Estratégia Concursos

Aula 00

CVM (Analista TI - Sistemas e Desenvolvimento - Perfil 8) Passo Estratégico de Conhecimentos Específicos - 2024 (Pós-Edital)

Autor:
Fernando Pedrosa Lopes

07 de Fevereiro de 2024

MÉTODOS ÁGEIS - XP, SCRUM E KANBAN

Sumário

CONTEÚDO

Método Ágeis. Manifesto Ágil. Princípios, conceitos e valores. XP. Práticas. Scrum. Papeis. Cerimônias. Artefatos. Kanban. Princípios e conceitos.

ANÁLISE ESTATÍSTICA

Inicialmente, convém destacar o percentual de incidência do assunto, dentro da disciplina **Engenharia de Software** em concursos/cargos similares. Quanto maior o percentual de cobrança de um dado assunto, maior sua importância.

Obs.: *um mesmo assunto pode ser classificado em mais de um tópico devido à multidisciplinaridade de conteúdo*.

Assunto	Relevância na disciplina em concursos similares
UML	11.0 %
Processos de Software - Desenvolvimento Ágil	8.4 %
Engenharia de Requisitos	8.2 %
Teste de Software	6.7 %
Métricas de Software	5.1 %
Desenvolvimento de Software	4.6 %
Inteligencia Artificial	4.6 %
Processos de Software	4.6 %
Orientação a Objetos	4.5 %
Gestão de Projetos em Engenharia de Software	3.6 %
Qualidade de Software	3.4 %
Metodologia de desenvolvimento de software	2.5 %
Gerência de Configuração	1.9 %
Geoprocessamento em Engenharia de Software	1.6 %
Prototipação	1.2 %
Conceitos Básicos em Engenharia de Software	0.7 %
Análise Estruturada	0.6 %
Ferramentas CASE	0.4 %
Ferramentas de Desenvolvimento de Software	0.4 %
Software livre	0.3 %

Manutenção de Software	0.1 %
Análise Essencial	0.1 %
Web 2.0	0.1 %
Portal Web	0.1 %
Refatoração	0.1 %
Engenharia da Informação	0.1 %

GLOSSÁRIO DE TERMOS

Faremos uma lista de termos que são relevantes ao entendimento do assunto desta aula. Caso tenha alguma dúvida durante a leitura, esta seção pode lhe ajudar a esclarecer.

Método Ágil: Método de desenvolvimento de software que valoriza a entrega de valor contínuo, a colaboração com os clientes, a adaptabilidade e a melhoria contínua.

Manifesto Ágil: Documento que estabelece os quatro valores e doze princípios do desenvolvimento de software ágil.

XP (Extreme Programming): Método ágil que enfatiza a excelência técnica, a comunicação direta e a satisfação do cliente.

História do usuário: Descrição de uma funcionalidade do ponto de vista do usuário, geralmente no formato: "Como [tipo de usuário], quero [alguma ação], para que eu possa [algum benefício ou resultado]".

Práticas do XP: Conjunto de práticas recomendadas para equipes que utilizam XP, incluindo programação em pares, desenvolvimento orientado a testes e integração contínua.

Ciclo de vida do XP: Ciclo de atividades em um projeto XP, incluindo planejamento, design, codificação e teste.

Scrum: Método ágil que utiliza ciclos fixos de trabalho, chamados Sprints, e enfatiza a transparência, inspeção e adaptação.

Product Owner: No Scrum, o Product Owner é a pessoa responsável por maximizar o valor do produto, gerenciando e priorizando o Product Backlog.

Developer: Membro da equipe Scrum que trabalha para entregar um incremento potencialmente entregável do produto ao final de cada Sprint.

Scrum Master: Papel no Scrum responsável por promover e suportar o Scrum, ajudando todos a entenderem a teoria, práticas, regras e valores do Scrum.

Product Backlog: Lista priorizada de itens ou funcionalidades desejadas para um produto, mantida pelo Product Owner.

Sprint Backlog: Conjunto de itens do Product Backlog selecionados para a Sprint, juntamente com um plano para entregá-los.

Product Increment: Versão utilizável e potencialmente entregável do produto ao final de cada Sprint.

Burndown Chart: Gráfico que mostra a quantidade de trabalho restante em relação ao tempo.

Sprint: Ciclo fixo de trabalho no Scrum, geralmente durando 2-4 semanas.

Sprint Planning: Reunião no início de cada Sprint para planejar o trabalho que será realizado.

Daily Scrum: Reunião diária de 15 minutos para a equipe sincronizar o progresso e planejar o trabalho do dia.

Sprint Review: Reunião no final de cada Sprint para inspecionar o Incremento e adaptar o Product Backlog.

Sprint Retrospective: Reunião no final de cada Sprint para a equipe refletir e planejar melhorias para a próxima Sprint.

Definition of Ready: Conjunto de critérios que um item do backlog deve atender antes que possa ser selecionado para uma Sprint.

Definition of Done: Conjunto de critérios que um item deve atender para ser considerado completo.

Planning Poker: Técnica usada para estimar o esforço de trabalho, onde cada membro da equipe "aposta" com um valor que representa a sua estimativa de esforço.

Kanban: Método ágil que visualiza o fluxo de trabalho, limita o trabalho em andamento e incentiva a melhoria contínua.

WIP (Work in Progress): Quantidade de trabalho atualmente em andamento. No Kanban, geralmente há limites no WIP para evitar sobrecargas.

ROTEIRO DE REVISÃO

A ideia desta seção é apresentar um roteiro para que você realize uma revisão completa do assunto e, ao mesmo tempo, destacar aspectos do conteúdo que merecem atenção.

Introdução às Metodologias Ágeis e o Manifesto Ágil

Métodos ágeis são uma categoria de metodologias de desenvolvimento de software que se baseiam em princípios **de flexibilidade, colaboração, melhoria contínua e entrega de valor** ao **cliente de forma rápida e eficiente**. Eles foram formalmente introduzidos no Manifesto Ágil em 2001 e incluem estruturas como Scrum, Kanban, e Extreme Programming (XP), entre outros.

Métodos Ágeis diferem de metodologias tradicionais (ou 'em cascata') de desenvolvimento de software de várias maneiras, como podemos ver na tabela a seguir:

	Métodos Ágeis	Metodologias Tradicionais
Flexibilidade	Aceitam e esperam mudanças ao longo do projeto.	São mais rígidas com pouca flexibilidade para mudanças posteriores.
Processo	Iterativo e incremental, com ciclos curtos (sprints).	Linear, com fases sequenciais (análise, design, implementação, teste, manutenção).
Entrega	Valor entregue de forma contínua ao cliente.	Geralmente, um único ponto de entrega, no final do projeto.
Organização	Equipes auto-organizadas e colaborativas.	Estrutura mais hierárquica e gerencial.
Relação com o Cliente	Colaboração contínua e resposta às necessidades do cliente.	Foco em cumprir as especificações do contrato acordado inicialmente.

Histórico, Evolução e Manifesto Ágil

Os métodos ágeis surgiram no final da década de 1990 e início dos anos 2000 como uma reação às limitações percebidas das metodologias de desenvolvimento de software tradicionais, muitas vezes chamadas de abordagens em cascata ou baseadas em planos (*plan-driven*). Esses métodos tradicionais foram vistos como excessivamente burocráticos, inflexíveis e lentos, dificultando a adaptação rápida às mudanças nos requisitos do projeto e às condições do mercado.

Diversos métodos ágeis diferentes foram desenvolvidos independentemente durante esse período, incluindo Scrum (Jeff Sutherland e Ken Schwaber, 1995), Extreme Programming (XP, Kent Beck, 1996), Crystal Clear (Alistair Cockburn, 1997), Adaptive Software Development (Jim Highsmith, 2000) e outros. Embora esses métodos diferissem em detalhes, eles compartilhavam uma ênfase comum na flexibilidade, na entrega rápida e frequente de software funcional, na colaboração próxima com os clientes e na capacidade de se adaptar a mudanças.

Em 2001, 17 proeminentes pensadores de software se reuniram em Snowbird, Utah, para discutir essas novas abordagens de desenvolvimento. Nesse encontro, eles formularam e assinaram o **Manifesto para o Desenvolvimento Ágil de Software**, mais conhecido como o Manifesto Ágil. O Manifesto Ágil articulava quatro valores fundamentais:

Indivíduos e interações mais do que processos e ferramentas.

Software funcionando mais do que documentação abrangente.

Colaboração com o cliente mais do que negociação de contratos.

Responder a mudanças mais do que seguir um plano.

Indivíduos e interações mais do que processos e ferramentas

Este valor enfatiza a **importância das pessoas e da maneira como elas se comunicam** e colaboram em um projeto ágil.

Isso significa que, embora os processos e as ferramentas sejam importantes para organizar o trabalho e manter a eficiência, eles não devem ser vistos como mais importantes do que as pessoas envolvidas no projeto e a maneira como elas interagem umas com as outras. O foco deve

ser na criação de um ambiente onde indivíduos talentosos possam se comunicar de maneira eficaz, colaborar e resolver problemas juntos.

Além disso, o valor reflete a crença de que o sucesso de um projeto depende muito mais da equipe e de como eles trabalham juntos do que dos processos ou ferramentas que eles usam. As melhores ferramentas e processos não podem compensar uma equipe desorganizada ou que não colabora, enquanto uma equipe forte e colaborativa pode superar limitações em ferramentas ou processos.

Por exemplo, em vez de confiar excessivamente em ferramentas de gerenciamento de projetos para coordenar o trabalho, uma equipe ágil pode se beneficiar mais de reuniões diárias em pé (também conhecidas como stand-ups) onde cada membro da equipe pode discutir brevemente o que eles estão trabalhando, quaisquer problemas que estão enfrentando, e como eles planejam avançar.

Vale ressaltar que este valor do Manifesto Ágil não sugere que processos e ferramentas sejam irrelevantes. Em vez disso, a ideia é que eles não devem ser considerados mais importantes do que as pessoas e as interações na condução do sucesso do projeto.

Software funcionando mais do que documentação abrangente

Este valor reflete a ênfase das metodologias ágeis na **entrega de um produto útil ao cliente**, em vez de se concentrar na criação de documentação extensa.

Nas metodologias de desenvolvimento de software tradicionais, é comum haver uma grande quantidade de documentação produzida em cada fase do processo. Isso pode incluir requisitos de software, especificações de design, planos de teste, manuais do usuário e mais. Embora a documentação possa ser útil para a comunicação e para rastrear o que foi feito, ela não fornece valor direto para o cliente se não estiver acompanhada de software funcional. A documentação por si só não resolve os problemas do cliente.

Métodos ágeis argumentam que é melhor se concentrar em entregar software funcional - isto é, um produto que realmente atenda às necessidades do cliente e resolva seus problemas. O tempo e os recursos dedicados à documentação devem ser equilibrados com a necessidade de entregar software funcional. Isso não significa que a documentação seja negligenciada em um ambiente ágil, mas que ela é minimizada para o que é realmente necessário para o projeto.

Tudo isso não significa que a documentação seja completamente dispensável. Uma documentação adequada ainda é necessária, especialmente para a manutenção e o entendimento do sistema a longo prazo. A diferença é que, em uma abordagem ágil, a

documentação é geralmente produzida de maneira "sob demanda" e é orientada para a necessidade real, em vez de ser produzida por padrão ou por hábito.

Colaboração com o cliente mais do que negociação de contratos

Este valor destaca a **importância de trabalhar em estreita colaboração com o cliente** durante todo o processo de desenvolvimento de software, em vez de confiar em acordos contratuais rígidos que definem o escopo do trabalho desde o início.

Em muitas abordagens tradicionais de desenvolvimento de software, uma grande quantidade de tempo é gasta na definição de um contrato que detalha exatamente o que será entregue, quando será entregue e quanto isso custará. Enquanto isso pode fornecer uma sensação de segurança, também pode limitar a flexibilidade e a capacidade de adaptação à medida que os requisitos mudam ou novas informações são descobertas.

No contexto ágil, acredita-se que é muito mais eficaz colaborar estreitamente com o cliente durante todo o projeto. Isso significa envolver o cliente em discussões de planejamento, revisões de trabalho e decisões de priorização. Dessa forma, a equipe de desenvolvimento pode entender melhor as necessidades do cliente e se adaptar rapidamente às mudanças.

Isso não significa que os contratos são irrelevantes no desenvolvimento ágil. Eles ainda são necessários para definir o relacionamento geral e as expectativas entre a equipe de desenvolvimento e o cliente. No entanto, os contratos ágeis tendem a ser mais flexíveis e abertos à mudança, em vez de tentar fixar todos os detalhes desde o início.

A chave é que a colaboração contínua com o cliente é vista como uma maneira muito mais eficaz de garantir que a equipe de desenvolvimento está construindo o que o cliente realmente precisa, em vez de apenas o que foi especificado no contrato inicial.

Responder a mudanças mais do que seguir um plano

Este valor enfoca a capacidade de ser **flexível e adaptável ao longo do projeto**, **em vez de se apegar rigidamente a um plano inicial** que pode não ser mais relevante ou útil à medida que o projeto avança.

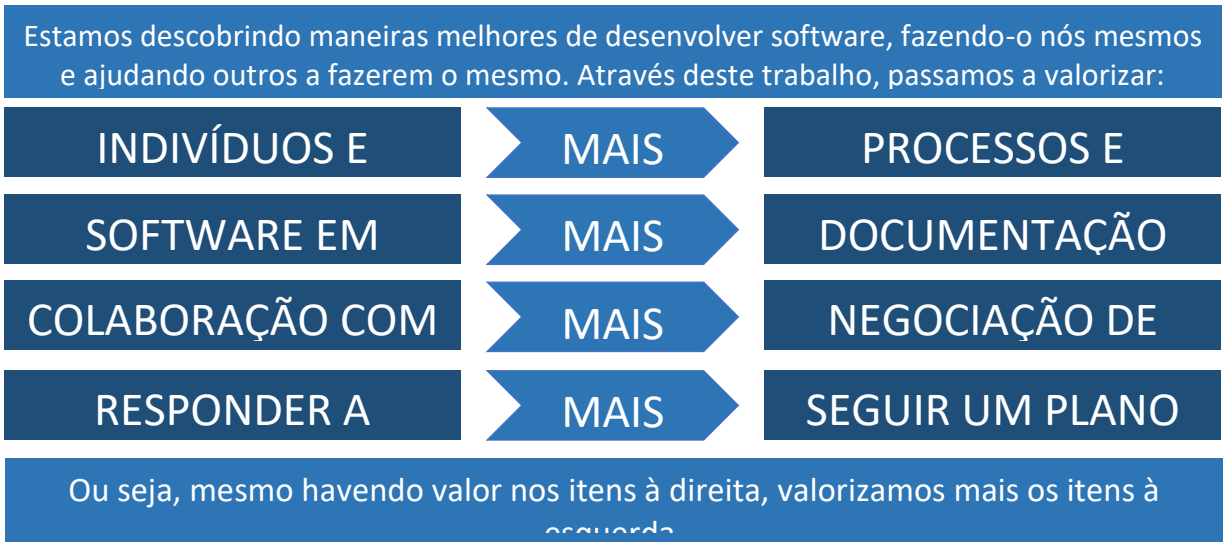
No desenvolvimento de software tradicional, geralmente há um esforço para planejar detalhadamente o projeto desde o início. O escopo do trabalho, o cronograma e os recursos necessários são definidos em detalhes e o projeto segue esse plano. No entanto, esse tipo de

abordagem pode ser problemático, pois raramente é possível prever todas as eventualidades e dificuldades que podem surgir durante um projeto.

Em contraste, as metodologias ágeis reconhecem que as circunstâncias podem mudar durante o curso do projeto - os requisitos do cliente podem evoluir, novas tecnologias podem se tornar disponíveis, os membros da equipe podem mudar, e assim por diante. Em vez de tentar resistir a essas mudanças, os métodos ágeis propõem que as equipes devem estar preparadas para responder e se adaptar a essas mudanças.

Este valor não significa que o planejamento seja irrelevante no desenvolvimento ágil. De fato, o planejamento é uma atividade contínua e crucial nas metodologias ágeis. O que é diferente é que os planos são vistos como sendo flexíveis e mutáveis, em vez de fixos. Em outras palavras, os planos servem como um guia, mas não são um conjunto fixo de instruções que devem ser seguidas, independentemente do que aconteça.

A ideia subjacente aqui é que a capacidade de responder eficazmente às mudanças é um fator mais importante para o sucesso do projeto do que a capacidade de criar e seguir um plano detalhado. Isso é particularmente relevante em ambientes incertos ou de rápido movimento, onde a capacidade de se adaptar rapidamente pode ser uma vantagem competitiva crucial.

