. Logo, se o cliente deseja mudar, é porque ele precisa. Seja porque aquele requisito não atende mais à necessidade ou ele aprendeu que o que foi solicitado pode ser melhorado para agregar mais valor ao produto.

3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo

Este princípio reforça a necessidade de software funcionando (produto) como medida de progresso, ainda que com períodos curtos entre a solicitação e a entrega, para rapidamente colhermos feedback do cliente, nos adaptarmos e melhorarmos o produto.

4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto

Durante o desenvolvimento de um produto em um projeto, quanto menor ruído de comunicação entre quem define os requisitos e quem desenvolve os incrementos de produto, menor será o risco de não aceitação da entrega. Assim, devemos promover constantemente a comunicação face a face, a fim de reduzirmos a perda de entendimento entre a definição de requisitos e o desenvolvimento do produto.

5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho

Este princípio demonstra a preocupação em criarmos um ambiente favorável ao trabalhador do conhecimento. O líder precisa entender o propósito de cada colaborador, se suas atividades estão alinhadas com esse propósito, se a pessoa que está desenvolvendo as atividades está motivada com o que está fazendo, para propiciar o engajamento, dedicação, comprometimento dos membros do time no desenvolvimento do produto.

6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face

Reforçado por este princípio, o conceito de times ágeis pressupõe equipes pequenas, para facilitar a comunicação. Só com equipes pequenas podemos ter a comunicação face a face. Quando precisamos executar grandes projetos com agilidade, é necessário escalarmos, ou seja, utilizarmos técnicas para que vários pequenos times trabalhem juntos em um mesmo produto.

7. Software funcionando é a medida primária de progresso

Ao contrário do que é apresentado geralmente em relatórios de status de projetos com percentual de evolução de atividades, este princípio preconiza que software funcionando ou produto na mão do cliente é a melhor medida de progresso. Melhor que uma informação de percentual de completude de uma atividade é a entrega para o cliente de parte do produto potencialmente utilizável.

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

Agilidade pressupõe qualidade. Qualidade no produto entregue, mas também qualidade de vida para as pessoas que trabalham no produto. Processos ágeis não incentivam sobrecarga de trabalho, como horas extras, por exemplo. Em processos ágeis, as pessoas devem manter um ritmo de trabalho sustentável constante indefinidamente.

9. Contínua atenção à excelência técnica e bom design aumentam a agilidade

Quando não é dada a devida atenção à qualidade técnica no desenvolvimento do produto, ocorre o que chamamos de dívidas técnicas. Em desenvolvimento de softwares, essas dívidas técnicas podem ser código mal feito, sem testes necessários, entre outros fatores, que com o passar do tempo irão consumir o prazo da equipe para corrigi-los, com isso a capacidade da equipe em entregar novas funcionalidades é comprometida com dívidas técnicas.

10. Simplicidade é essencial

Muitas vezes a resolução de um problema está em uma solução simples. Ao apresentar uma solução simples, o time estará sendo Lean (enxuto), quer dizer: Entregando valor com menos recursos e menos tempo. Soluções elaboradas podem requerer um custo agregado de testes elaborados e risco de aumento de dívida técnica.

11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis

Em times ágeis, devemos buscar a auto-organização. Times auto-organizados recebem o objetivo da entrega e se auto-organizam para definirem a melhor forma de construir o produto. Com isso, as melhores arquiteturas emergem dessa união de perfis diferentes em prol do objetivo comum de construir uma melhor arquitetura para suportar a entrega do produto.

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

Este princípio preconiza que os times ágeis, por exemplo ao final de um ciclo de entrega, devem realizar uma retrospectiva com o objetivo de entender o que deu certo, o que deu errado e como podem melhorar para o próximo ciclo.

Atenção

O manifesto ágil, com sua definição de valores e princípios, configurou um marco no contexto do desenvolvimento de software e, consequentemente, na gestão de projetos para esta indústria. Posteriormente, entendeu-se que em outras áreas, os métodos ágeis também poderiam ser utilizados com sucesso, e atualmente temos agilidade em processos financeiros, de RH, comercial, marketing, inovação entre outros.

Verificando o aprendizado

Questão 1

De acordo com o que você estudou neste módulo, selecione a sentença que não representa um valor do Manifesto Ágil:

A

Em detrimento de muita documentação, entregue software funcionando.

B

Priorize a relação com seu cliente ao invés de recorrer constantemente ao contrato.

C

Siga seu plano à risca e defina um processo que não acate mudanças.

D

O processo não é mais importante que as pessoas e suas relações.

A alternativa C está correta.

O quarto valor do Manifesto Ágil diz para respondermos às mudanças mais que seguirmos um plano. As mudanças devem ser acatadas, porque os ciclos curtos permitem a adaptação para responder às mudanças. Este valor não fala de não seguir um plano, mas, sim, de responder às mudanças ser mais importante que seguir um plano.

Questão 2

Assinale o princípio do Manifesto Ágil que não está de acordo com a realização de excessivas horas extras.

A

Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

B

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

C

Simplicidade é essencial.

D

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

A alternativa B está correta.

O oitavo princípio do Manifesto Ágil diz que a equipe deve manter um ritmo sustentável indefinitivamente, logo não devemos ter variações de carga de trabalho, como horas extras, para que a equipe possa manter esse ritmo. O ritmo sustentável de um time ágil é determinado pelo respeito à sua capacidade durante o tempo da construção do produto. Este ritmo deve evitar variações como sobre ou subcarga de trabalho para que não tenhamos períodos de ociosidade, bem como não devemos permitir sobrecarga para que não tenhamos problemas de saúde dos membros do time e também má qualidade na entrega do produto.

Extreme Programming

Assista ao vídeo abaixo e tenha mais clareza sobre o que é Extreme Programmig e seus valores.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O Extreme Programmig (XP) é um método de desenvolvimento de software criado por três dos signatários do Manifesto Ágil: Kent Beck, Ward Cunningham e Ron Jeffries. O XP, além de ser inteiramente aderente aos valores e princípios do Manifesto Ágil, fato que o credencia como um método ágil, também é fundamentado em seus próprios valores e práticas.