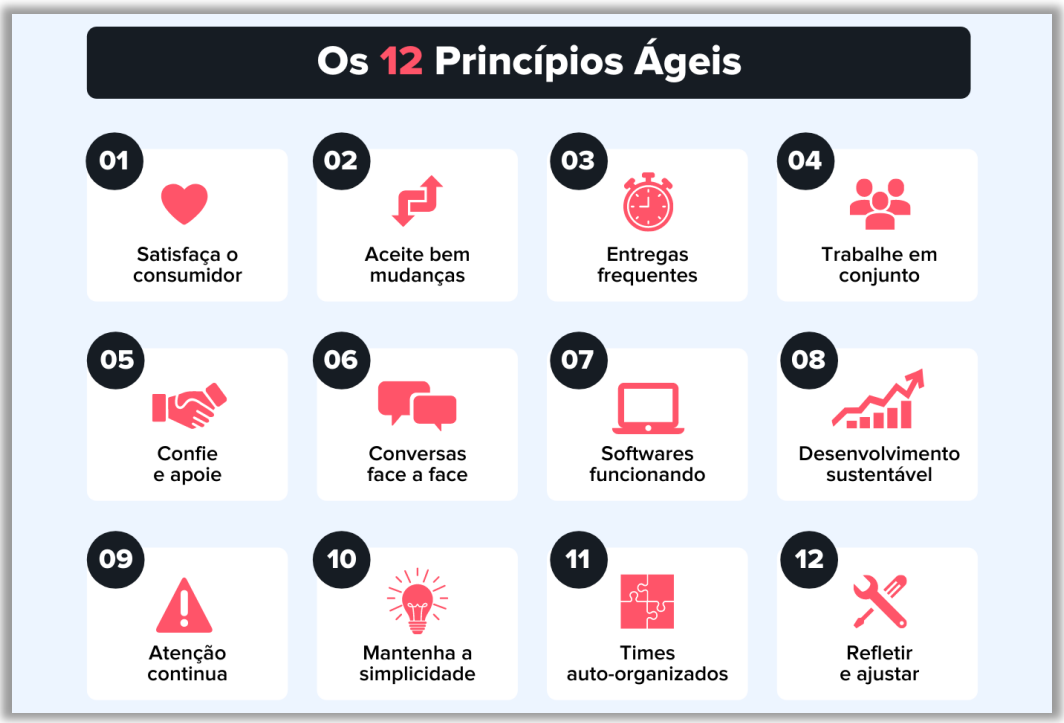


Os 12 princípios

Os princípios do Manifesto refletem a filosofia ágil que enfatiza a colaboração, a adaptabilidade, a entrega contínua de valor e a satisfação do cliente. São eles:

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 em desenvolvimento. Os processos ágeis aproveitam a mudança para proporcionar vantagem competitiva ao cliente.
3. Entregar frequentemente software funcionando, com preferência à menor escala de tempo possível.
4. Colaboração diária cara a cara com os envolvidos no projeto é o meio mais eficiente e eficaz de transmitir informações.
5. Projetos são construídos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessários 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 a conversa cara a cara.
7. Software funcionando é a principal medida de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Atenção contínua 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, então, sintoniza e ajusta seu comportamento de acordo.



Hoje em dia, Métodos Ágeis são uma grande "família" de processos de gestão e desenvolvimento de software. Em nossa aula, veremos os métodos XP, Scrum e Kanban, que hoje em dia são os

mais utilizados pela indústria e mais cobrados em provas. Como exemplo, a tabela a seguir mostra algumas das principais metodologias ágeis existentes atualmente:

Principais METODOLOGIAS ÁGEIS		
SCRUM	CRYSTAL	XP
TDD	ATDD	BDD
FDD	DDD	MDD
DSDM	ASD	KANBAN
LEAN	AUP	AGILE MODELING
OSSD	SCRUMBAN	BADM

Extreme Programming (XP)

Extreme Programming (XP) é uma metodologia de desenvolvimento de software ágil que tem como foco a produção de software de alta qualidade e a capacidade de se adaptar a mudanças de requisitos. Foi criada por Kent Beck, Ward Cunningham e Ron Jeffries durante o projeto Chrysler Comprehensive Compensation System (C3) no início dos anos 90.

O XP se destaca pela abordagem que dá ao desenvolvimento de software, que enfatiza a comunicação contínua com e entre os desenvolvedores, bem como a simplicidade no design de software e a obtenção de feedback rápido do cliente.

Por que utilizar o termo "*extremo*"? Porque XP recomenda que as boas práticas sejam levadas ao extremo. Por exemplo:

- **Testar é bom?** Então vamos testar toda hora (prática de TDD).
- **Integrar é bom**? Então vamos integrar toda hora (prática de integração contínua).
- **Melhorar o código é bom**? Então vamos melhorar o código sempre que possível (prática de refactoring).

Os princípios fundamentais do XP são:

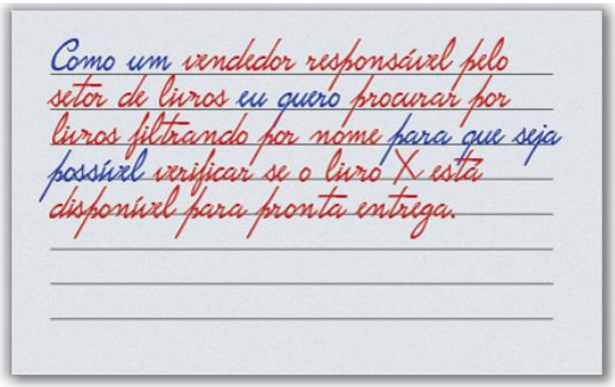
1. **Comunicação**: XP enfatiza a comunicação face a face e a colaboração, tanto entre os desenvolvedores quanto com os clientes.
2. **Simplicidade**: XP incentiva a criação do design mais simples possível para atender aos requisitos atuais e a evitar o excesso de planejamento para o futuro. Isso facilita as mudanças quando elas são necessárias.

3. **Feedback**: XP usa várias práticas para fornecer feedback rápido e constante sobre o projeto. Isso inclui testes contínuos, revisões de código e a entrega frequente de versões funcionais do software para os clientes.
4. **Coragem**: XP enfatiza a disposição de fazer mudanças no projeto, mesmo quando isso é difícil. Isso inclui a mudança de requisitos, a refatoração de código para melhorá-lo, ou a disposição de abandonar práticas que não estão funcionando.
5. **Respeito**: Cada membro da equipe deve respeitar os outros e suas contribuições. XP promove um ambiente onde todos os membros da equipe se sentem valorizados e são encorajados a contribuir com suas habilidades e perspectivas únicas.

Histórias do Usuário

Histórias do usuário são uma técnica utilizada em metodologias ágeis de desenvolvimento de software, como Scrum e Extreme Programming, para **capturar uma descrição de alto nível de um requisito do sistema a partir da perspectiva do usuário final**. Elas são uma maneira simples de capturar o que o usuário precisa fazer sem especificar como o sistema deve fornecer a funcionalidade.

Uma história do usuário normalmente segue um formato simples: "Como um [tipo de usuário], eu quero [objetivo] para que eu possa [benefício/valor]". Isso ajuda a manter o foco na perspectiva do usuário e no valor que ele obterá do sistema.



Histórias do usuário servem a vários propósitos:

Foco no usuário: Ao se concentrar no que o usuário precisa alcançar, elas ajudam a garantir que o sistema seja desenvolvido a partir da perspectiva do usuário.

Comunicação e colaboração: Histórias do usuário são uma maneira eficaz de facilitar a discussão entre os membros da equipe e com o cliente. Elas são fáceis de entender e servem como um ponto de partida para conversas sobre o que o sistema precisa fazer.

Flexibilidade: Histórias do usuário são de alto nível e não especificam uma solução, o que dá à equipe de desenvolvimento a flexibilidade para determinar a melhor maneira de implementar a funcionalidade.

Planejamento e priorização: Podem ser estimadas e priorizadas para ajudar no planejamento do projeto. Elas também podem ser divididas em tarefas menores quando a equipe está pronta para começar a trabalhar nelas.

Testabilidade: Cada história do usuário pode ser associada a um ou mais critérios de aceitação, que são usados para confirmar que a história foi implementada corretamente.

Veja exemplos de histórias de usuário de um sistema bancário:

Pagamento – Boleto Como um Cliente, quero utilizar a forma de pagamento Boleto Bancário Para pagar meus pedidos.	Pagamento – Débito em Conta – Banco 1 Como um Cliente, quero utilizar a forma de pagamento Débito em Conta do Banco 1 Para pagar meus pedidos.	Pagamento – Débito em Conta – Banco 2 Como um Cliente, quero utilizar a forma de pagamento Débito em Conta do Banco 2 Para pagar meus pedidos.
Pagamento – Cartão de Crédito – Bandeira A Como um Cliente, quero utilizar a forma de pagamento Cartão de Crédito da Bandeira A Para pagar meus pedidos.	Pagamento – Cartão de Crédito – Bandeira B Como um Cliente, quero utilizar a forma de pagamento Cartão de Crédito da Bandeira B Para pagar meus pedidos.	Pagamento – Cartão de Crédito – Bandeira C Como um Cliente, quero utilizar a forma de pagamento Cartão de Crédito da Bandeira C Para pagar meus pedidos.

Práticas do XP

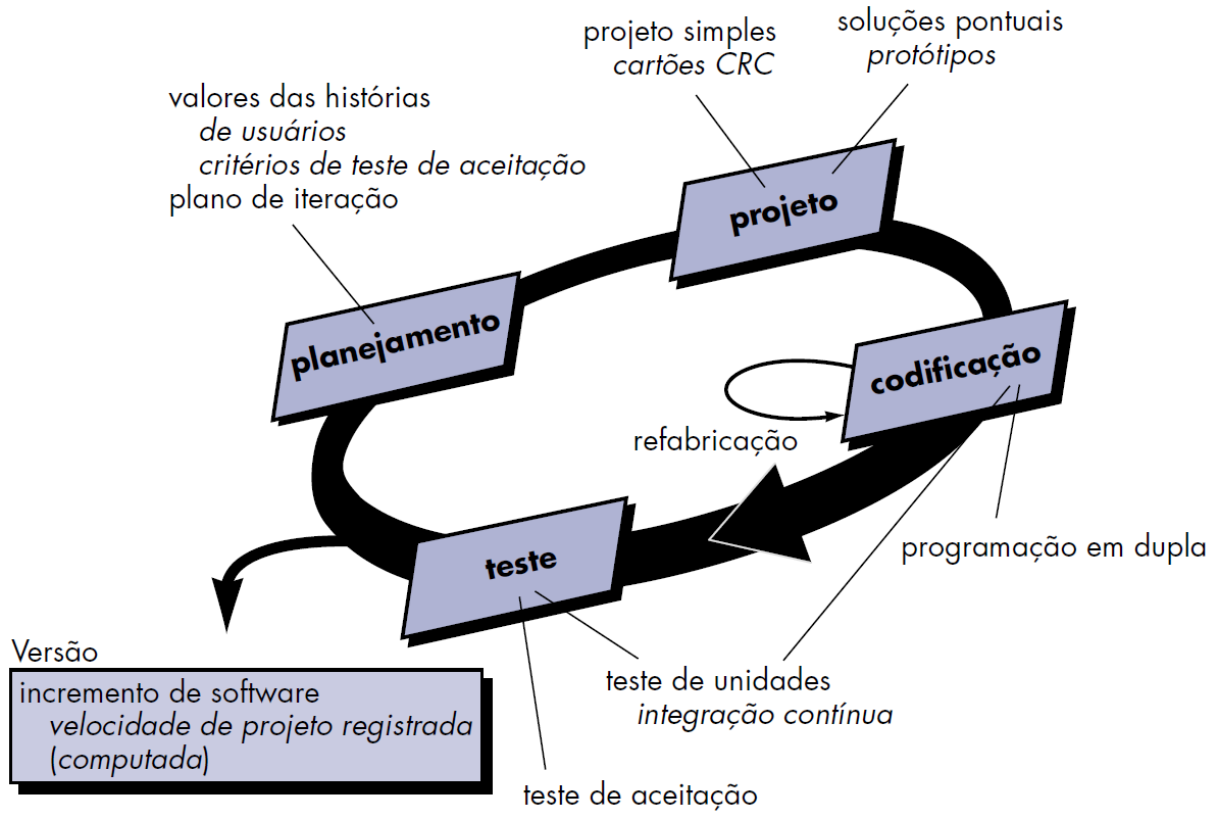
Extreme Programming possui um conjunto de práticas, a maioria focadas em Engenharia de Software, para aplicar os princípios da metodologia. A tabela a seguir lista as principais práticas do XP:

PRÁTICA	DESCRIÇÃO
Planejamento Incremental	Os requisitos são registrados em cartões de histórias e as histórias a serem incluídas em um release são determinadas pelo tempo disponível e sua prioridade relativa. Os desenvolvedores dividem essas histórias em tarefas.
Pequenos Releases	O conjunto mínimo útil de funcionalidade que agrega valor ao negócio é desenvolvido primeiro. Releases do sistema são frequentes e adicionam funcionalidade incrementalmente ao primeiro release.
Projeto Simples	É realizado um projeto suficientemente simples de modo que atenda aos requisitos atuais e nada mais. Deve-se lembrar que um código simples não é código fácil (KIS – Keep It Simple).
Desenvolvimento Test-First	Um framework automatizado de teste unitário é usado para escrever os testes para uma nova parte da funcionalidade antes que esta seja implementada. Portanto, primeiro se escreve o teste, depois faz-se a implementação.
Refactoring	Espera-se que todos os desenvolvedores recriem o código continuamente tão logo os aprimoramentos do código forem encontrados. Isso torna o código simples de entender e fácil de manter.
Programação em Pares	Os desenvolvedores trabalham em pares, um verificando o trabalho do outro e fornecendo apoio para realizar sempre um bom trabalho. Eles utilizam o mesmo mouse, teclado e monitor.
Propriedade Coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de tal maneira que não se formem ilhas de conhecimento, com todos os desenvolvedores de posse de todo o código. Qualquer um pode mudar qualquer coisa.
Integração Contínua	Tão logo o trabalho em uma tarefa seja concluído, este é integrado ao sistema como um todo. Depois de qualquer integração, todos os testes unitários do sistema devem ser realizados.
Ritmo Sustentável	Grandes quantidades de horas extras não são consideradas aceitáveis, pois, no médio prazo, há uma redução na qualidade do código e na produtividade. Trabalhar por longos períodos se torna contraproducente – recomenda-se 40 horas semanais.
Metáforas	A equipe se comunica sobre o desenvolvimento de software por meio de metáforas, caso consiga encontrar uma que realmente faça sentido dentro do contexto e possa facilitar a comunicação.
Cliente On-site	Um representante do usuário final do sistema deve estar disponível em tempo integral para apoiar a equipe. No XP, o cliente é um membro da equipe de desenvolvimento e é responsável por trazer os requisitos do sistema.
Reuniões em Pé	Reuniões são realizadas em pé para não se perder o foco nos assuntos, produzindo reuniões mais rápidas, somente abordando as tarefas realizadas e tarefas a realizar pela equipe no futuro.
Time Coeso	A equipe de desenvolvimento é formada por pessoas engajadas e multidisciplinares, i.e., elas possuem habilidades para diversas áreas do projeto.

Jogo do Planejamento	O planejamento de um release e das iterações são feitos com base nas histórias e conta com a colaboração de toda equipe de desenvolvimento, inclusive o cliente, divididos em papeis: negócio e técnico. Os clientes priorizam e os desenvolvedores avaliam e estimam.

Processo (Ciclo de Vida) do XP

O ciclo de vida de um projeto utilizando XP envolve várias fases que se repetem continuamente durante o desenvolvimento do software. As principais etapas desse ciclo incluem planejamento, projeto, codificação e teste.



Planejamento:

No início de cada iteração (geralmente duração de uma a duas semanas), a equipe realiza uma reunião de planejamento para determinar quais histórias do usuário serão trabalhadas durante esse período. Isso é feito em estreita colaboração com o cliente ou representante do cliente, que ajuda a priorizar as histórias de acordo com o valor de negócio. A equipe então estima o esforço necessário para cada história, que ajuda a determinar quantas histórias podem ser concluídas na iteração.

Projeto:

Na fase de projeto, a equipe decide a melhor maneira de implementar as histórias selecionadas. Isso pode envolver a criação de diagramas ou outras representações visuais para ajudar a entender como o código deve ser estruturado. XP enfatiza um design simples que faz exatamente o que é necessário e nada mais. Também promove a melhoria contínua do design através de refatoração, onde o código é continuamente revisado e melhorado.

Codificação:

Durante a fase de codificação, a equipe implementa as histórias do usuário. Uma prática chave de XP é a programação em pares, onde dois desenvolvedores trabalham juntos em um computador. Isso ajuda a melhorar a qualidade do código e facilita a transferência de conhecimento. Além disso, o XP enfatiza a integração contínua, onde as alterações no código são testadas e combinadas com o código principal frequentemente.

Teste:

A fase de teste é crucial em XP e ocorre em paralelo com a codificação. A equipe escreve testes antes de escrever o código (uma prática conhecida como desenvolvimento orientado a testes ou TDD). Os testes são executados sempre que o código é alterado para garantir que tudo ainda esteja funcionando corretamente. Isso ajuda a identificar e corrigir problemas rapidamente e dá à equipe a confiança para fazer mudanças.

Essas fases não são estritamente sequenciais e podem ocorrer simultaneamente ou se sobrepor. Por exemplo, a equipe pode começar a planejar a próxima iteração antes de terminar a codificação da iteração atual. Da mesma forma, o teste ocorre durante todo o processo de codificação, e o design é uma atividade contínua que ocorre sempre que a equipe está codificando.

Além dessas fases, o XP também enfatiza a importância da revisão e adaptação contínuas. A equipe deve regularmente refletir sobre seu desempenho e procurar maneiras de melhorar. Isso pode envolver a adaptação de suas práticas ou a introdução de novas práticas para resolver quaisquer problemas que tenham sido identificados.

Scrum

Scrum é um dos métodos ágeis mais amplamente utilizados atualmente, talvez o maior. Foi inicialmente apresentado por Ken Schwaber e Jeff Sutherland em um artigo em 1995, mas as

ideias e conceitos por trás dele foram inspirados por um artigo de 1986 na Harvard Business Review escrito por Hirotaka Takeuchi e Ikujiro Nonaka.