Valores

Os valores denotam o grau de importância de um determinado tema e representam as melhores ações a serem tomadas no contexto deste tema. De acordo com Wildt (2015), o XP define cinco valores para que seus papéis e práticas atuem de acordo com sua essência ágil: A comunicação, o feedback, a simplicidade, o respeito e a coragem.

Comunicação

A comunicação figura como a principal causa em fracasso em projetos de software na maioria das pesquisas do ramo. O contexto de desenvolvimento de software envolve muitas incertezas, logo precisamos dar muita atenção à comunicação, ainda mais em ambientes de equipes remotas onde a comunicação face a face ocorre por meio de recursos eletrônicos, podendo acarretar perda de informação.

Feedback

O feedback rápido e frequente é outra forma de minimizar riscos em ambiente de desenvolvimento de software. Muitas vezes, o cliente só percebe o que realmente precisa ao receber as primeiras versões do que pediu, e quanto mais cedo o time receber o feedback, mais cedo fará as adaptações necessárias.

Simplicidade

A simplicidade é um valor que ajuda o time a manter o foco no valor que realmente precisa entregar. O foco é essencial para evitar desperdícios no processo de desenvolvimento de software.

Respeito

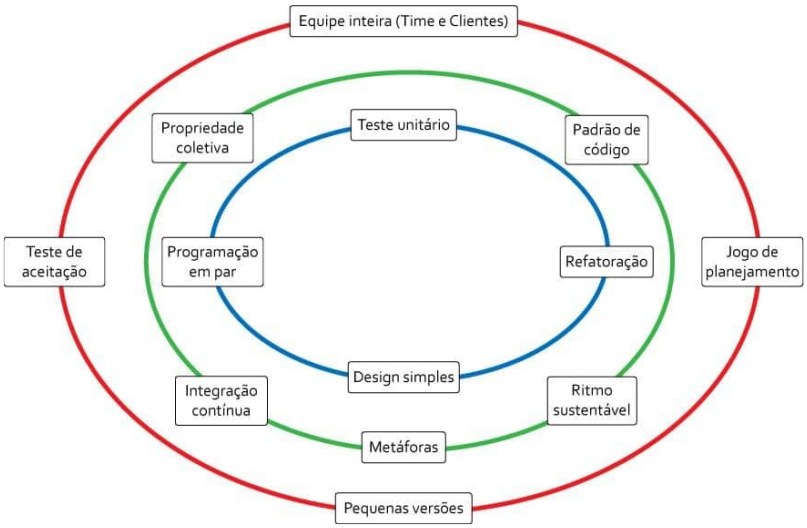
Este valor deve ser praticado por todos. Tanto os membros dos times quanto clientes e outras partes interessadas. Criar um ambiente onde se pratica o respeito, sem necessidade de encontrar culpados e focado em criar um ambiente agradável de trabalho certamente contribuirá para a formação de um time de alta performance, mais motivado e com empatia, que pratica o respeito tanto com seus pares, quanto com a qualidade do produto que está sendo desenvolvido, que se traduz em respeito com o cliente.

Coragem

As práticas dos valores anteriores, principalmente o respeito, estimulam a formação de um ambiente onde os membros do time tenham coragem para fazer o que é preciso, por exemplo, refatorar um código (ver prática: refatoração). Os membros dos times precisam se sentir em um ambiente safe to fail, que significa, em livre tradução, um ambiente seguro para falhar, para que possam ter coragem de fazer o que é preciso, sem se sentirem inseguros e com medo de serem penalizados.

Práticas

No Extreme Programmig, membros do time e cliente interagem para definir, priorizar e entregar o produto utilizando as práticas de desenvolvimento de software. Segundo Teles (2004), o time precisa ter coragem e acreditar que a utilização dessas práticas e valores do XP fará o software evoluir com segurança e agilidade.



Práticas Extreme Programming.

Veja um pouco mais sobre cada uma dessas práticas:

Equipe inteira

Colaboração com partes interessadas, como cliente, deve ser uma prática constante da equipe com o objetivo de promover a interação com o time para fornecer requisitos e prioridades para o time de desenvolvimento.

> Segundo Massari (2014), o princípio de equipe inteira ou equipe unida parte da premissa de que a equipe deve ser formada por generalistas e não por especialistas.

Jogo de planejamento

Oportunidade na qual cliente e time de desenvolvimento se reúnem para decidir quais entregas entrarão na próxima release e iterações. Uma release pode ser uma versão de um software ou um módulo do software.

No jogo do planejamento, o cliente e o time trabalham juntos, colaborativamente para planejarem a entrega de uma release. Uma release poderá ter várias iterações. No jogo, o cliente definirá o escopo, a prioridade e a expectativas de data de entrega. O time estima o trabalho apresentado em forma de história de usuário e aponta dependências técnicas, caso existam. Os desenvolvedores puxam as histórias e se comprometem com sua entrega.

Atenção

Os incrementos de produto são definidos em um plano de release e as iterações são definidas em um plano de iterações. Um incremento pode ser desenvolvido em uma ou mais iterações.

Pequenas entregas

Entregas pequenas e frequentes são essenciais para a obtenção de feedback.

Comentário

Com essa prática, o time valida se está indo no caminho correto ou precisa corrigir parte do produto para atender à expectativa do cliente.

Testes de aceitação (cliente)

Ao definir o requisito, o cliente define os critérios de aceitação para o requisito. A equipe deve desenvolver o objeto do requisito à luz dos critérios de aceitação, e implementar testes automatizados para validar se o que foi desenvolvido está de acordo com o que foi pedido pelo cliente.

Propriedade coletiva do código

Os códigos desenvolvidos em times XP são compartilhados entre os membros da equipe

evitando que indivíduos sejam seus proprietários. Isso é muito importante para que a produtividade não caia quando, por algum motivo, um membro esteja indisponível. Outro ponto importante é que a qualidade aumenta quando mais de uma pessoa trabalha no código, em razão do compartilhamento do conhecimento.

Padrão do código

Os times XP seguem um padrão de codificação para manter a produtividade coletiva.

Comentário

Assim, quando um membro for trabalhar em um código desenvolvido por outro membro, não haverá problema de entendimento.

Ritmo sustentável

De acordo com o WHAT IS EXTREME PROGRAMMING, assim como abordamos no módulo anterior, os times XP devem manter um ritmo de trabalho que possa perdurar por muito tempo, evitando grandes variações de carga de trabalho, como, por exemplo, com a realização de horas extras.