O artigo de Takeuchi e Nonaka comparou equipes de alta performance a um scrum de rugby, onde a bola é passada para frente e para trás dentro da equipe, em vez de ser passada de um departamento para outro como em um jogo de revezamento. Eles argumentaram que as empresas de sucesso eram aquelas que usavam este estilo "scrum" de trabalho em equipe e de desenvolvimento de produtos.

Schwaber e Sutherland adotaram este conceito e formalizaram o método Scrum, que tem evoluído ao longo dos anos e é agora usado por equipes de desenvolvimento de software em todo o mundo.

De maneira geral, Scrum é uma estrutura simples que enfatiza a colaboração, a adaptabilidade e a entrega contínua de valor. Embora tenha sido originalmente concebido para o desenvolvimento de software, tem sido adaptado e utilizado em muitos outros tipos de projetos.

Veja a definição oficial do *Scrum Guide* (Guia do Scrum) 2020:

[Guia Scrum - Versão 2020]

Scrum é um framework leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos.

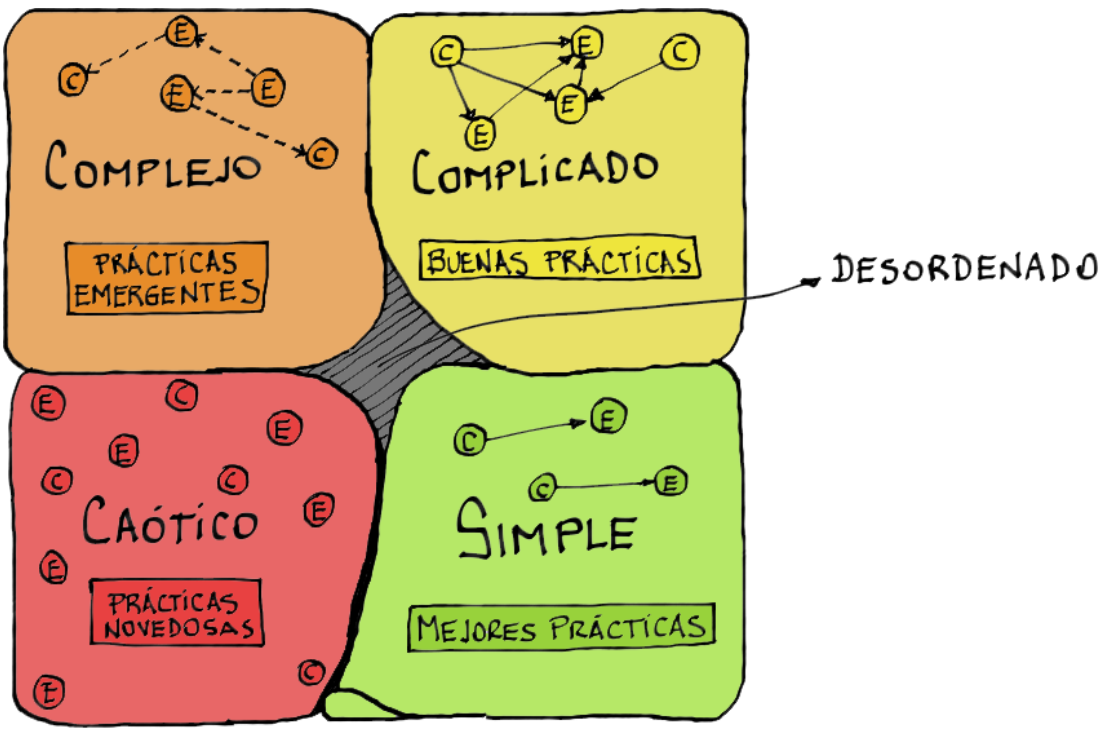
Em suma, Scrum requer um Scrum Master para promover um ambiente onde:

1. Um Product Owner ordena o trabalho para um problema complexo em um Product Backlog.
2. O Scrum Team transforma uma seleção do trabalho em um incremento de valor durante uma Sprint.
3. O Scrum Team e seus stakeholders inspecionam os resultados e se ajustam para a próxima Sprint.
4. Repita

Teoria da Complexidade

O Scrum tem sua aplicação recomendada para "ambientes complexos". Para entender o que é um ambiente complexidade, precisamos falar sobre a teoria da complexidade e o Modelo Cynefin.

A teoria de Cynefin é um framework conceitual usado para auxiliar na tomada de decisões e no gerenciamento de problemas. Foi criada por Dave Snowden enquanto trabalhava para a IBM no início dos anos 2000. O modelo Cynefin classifica os problemas e sistemas em cinco categorias ou domínios: simples, complicado, complexo, caótico e desordenado.



- **Simples (também chamado de Óbvio)**: Neste domínio, as relações entre causa e efeito são claras e podem ser identificadas com facilidade. Problemas são solucionados por meio de práticas estabelecidas e replicáveis. Este é o domínio do "sentido - categorizar - responder": você percebe a situação, categoriza-a e responde de acordo.

- **Complicado**: Aqui, a relação entre causa e efeito existe, mas não é imediatamente aparente para todos. Tais problemas podem requerer a ajuda de especialistas para serem solucionados. Este é o domínio do "analisar - sentido - responder": é preciso analisar a situação, entender o que está acontecendo e então responder de acordo.
- **Complexo**: Neste domínio, **a relação entre causa e efeito é vista apenas em retrospecto** e não é replicável. O caminho certo a seguir só pode ser determinado através de experimentação e observação. Este é o domínio do "sondar - sentido - responder": testa-se uma abordagem, observa-se o resultado e, com base nessa observação, decide-se o próximo passo.
- **Caótico**: No domínio caótico, não há relações claras entre causa e efeito. Decisões devem ser tomadas imediatamente para trazer a situação de volta ao controle. Este é o domínio do "agir - sentido - responder": toma-se uma ação para estabilizar a situação, avalia-se o resultado dessa ação e então responde-se com ações adicionais para trazer a situação de volta a um estado controlável.
- **Desordenado**: No domínio desordenado, não está claro qual dos outros quatro domínios se aplica à situação. O principal objetivo é descobrir em qual domínio você está para, então, poder agir adequadamente.

Recapitulando, Scrum é recomendado para ambientes complexos, ou seja, aqueles em que é mais importante testar e verificar resultados práticos e, com base nessa observação, melhorar o processo continuamente.

Pilares Fundamentais

Scrum é baseado em três pilares fundamentais: transparência, inspeção e adaptação. Eles são princípios que sustentam o framework do Scrum e essenciais para a sua implementação eficaz.

Transparência:

Significa que todos os aspectos do trabalho devem ser visíveis para todos que estão envolvidos no projeto. Isso inclui o progresso do trabalho durante um sprint, o backlog do produto e as definições de "pronto". Essa transparência é necessária para que todos possam entender exatamente o que está sendo feito e como está progredindo. Isso também permite que os problemas sejam identificados e tratados rapidamente.

Inspeção:

Regularmente, os membros da equipe de Scrum e as partes interessadas devem inspecionar o que está sendo desenvolvido para garantir que é o que o cliente quer e que está sendo feito corretamente. As inspeções ocorrem em várias ocasiões, como durante as reuniões diárias de Scrum, nas revisões de sprint e nas retrospectivas de sprint. A ideia é garantir que o produto está no caminho certo e que o trabalho está progredindo como planejado.

Adaptação:

Se durante a inspeção for determinado que um ou mais aspectos de um projeto desviam das expectativas, e o resultado não é mais o que o cliente ou o usuário quer, o processo ou o material sendo trabalhado deve ser ajustado. As adaptações devem ser feitas o mais rápido possível para minimizar qualquer impacto negativo adicional.

Esses três pilares funcionam juntos para garantir que o trabalho está sendo feito corretamente, que está progredindo conforme esperado e que está resultando em um produto que é valioso para o cliente ou usuário. Eles também ajudam a garantir que a equipe esteja sempre melhorando e aprendendo com seus erros e sucessos.

Valores

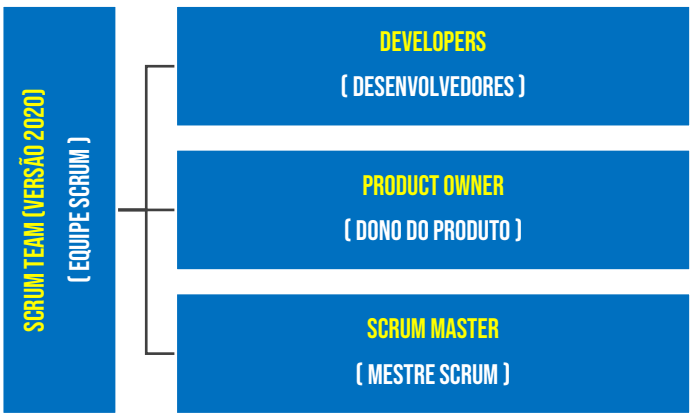
Scrum possui basicamente cinco valores principais: **coragem, foco, comprometimento, respeito e abertura**. O sucesso do uso do framework dependerá de como os membros da equipe entendem e aplicam cada um desses valores. Vejamos:

VALOR	DESCRIÇÃO
CORAGEM	Os integrantes de um projeto precisam ter coragem para fazer a coisa certa e trabalharem juntos removendo impedimentos, buscando soluções.
FOCO	Os integrantes de um projeto precisam focar no trabalho durante a sprint e nas metas designadas – time disperso perde produtividade e não alcança os objetivos.
COMPROMETIMENTO	Os integrantes se comprometem com o trabalho que se responsabilizou em fazer, envolvendo-se e não abandonando pela metade ou entregando sem qualidade.
RESPEITO	Os integrantes se respeitam entre si a fim de manter a colaboração, a integração e o bom ambiente de trabalho.
ABERTURA	Os integrantes devem poder ser francos, expor ideias e propostas mesmo que elas não sejam proveitosas. Momentos de debates, discussões e sugestões são ideais.

Papéis

O Scrum possui apenas três papéis, porém muito bem definidos. As pessoas que desempenham esses papéis são igualmente responsáveis e responsabilizadas pelos resultados do trabalho e, assim, se comprometem com o projeto. Eles são membros de um mesmo time e trabalham juntos, de forma colaborativa, para alcançarem seus resultados

Os papéis do Scrum são o Product Owner, os Developers e o Scrum Master.



Product Owner

O Product Owner, ou proprietário do produto, é fundamental para garantir que o valor do negócio seja entendido e priorizado durante o desenvolvimento do produto.

O PO tem várias responsabilidades importantes:

- **Definição de Visão**: O Product Owner deve ter uma visão clara e articulada do produto a ser desenvolvido. Eles devem entender as necessidades e desejos dos usuários e do negócio e ser capazes de comunicar essa visão à equipe de desenvolvimento.
- **Gestão do Backlog do Produto**: É responsável por criar e manter o backlog do produto, uma lista priorizada de tudo o que precisa ser feito para o produto. Ele precisa garantir que o backlog seja claro para todos e que os itens sejam priorizados de acordo com o valor que eles trazem para o negócio e para os usuários.
- **Entrega de Valor**: Deve garantir que a equipe de desenvolvimento esteja sempre trabalhando nos itens de maior valor. Ele também é responsável por aceitar ou rejeitar o trabalho realizado pela equipe de desenvolvimento.
- **Representação dos Stakeholders**: O Product Owner atua como uma ponte entre a equipe de desenvolvimento e os stakeholders do projeto (por exemplo, usuários, gerentes, outros departamentos da empresa). Ele deve compreender e defender as necessidades dos stakeholders e comunicá-las à equipe de desenvolvimento.
- **Adaptação**: Conforme o produto é desenvolvido e mais informações são aprendidas, o Product Owner deve ser capaz de se adaptar e mudar o plano conforme necessário. Ele deve ser capaz de incorporar o feedback dos usuários e das partes interessadas no produto e repriorizar o backlog do produto conforme necessário.

Para ser eficaz, o Product Owner precisa ter uma boa compreensão do negócio e do mercado, habilidades fortes de comunicação e capacidade de tomar decisões difíceis. Eles também precisam estar disponíveis para a equipe de desenvolvimento para responder a perguntas e tomar decisões rápidas.

Veja o que o Guia do Scrum fala:

[Guia Scrum - Versão 2020]

O Product Owner é responsável por maximizar o valor do produto resultante do trabalho do Scrum Team. A forma como isso é feito pode variar amplamente entre organizações, Scrum Teams e indivíduos. O Product Owner também é responsável pelo gerenciamento eficaz do Product Backlog , que inclui:

• Desenvolver e comunicar explicitamente a meta do produto;
• Criar e comunicar claramente os itens do Product Backlog;
• Ordenar os itens do Product Backlog; e,
• Garantir que o Product Backlog seja transparente, visível e compreensível.

O Product Owner pode fazer o trabalho acima ou pode delegar a responsabilidade a outros. Independentemente disso, o Product Owner ainda é o responsável. Para que os Product Owners tenham sucesso, toda a organização deve respeitar suas decisões. Essas decisões são visíveis no conteúdo e na ordem do Product Backlog e por meio do incremento inspecionável na revisão da sprint. O Product Owner é uma pessoa, não um comitê. O Product Owner pode representar as necessidades de muitos stakeholders

no Product Backlog. Aqueles que desejam alterar o Product Backlog podem fazê-lo tentando convencer o Product Owner.

Developer

Os Developers, são responsáveis pela entrega das funcionalidades do produto e têm várias responsabilidades importantes, tais como:

1. **Implementação do Produto**: Os Developers são responsáveis por implementar as funcionalidades descritas pelo Product Owner. Eles criam o produto utilizando as habilidades e ferramentas à sua disposição.
2. **Estimativa de Trabalho**: Durante o planejamento da Sprint, os Developers ajudam a estimar o trabalho necessário para implementar cada funcionalidade ou história de usuário. Eles precisam ter um entendimento sólido das tarefas necessárias para cada item do backlog do produto.
3. **Qualidade Técnica do Produto**: São responsáveis pela qualidade técnica do produto. Eles devem seguir as melhores práticas e padrões de codificação para garantir que o produto seja de alta qualidade e funcione como esperado.
4. **Participação em Cerimônias do Scrum**: Os Developers participam de todas as cerimônias do Scrum, incluindo as Reuniões Diárias de Scrum (Daily Stand-ups), Planejamento da Sprint, Revisão da Sprint e Retrospectiva da Sprint. Eles compartilham atualizações sobre o progresso e colaboram com o restante da equipe para resolver problemas.
5. **Autogerenciamento**: No Scrum, a equipe de Developers é auto-organizada e autogerenciada, o que significa que decide internamente quem faz o que, quando e como.

Diretamente do Guia do Scrum:

[Guia Scrum - Versão 2020]

Developers são as pessoas do Scrum Team que estão comprometidas em criar qualquer aspecto de um Incremento utilizável a cada Sprint. As habilidades específicas necessárias pelos Developers geralmente são amplas e variam de acordo com o domínio de trabalho. No entanto, os Developers são sempre responsáveis por: criar um plano para a Sprint, o Sprint Backlog; introduzir gradualmente qualidade aderindo a uma Definição de Pronto; adaptar seu plano a cada dia em direção à meta da Sprint; e responsabilizar-se mutuamente como profissionais.

Scrum Master

O Scrum Master não é o líder tradicional ou o gerente do projeto, mas sim um facilitador e um servidor-líder que ajuda a equipe a seguir o processo Scrum.

As responsabilidades do Scrum Master incluem:

- **Defender o Scrum**: O Scrum Master deve educar a equipe e a organização sobre o Scrum e as práticas ágeis. Ele ajuda a entender a teoria, as práticas, as regras e os valores do Scrum.
- **Facilitar Cerimônias Scrum**: O Scrum Master organiza e facilita as cerimônias Scrum, como as reuniões de planejamento de Sprint, Revisões de Sprint, Retrospectivas de Sprint e as reuniões diárias de Scrum (Daily Stand-ups).
- **Remover Obstáculos**: É responsável por remover qualquer obstáculo ou bloqueio que possa impedir a equipe de alcançar seus objetivos. Isso inclui questões técnicas, conflitos de pessoal, problemas com ferramentas ou software, e assim por diante.
- **Facilitar a Colaboração**: Promove a colaboração entre a equipe de desenvolvimento e o Product Owner, bem como com outras partes interessadas e equipes.
- **Promover Melhoria Contínua**: Ajuda a equipe a melhorar continuamente através de práticas como a retrospectiva da Sprint, onde a equipe reflete sobre o que correu bem e o que pode ser melhorado para a próxima Sprint.
- **Proteger a Equipe**: Protege a equipe de interrupções desnecessárias ou de demandas externas que podem distrair a equipe de seus objetivos para a Sprint. Ele também garantem que a equipe tenha um ambiente de trabalho seguro e positivo.

Para ser um bom Scrum Master, a pessoa precisa ter uma compreensão sólida dos princípios e práticas do Scrum, ser um bom comunicador e ter habilidades de resolução de conflitos. Além disso, eles devem ser capazes de liderar e motivar a equipe, bem como promover uma cultura de transparência, inspeção e adaptação.

Do Guia do Scrum:

[Guia Scrum - Versão 2020]

O Scrum Master é responsável por estabelecer o Scrum conforme definido no Guia do Scrum. Eles fazem isso ajudando todos a entender a teoria e a prática do Scrum, tanto no Scrum Team quanto na organização. O Scrum Master é responsável pela eficácia do Scrum Team. Eles fazem isso permitindo que o Scrum Team melhore suas práticas, dentro do framework Scrum Scrum Masters são verdadeiros líderes que servem ao Scrum Team e à organização como um todo.

O Scrum Master serve ao Scrum Team de várias maneiras, incluindo:

• Treinar os membros do time em autogerenciamento e cross-funcionalidade;
• Ajudar o Scrum Team a se concentrar na criação de incrementos de alto valor que atendem à Definição de Pronto;
• Provocando a remoção de impedimentos ao progresso do Scrum Team; e,
• Garantir que todos os eventos Scrum ocorram e sejam positivos, produtivos e mantidos dentro do Timebox.

O Scrum Master serve o Product Owner de várias maneiras, incluindo:

• Ajudar a encontrar técnicas para a definição eficaz de meta do Produto e gerenciamento do Product Backlog;
• Ajudar o Scrum Team a entender a necessidade de itens do Product Backlog claros e concisos;
• Ajudar a estabelecer o planejamento empírico do produto para um ambiente complexo; e,
• Facilitar a colaboração dos stakeholder, conforme solicitado ou necessário.

O Scrum Master serve a organização de várias maneiras, incluindo:

• Liderar, treinar e orientar a organização na adoção do Scrum;
• Planejar e aconselhar implementações de Scrum dentro da organização;
• Ajudar os funcionários e os stakeholders a compreender e aplicar uma abordagem empírica para trabalhos complexos; e, • Remover barreiras entre stakeholders e Scrum Teams.

Uma analogia comumente usada para explicar os três papéis em Scrum é a de uma orquestra.

- **Product Owner - Maestro**: O Maestro (Product Owner) não toca nenhum instrumento, mas conhece a partitura de cor (a visão do produto). Ele conduz a orquestra, garantindo que todos toquem no tempo certo para criar a sinfonia desejada. O Maestro interpreta a música da maneira que acha que deveria soar (define e prioriza o backlog do produto) e faz ajustes conforme necessário para que a orquestra soe bem (adaptação).
- **Developers - Músicos**: Os Músicos (Developers) são especialistas em seus instrumentos. Eles sabem como tocar as notas na partitura (implementam as funcionalidades). Eles praticam individualmente, mas também ensaiam juntos para garantir que tudo se encaixe (colaboração e auto-organização). Eles são guiados pelo Maestro, mas também têm a liberdade de fazer sugestões e melhorias para tornar a música ainda melhor (contribuição para a visão do produto).
- **Scrum Master - Coordenador de Palco**: O Coordenador de Palco (Scrum Master) garante que todos os músicos tenham o que precisam para a performance. Ele garante que os instrumentos estejam afinados, que as partituras estejam disponíveis, que o palco esteja pronto, etc. (remove obstáculos). Ele também ajuda o Maestro a manter a orquestra funcionando suavemente (facilita as cerimônias Scrum e ajuda a equipe a seguir os princípios Scrum).

Artefatos

Segundo o Guia do Scrum, o framework possui apenas três artefatos oficiais: Product Backlog, Sprint Backlog e Product Increment.

[Guia Scrum - Versão 2020]

Os artefatos do Scrum representam trabalho ou valor. Eles são projetados para maximizar a transparência das principais informações. Assim, todos os que os inspecionam têm a mesma base para adaptação. Cada artefato contém um compromisso para garantir que ele forneça informações que aumentem a transparência e o foco contra o qual o progresso pode ser medido:

• Para o Product Backlog, é a Meta do produto.
• Para o Sprint Backlog, é a Meta da Sprint.
• Para o incremento, é a Definição de Pronto.

Esses compromissos existem para reforçar o empirismo e os valores Scrum para o Scrum Team, e seus stakeholders.

Product Backlog

O Product Backlog é um dos três artefatos do Scrum e consiste essencialmente de uma "lista de necessidades" para o projeto. Ele contém todas as características, funcionalidades, requisitos, melhorias e correções que são necessárias para o produto e que ainda não foram implementadas.

Características do Product Backlog:

- **Listagem Priorizada**: O Product Backlog é uma lista priorizada de itens. Os itens de maior prioridade, que geralmente são aqueles que fornecem o maior valor para o negócio ou para o usuário, estão no topo da lista. O Product Owner é responsável por definir a prioridade dos itens no backlog.
- **Itens do Backlog**: Cada item do backlog, frequentemente chamado de "história de usuário" ou "épico", contém uma descrição do que precisa ser feito, a razão para isso e qualquer outra informação que a equipe de desenvolvimento possa precisar para implementá-lo.

Backlog do Produto: Twitter
Login de usuários já cadastrados
Cadastrar novo usuário
Tuitar
Visualizar tuítes de quem sigo
Visualizar número de seguidores que possuo
Visualizar histórico de um tuíte
Mostrar banner promocional
Remover tuíte
Ver sugestões de usuários a seguir
Listar trends
Compor e alterar meu perfil de usuário
(...)

- **Vivo e Adaptável**: O Product Backlog é um documento vivo. Ele é constantemente atualizado e modificado conforme o projeto avança. Itens podem ser adicionados, removidos ou repriorizados com base em feedback, mudanças no mercado, novas ideias, e assim por diante.
- **Refinamento do Backlog**: É uma prática regular em Scrum revisar e refinar o backlog, um processo às vezes chamado de "grooming". Durante a reunião de refinamento, o Product Owner e a equipe de desenvolvimento revisam os itens do backlog, discutem detalhes, esclarecem dúvidas e ajustam a prioridade dos itens, se necessário.
- **Estimativas de Esforço**: Para cada item do backlog, a equipe de desenvolvimento geralmente fornece uma estimativa do esforço necessário para implementá-lo. Isso ajuda o Product Owner e a equipe de desenvolvimento a planejar o que pode ser realizado em cada Sprint.

O Product Backlog é a principal fonte de requisitos para qualquer mudança a ser feita no produto. É a única lista de trabalho e tudo o que a equipe de Scrum trabalha deve ser derivado dele.

Idealmente, cada item do Backlog do Produto, para ser considerado em uma Sprint, deve estar em um estado de "preparado", o que nos leva ao conceito de *Definition of Ready.*

Definition of Ready (Definição de Pronto): Um conjunto de critérios que um item do Product Backlog deve atender antes que ele possa ser considerado pronto para ser incluído em uma Sprint. A "Definition of Ready" pode variar de equipe para equipe, mas geralmente inclui coisas como ter uma descrição clara, ter critérios de aceitação definidos, ser de um tamanho gerenciável para ser concluído em uma Sprint e ser valorizado pelo Product Owner. O objetivo da "Definition of Ready" é garantir que a equipe tenha todas as informações necessárias para iniciar o trabalho em um item, reduzindo a possibilidade de interrupções durante a Sprint.

Veja a definição de Product Backlog no Guia do Scrum:

[Guia Scrum - Versão 2020]

O Product Backlog é uma lista ordenada e emergente do que é necessário para melhorar o produto. É a única fonte de trabalho realizado pelo Scrum Team. Os itens do Product Backlog que podem ser realizados pelo Scrum Team em uma Sprint são considerados preparados para seleção no evento Sprint Planning. Eles geralmente adquirem esse grau de transparência após as atividades de refinamento. O Product Backlog refinement é o ato de quebrar e incluir definição adicional aos itens do Product Backlog para ter itens menores e mais precisos.

Esta é uma atividade contínua para adicionar detalhes, como descrição, ordem e tamanho. Os atributos geralmente variam de acordo com o domínio de trabalho. Os Developers que farão o trabalho são responsáveis pelo dimensionamento. O Product Owner pode influenciar os Developers, ajudando-os a entender e selecionar trade-offs (trocas de itens).

***Compromisso: Meta do Produto.** A Meta do Produto descreve um estado futuro do produto que pode servir como um alvo para o Scrum Team planejar. A Meta do produto está no Product Backlog. O restante do Product Backlog emerge para definir "o que" cumprirá a Meta do Produto. Um produto é um veículo para entregar valor. Tem um limite claro, stakeholders conhecidos, usuários ou clientes bem definidos. Um produto pode ser um serviço, um produto físico ou algo mais abstrato. A Meta do Produto é o objetivo de longo prazo para o Scrum Team. Eles devem cumprir (ou abandonar) um objetivo antes de assumir o próximo.*

Sprint Backlog

O Sprint Backlog é uma lista de itens do Product Backlog que a equipe de desenvolvimento se compromete a completar durante um Sprint específico.

Características do Sprint Backlog:

- **Seleção de Trabalho**: Durante a reunião de Planejamento da Sprint, a equipe de desenvolvimento seleciona os itens do Product Backlog que acredita ser capaz de completar durante a próxima Sprint. Esses itens selecionados compõem o Sprint Backlog.
- **Objetivo do Sprint**: Cada Sprint tem um Objetivo da Sprint, que é uma descrição curta e flexível do que a equipe planeja alcançar durante a Sprint. O Objetivo da Sprint é determinado durante a reunião de Planejamento da Sprint e fornece à equipe uma orientação clara e compartilhada.
- **Flexibilidade Limitada**: Uma vez que uma Sprint começa, seu Sprint Backlog é geralmente considerado bloqueado ou fixo, o que significa que não são permitidos novos itens. Isso permite que a equipe de desenvolvimento se concentre em completar o trabalho que se comprometeu a fazer, sem interrupções. No entanto, o Sprint Backlog pode ser renegociado entre a equipe de desenvolvimento e o Product Owner como parte do controle de processos empíricos.
- **Visibilidade e Transparência**: O Sprint Backlog é uma ferramenta visual que mostra todo o trabalho que a equipe se comprometeu a fazer em uma Sprint. Ele deve ser facilmente acessível e visível para todos os membros da equipe.

O Sprint Backlog é um plano com detalhes suficientes que podem ser entendidos pelo menos pelos desenvolvedores. Não é apenas uma lista de itens a fazer, mas um plano altamente visível e de fácil acesso que mostra o trabalho da equipe para alcançar o Objetivo da Sprint.

Veja um exemplo de Sprint Backlog:

Backlog da Sprint #2		
META DA SPRINT: USUÁRIOS PODERÃO ESTAR NO TWITTER		
Login de usuários já cadastrados	Ativar login com usuário GMail	Montar layout do box de login
	Montar plano de segurança	Testar integrado
	Estruturar log	Revisar código
	Criar comportamento de login	Inserir hint explicativo de funcionamento
	Atualizar documentação técnica	(...)
Cadastrar novo usuário	Criar tabelas no banco de dados	Estruturar persistência
	Definição sobre uso de templates	Quando validado, ativar usuário
	Escrever testes	Definir padrões para cadastros
	Atualizar documentação técnica	Validar e-mail cadastrado
	Testar integrado	(...)

Definição do Guia do Scrum:

[Guia Scrum - Versão 2020]

O Sprint Backlog é composto pela Meta da Sprint (por que), o conjunto de itens do Product Backlog selecionados para a Sprint (o que), bem como um plano de ação para entregar o Incremento (como). O Sprint Backlog é um plano feito por e para os Developers. É uma imagem altamente visível, em tempo real do trabalho que os Developers planejam realizar durante a Sprint para atingir a Meta da Sprint. Consequentemente, o Sprint Backlog é atualizado ao longo da Sprint conforme mais é aprendido. Deve ter detalhes suficientes para que eles possam inspecionar seu progresso na Daily Scrum.

Compromisso: Meta da Sprint

A Meta da Sprint é o único objetivo da Sprint. Embora a Meta da Sprint seja um compromisso dos Developers, esta fornece flexibilidade em termos do trabalho exato necessário para alcançá-la. A Meta da Sprint também cria coerência e foco, encorajando o Scrum Team a trabalhar junto ao invés de iniciativas separadas. A Meta da Sprint é criada durante o evento Sprint Planning e então adicionada ao Sprint Backlog. Conforme os Developers trabalham durante a Sprint, eles mantêm a Meta da Sprint em mente. Se o trabalho acabar sendo diferente do que eles esperavam, eles colaboram com o Product Owner para negociar o escopo do Sprint Backlog dentro da Sprint sem afetar a Meta da Sprint.

Product Increment

O Incremento do Produto é o terceiro e último artefato no Scrum. Ele é a soma de todos os itens do Product Backlog concluídos durante uma Sprint e todos os incrementos anteriores. Em outras palavras, é a versão mais recente do produto, que é potencialmente liberável ou pronta para ser entregue ao usuário final.

Aqui estão algumas características importantes do Incremento do Produto:

- **Potencialmente Liberável**: Uma das principais características de um Incremento é que ele deve estar em um estado "Potencialmente Liberável". Isso significa que ele atende a todas as normas de qualidade necessárias e está pronto para ser entregue ao usuário final, se o Product Owner decidir que é o momento certo para isso.
- **Soma de Trabalho**: O Incremento é a soma de todo o trabalho concluído durante a Sprint atual e todos as Sprints anteriores. Cada novo incremento se baseia no anterior, acrescentando novas funcionalidades e melhorias.
- **Definição de Pronto**: Cada equipe Scrum deve ter uma "Definição de Pronto" que define claramente o que significa para um item do Product Backlog estar completo. Essa definição ajuda a equipe a saber quando um item está pronto para ser adicionado ao Incremento.
- **Transparência**: O Incremento fornece uma forma clara e transparente para todos os envolvidos no projeto verem o progresso que está sendo feito. Ele é uma representação tangível do trabalho que foi realizado.
- **Inspeção e Adaptação**: No final de cada Sprint, o Incremento do Produto é inspecionado na Revisão da Sprint. Isso permite que a equipe e os stakeholders vejam o que foi realizado e adaptam o plano conforme necessário para a próxima Sprint.

O Incremento do Produto é essencialmente a razão pela qual a equipe Scrum existe - para produzir um produto de valor incrementado Sprint após Sprint, proporcionando valor regular e contínuo para o cliente ou usuário final.

O Incremento do Produto só é considerado "pronto" quando atender ao critério de *Definition of Done*.

Definition of Done (Definição de Concluído): Um conjunto de critérios que um item do Product Backlog deve atender para ser considerado concluído. A "Definition of Done" também varia de equipe para equipe, mas pode incluir coisas como passar em todos os testes, ter documentação completa, ser revisado e aprovado pelo Product Owner e não ter defeitos conhecidos. O objetivo da "Definition of Done" é garantir que todos na equipe tenham a mesma compreensão do que significa para um trabalho ser concluído, garantindo a qualidade do Incremento do Produto.

Veja a definição de Incremento do Produto no Guia do Scrum:

[Guia Scrum - Versão 2020]

Um incremento é um trampolim concreto em direção a Meta do produto. Cada incremento é adicionado a todos os incrementos anteriores e completamente verificado, garantindo que todos os incrementos funcionem juntos. A fim de fornecer valor, o incremento deve ser utilizável. Vários incrementos podem ser criados em uma Sprint. A soma dos incrementos é apresentada na Sprint Review, apoiando assim o empirismo. No entanto, um incremento pode ser entregue aos stakeholders antes do final da Sprint. A Sprint Review nunca deve ser considerada um marco para liberar valor. O trabalho não pode ser considerado parte de um incremento a menos que atenda a Definição de Pronto.

Compromisso: Definição de Pronto

A Definição de Pronto é uma descrição formal do estado do Incremento quando ela atende às medidas de qualidade exigidas para o produto. No momento em que um item do Product Backlog atende a Definição de Pronto, um incremento nasce. A Definição de Pronto cria transparência ao fornecer a todos um entendimento compartilhado de qual trabalho foi concluído como parte do Incremento. Se um item do Product Backlog não atender à Definição de Pronto, ele não poderá ser liberado ou mesmo apresentado na Sprint Review. Em vez disso, ele retorna ao Product Backlog para consideração futura.

Se a Definição de Pronto para um incremento faz parte dos padrões da organização, todos os Scrum Teams devem segui-la como mínimo. Se não for um padrão organizacional, o Scrum Team deve criar uma Definição de Pronto apropriada para o produto. Os Developers devem estar em conformidade com a Definição de Pronto. Se houver vários
Scrum Teams trabalhando juntos em um produto, eles devem definir e cumprir mutuamente a mesma Definição de Pronto.

Gráfico de Burndown

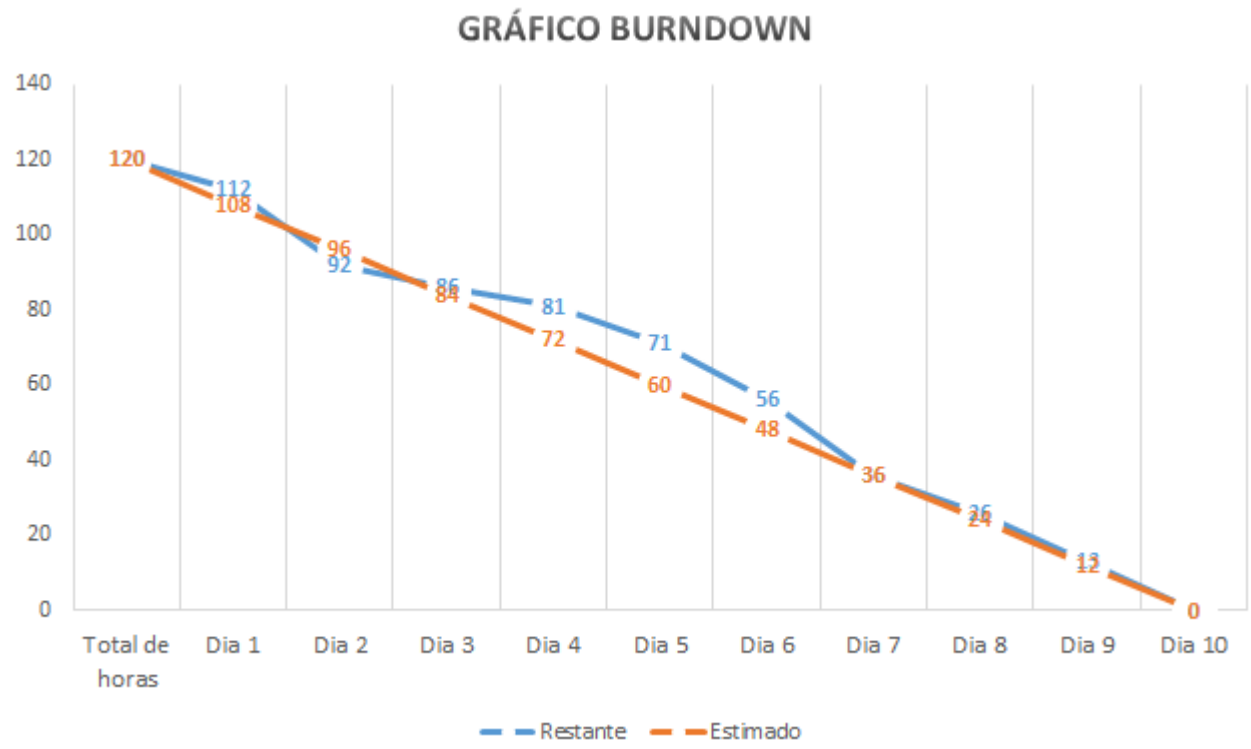
Um último artefato que vale a pena mencionarmos é o Gráfico de Burndown, que não aparece explicitamente no Guia do Scrum, mas é muito utilizado por equipes ágeis.