Metáforas

As metáforas são utilizadas para manter uma linguagem comum entre os membros do time XP.

> É uma técnica para que todos saibam identificar rapidamente uma determinada rotina ou sistema.

Quando alguém se referir a determinado nome, não deixará dúvida sobre o que ele está mencionando. Isso é mais fácil do que explicar sobre o que se está falando.

Comentário

Segundo Wildt (2015) usamos metáforas em nosso cotidiano para facilitar nosso entendimento, como o carrinho de compras de uma loja virtual ou a área de trabalho de nosso computador com nossas pastas.

Integração contínua

Os times XP integram seus códigos várias vezes ao dia.

Atenção

Quanto mais tempo levar para integrar os códigos escritos, mais difícil será para ajustar erros. Por isso, é muito importante integrar continuamente o código.

Testes unitários

Os testes unitários servem para fornecer feedback imediato ao desenvolvedor ou ao par de desenvolvedores. Na execução do código principal, o código do teste também é executado, informando se o código feito está com erro ou não.

Refatoração

A melhoria contínua é uma preocupação dos métodos ágeis e no XP não é diferente.

A refatoração ou melhoria contínua é uma técnica para melhorar o código minimizando problemas como duplicidade de código.

Para Massari (2014), o princípio da refatoração consiste em melhorar a qualidade do código existente, tornando-o mais elegante e legível, porém sem alterar o comportamento da funcionalidade.

Design simples

Alinhado ao valor simplicidade, o XP preconiza que o design do software comece simples e continue assim, mesmo com o incremento de funcionalidades no sistema.

Comentário

As práticas como testes unitários, programação em par e refatoração ajudam a evoluir com o software e manter seu design simples.

Programação em par

De acordo com Massari (2014), os códigos produzidos por times que adotam o XP são desenvolvidos em par, mas em uma mesma máquina.

Atenção

Está técnica propicia não só um melhor compartilhamento de código, mas também de conhecimentos, o que propicia alcançar melhor qualidade, identificar riscos e formar equipes de alto desempenho.

Outra técnica muito utilizada no Extreme Programmig, que também aparece no Scrum, é a história de usuário.

Seu formato simples requer uma conversação face a face para coletar detalhes do cliente pelo time de desenvolvimento. Exemplos de histórias de usuários:

- Eu, como médico, quero pesquisar prontuário do paciente pelo número do CPF.

- Eu, como cliente, quero pesquisar restaurantes próximos à minha casa.

As práticas do Extreme Programmig de desenvolvimento de software foram tão consolidadas pelo mercado que atualmente é comum serem utilizadas em conjunto com outros métodos ágeis, por exemplo, o Scrum.

Papéis do XP

O Extreme Programming preconiza alguns papéis que objetivam equilibrar as responsabilidades inerentes ao desenvolvimento de software. Conforme Wildt (2015), em alguns casos, um indivíduo pode acumular papéis, desde que não haja conflito de interesses, por exemplo, um indivíduo que atua em um papel de gerente, sendo pressionado para realizar uma determinada entrega, e este mesmo indivíduo atuando como coach com necessidade de trabalhar com mais paciência e foco na evolução comportamental do time.

Desenvolvedor

O desenvolvedor, também conhecido como programador, é um papel com atribuições ligadas diretamente à criação do software com qualidade. As atribuições do desenvolvedor vão desde a estimativa das atividades até a entrega do produto. Em times ágeis multidisciplinares é comum termos desenvolvedores com especialidades específicas como front-end ou back-end. Mas podemos ainda ter indivíduos que possuem competência para atuar em todas essas áreas, como os chamados atualmente de full stacks. Obviamente, este perfil é muito interessante para um time ágil, mas não é fácil montar times com vários indivíduos assim, pelo fato de geralmente pedirem uma remuneração maior e serem mais difíceis de ser encontrados no mercado para contratação, uma vez que são profissionais que costumam possuir um nível mais elevado de conhecimentos.

Cliente

O cliente é o papel que mais conhece do negócio. No Scrum, existe um papel semelhante a este, que é conhecido como Product Owner. O cliente do Extreme Programming é quem define os requisitos em formato de histórias de usuário, bem como sua priorização para os desenvolvedores materializá-las em produto. O papel do cliente é fundamental para fornecer o feedback rápido em relação ao produto. Sabemos que mesmo com uma boa definição do requisito e frequente comunicação do cliente com a equipe, ainda é possível que, ao utilizar o produto, a experiência possa não atender à expectativa do cliente, por isso é importante que o cliente alimente o time de desenvolvimento com constantes feedbacks sobre a sua percepção de utilização do produto.

Coach

O XP é intimamente orientado ao lado comportamental do time. Isso quer dizer que além da capacidade técnica, o time precisa respeitar valores, aderir às práticas, ter disciplina com os ritos necessários do método. Para ajudar o time a criar essa cultura, o XP conta com o papel do coach. O coach precisa conhecer bem de desenvolvimento de sistemas e ser um técnico capaz de ajudar o time a assimilar os valores e utilizar as práticas do XP. O coach é o líder servidor, ele facilita cerimônias, como jogo do planejamento, reuniões diárias e retrospectivas e é o guardião dos valores do XP. Ele serve à equipe XP, identifica necessidades de cada membro, como a realização de um treinamento para que um desenvolvedor possa melhor realizar suas atividades. O coach também pode acumular o papel com um desenvolvedor, por exemplo.

Testador

O testador é um papel importante para garantir a qualidade do software. Dentre suas responsabilidades, ele apoia o cliente a escrever testes de aceitação. O testador faz parte da equipe e pode apoiar a automatização dos testes, bem como realizar testes funcionais para minimizar a probabilidade de liberação de software com erros.

Cleaner

O cleaner é o indivíduo responsável por melhorar o código sempre apoiado nas práticas do XP. Ele deve ter como característica a excelência técnica porque precisa enxugar o código sem obviamente perder sua consistência. O cleaner deve encorajar o time a desenvolver códigos mais limpos e com qualidade para minimizar problemas com dívida técnica. A dívida técnica ocorre quando o time não prima pela qualidade do código e posterga ações de melhoria de qualidade. Com o tempo, os problemas vão se acumulando e consumindo a capacidade do time de desenvolver novas funcionalidades, pelo fato de ficarem ocupados corrigindo problemas causados por falta de atenção à qualidade no passado.

Tracker

O tracker é responsável por coletar medições do time durante o projeto. Essas medições servem para avaliar a evolução do time e suas entregas. Por exemplo, a medição da quantidade de histórias de usuários entregues com sucesso. Quando um time começa a entender seu padrão de entregas por iteração (ciclos de entregas), é possível utilizar essa métrica para prever entregas de produtos no futuro.

Gerente

O gerente é um papel que auxilia o time XP com a comunicação com clientes e partes interessadas do projeto. Ele é responsável pela elaboração e manutenção de relatórios sobre a evolução do projeto. Seu papel não é o de chefe do time XP, mas, sim, o de líder servidor, sempre procurando avaliar as necessidades do time para prestar suporte naquilo que for necessário para melhorar o seu desempenho e motivação para a entrega de produtos de qualidade.

Verificando o aprendizado

Questão 1

Carlos é um desenvolvedor em um time que utiliza o método Extreme Programming. Ele desenvolveu um código e está muito orgulhoso do que fez, a ponto de não querer mostrar para ninguém seu código com receio que copiem suas ideias. Qual das práticas listadas abaixo foi desrespeitada no cenário apresentado?

A

Metáforas.

B

Design simples.

C

Padrão de código.

D

Propriedade coletiva do código.

A alternativa D está correta.

O código em equipes Extreme Programming deve ser desenvolvido em par para que não haja propriedade de código. Este deve ser compartilhado para que o time tenha conhecimento que o risco de só uma pessoa conhecer o código seja minimizado.

Questão 2

Os times de desenvolvimento de empresa que utilizam o método ágil Extreme Programming referem-se ao sistema que mais gera receita para a empresa como "mina de ouro" para que todos saibam sobre qual sistema estão falando. Dado o contexto acima, qual prática do Extreme Programming está sendo utilizada?

A

Jogo do planejamento.

B

Pequenas entregas.

C

Metáforas.

D

Ritmo sustentável.

A alternativa C está correta.

As metáforas são utilizadas pelo time para relacionar algo no ambiente de desenvolvimento de sistemas a uma coisa fora deste ambiente. Por exemplo, a lixeira na área de trabalho, a própria área de trabalho, o carrinho de compras em um site de buscas, a lupa como símbolo de buscas. As metáforas auxiliam na criação de uma linguagem comum entre o time para referenciar objetos no sistema.

Scrum

Agora, aprenda um pouco mais sobre o Scrum assistindo ao vídeo abaixo:

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Segundo Sutherland (2016), Ken Schwaber e Jeff Sutherland, dois dos signatários do Manifesto Ágil criaram o Scrum sob influência do artigo de nome *The New New Product Development Game*, de Nonaka e Takeuchi. Neste artigo, eles apresentaram um estudo de equipes de algumas das empresas mais produtivas e inovadoras do mundo: Honda, Fuji-Xerox, 3M, Hewlett-Packard, entre outras. O desenvolvimento de software precisava de uma nova abordagem, pois o ciclo de vida em cascata já não era eficiente para um cenário de complexidade.

Saiba mais

O nome Scrum vem de uma forma de reinício de jogada no rugby, na qual os jogadores dos dois times se juntam e se empurram com o objetivo de ganhar a posse de bola. Esta jogada onde o time trabalha unido com um objetivo comum influenciou a atribuição do nome da jogada para o método ágil.

A primeira apresentação pública formal do Scrum ocorreu quando Ken Schwaber e Jeff Sutherland fizeram uma palestra do Scrum na Conferência OOPSLA de 1995.

> De acordo com o *Scrum Guide* (2017), O Scrum pode ser considerado um framework usado para gerenciar o trabalho em produtos complexos. O Scrum é leve, simples de entender, mas difícil de aplicar.

Teoria do Scrum

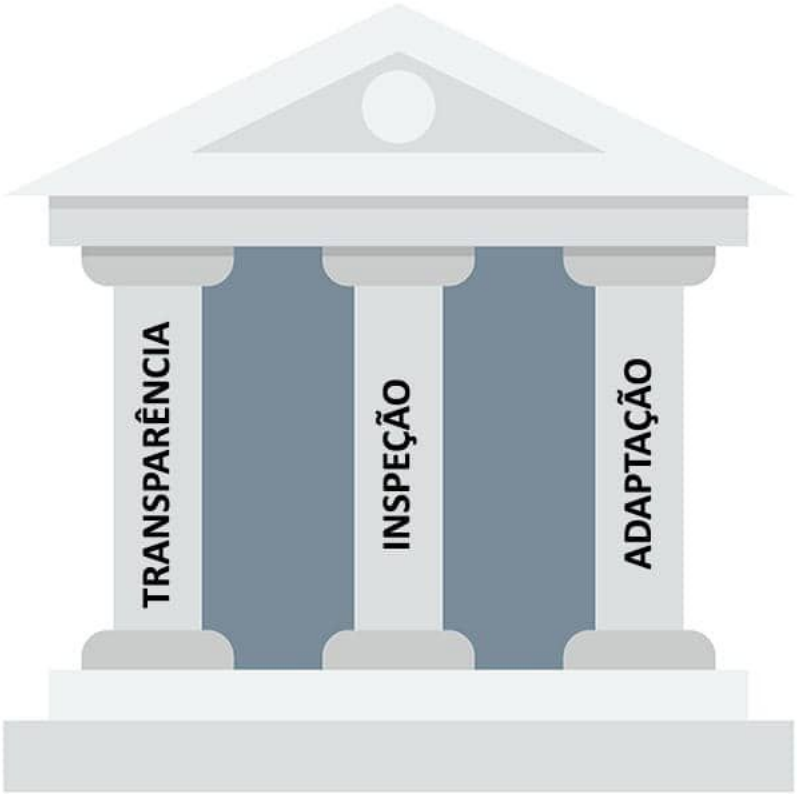
O Scrum é baseado no empirismo para controle de processo, isso quer dizer que o conhecimento vem da experiência. Ao invés de fazermos previsões de entregas sem grande função em ambientes complexos, cuja principal característica é a incerteza, com o empirismo, nós fazemos, aprendemos e aí sim temos a condição de projetar uma entrega futura, mas como norte, como um objetivo a ser buscado, pois não esqueçamos que ainda estamos em contexto complexo com muitas incertezas.