O gráfico de burndown é uma ferramenta visual usada em métodos ágeis, como o Scrum, para rastrear o progresso da equipe ao longo de uma Sprint ou de um projeto. Ele fornece uma maneira rápida e fácil de ver o trabalho restante e como a equipe está progredindo em direção à conclusão da Sprint ou do projeto.

O eixo vertical do gráfico representa a quantidade de trabalho a ser feita, geralmente medido em unidades como pontos de história ou horas de trabalho. O eixo horizontal representa o tempo, geralmente dividido em dias da Sprint ou fases do projeto.

No início de um Sprint, a linha de burndown começa no topo, representando todo o trabalho que precisa ser feito. Conforme o trabalho é concluído, a linha de burndown cai, idealmente chegando ao fundo do gráfico no final da Sprint.

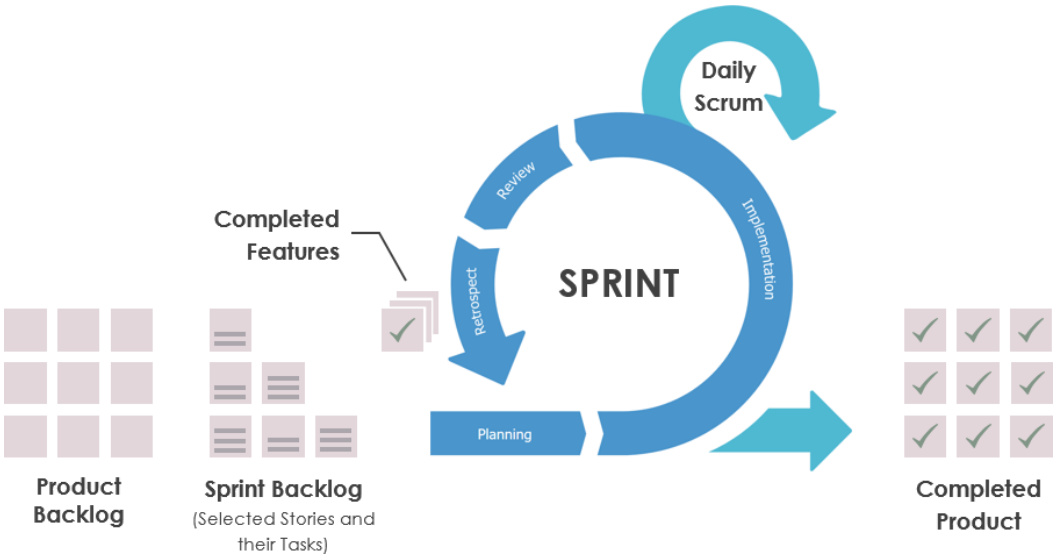
Veja um exemplo:



Eventos e Cerimônias

No Scrum, eventos ou cerimônias são momentos estruturados destinados a formalizar e regular o processo de trabalho durante cada Sprint. Eles são essenciais para promover a colaboração, monitorar o progresso e fazer ajustes conforme necessário. Existem cinco eventos principais no

Scrum: Sprint, Planejamento da Sprint, Reunião Diária, Revisão da Sprint e Retrospectiva da Sprint.



Sprint

A Sprint é o coração do Scrum, uma unidade de tempo fixa durante a qual um conjunto de trabalho é concluído. Geralmente, Sprints duram entre uma e quatro semanas (no máximo), embora a duração possa variar dependendo da equipe e do projeto. A duração da Sprint é determinada durante o planejamento da Sprint e não deve ser alterada depois que a Sprint começa.

O objetivo da Sprint é produzir um Incremento do Produto "Potencialmente Liberável", que é uma versão do produto que está em um estado suficientemente alto de qualidade e funcionalidade para ser liberado para o cliente, se desejado.

Características importantes de uma Sprint:

- **Objetivos de Sprint**: No início de cada Sprint, a equipe e o Product Owner concordam com um Objetivo da Sprint, que é uma descrição breve e flexível do que a equipe espera alcançar durante o Sprint.
- **Trabalho Consistente**: Uma vez que uma Sprint começa, a quantidade de trabalho (ou o escopo) deve permanecer consistente. Isso permite que a equipe se concentre em concluir o trabalho que se comprometeu a fazer, sem a interrupção de novas demandas ou alterações.
- **Timebox (Fim definido)**: Cada Sprint tem um fim definido. Quando o tempo acaba, a Sprint termina, independentemente de todo o trabalho ter sido concluído.

A Sprint fornece um ritmo constante para o trabalho da equipe, incentivando um ciclo de planejamento, trabalho, revisão e melhoria. Ao trabalhar em Sprints, a equipe pode entregar valor regularmente, inspecionar e adaptar seu trabalho e processos de forma consistente, e responder rapidamente às mudanças ou novas informações.

Veja a definição do Guia do Scrum:

[Guia Scrum - Versão 2020]

A Sprint é um contêiner para todos os outros eventos. Cada evento no Scrum é uma oportunidade formal para inspecionar e adaptar os artefatos do Scrum. Esses eventos são projetados especificamente para permitir a transparência necessária. A falha em operar quaisquer eventos conforme prescrito resulta em oportunidades perdidas de inspeção e adaptação. Os eventos são usados no Scrum para criar regularidade e minimizar a necessidade de reuniões não definidas no Scrum. O ideal é que todos os eventos sejam realizados no mesmo horário e local para reduzir a complexidade.

Sprints são o coração do Scrum, onde ideias são transformadas em valor. São eventos de duração fixa de um mês ou menos para criar consistência. Uma nova Sprint começa imediatamente após a conclusão da Sprint anterior. Todo o trabalho necessário para atingir a meta do Produto, incluindo Sprint Planning, Daily Scrums, Sprint Review e Sprint Retrospective, acontece dentro de Sprints. Durante a Sprint:

• Nenhuma mudança é feita que coloque em risco a meta da Sprint;
• A qualidade não diminui;
• O Product Backlog é refinado conforme necessário; e,
• O escopo pode ser esclarecido e renegociado com o Product Owner conforme mais é aprendido.

Sprints permitem previsibilidade, garantindo a inspeção e adaptação do progresso em direção a uma meta do Produto ao menos uma vez por mês. Quando o horizonte de uma Sprint é muito longo, a meta da Sprint pode se tornar inválida, a complexidade pode aumentar e o risco pode aumentar. Sprints mais curtas podem ser empregados para gerar mais ciclos de aprendizagem e limitar os riscos de custo e esforço a um período de tempo menor. Cada Sprint pode ser considerado um projeto curto. Existem várias práticas para prever o progresso, como burn-downs, burn-ups ou cumulative flows. Embora comprovadamente úteis, eles não substituem a importância do empirismo. Em ambientes complexos, o que acontecerá é desconhecido. Somente o que já aconteceu pode ser usado para a tomada de decisão voltada para o futuro. Uma Sprint pode ser cancelada se a Meta da Sprint se tornar obsoleta. Apenas o Product Owner tem autoridade para cancelar a Sprint.

Planejamento da Sprint

O Planejamento da Sprint é uma cerimônia que ocorre no início de cada Sprint. A duração do Planejamento da Sprint é geralmente proporcional à duração da Sprint; para uma Sprint de duas semanas, a reunião de Planejamento da Sprint pode durar até quatro horas.

O objetivo da Planejamento da Sprint é estabelecer um entendimento comum do que será trabalhado durante a Sprint. Isso é alcançado por meio da colaboração entre o Product Owner, a equipe de desenvolvimento e o Scrum Master.

O Planejamento da Sprint geralmente envolve os seguintes componentes:

1. O que será entregue como resultado do incremento da próxima Sprint?

2. Como o trabalho necessário para entregar o incremento será realizado?

- **Estabelecer o Objetivo da Sprint**: O Product Owner apresenta o objetivo da Sprint, que é a principal meta ou resultado desejado para ela.
- **Seleção de Itens do Backlog**: O Product Owner apresenta os itens de maior prioridade do Product Backlog à equipe. A equipe de desenvolvimento então estima o esforço necessário para cada item e seleciona o que acredita que pode ser concluído durante a Sprint, levando em consideração sua capacidade.
- **Quebrar Itens do Backlog em Tarefas**: A equipe de desenvolvimento então divide os itens do Backlog selecionados em tarefas menores. Cada tarefa é estimada em termos de tempo ou esforço necessário para concluí-la.
- **Compromisso**: No final do Planejamento da Sprint, a equipe se compromete com o conjunto de trabalho que planeja concluir durante a Sprint.

Uma técnica muito famosa utilizada para estimar o esforço necessário para determinados Itens do Backlog durante o Planejamento da Sprint é a técnica de **Planning Poker**.

Essa técnica combina a opinião de desenvolvedores, analogia e desagregação em uma abordagem divertida na estimativa de Histórias de Usuário, e elimina a influência que um desenvolvedor possa exercer sobre outros.

Primeiramente, cada desenvolvedor faz uma estimativa de esforço para cada Item de Backlog (História de Usuário). Feita a estimativa, os desenvolvedores decidirão a quantidade de itens do backlog a serem realizados na sprint, conhecidos como "Story Points". 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 uma User Story, os desenvolvedores os comparam com outros já estimados. Caso não haja ainda nada estimado no Product Backlog, os desenvolvedores localizam o User Story com o menor esforço para o desenvolvimento, e o utiliza como base para comparações futuras. Uma das melhores formas de se estimar Story Points é utilizando o Planning Poker.

Vamos entender um pouco melhor como ele funciona?

No início do Planning Poker, cada membro do time recebe um conjunto de cartas. Cada carta exibe um dos valores válidos para a estimativa (Ex: 0, 1, 2, 3, 5, 8, 13, 20, 40, e 100).

Em geral, os valores seguem uma escala baseada na Sequência de Fibonacci. Observem: outra sequência pode ser escolhida, porém Fibonacci é a mais utilizada. Para cada User Story a ser estimativa, 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 significativamente diferentes. E na realidade é um bom sinal.

Quando as estimativas diferem, os membros 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. O Planning Poker funciona porque utiliza a opinião de diversos desenvolvedores na estimativa. Como estes experts compõem um time multidisciplinar, 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. Mais uma vez: Fibonacci é a escala mais utilizada na estimativa de User Story por Planning Poker. Isso se deve ao fato de a Sequência de Fibonacci ser uma função quadrática, em vez de uma função linear. Detalhe: algumas vezes as histórias de usuário são muito grandes para serem desenvolvidas em uma sprint – são chamadas de Épicos.

Veja o planejamento da Sprint como definido no Guia do Scrum:

[Guia Scrum - Versão 2020]

A Sprint Planning inicia a Sprint ao definir o trabalho a ser realizado na Sprint. Este plano resultante é criado pelo trabalho colaborativo de todo o Scrum Team. O Product Owner garante que os participantes estejam preparados para discutir os itens mais importantes do Product Backlog e como eles são mapeados para a Meta do Produto. O Scrum
Team também pode convidar outras pessoas para participar da Sprint Planning para fornecer conselhos.

A Sprint Planning aborda os seguintes tópicos:

Tópico um: Por que esta Sprint é valiosa?

O Product Owner propõe como o produto pode aumentar seu valor e utilidade na Sprint atual. Todo o Scrum Team então colabora para definir uma Meta da Sprint que comunica porque a Sprint é valiosa para os stakeholders. A meta da Sprint deve ser finalizada antes do final da Sprint Planning.

Tópico dois: O que pode ser feito nesta Sprint?

Por meio de discussão com o Product Owner, os Developers selecionam itens do Product Backlog para incluir na Sprint atual. O Scrum Team pode refinar esses itens durante este processo, o que aumenta a compreensão e a confiança. Selecionar o quanto pode ser concluído em uma Sprint pode ser um desafio. No entanto, quanto mais os Developers sabem sobre seu desempenho anterior, sua capacidade futura e sua Definição de Pronto, mais confiantes eles estarão em suas previsões quanto a Sprint.

Tópico três: Como o trabalho escolhido será realizado?

Para cada item do Product Backlog selecionado, os Developers planejam o trabalho necessário para criar um Incremento que atenda à Definição de Pronto. Isso geralmente é feito decompondo itens do Product Backlog em itens de trabalho menores de um dia ou menos. A forma como isso é feito fica a critério exclusivo dos Developers . Ninguém mais diz a eles como transformar itens do Product Backlog em incrementos de valor. A Meta da Sprint, os itens do Product Backlog selecionados para a Sprint, mais o plano para entregá-los são chamados juntos de Sprint Backlog.

A Sprint Planning tem um Timebox definido com duração máxima de oito horas para uma Sprint de um mês. Para Sprints mais curtas, o evento geralmente é mais curto.

Reunião Diária

A Reunião Diária, também conhecida como Daily Scrum ou Stand-up, é uma cerimônia curta que acontece todos os dias no mesmo horário e lugar durante um Sprint. Esta reunião é geralmente limitada a 15 minutos, independentemente do tamanho da equipe, para manter a eficiência.

Durante a Reunião Diária, cada membro da equipe de desenvolvimento pode responder às seguintes três perguntas:

1. **O que eu fiz ontem para ajudar a equipe a alcançar o Objetivo do Sprint?**
2. **O que eu planejo fazer hoje para ajudar a equipe a alcançar o Objetivo do Sprint?**
3. **Existem quaisquer impedimentos no meu caminho?**

O propósito da Reunião Diária é promover a comunicação aberta e eficiente entre os membros da equipe, permitindo que todos estejam cientes do progresso em direção ao Objetivo da Sprint e de quaisquer problemas ou obstáculos que possam surgir.

O Scrum Master facilita a reunião, mas não precisa ser o único a falar. O objetivo é permitir que a equipe de desenvolvimento se sincronize, não fornecer um status para o Scrum Master ou Product Owner.

Importante mencionar que a Reunião Diária não é uma reunião de resolução de problemas. Se surgirem problemas que exigem discussão adicional, uma reunião separada deve ser agendada com as pessoas relevantes.

[Guia Scrum - Versão 2020]

O propósito da Daily Scrum é inspecionar o progresso em direção a Meta da Sprint e adaptar o Sprint Backlog conforme necessário, ajustando o próximo trabalho planejado. A Daily Scrum é um evento de 15 minutos para os Developers do Scrum Team. Para reduzir a complexidade, é realizado no mesmo horário e local, todos os dias úteis da Sprint. Se o Product Owner ou o Scrum Master estão trabalhando ativamente nos itens do Sprint Backlog, eles participam como Developers.

Os Developers podem selecionar qualquer estrutura e técnicas que quiserem, desde que seu Daily Scrum se concentre no progresso em direção a Meta da Sprint e produza um plano de ação para o próximo dia de trabalho. Isso cria foco e melhora o autogerenciamento. As Daily Scrums melhoram as comunicações, identificam os impedimentos, promovem a rápida tomada de decisões e consequentemente, eliminam a necessidade de outras reuniões. A Daily Scrum não é o único momento em que os Developers podem ajustar seu plano. Eles costumam se reunir ao longo do dia para discussões mais detalhadas sobre a adaptação ou replanejamento do resto do trabalho da Sprint.

Revisão da Sprint

A Revisão da Sprint é uma cerimônia que acontece no final de cada Sprint, antes da Retrospectiva da Sprint. A duração da Revisão da Sprint é geralmente proporcional à duração da Sprint; para uma Sprint de duas semanas, a Revisão da Sprint pode durar até duas horas.

O objetivo da Revisão da Sprint é inspecionar o Incremento do Produto concluído e adaptar o Product Backlog, se necessário. Em outras palavras, é um momento para a equipe mostrar o que foi feito durante a Sprint.

A Revisão da Sprint geralmente inclui os seguintes componentes:

1. **Demonstração do trabalho**: A equipe de desenvolvimento demonstra o trabalho concluído e responde a quaisquer perguntas sobre o Incremento do Produto.
2. **Revisão do Product Backlog**: O Product Owner discute o status atual do Product Backlog e projeta datas de entrega prováveis com base no progresso até agora.
3. **Feedback e adaptação**: Todos os participantes colaboram sobre o que foi visto na revisão. Com base nesse feedback, o Product Owner pode fazer ajustes no Product Backlog, o que pode incluir detalhamento adicional de itens do backlog, ajustes na priorização ou até a introdução de novos itens.