Exemplo

Por exemplo: Para que um time de desenvolvimento forneça uma estimativa para término de um produto, ele precisará conhecer a velocidade média do time nas últimas iterações para poder projetar a previsão.

Pilares

A implementação do controle de processo empírico é apoiada em três pilares.



Transparência

A transparência deve ser perseguida constantemente por todo o time Scrum. Por exemplo, não deve haver dúvida sobre a situação da entrega do incremento de produto ao final de uma iteração (sprint). Todos devem saber o que é um incremento pronto, isso é fornecido pela transparência. Por exemplo, se o Product Owner (PO) reclamar ao final de uma sprint que o produto não está com a qualidade esperada, provavelmente temos um problema de transparência sobre o que é uma entrega pronta. Se houvesse transparência nesse time Scrum, o PO saberia qual era o acordo para entrega pronta. A transparência garante que todos ao final de uma sprint saibam o que é um incremento pronto.

Inspeção

O Scrum, como veremos mais adiante, possui cerimônias que possibilitam a inspeção do incremento do produto e do atingimento da meta da sprint. Por exemplo, em uma reunião diária, o time possui a oportunidade de inspecionar se o trabalho que está sendo realizado está evoluindo no sentido de atingir a meta, e caso não esteja, o time deverá adaptar seu plano para maximizar as chances de atingimento do objetivo. Na reunião de revisão, o time possui outra oportunidade de inspecionar a entrega.

Adaptação

Como vimos nos módulos anteriores, os métodos ágeis são utilizados em ambientes complexos, com muita incerteza. Nesses contextos, é necessário validar rapidamente se estamos no caminho certo e corrigir o rumo. O Scrum não é diferente, o pilar adaptação possibilita a rápida adequação do que não está de acordo. O framework Scrum preconiza iterações de até 30 dias para que possamos ter ciclos de desenvolvimento curtos e consequentemente feedbacks curtos. A resposta rápida sobre o produto propicia a adaptação do mesmo no sentido de se adequar à expectativa do cliente.

Valores

Como vimos, no Extreme Programming os valores denotam o grau de importância de um determinado tema e representam as melhores ações a serem tomadas no contexto deste tema. No Scrum não é diferente, os valores precisam ser vividos e praticados por todos para que possamos criar um ambiente de confiança e, consequentemente, fortalecer os pilares transparência, inspeção e adaptação.

Coragem

Os membros de um time Scrum precisam ter coragem para fazer a coisa certa, fazer o que é preciso. Por exemplo: Informar que não estão confiantes em entregar a meta e apresentar os motivos ou ouvir opiniões mesmo que sejam contrárias às suas.

Foco

O time Scrum durante um ciclo de desenvolvimento mantém o foco nas entregas prioritárias ao negócio. O time procura entregar o que é esperado com simplicidade e objetividade.

Comprometimento

Como o Scrum é um método ágil, e como já vimos que métodos ágeis geralmente são utilizados para problemas complexos, cuja principal característica é o contexto com muitas incertezas, o comprometimento refere-se ao empenho, atitude e engajamento do time com os resultados, mas não necessariamente com o atingimento dos resultados.

Respeito

Este valor refere-se não somente ao evidente respeito que devemos ter com qualquer pessoa com a qual nos relacionamos, mas também respeito para com o cliente ao desenvolver software de qualidade.

Abertura

O time Scrum deve estar aberto a ser transparente constantemente durante a construção do produto; a colaborar sem problemas em ser assertivo para dizer o que precisa ser dito. Abertura para ouvir críticas construtivas que podem melhorar o comportamento profissional, bem como melhorar o processo ou o produto.

Time Scrum

O time Scrum compõe um modelo desenhado para otimizar a flexibilidade, criatividade e produtividade. Este modelo composto de papéis muito bem equilibrados com responsabilidades inerentes às atividades de construção de produto em ambientes complexos.

Os times Scrum são:

- auto-organizáveis - no sentido de se organizarem para desenvolver o produto necessário, sem precisar de chefe ou gerente para dizer como desenvolver as atividades;
- multifuncionais - no sentido que o time deve possuir os perfis necessários para o desenvolvimento do produto de modo iterativo e incremental, maximizando o valor de suas entregas.

Muito embora o time Scrum não possua outros papéis além dos citados acima, o Scrum pode ser utilizado em projetos com um gerente de projetos, por exemplo. O gerente de projetos pode trabalhar com o Product Owner ou Scrum Master para apoiar no relacionamento com o restante da organização; pode criar relatórios de evolução de projetos e pode trabalhar com o time Scrum para criar previsões para entregas de projetos.

Os papéis de um time Scrum são: Product Owner, Scrum Master e Time de Desenvolvimento.

Product Owner (dono do produto)

O Product Owner ou PO é o dono do produto. Ele define, transmite a visão do produto ao time de desenvolvimento e é responsável por representar a visão do produto em um product backlog.

O PO é responsável por gerenciar o product backlog. Ele deve garantir que a necessidade mais premente do negócio esteja refletida no topo da pilha de forma a garantir o retorno sobre o investimento feito no produto. Imaginem que o patrocinador está financiando o produto e espera receber em entregas parciais e frequentes o que mais representa valor naquele momento, por exemplo, uma entrega para atender a uma promoção.

Atenção

O PO deve interagir sempre com stakeholders, como clientes ou departamento de marketing para identificar novas features ou tendências que poderão influenciar o produto. Ele deve também identificar, por exemplo, com o help desk as reclamações dos usuários para retroalimentar o backlog com essas necessidades. Com essa dinâmica, o PO gerencia a entrada e saída dos itens no backlog.

O PO é responsável por definir os PBI (product backlog items) e esclarecer as dúvidas do time de desenvolvimento durante a sprint planning meeting (reunião de planejamento) sobre os itens que entrarão na próxima sprint (ciclo de desenvolvimento).