A Revisão da Sprint fornece uma oportunidade para a equipe receber feedback valioso sobre o produto de partes interessadas, incluindo outros membros da equipe, outros membros da organização e, se possível, o cliente ou usuário final. É uma cerimônia crucial para garantir que o produto continue a fornecer o maior valor possível ao cliente.

[Guia Scrum - Versão 2020]

O propósito da Sprint Review é inspecionar o resultado da Sprint e determinar as adaptações futuras. O Scrum Team apresenta os resultados de seu trabalho para os principais stakeholders e o progresso em direção a Meta do Produto é discutido. Durante o evento, o Scrum Team e os stakeholders revisam o que foi realizado na Sprint e o que mudou em seu ambiente. Com base nessas informações, os participantes colaboram sobre o que fazer a seguir. O Product Backlog também pode ser ajustado para atender a novas oportunidades.

A Sprint Review é uma sessão de trabalho e o Scrum Team deve evitar limitá-la a uma apresentação. A Sprint Review é o penúltimo evento da Sprint e tem um Timebox com prazo máximo de quatro horas para uma Sprint de um mês. Para Sprints mais curtas, o evento geralmente é mais curto.

Retrospectiva da Sprint

A Retrospectiva da Sprint é uma reunião que ocorre após a Revisão da Sprint e antes do próximo Planejamento da Sprint. Esta cerimônia marca o fim de uma Sprint e serve como uma

oportunidade para a equipe inspecionar a si mesma e criar um plano para melhorias a serem aplicadas na próxima Sprint.

Durante a Retrospectiva da Sprint, a equipe reflete sobre o que funcionou bem e o que pode ser melhorado. Os membros da equipe discutem suas experiências, compartilham feedback e identificam ações para melhorar seu desempenho na próxima Sprint.

A Retrospectiva da Sprint geralmente inclui três principais tópicos de discussão:

1. **O que correu bem durante a Sprint**: Esta parte da discussão se concentra em identificar os sucessos. Isso ajuda a equipe a entender o que deve continuar fazendo.
2. **O que pode ser melhorado**: Aqui, a equipe discute os desafios enfrentados durante a Sprint e identifica maneiras de superar esses desafios no futuro.
3. **Quais ações podem ser tomadas para implementar melhorias**: Com base nas discussões anteriores, a equipe define um plano de ação concreto para a próxima Sprint. Isso pode incluir práticas de trabalho a serem alteradas, experimentos a serem realizados ou novas ferramentas a serem usadas.

A Retrospectiva da Sprint é um componente essencial do Scrum, pois promove a melhoria contínua e a auto-organização, que são princípios centrais do Scrum. O Scrum Master geralmente facilita esta reunião, garantindo que todos os membros da equipe tenham a oportunidade de falar e que a discussão permaneça construtiva e focada.

[Guia Scrum - Versão 2020]

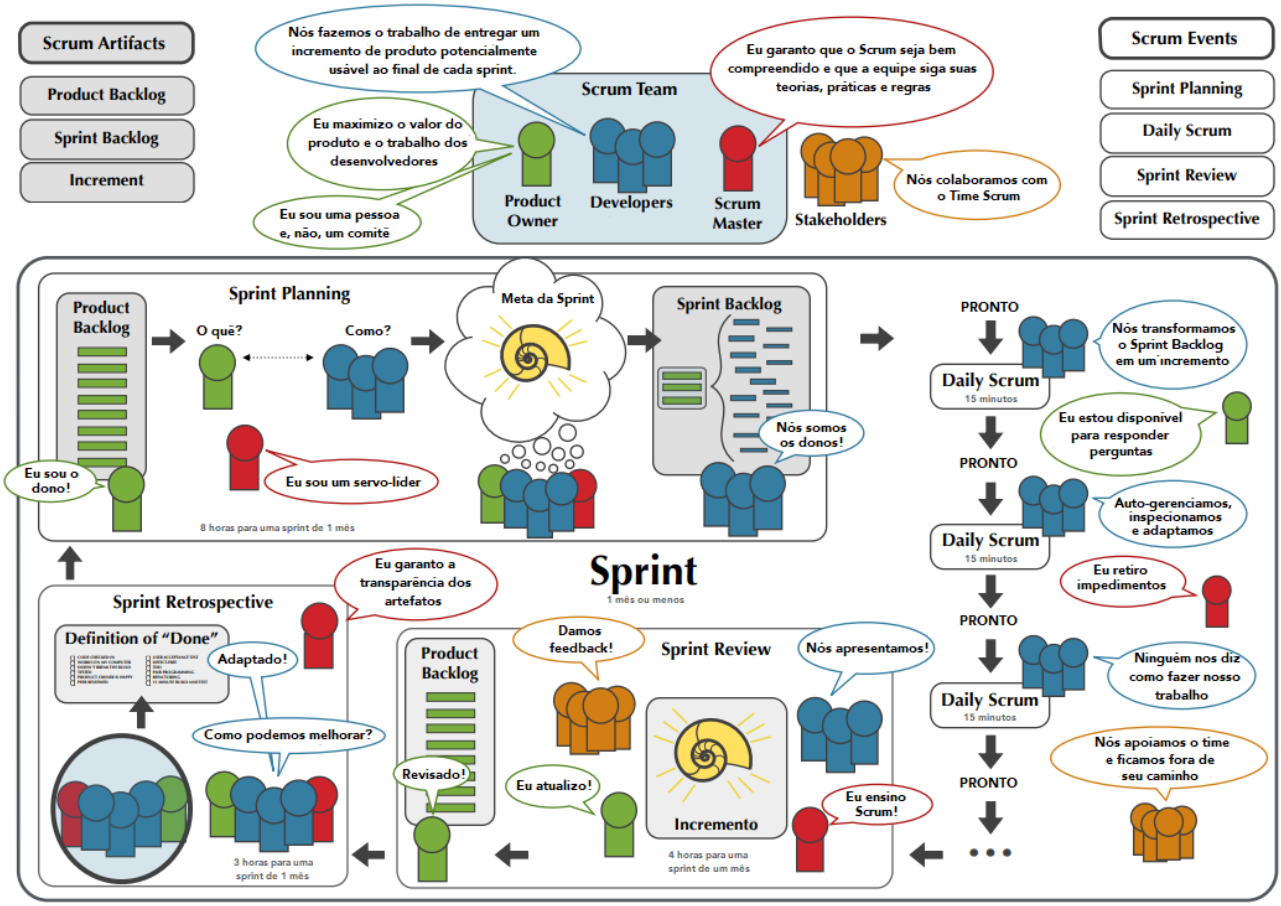
O propósito da Sprint Retrospective é planejar maneiras de aumentar a qualidade e a eficácia. O Scrum Team inspeciona como foi a última Sprint em relação a indivíduos, interações, processos, ferramentas e sua Definição de Pronto. Os elementos inspecionados geralmente variam com o domínio de trabalho. As suposições que os desviaram são identificadas e suas origens exploradas. O Scrum Team discute o que deu certo durante a Sprint, quais problemas
encontraram e como esses problemas foram (ou não) resolvidos. O Scrum Team identifica as mudanças mais úteis para melhorar sua eficácia. As melhorias mais impactantes são endereçadas o mais rápido possível. Essas podem até ser adicionadas ao Sprint Backlog para a próxima Sprint. A Sprint Retrospective conclui a Sprint. É limitada pelo Timebox de no máximo três horas para uma Sprint de um mês. Para Sprints mais curtas, o evento geralmente é mais curto.

A tabela a seguir resume os eventos do Scrum:

Evento	Descrição	Duração
Planejamento da Sprint	Reunião onde a equipe determina o que pode ser entregue no próximo Sprint	Para uma Sprint de duas semanas, até 4 horas.

	e como o trabalho necessário para entregar isso será realizado.	
Sprint	Período de tempo durante o qual um "Incremento" de trabalho utilizável e potencialmente lançável é criado.	Normalmente 2-4 semanas.
Reunião Diária	Uma reunião diária de 15 minutos onde a equipe se sincroniza sobre o progresso e identifica obstáculos.	15 minutos.
Revisão da Sprint	Reunião onde o Scrum Team e os stakeholders inspecionam o Incremento e adaptam o Product Backlog, se necessário.	Para uma Sprint de duas semanas, até 2 horas.
Retrospectiva da Sprint	Reunião onde a equipe inspeciona a si mesma e cria um plano para melhorias a serem aplicadas durante a próxima Sprint.	Para uma Sprint de duas semanas, até 1,5 horas.

Lembre-se de que a duração desses eventos é máxima e pode ser reduzida se a equipe achar adequado e a qualidade das discussões não for afetada.



Kanban

Kanban é uma metodologia ágil surgida no Japão, inicialmente utilizada pela Toyota na década de 1950 para gerenciar o fluxo de produção em suas fábricas de automóveis. No contexto do desenvolvimento de software, Kanban é uma abordagem que se concentra **no fluxo de trabalho e na entrega contínua de valor**.

O termo "Kanban" traduz-se literalmente como "cartão visual" ou "placa", que é uma referência à maneira como o sistema visualiza o fluxo de trabalho.

Antes de falarmos sobre as características do Kanban, é importante definirmos o que Kanban NÃO é. David J. Anderson – pioneiro do Kanban – acha que Kanban não é uma metodologias de desenvolvimento de software (conforme podemos ver nas declarações apresentadas abaixo). No entanto, é bastante comum ver algumas bancas o tratando como uma metodologia de desenvolvimento de software.

*"Kanban **is not a software development life cycle or project management methodology**! It is not a way of making software or running projects that make software!" – David J. Anderson*

*"There is **no kanban process for software development**. At least I am not aware of one. I have never published one" – David J. Anderson*

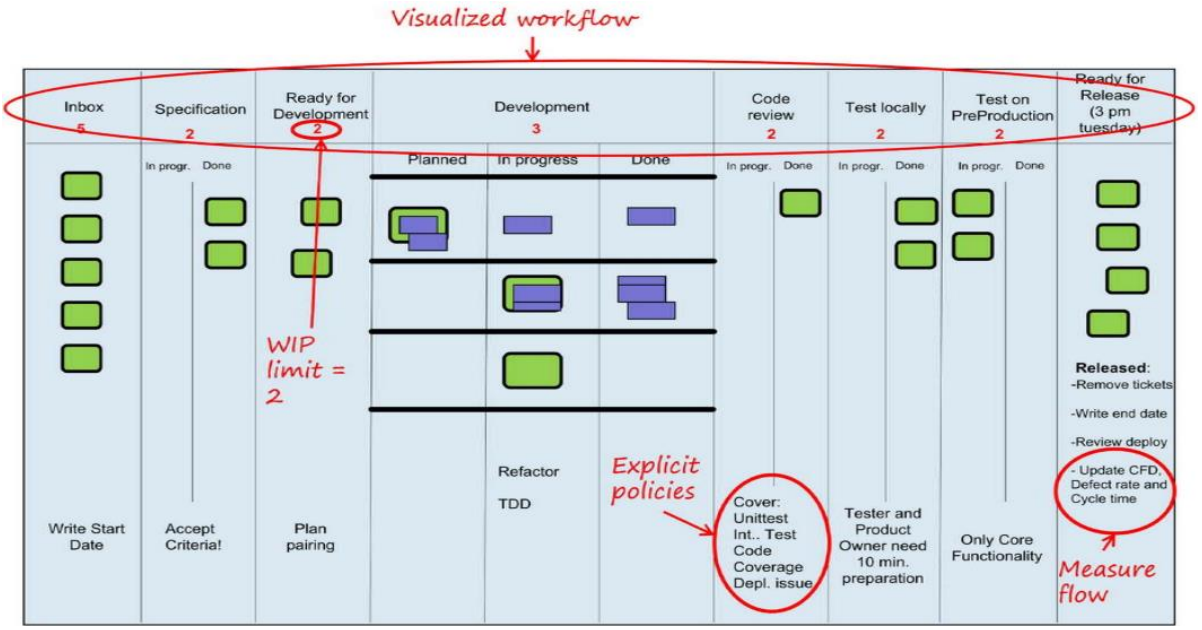
*"It is actually **not possible to develop with only Kanban**. The Kanban Method by itself does not contain practices sufficient to do product development" – David J. Anderson*

De forma geral, Kanban pode ser visto como um acelerador para a **condução de mudanças ou até mesmo um método para implantação de mudanças em uma organização**. Ele não prescreve papéis, práticas ou cerimônias específicas (como faz, por exemplo, Scrum). Em vez disso, ele oferece uma série de princípios para otimizar o fluxo e a geração de valor dos sistemas de entrega de software

As características principais do Kanban são:

- **Visualização do fluxo de trabalho**: Kanban utiliza um quadro (Kanban board) para visualizar o estado de todas as tarefas. O quadro é dividido em colunas, que representam as diferentes etapas do fluxo de trabalho.
- **Limitação do trabalho em andamento (WIP - Work in Progress)**: Para evitar a sobrecarga de trabalho e promover a eficiência, o Kanban limita a quantidade de trabalho que pode estar em andamento em cada estágio do fluxo de trabalho.
- **Fluxo de trabalho "pull"**: Em vez de empurrar o trabalho para a próxima etapa do processo quando ele é concluído, o trabalho é "puxado" (pull) para a próxima etapa quando há capacidade disponível.
- **Gestão do fluxo**: Monitorar e otimizar o fluxo de trabalho é fundamental para o Kanban. O tempo que uma tarefa leva para mover-se do início ao fim (lead time) é uma métrica-chave.
- **Políticas explícitas**: As regras do processo de trabalho são claramente definidas e visíveis para todos.
- **Melhoria contínua**: Como outras abordagens ágeis, o Kanban se concentra na inspeção e adaptação para melhorar constantemente.

Veja um exemplo de Quadro Kanban:



Kanban versus XP e Scrum

Scrum, XP (Extreme Programming) e Kanban são todos métodos ágeis, mas têm diferenças significativas na forma como abordam o desenvolvimento de software. Veja alguns ponto onde Kanban difere das outras abordagens:

- **Ciclos de tempo (Sprints vs Fluxo Contínuo)**: Scrum trabalha com Sprints, que são ciclos de tempo fixos para o trabalho ser concluído (geralmente 2-4 semanas). O XP também adota a ideia de iterações. O Kanban, por outro lado, se concentra no **fluxo contínuo de trabalho** e não tem iterações fixas.
- **Papéis da equipe**: Scrum tem papéis definidos (Product Owner, Scrum Master, Development Team), enquanto XP tem papéis adicionais (como o Tracker e o Coach), cada um com responsabilidades específicas. Em contraste, o **Kanban não define papéis específicos**, embora possam ser adotados se necessário.
- **Estimativa e Planejamento**: Scrum e XP usam estimativas para planejar o que será realizado em cada Sprint ou iteração. Em contrapartida, o Kanban não exige estimativas e, em vez disso, se concentra em acompanhar o tempo de ciclo das tarefas para prever a conclusão.
- **Trabalho em Andamento (WIP)**: Uma das principais características do Kanban é a limitação do trabalho em andamento. Scrum e XP limitam indiretamente o WIP ao determinar a quantidade de trabalho planejado para cada iteração, mas não têm a mesma ênfase estrita na limitação do WIP em cada etapa do fluxo de trabalho.
- **Mudanças durante o ciclo**: No Scrum, uma vez que uma Sprint começou, as mudanças normalmente não são feitas naquilo que está sendo desenvolvido durante essa Sprint. XP é um pouco mais flexível, mas ainda foca nas iterações. Em contraste, o Kanban permite a re-

priorização de itens de trabalho em qualquer ponto do tempo, desde que o limite de WIP não seja ultrapassado.

- **Abordagem para a melhoria contínua**: Todas as três metodologias enfatizam a melhoria contínua, mas abordam de maneira diferente. Scrum tem a Retrospectiva da Sprint, XP tem várias práticas (como a programação em pares e a integração contínua) e Kanban enfatiza a melhoria contínua do fluxo de trabalho visualizando-o e limitando o WIP.

Princípios e Práticas

A tabela a seguir resume um conjunto de princípios ou "restrições" do Kanban:

PRINCÍPIOS OU RESTRIÇÕES DO KANBAN
- Comece com o que você tem hoje;
- Estimule a liderança em todos os níveis da organização;
- Visualize cada passo em sua cadeia de valor, do conceito geral até o software que se possa lançar;
- Limite o Trabalho em Progresso (WIP), restringindo o total de trabalho permitido para cada estágio;
- Torne explícitas as políticas sendo seguidas;
- Meça e gerencie o fluxo, para poder tomar decisões bem embasadas, além de visualizar as consequências;
- Identifique oportunidades de melhorias, na qual a melhoria contínua é responsabilidade de todos.

Já a tabela abaixo resume um conjunto de práticas do Kanban:

PRÁTICAS DO KANBAN
- Implemente mecanismos de feedback;
- Gerencie e meça o fluxo de trabalho;
- Visualize o processo;
- Limite o WIP (Work In Progress);
- Torne as políticas dos processos explícitas;
- Melhore colaborativamente e com métodos científicos.

QUESTÕES ESTRATÉGICAS

Nesta seção, apresentamos e comentamos uma amostra de questões objetivas selecionadas estrategicamente: são questões com nível de dificuldade semelhante ao que você deve esperar para a sua prova e que, em conjunto, abordam os principais pontos do assunto.