Recomendação

Ainda nesta reunião, o PO entra em acordo com o time de desenvolvimento sobre o objetivo da sprint. Muito embora a sprint possa conter vários PBI, ela deve possuir um objetivo de entregar um incremento de produto.

O PO é o único papel com autoridade para cancelar uma sprint. Ao detectar que o objetivo, meta da sprint não faz mais sentido para o negócio, o PO pode cancelar a sprint.

Scrum Master (líder Scrum)

O Scrum Master, SM, é quem mais conhece o Scrum dentre todos os papéis do Scrum.

> Ele é um líder servidor e deve apoiar todos no entendimento do Scrum, tanto o time Scrum quanto a organização.

Assim como o PO está focado em criar um product backlog alinhado com as necessidades do cliente para o time de desenvolvimento criar o produto certo, o SM é focado em apoiar o time Scrum a abraçar os valores, princípios e práticas do Scrum, além de remover impedimentos e interagir com stakeholders no sentido de apoiar o uso do Scrum dentro da organização. Isso tudo como líder servidor, sem fazer uso de autoridade.

Atenção

O SM atua para o PO e time de desenvolvimento líder servidor, coach, facilitador dos eventos Scrum, removedor de impedimentos e agente de mudança. Para o PO, por exemplo, o SM pode apoiar nas atividades de criação e refinamento e ordenação do product backlog, que proporcione o desenvolvimento do produto certo; para o time de desenvolvimento, pode atuar fazendo que ele descubra as melhores formas para resolver os problemas encontrados na utilização do Scrum.

DEV Team (time de desenvolvimento)

Um time de desenvolvimento Scrum deve ter entre 3 e 9 componentes.

> Os times precisam ser pequenos para não perderem a capacidade de comunicação e também a auto-organização.

Fórmula de canais de comunicação:

n*(n-1)/2

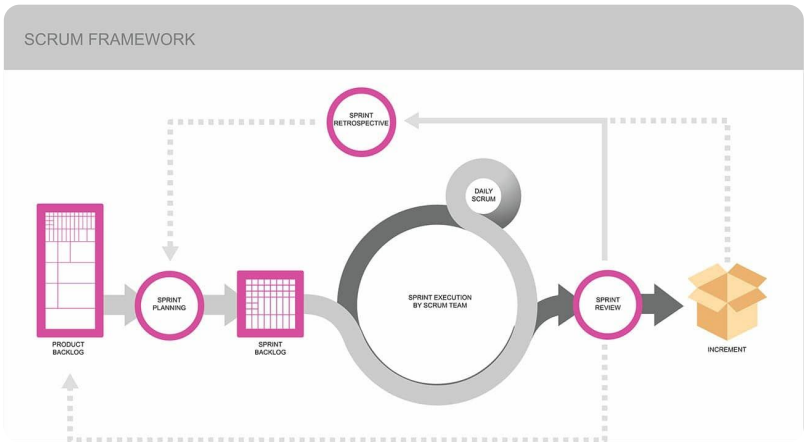
Logo:

Para um time com 5 pessoas, temos: $5*(5-1)/2 = 10$ canais de comunicação, mas se colocarmos uma pessoa a mais no time, temos $6*(6-1)/2 = 15$ canais de comunicação.

Percebam neste exemplo que, ao aumentarmos uma pessoa no time, aumentamos em 50% a quantidade de canais de comunicação.

O time de desenvolvimento auto-organizado não precisa de um gerente para fazer microgestão de atividades. Após a criação do forecast (previsão) da meta da sprint durante a sprint planning o time se auto-organiza para a criação de um plano para desenvolvimento das entregas que compõem a meta da sprint, sem precisar que um gerente diga como fazer.

O time de desenvolvimento pode incrementar o framework Scrum utilizando técnicas de desenvolvimento de software como algumas abordadas no Módulo 2.



Framework scrum.

A figura acima apresenta o ciclo do Scrum com os artefatos: Product Backlog, sprint backlog e incremento, bem como os eventos sprint planning, daily, sprint review e sprint retrospective.

Eventos

O Scrum preconiza eventos que possibilitam oportunidades de transparência e inspeção. Cada evento possui seu propósito claro dentro do Scrum e também um tempo limite (time-boxed), para que não se perca o foco em cada evento.

Sprint planning (reunião de planejamento)

A reunião de planejamento da sprint é o primeiro evento dentro da sprint. Nele o time e o PO debatem sobre a meta da sprint e os PBI, ou seja, os itens que farão parte do backlog da sprint. Esta reunião de time-box dura no máximo oito horas para uma sprint de um mês de duração.

Meta ou objetivo da sprint

O conceito de meta ou objetivo da sprint é bastante relevante para entendermos o que deve ser entrega na sprint. Quando o time Scrum não trabalha com esse conceito, ele atua como um time “tarefeiro” ou seja, preocupa-se com entrega de histórias do sprint backlog (histórias que compõem o escopo da sprint). Desta forma, o PO chega a um acordo com o time sobre quantas histórias eles puxam para a sprint e, ao final, na reunião de revisão, é verificado quantas histórias foram entregues. Já um time orientado à meta define com o PO a meta, o objetivo, ou seja, qual a entrega relevante para o negócio será realizada ao término da sprint. Essa entrega obviamente é composta por histórias de usuário, mas podemos ter dentro da sprint histórias que não fazem parte da meta. Assim, o time deve priorizar a entrega da meta, o compromisso é com a meta. Esta é a diferença de um time orientado a meta e um time orientado a tarefas.

É importante entender que o time possui uma capacidade de trabalho durante a sprint. O trabalho puxado pelo time de desenvolvimento deverá ser factível de ser executado dentro do período da sprint. Percebam que usei o verbo “puxar”. Em agilidade não se empurra trabalho. Isso significa que o time de desenvolvimento avalia os itens com o PO, estima um a um e só puxa do backlog do produto para o backlog da sprint a quantidade de itens que a capacidade de trabalho da equipe comporta realizar com a qualidade acordada (definição de pronto) dentro da duração da sprint.