A ideia, aqui, não é que você fixe o conteúdo por meio de uma bateria extensa de questões, mas que você faça uma boa revisão global do assunto a partir de, relativamente, poucas questões

1. **(FGV / MPE-MS – 2013)** Considerando a caracterização de agilidade e processo de desenvolvimento ágil, segundo Pressman, analise as afirmativas a seguir.

   I. Um processo ágil de software deve ser incrementalmente adaptável.
   II. Um processo ágil de software permite que as pessoas e a equipe se moldem a ele com facilidade.
   III. Os conceitos ágeis são efetivos, pois diminuem a imprevisibilidade sistêmica ao enfatizar entregas em prazos curtos.

   a) se somente a afirmativa I estiver correta.
   b) se somente a afirmativa II estiver correta.
   c) se somente a afirmativa III estiver correta.
   d) se somente as afirmativas I e II estiverem corretas.
   e) se todas as afirmativas estiverem corretas.

   Comentários:

   (I) Correto, idealmente ele deve se adaptar de forma incremental; (II) Errado, a agilidade não tem relação com a facilidade da equipe de se moldar; (III) Errado, mas questão polêmica! Eu acredito que os conceitos ágeis diminuem a imprevisibilidade. Já alguns argumentam que conceitos ágeis não diminuem a imprevisibilidade, na verdade eles aceitam que a imprevisibilidade é inevitável e, dessa forma, provêm métodos de se adaptar às mudanças rapidamente.

   Gabarito: A

2. **(FGV / PGE-RO – 2015)** Durante 5 anos gerenciando o desenvolvimento de sistemas de informação, Claudia teve que lidar com diversas insatisfações de seus usuários pois os sistemas não atendiam as suas necessidades. Claudia decidiu, então, implantar métodos ágeis de desenvolvimento e definiu os seguintes princípios:

   I. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

   II. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através da documentação.

   III. Simplicidade é essencial.

Dentre os princípios definidos por Claudia, o que infringe os princípios do manifesto para Desenvolvimento Ágil de Software é o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente I e III;
e) I, II e III.

Comentários:

NÓS SEGUIMOS ESSES PRINCÍPIOS...
Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.

(I) Correto, mudanças são sempre bem-vindas; (II) Errado, o método mais eficiente é frente-a-frente; (III) Correto. Como a questão pede os princípios que infringem o manifesto ágil, trata-se da segunda opção.

Gabarito: B

3. **(FGV / BANESTES – 2018)** Um dos valores relacionados ao ambiente ágil de desenvolvimento é:

a) documentação abrangente mais que software funcional;
b) negociação de contratos mais que colaboração do cliente;
c) processos e ferramentas mais que indivíduos e iterações;
d) rapidez na construção mais que excelência técnica;
e) responder a mudanças mais que seguir um plano.

Comentários:

(1) Indivíduos e interações acima de processos e ferramentas; (2) Software em funcionamento acima de documentação abrangente; (3) Colaboração com o cliente acima de negociação de contratos; (4) Responder a mudanças acima de seguir fielmente um plano.

Gabarito: E

4. **(FGV / BANESTES – 2018)** Com relação aos valores relacionados ao desenvolvimento ágil de software, NÃO se pode incluir:

a) colaboração do cliente mais que negociação de contratos;
b) indivíduos e iterações mais que processos e ferramentas;
c) rapidez na construção mais que excelência técnica;
d) responder a mudanças mais que seguir um plano;
e) software funcional mais que documentação abrangente.

Comentários:

(1) Indivíduos e interações acima de processos e ferramentas; (2) Software em funcionamento acima de documentação abrangente; (3) Colaboração com o cliente acima de negociação de contratos; (4) Responder a mudanças acima de seguir fielmente um plano. Notem que rapidez na construção mais que excelência técnica não é um dos valores do manifesto ágil.

Gabarito: C

5. **(FGV / AL-RO – 2018)** Para o desenvolvimento do Sistema de Informações ao Cidadão (SIC), foi decidida a utilização de uma metodologia ágil. Segundo o Manifesto Ágil, esta decisão indica que foi dado maior valor:

a) aos processos e ferramentas.
b) à resposta a modificações.
c) à documentação abrangente.
d) à negociação do contrato.
e) ao cumprimento do plano.

Comentários:

(1) Indivíduos e interações acima de processos e ferramentas; (2) Software em funcionamento acima de documentação abrangente; (3) Colaboração com o cliente acima de negociação de contratos; (4) Responder a mudanças acima de seguir fielmente um plano.

Gabarito: B

6. **(VUNESP / Câmara de Piracicaba - SP – 2019)** Um dos processos ágeis de desenvolvimento de software é a programação extrema (extreme programming – XP), cuja fase ou atividade inicial é composta pela descrição dos cenários (características e funcionalidades) requisitadas para o software a ser desenvolvido. Essa atividade recebe a denominação de:

a) métodos práticos.
b) histórias de usuário.
c) estruturas de apoio.
d) classes de projeto.
e) artefatos de usuário

Comentários:

No Extreme Programming (XP), as "histórias de usuário" são usadas para descrever características e funcionalidades do software do ponto de vista do usuário final. Elas ajudam a equipe de desenvolvimento a entender melhor as necessidades do usuário e a orientar o desenvolvimento para atender a essas necessidades.

Gabarito: B

7. **(FGV / Prefeitura de Paulínia - SP – 2016)** A empresa de desenvolvimento de sistemas “Inovation” tem ampla experiência no mercado e, até o momento, utilizou diversos modelos de ciclo de vida para o desenvolvimento de sistemas. A “Inovation” já recebeu diversas reclamações dos seus clientes por causa da demora em apresentar alguma tela em funcionamento, bem como da falta de envolvimento dos clientes no desenvolvimento. A empresa, assim, decidiu passar a utilizar um novo modelo de ciclo de vida. Esta decisão visa aproveitar a grande experiência de sua equipe e trazer o cliente para a equipe de desenvolvimento, com iterações de desenvolvimento extremamente curtas. Qualquer membro da equipe implementa parte do código, que pode ser evoluído por qualquer outro membro.
O novo modelo adotado pela “Inovation” é denominado:

a) Extreme Programming (XP).
b) Modelo V.
c) Evolutivo.
d) Incremental.
e) Espiral.

Comentários:

A empresa demora a apresentar incrementos visíveis para o usuário e peca na falta de envolvimento com os clientes. Ela deseja utilizar um novo modelo que traz o cliente para perto da equipe e realiza iterações de desenvolvimento extremamente curtas (olha a dica!). Por fim, ela quase dá a resposta para a questão afirmando que qualquer membro da equipe implementa parte do código, que pode ser evoluído por qualquer outro membro.

A questão apresentou diversas características do modelo Extreme Programming (XP):

PRÁTICAS	DESCRIÇÃO
PEQUENOS RELEASES	O conjunto mínimo útil de funcionalidade que agrega valor ao negócio é desenvolvido primeiro. Releases do sistema são frequentes e adicionam funcionalidade incrementalmente ao primeiro release.
PROPRIEDADE COLETIVA	Os pares de desenvolvedores trabalham em todas as áreas do sistema, de tal maneira que não se formem ilhas de conhecimento, com todos os desenvolvedores de posse de todo o código. Qualquer um pode mudar qualquer coisa.
CLIENTE ON-SITE	Um representante do usuário final do sistema deve estar disponível em tempo integral para apoiar a equipe. No XP, o cliente é um membro da equipe de desenvolvimento e é responsável por trazer os requisitos do sistema.

Gabarito: A

8. **(FGV / Câmara Municipal do Recife-PE – 2014)** Uma das práticas do método ágil XP (eXtreme Programming) é:

a) documentação extensiva;
b) prototipação;
c) ciclos longos de desenvolvimento;
d) desenvolvimento orientado a testes (TDD);
e) utilização de todos os artefatos do RUP.

Comentários:

O TDD é uma prática XP em que se escreve o teste antes de escrever o código do componente.

Gabarito: D

9. **(VUNESP / Prefeitura de Campinas – SP – 2019)** No método ágil Scrum, há um artefato denominado backlog, aplicado a diversas etapas do método. Em particular, o backlog do produto corresponde a:

a) um conjunto de todos os produtos de software já desenvolvidos com o uso do Scrum.
b) uma lista das funcionalidades a serem implementadas no produto.
c) uma relação dos analistas envolvidos no desenvolvimento do produto.
d) um conjunto de normas seguidas pela empresa proprietária do produto.
e) uma relação dos futuros usuários do produto.

Comentários:

O Backlog do Produto é uma lista das funcionalidades a serem implementadas no produto – nenhum dos outros itens faz qualquer sentido.

Gabarito: B

10. **(FGV / COMPESA – 2018)** O SCRUM é um framework para gerenciamento de projetos complexos, sendo um dos métodos ágeis mais populares do mundo. Uma das dinâmicas definidas no SCRUM é a retrospectiva. Assinale a opção que melhor descreve o objetivo da retrospectiva definida no SCRUM.

a) Planejar medidas que possam trazer, no próximo Sprint, melhorias relacionadas à colaboração entre as pessoas, processos ou ferramentas.
b) Inspecionar o resultado do trabalho realizado em um Sprint e ajustar o Backlog do produto, se necessário.
c) Rever as prioridades dos itens que compõem o Backlog do produto.
d) Planejar como o time construirá as funcionalidades definidas para um Sprint com base nos resultados obtidos nos Sprints anteriores.
e) Disseminar conhecimento sobre o que foi feito no dia anterior para poder priorizar o trabalho a ser realizado no dia que se inicia.

Comentários:

A Retrospectiva da Sprint (Proporcional a 3 horas) é uma chance para o Scrum Team inspecionar a si próprio e criar um plano de melhorias para a próxima sprint. Ela

inspeciona como foi a última sprint em relação às pessoas, às relações, aos processos e às ferramentas. Pode identificar e ordenar os itens que se tornaram potenciais de melhorias e cria um plano para implementar melhorias no trabalho.

Dessa forma, temos que: (a) Correto; (b) Errado, isso é a Revisão da Sprint; (c) Errado, isso é Planejamento da Sprint; (d) Errado, isso é Planejamento da Spring; (e) Errado, isso é Reunião Diária.

Gabarito: A

11. **(FGV / AL-RO – 2018)** Para o desenvolvimento do Sistema de Informações ao Cidadão (SIC), foi decidida a utilização de uma metodologia ágil. Segundo o Manifesto Ágil, esta decisão indica que foi dado maior valor:

a) aos processos e ferramentas.
b) à resposta a modificações.
c) à documentação abrangente.
d) à negociação do contrato.
e) ao cumprimento do plano

Comentários:

Segundo o Manifesto Ágil, esta decisão indica que foi dado maior valor à resposta a modificações. Vamos relembrar: (1) os indivíduos e suas interações acima de procedimentos e ferramentas; (2) o funcionamento do software acima de documentação abrangente; (3) a colaboração com o cliente acima da negociação e contrato; (4) a capacidade de resposta a mudanças acima de um plano pré-estabelecido.

Gabarito: B

12. **(FGV / TJ-SC – 2015)** O SCRUM, processo para o desenvolvimento de software ágil, estrutura-se sobre:

a) plan, documentaton, test;
b) roles, artifacts, activities.
c) requisites, code, products;
d) client team, development team, deliverables;

e) interface, data, code.

Comentários:

O Scrum trata de Papeis (Roles), Artifacts (Artefatos) e Eventos (Activities). A tradução desse último termo ficou péssima infelizmente.

Gabarito: B

13. **(FGV / BANESTES – 2021)** Observe o quadro comparativo a seguir, publicado em sites ligados ao estudo e à investigação de diferentes estratégias/metodologias para implementar um sistema ágil de desenvolvimento ou gestão de projetos.

Aspectos	X	Y
Ritmo	Sprints	Fluxo contínuo
Funções	Funções bem definidas	Sem funções necessárias
Entregas	Final de cada sprint	Entrega contínua
Mudanças	Evitar durante sprint	A qualquer momento

É correto identificar que X e Y representam, respectivamente:

a) Crystal e Scrum;
b) Extreme Programming e Crystal;
c) Kanban e Lean;
d) Lean e Extreme Programming;
e) Scrum e Kanban.

Comentários:

Scrum é iterativo e incremental, e o nome que se dá a Iterações no Scrum é Sprints. Já o Kanban é de fluxo contínuo e sem papeis ou funções definidos.

Gabarito: E

14. **(FGV / TJ-GO – 2014)** Scrum e Kanban são metodologias de gerenciamento de projetos de software populares entre praticantes do desenvolvimento ágil. Um aspecto de divergência entre as duas metodologias é:

a) processo incremental;
b) processo iterativo;
c) uso de quadro de tarefas;
d) apresentação do estágio de desenvolvimento de uma tarefa;
e) valorização de feedback.

Comentários:

O Kanban não é necessariamente iterativo como é o Scrum! Na prática, esse assunto é rodeado de polêmicas e divergências. Já os outros itens tratam de convergências!

Gabarito: B

15. **(FGV / MPE-MS – 2013)** Kanban é um dos métodos ágeis mais recentes e sofreu grande influência do movimento "Lean", surgido nos anos 1980. São práticas comuns a esse método:

a) limitar o WIP (Work In Progress) e uma visualização explícita do fluxo de trabalho.
b) integração Contínua e gerenciamento de configuração.
c) limitar o WIP (Work In Progress) e gerenciamento de configuração.
d) gerenciar o fluxo de trabalho e manter estimativas previamente definidas.
e) melhoria contínua e nunca limitar o WIP para evitar folgas no sistema de trabalho.

Comentários:

As práticas são: implemente mecanismos de feedback; gerencie e meça o fluxo de trabalho; visualize o processo; limite o WIP (Work In Progress); torne as políticas dos processos explícitas; melhore colaborativamente e com métodos científicos.

Gabarito: A

16. **(FCC / SEFAZ-AP – 2022)** No modelo ágil de gestão de projetos Scrum, um dos conceitos mais importantes é o:

a) *product backlog*, um diagrama que mostra a quantidade de trabalho a fazer e a quantidade feita ao longo do tempo.

b) *daily sprint meeting*, uma reunião de cerca de uma hora para se obter ideias para solução de problemas encontrados no projeto.

c) *sprint*, que consiste em um ciclo de desenvolvimento que, em geral, vai de duas semanas a um mês.

d) *pair programming*, prática que consiste na programação realizada por duas pessoas em cada computador.

e) *test driven development*, prática que consiste em definir e implementar os testes de unidade antes da programação.

Comentários:

(a) Errado, a questão trata do Gráfico de Burndown; (b) Errado, a descrição se aproxima mais da Sprint Retrospective e ela dura um máximo de três horas; (c) Correto, elas são eventos de um mês ou menos em que ideias se tornam efetivamente valor para o cliente; (d) Errado, são duas pessoas em um único computador e isso não é uma prática preconizada pelo Scrum; (e) Errado, isso também não é uma prática preconizada pelo Scrum.

Gabarito: Letra C

17. **(FCC / TRF - 3ª REGIÃO – 2019)** SCRUM atende aos princípios do Manifesto Ágil porque:

a) pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
b) não aceita mudanças nos requisitos durante o desenvolvimento e por isso as entregas são mais ágeis.
c) as entregas ocorrem sempre no prazo, nunca adiantadas ou atrasadas.
d) mais importante que a motivação dos desenvolvedores é a disciplina gerencial imposta que organiza e agiliza o desenvolvimento.
e) não admite a comunicação direta entre os desenvolvedores, pessoalmente. Isso só pode ser feito por intermédio de um gerente ou coordenador.

Comentários:

(a) Correto, indivíduos e interações valorizam mais processos e ferramentas; (b) Errado, ele aceita mudanças nos requisitos; (c) Errado, entregas podem ser adiadas ou atrasadas; (d) Errado, indivíduos são mais valorizados do que processos; (e) Errado, a comunicação direta ocorre frequentemente.

Gabarito: Letra A

18. (FCC / TRF - 3ª REGIÃO – 2019) A Reunião Diária do Scrum é:

a) executada no final da Sprint para inspecionar o incremento e adaptar o Backlog do Produto, se necessário.
b) um time-boxed de 15 minutos, durante o qual um "Pronto", versão incremental potencialmente utilizável do produto, é criado.
c) uma oportunidade para o Time Scrum inspecionar a si próprio e criar um plano para melhorias a serem aplicadas na próxima Sprint.
d) um time-boxed de 15 minutos, para que o Time de Desenvolvimento possa sincronizar as atividades e criar um plano para as próximas 24 horas.
e) um time-boxed de 60 minutos, durante o qual os produtos de uma Sprint são definidos.

Comentários:

(a) Errado, essa seria a Revisão da Sprint; (b) Errado, ela realmente dura 15 minutos, mas não busca criar uma versão potencialmente utilizável do produto; (c) Errado, essa seria a Retrospectiva da Sprint; (d) Correto; (e) Errado, ela dura 15 minutos e não busca definir os produtos de uma sprint.

Gabarito: Letra D

19. (CESGRANRIO / Banco da Amazônia – 2021)

"O Scrum é um arcabouço que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos."

SCHWABER, K. ; SUTHERLAND, J. O Guia do Scrum, O Guia Definitivo para o Scrum: As Regras do Jogo. Nov. 2020. p 3. Adaptado.

Para cumprir seu objetivo, o Scrum se baseia em quatro eventos formais, contidos dentro de um evento de maior duração: a Sprint. Tais eventos formais implementam os três pilares empíricos do Scrum, que são:

a) compromisso, abertura e adaptação
b) respeito, coragem e foco
c) respeito, inspeção e adaptação
d) transparência, compromisso e respeito
e) transparência, inspeção e adaptação

Comentários:

Os três pilares são o TIA (Transparência, Inspeção e Adaptação).