Estimativa

Não existe um padrão para estimativa, mas uma forma muito usada são os pontos por história. Os pontos por história são uma forma de estimativa relativa na qual o time de desenvolvimento atribui tamanho de cada história de usuário. Exemplo: Para um item simples o time pode atribuir tamanho 1, para um maior pode atribuir tamanho 3. Uma técnica muito utilizada para isso é o planning poker, na qual o time utiliza um baralho com uma numeração, geralmente com a série de Fibonacci e cada membro, após entender o que é para ser feito na história, joga uma carta com uma numeração, sem que os outros vejam para não influenciá-los. Caso haja discrepância na numeração jogada, os membros argumentam sobre os motivos da estimativa de cada e jogam novamente até chegaram ao consenso, com isso o tamanho da história de usuário é atribuído.

Como a duração da sprint é fixa e o time só puxa para o sprint backlog a quantidade de itens que consegue entregar, o time de desenvolvimento começa a conhecer sua velocidade (velocity). A velocidade é uma métrica que serve para estimativas de demandas futuras.

> Uma vez que conhecemos a capacidade de trabalho do time de desenvolvimento em uma sprint, podemos fazer previsões.

Segundo Kenneth (2017), é importante determinar a capacidade do time porque só conhecendo a capacidade poderemos saber o que o time pode entregar.

Após definir os itens que entrarão no backlog da sprint, o time se auto-organiza para definir um plano para atingimento da meta ao final da sprint.

Sprint

> A sprint é o evento que contém outros eventos, ou seja, os outros eventos ocorrem dentro de uma sprint.

A sprint deve ter no máximo um mês de duração e, ao definir esta duração, o ideal é que não seja alterada durante a evolução do produto ou projeto. Este tempo máximo de um mês é para permitir ciclos de feedbacks curtos, para validação da entrega, que possibilitem inspeção e adaptação.

Saiba mais

Assim como não se deve alterar a duração sprint, da mesma forma, não se altera escopo que possa comprometer a meta da sprint, também não se altera o padrão de qualidade acordado para as entregas. A sprint só pode ser cancelada pelo PO se a meta (objetivo) da sprint se tornar obsoleta.

Durante a sprint, geralmente o time utiliza um quadro físico ou virtual para acompanhamento da evolução do trabalho. Não existe regra para a confecção desses quadros, mas normalmente são criadas as colunas: **sprint backlog** para colocar os post-its com as histórias de usuário contidas na sprint backlog; **Fazendo**, para as histórias de usuários que estão sendo feitos e **Feitas** para as histórias de usuário prontas. À medida que as histórias vão sendo desenvolvidas, os post-its são movidos nas

colunas, até todas migrarem para a coluna **Feitas**.

Daily meeting (reunião diária)

É um evento de time-box de no máximo 15 minutos no qual o time de desenvolvimento aproveita a oportunidade para inspecionar a evolução do trabalho em direção ao atingimento da meta da sprint. Neste evento, cada membro do time de desenvolvimento responde:

- O que eu fiz ontem que ajudou a atingir a meta da sprint?
- O que eu farei hoje para ajudar a atingir a meta da sprint?
- Existe impedimento que atrapalhe a mim ou o time de desenvolvimento para atingir a meta da sprint?

A reunião diária é um evento do time de desenvolvimento, logo não há necessidade de mais ninguém participar além do time, à exceção se for convidado pelo time ou o Scrum Master perceber que o time precisa para seguir os valores e pilares do Scrum. O time também pode pedir apoio ao PO para participar e esclarecer algo da sprint backlog que, por ventura, não tenha ficado claro.

Atenção

Obviamente, a meta da sprint é obtida por meio da conclusão das histórias de usuário contidas na sprint, mas o time deve sempre se focar no atingimento da meta. Este é um ponto de atenção, pois pode acontecer de o time se focar em atividades e descolar da meta. A priorização da execução das atividades dentro da sprint deve ser orientada à meta.

Sprint review (revisão da sprint)

É um evento de time-box de no máximo 4 horas para uma sprint de um mês.

> Neste evento, o time de desenvolvimento apresenta para o PO e stakeholders o que foi feito durante a sprint e o PO aprova ou não o incremento do produto que visa a atender a meta da sprint.

Atenção

O Scrum quando é implementado por um responsável sem experiência pode gerar várias disfunções capazes de prejudicar muito o entendimento do framework. Não é difícil encontrarmos no mercado sprints de testes, ou após a Reunião de Revisão da Sprint fazerem a “homologação”. Como disse, são disfunções, porque ao final de uma sprint, deve ser entregue um incremento de produto potencialmente utilizável, ou seja, os itens contidos da Definição de Pronto (ver definição de pronto) devem ser executados. O padrão de qualidade definido deve ter sido atingido para o item ser aceito na reunião de revisão.

Sprint retrospective (retrospectiva da sprint)

É o último evento de uma sprint e possui um time-box de no máximo 3 horas para uma sprint de um mês. Seu propósito é inspeção do próprio time. É comum nesse evento o time colaborar para identificar coisas que deram certo, que deram errado e criar um plano de melhoria para as próximas sprints.

A retrospectiva da sprint é uma oportunidade para o time refletir sobre o processo, sobre o como foi a sprint que se encerra com o evento desta reunião. O Scrum Master deve facilitar esta reunião de modo que o time encontre um ambiente favorável para falar. Muitas vezes, os membros não falam dos problemas para evitarem o conflito, mas o conflito respeitoso é saudável, ele traz foco ao problema e fornece transparência (vejam o pilar transparência novamente) aos pontos de atenção. Detectada a causa raiz, é possível traçar plano de ação para o time corrigir o problema e melhorar continuamente.

Artefatos

Segundo o *Scrum Guide* (2017), os artefatos representam o trabalho ou o valor para o fornecimento de transparência e oportunidades para inspeção e adaptação. Os artefatos definidos para o Scrum são especificamente projetados para maximizar a transparência das informações-chave de modo que todos tenham o mesmo entendimento dos artefatos.

Backlog do produto

Consiste em uma pilha de PBI (product backlog items) que pode conter histórias de usuários, itens técnicos e correções, ordenados por valor, onde o que está no topo da pilha corresponde aos itens com mais valor para o negócio.

O backlog do produto é um artefato em constante mudança. O cliente está cada vez mais exigente tanto em qualidade quanto em redução de tempo para receber seus produtos. Por isso, o Product Owner (PO) deve ficar atento às constantes mudanças de prioridades de negócio exigidas pelo mercado ou clientes e fazer com que essas mudanças sejam refletidas no Product Backlog. Por isso, constantemente o PO deve refinar o product backlog.

Refinamento do backlog do produto

O refinamento é um processo para deixar o backlog sempre com PBI preparados para serem apresentados para a reunião de planejamento da sprint. Enquanto o time de desenvolvimento trabalha na Sprint atual para desenvolver as entregas, o PO deve preparar os PBI candidatos para as próximas sprints. Os PBI candidatos são os prioritários que representam mais valor para o negócio naquele momento.

Backlog da sprint

Consiste nos PBI que foram selecionados do product backlog na reunião de planejamento para compor o trabalho a ser realizado na sprint.

Incremento (increment)

É o produto do trabalho realizado na sprint. Este incremento deve estar pronto segundo o padrão de qualidade estipulado na definição de pronto (DoD). O incremento é um produto potencialmente utilizável após sua aceitação na reunião de revisão da sprint.

Definição de “pronto”

A definição de pronto (definition of done) fornece transparência para o time Scrum e partes interessadas sobre o trabalho da sprint. É uma definição sobre o padrão de qualidade que a entrega possuirá. Na estimativa das histórias, o time de desenvolvimento deve ter em mente o esforço que precisará dispender para aplicar em cada entrega os critérios de qualidade definidos na definição de pronto. Na review, todos sabem que os itens entregues devem atender à definição de pronto.

Como rodar o Scrum

- Pessoas com perfil de SM e PO, devem ser identificadas e, se necessário, capacitadas para exercerem as responsabilidades de cada papel.
- Os especialistas devem decidir os melhores componentes para a composição do time de desenvolvimento.
- O SM deve apoiar o PO na criação do product backlog que represente valor para o negócio.
- O PO deve apresentar as histórias de usuário candidatas a entrarem na sprint, e na reunião de planejamento do time, puxa as histórias de usuário para comporem a sprint backlog.
- O PO e o time devem definir a meta da sprint.
- O time desenvolve o incremento do produto durante a sprint e realiza as reuniões diárias para inspecionar o andamento do desenvolvimento do incremento.
- Ao final da sprint, na reunião de revisão, o time apresenta a entrega para o PO, que pode aceitar ou rejeitar os itens.
- O time Scrum realiza uma reunião de retrospectiva para identificar oportunidade de melhoria no processo.
- O ciclo se repete até a entrega de todo o backlog e, consequentemente, o produto.

Verificando o aprendizado

Questão 1

Em uma reunião de revisão do produto, o dono do produto informa que a entrega não está totalmente diferente do que foi pedido e ainda diz que não sabe do que se trata a entrega.

Dado o contexto acima, qual pilar do Scrum foi desconsiderado?

A

Transparência

B

Respeito

C

Inspeção

D

Adaptação

A alternativa A está correta.

A transparência é um pilar que deve ser perseguido constantemente por todo o time Scrum. A transparência faz com que não existam surpresas na revisão do produto. No cenário acima, se o dono do produto tivesse participado da reunião de planejamento, respondido às perguntas do time, ajudado a elucidar as dúvidas e ainda ficado disponível para o time durante a sprint, provavelmente não ocorreria problema de falta de transparência sobre o que foi entregue.

Questão 2

Qual item abaixo não é um papel do Scrum?

A

Scrum Master.

B

Product Owner.

C

Gerente de projeto.

D

Time de desenvolvimento.

A alternativa C está correta.

O time Scrum é composto dos seguintes papéis: Scrum Master, Product Owner e Time de Desenvolvimento. Isso não quer dizer que em um projeto não possamos ter um gerente de projeto, mas ele não é um papel preconizado no Scrum.

Considerações finais

Durante nossa jornada, vimos um marco importante da criação do conceito da Agilidade, o Manifesto Ágil. Assinado por 17 expoentes do ramo de desenvolvimento de software, os valores e princípios do Manifesto apontam a necessidade de pensarmos primeiramente nas pessoas, tanto as que desenvolvem produtos, quanto as que usam os produtos; a necessidade de colaborarmos com o cliente; a necessidade de frequentemente validarmos se o que está sendo entregue está alinhado com as expectativas do cliente, entre outras.

Na sequência, vimos o método ágil de desenvolvimento Extreme Programming (XP), que também possui seus valores, mas inteiramente aderentes aos valores do Manifesto. Estudamos as práticas preconizadas pelo XP que apoiam o desenvolvimento de software com qualidade. Por fim, vimos o método ágil Scrum, seus valores, pilares, papéis, responsabilidades, eventos e artefatos.

Podcast

Para encerrar, ouça sobre projeto ágil.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore+

Para saber mais sobre os assuntos tratados neste tema, pesquise na internet:

- Scrum e XP direto das Trincheiras, Infoq.

Leia:

- Aplicação do método ágil Scrum no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica, Scielo.
- Uma estratégia baseada em aquisição de conhecimento para o gerenciamento de riscos nos requisitos em um desenvolvimento XP distribuído, Scielo.
- Orientações para o processo de certificação SCM (Scrum Master Certified) da Scrum Alliance.
- Orientações para o processo de certificação PSM (Professional Scrum Master) da Scrum.org.
- Orientações para o processo de certificação EXIN Agile Scrum Foundation.

Referências

AGILE MANIFESTO. Agile Manifesto. Consultado em meio eletrônico em: 29 set. 2020.

JEFFRIES, R. What is extreme programming? In: Ronjeffries. Publicado em: 16 mar. 2011.

KENNETH, S. R. Scrum essencial: Um guia prático para o mais popular processo ágil. Traduzido por Roberto Rezende. Rio de Janeiro: Alta Books, 2017.

MASSARI, V. Gerenciamento Ágil de Projetos. Rio de Janeiro: Brasport, 2014.

SCRUM GUIDES. Guia do Scrum. Publicado em: out. 2017.

SUTHERLAND, J. Scrum: A arte de fazer o dobro do trabalho na metade do tempo. São Paulo: LeYa, 2016.

TELES, V. M. Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade. Rio de Janeiro: Novatec, 2004.

WILDT, D. eXtreme Programming: Práticas para o dia a dia no desenvolvimento ágil de software. São Paulo: Casa do Código, 2015.