Gabarito: E

20. (CESGRANRIO / Banco da Amazônia – 2014)

Uma prática que NÃO é adotada por Extreme Programming (XP) é

a) usar duas pessoas trabalhando juntas em um único computador para produzir todo o código que será enviado para a produção.

b) criar os testes antes do código que será testado.

c) refatorar frequentemente, e ao longo de todo o projeto, o código produzido pelos desenvolvedores.

d) integrar continuamente o código recém-produzido com o código existente no repositório.

e) variar a duração de cada iteração durante todo o projeto para acomodar eventuais mudanças de prioridade dos requisitos, definidas pelo cliente.

Comentários:

Lembrem-se: tempo fixo e escopo variável e, não, o contrário! A última opção afirma que o tempo muda de acordo com o escopo/mudanças e isso não é verdade.

Gabarito: E

21. (CESGRANRIO / Banco da Amazônia – 2018)

O Manifesto Ágil se tornou um marco da Engenharia de Software, chamando a atenção de que vários processos propostos de forma independente tinham valores em comum. Além disso, foram definidos 12 princípios. Entre eles, figura o seguinte princípio:

a) cada pessoa em um projeto deve ter sua função predeterminada para acelerar o desenvolvimento em conjunto.
b) a contínua atenção à simplicidade do trabalho feito aumenta a agilidade.
c) software funcionando é a medida primária de progresso.
d) os indivíduos, clientes e desenvolvedores, são mais importantes que processos e ferramentas.

e) o software funcional emerge de times auto-organizáveis.

Comentários:

Nenhum desses faz parte dos princípios, exceto: software funcionando é a medida primária de progresso.

Gabarito: C

QUESTIONÁRIO DE REVISÃO E APERFEIÇOAMENTO

A ideia do questionário é elevar o nível da sua compreensão no assunto e, ao mesmo tempo, proporcionar uma outra forma de revisão de pontos importantes do conteúdo, a partir de perguntas que exigem respostas subjetivas.

São questões um pouco mais desafiadoras, porque a redação de seu enunciado não ajuda na sua resolução, como ocorre nas clássicas questões objetivas.

O objetivo é que você realize uma auto explicação mental de alguns pontos do conteúdo, para consolidar melhor o que aprendeu ;)

Além disso, as questões objetivas, em regra, abordam pontos isolados de um dado assunto. Assim, ao resolver várias questões objetivas, o candidato acaba memorizando pontos isolados do conteúdo, mas muitas vezes acaba não entendendo como esses pontos se conectam.

Assim, no questionário, buscaremos trazer também situações que ajudem você a conectar melhor os diversos pontos do conteúdo, na medida do possível.

É importante frisar que não estamos adentrando em um nível de profundidade maior que o exigido na sua prova, mas apenas permitindo que você compreenda melhor o assunto de modo a facilitar a resolução de questões objetivas típicas de concursos, ok?

Nosso compromisso é proporcionar a você uma revisão de alto nível!

Vamos ao nosso questionário:

Perguntas

1. O que são métodos ágeis?
2. Quais são os quatro valores fundamentais do Manifesto Ágil?
3. O que é Extreme Programming (XP)?
4. O que é uma história do usuário?
5. O que é Scrum?
6. Quem é o Product Owner no Scrum?
7. O que é o Product Backlog?
8. O que é uma Sprint?
9. O que é feito durante o planejamento da Sprint?
10. O que é o Daily Scrum?
11. O que é a revisão da Sprint?
12. O que é a retrospectiva da Sprint?
13. O que é o Planning Poker?
14. O que é Kanban?
15. Como o Scrum difere do Kanban?
16. O que é a definição de pronto (Definition of Ready)?
17. O que é a definição de concluído (Definition of Done)?
18. O que é um Gráfico de Burndown?
19. O que é um incremento do produto (Product Increment)?
20. O que é WIP (Work in Progress) no Kanban?

Perguntas e Respostas

1. O que são métodos ágeis?
   Resposta: Os métodos ágeis são abordagens de desenvolvimento de software que enfatizam a entrega contínua de valor, a colaboração com os clientes, a adaptabilidade e a melhoria contínua.
2. Quais são os quatro valores fundamentais do Manifesto Ágil?
   Resposta: Os quatro valores são: indivíduos e interações mais que processos e ferramentas, software funcionando mais que documentação abrangente, colaboração com o cliente mais que negociação de contratos e responder a mudanças mais que seguir um plano.
3. O que é Extreme Programming (XP)?
   Resposta: XP é um método ágil que enfatiza a excelência técnica, a comunicação direta e a satisfação do cliente.

4. O que é uma história do usuário?
   Resposta: Uma história do usuário é uma descrição de uma funcionalidade do ponto de vista do usuário, geralmente no formato: "Como [tipo de usuário], quero [alguma ação], para que eu possa [algum benefício ou resultado]".
5. O que é Scrum?
   Resposta: Scrum é um método ágil que utiliza ciclos fixos de trabalho, chamados Sprints, e enfatiza a transparência, inspeção e adaptação.
6. Quem é o Product Owner no Scrum?
   Resposta: No Scrum, o Product Owner é a pessoa responsável por maximizar o valor do produto, gerenciando e priorizando o Product Backlog.
7. O que é o Product Backlog?
   Resposta: O Product Backlog é uma lista priorizada de itens ou funcionalidades desejadas para um produto, mantida pelo Product Owner.
8. O que é uma Sprint?
   Resposta: Uma Sprint é um ciclo fixo de trabalho no Scrum, geralmente durando 2-4 semanas.
9. O que é feito durante o planejamento da Sprint?
   Resposta: Durante o planejamento da Sprint, a equipe seleciona itens do Product Backlog para trabalhar durante a Sprint e cria um plano para entregar esses itens.
10. O que é o Daily Scrum?
    Resposta: O Daily Scrum é uma reunião diária de 15 minutos para a equipe sincronizar o progresso e planejar o trabalho do dia.
11. O que é a revisão da Sprint?
    Resposta: A revisão da Sprint é uma reunião no final de cada Sprint para inspecionar o incremento do produto e adaptar o Product Backlog.
12. O que é a retrospectiva da Sprint?
    Resposta: A retrospectiva da Sprint é uma reunião no final de cada Sprint para a equipe refletir sobre o que foi bem, o que pode ser melhorado e como implementar as melhorias na próxima Sprint.
13. O que é o Planning Poker?
    Resposta: O Planning Poker é uma técnica usada para estimar o esforço de trabalho, onde cada membro da equipe "aposta" com um valor que representa a sua estimativa de esforço.
14. O que é Kanban?
    Resposta: Kanban é um método ágil que visualiza o fluxo de trabalho, limita o trabalho em andamento e incentiva a melhoria contínua.

15. Como o Scrum difere do Kanban?
    Resposta: Scrum tem Sprints, papéis definidos e planejamento baseado em tempo. Kanban flui continuamente, não tem papéis definidos e o planejamento é baseado na capacidade.
16. O que é a definição de pronto (Definition of Ready)?
    Resposta: É o conjunto de critérios que um item do backlog deve atender antes que possa ser selecionado para uma Sprint.
17. O que é a definição de concluído (Definition of Done)?
    Resposta: É o conjunto de critérios que um item deve atender para ser considerado concluído.
18. O que é um Gráfico de Burndown?
    Resposta: O Gráfico de Burndown é um gráfico que mostra a quantidade de trabalho restante em relação ao tempo.
19. O que é um incremento do produto (Product Increment)?
    Resposta: É a versão utilizável e potencialmente entregável do produto ao final de cada Sprint.
20. O que é WIP (Work in Progress) no Kanban?
    Resposta: WIP é a quantidade de trabalho atualmente em andamento. No Kanban, geralmente há limites no WIP para evitar sobrecargas.

LISTA DE QUESTÕES ESTRATÉGICAS

1. **(FGV / MPE-MS – 2013)** Considerando a caracterização de agilidade e processo de desenvolvimento ágil, segundo Pressman, analise as afirmativas a seguir.

   I. Um processo ágil de software deve ser incrementalmente adaptável.
   II. Um processo ágil de software permite que as pessoas e a equipe se moldem a ele com facilidade.
   III. Os conceitos ágeis são efetivos, pois diminuem a imprevisibilidade sistêmica ao enfatizar entregas em prazos curtos.

   a) se somente a afirmativa I estiver correta.
   b) se somente a afirmativa II estiver correta.
   c) se somente a afirmativa III estiver correta.
   d) se somente as afirmativas I e II estiverem corretas.
   e) se todas as afirmativas estiverem corretas.

2. **(FGV / PGE-RO – 2015)** Durante 5 anos gerenciando o desenvolvimento de sistemas de informação, Claudia teve que lidar com diversas insatisfações de seus usuários pois os sistemas não atendiam as suas necessidades. Claudia decidiu, então, implantar métodos ágeis de desenvolvimento e definiu os seguintes princípios:

   I. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

   II. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através da documentação.

   III. Simplicidade é essencial.

   Dentre os princípios definidos por Claudia, o que infringe os princípios do manifesto para Desenvolvimento Ágil de Software é o que se afirma em:

   a) somente I;
   b) somente II;
   c) somente III;
   d) somente I e III;
   e) I, II e III.

3. **(FGV / BANESTES – 2018)** Um dos valores relacionados ao ambiente ágil de desenvolvimento é:

   a) documentação abrangente mais que software funcional;
   b) negociação de contratos mais que colaboração do cliente;
   c) processos e ferramentas mais que indivíduos e iterações;
   d) rapidez na construção mais que excelência técnica;
   e) responder a mudanças mais que seguir um plano.

4. **(FGV / BANESTES – 2018)** Com relação aos valores relacionados ao desenvolvimento ágil de software, NÃO se pode incluir:

   a) colaboração do cliente mais que negociação de contratos;
   b) indivíduos e iterações mais que processos e ferramentas;
   c) rapidez na construção mais que excelência técnica;
   d) responder a mudanças mais que seguir um plano;
   e) software funcional mais que documentação abrangente.

5. **(FGV / AL-RO – 2018)** Para o desenvolvimento do Sistema de Informações ao Cidadão (SIC), foi decidida a utilização de uma metodologia ágil. Segundo o Manifesto Ágil, esta decisão indica que foi dado maior valor:

   a) aos processos e ferramentas.
   b) à resposta a modificações.
   c) à documentação abrangente.
   d) à negociação do contrato.
   e) ao cumprimento do plano.

6. **(VUNESP / Câmara de Piracicaba - SP – 2019)** Um dos processos ágeis de desenvolvimento de software é a programação extrema (extreme programming – XP), cuja fase ou atividade inicial é composta pela descrição dos cenários (características e funcionalidades) requisitadas para o software a ser desenvolvido. Essa atividade recebe a denominação de:

   a) métodos práticos.
   b) histórias de usuário.
   c) estruturas de apoio.
   d) classes de projeto.
   e) artefatos de usuário

7. **(FGV / Prefeitura de Paulínia - SP – 2016)** A empresa de desenvolvimento de sistemas “Inovation” tem ampla experiência no mercado e, até o momento, utilizou diversos modelos de ciclo de vida para o desenvolvimento de sistemas.
   A “Inovation” já recebeu diversas reclamações dos seus clientes por causa da demora em apresentar alguma tela em funcionamento, bem como da falta de envolvimento dos clientes no desenvolvimento. A empresa, assim, decidiu passar a utilizar um novo modelo de ciclo de vida. Esta decisão visa aproveitar a grande experiência de sua equipe e trazer o cliente para a equipe de desenvolvimento, com iterações de desenvolvimento extremamente curtas. Qualquer membro da equipe implementa parte do código, que pode ser evoluído por qualquer outro membro.
   O novo modelo adotado pela “Inovation” é denominado:

   a) Extreme Programming (XP).
   b) Modelo V.
   c) Evolutivo.
   d) Incremental.
   e) Espiral.

8. **(FGV / Câmara Municipal do Recife-PE – 2014)** Uma das práticas do método ágil XP (eXtreme Programming) é:

a) documentação extensiva;
b) prototipação;
c) ciclos longos de desenvolvimento;
d) desenvolvimento orientado a testes (TDD);
e) utilização de todos os artefatos do RUP.

9. **(VUNESP / Prefeitura de Campinas – SP – 2019)** No método ágil Scrum, há um artefato denominado backlog, aplicado a diversas etapas do método. Em particular, o backlog do produto corresponde a:

a) um conjunto de todos os produtos de software já desenvolvidos com o uso do Scrum.
b) uma lista das funcionalidades a serem implementadas no produto.
c) uma relação dos analistas envolvidos no desenvolvimento do produto.
d) um conjunto de normas seguidas pela empresa proprietária do produto.
e) uma relação dos futuros usuários do produto.

10. **(FGV / COMPESA – 2018)** O SCRUM é um framework para gerenciamento de projetos complexos, sendo um dos métodos ágeis mais populares do mundo. Uma das dinâmicas definidas no SCRUM é a retrospectiva. Assinale a opção que melhor descreve o objetivo da retrospectiva definida no SCRUM.

a) Planejar medidas que possam trazer, no próximo Sprint, melhorias relacionadas à colaboração entre as pessoas, processos ou ferramentas.
b) Inspecionar o resultado do trabalho realizado em um Sprint e ajustar o Backlog do produto, se necessário.
c) Rever as prioridades dos itens que compõem o Backlog do produto.
d) Planejar como o time construirá as funcionalidades definidas para um Sprint com base nos resultados obtidos nos Sprints anteriores.
e) Disseminar conhecimento sobre o que foi feito no dia anterior para poder priorizar o trabalho a ser realizado no dia que se inicia.

11. **(FGV / AL-RO – 2018)** Para o desenvolvimento do Sistema de Informações ao Cidadão (SIC), foi decidida a utilização de uma metodologia ágil. Segundo o Manifesto Ágil, esta decisão indica que foi dado maior valor:

a) aos processos e ferramentas.
b) à resposta a modificações.
c) à documentação abrangente.
d) à negociação do contrato.
e) ao cumprimento do plano

12. **(FGV / TJ-SC – 2015)** O SCRUM, processo para o desenvolvimento de software ágil, estrutura-se sobre:

a) plan, documentaton, test;
b) roles, artifacts, activities.
c) requisites, code, products;
d) client team, development team, deliverables;
e) interface, data, code.

13. **(FGV / BANESTES – 2021)** Observe o quadro comparativo a seguir, publicado em sites ligados ao estudo e à investigação de diferentes estratégias/metodologias para implementar um sistema ágil de desenvolvimento ou gestão de projetos.

Aspectos	X	Y
Ritmo	Sprints	Fluxo contínuo
Funções	Funções bem definidas	Sem funções necessárias
Entregas	Final de cada sprint	Entrega contínua
Mudanças	Evitar durante sprint	A qualquer momento

É correto identificar que X e Y representam, respectivamente:

a) Crystal e Scrum;
b) Extreme Programming e Crystal;
c) Kanban e Lean;
d) Lean e Extreme Programming;
e) Scrum e Kanban.

14. **(FGV / TJ-GO – 2014)** Scrum e Kanban são metodologias de gerenciamento de projetos de software populares entre praticantes do desenvolvimento ágil. Um aspecto de divergência entre as duas metodologias é:

a) processo incremental;

b) processo iterativo;
c) uso de quadro de tarefas;
d) apresentação do estágio de desenvolvimento de uma tarefa;
e) valorização de feedback.

15. **(FGV / MPE-MS – 2013)** Kanban é um dos métodos ágeis mais recentes e sofreu grande influência do movimento "Lean", surgido nos anos 1980. São práticas comuns a esse método:

a) limitar o WIP (Work In Progress) e uma visualização explícita do fluxo de trabalho.
b) integração Contínua e gerenciamento de configuração.
c) limitar o WIP (Work In Progress) e gerenciamento de configuração.
d) gerenciar o fluxo de trabalho e manter estimativas previamente definidas.
e) melhoria contínua e nunca limitar o WIP para evitar folgas no sistema de trabalho.

16. **(FCC / SEFAZ-AP – 2022)** No modelo ágil de gestão de projetos Scrum, um dos conceitos mais importantes é o:

a) *product backlog*, um diagrama que mostra a quantidade de trabalho a fazer e a quantidade feita ao longo do tempo.

b) *daily sprint meeting*, uma reunião de cerca de uma hora para se obter ideias para solução de problemas encontrados no projeto.

c) *sprint*, que consiste em um ciclo de desenvolvimento que, em geral, vai de duas semanas a um mês.

d) *pair programming*, prática que consiste na programação realizada por duas pessoas em cada computador.

e) *test driven development*, prática que consiste em definir e implementar os testes de unidade antes da programação.

17. **(FCC / TRF - 3ª REGIÃO – 2019)** SCRUM atende aos princípios do Manifesto Ágil porque:

a) pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
b) não aceita mudanças nos requisitos durante o desenvolvimento e por isso as entregas são mais ágeis.
c) as entregas ocorrem sempre no prazo, nunca adiantadas ou atrasadas.

d) mais importante que a motivação dos desenvolvedores é a disciplina gerencial imposta que organiza e agiliza o desenvolvimento.
e) não admite a comunicação direta entre os desenvolvedores, pessoalmente. Isso só pode ser feito por intermédio de um gerente ou coordenador.

18. (FCC / TRF - 3ª REGIÃO – 2019) A Reunião Diária do Scrum é:

a) executada no final da Sprint para inspecionar o incremento e adaptar o Backlog do Produto, se necessário.
b) um time-boxed de 15 minutos, durante o qual um "Pronto", versão incremental potencialmente utilizável do produto, é criado.
c) uma oportunidade para o Time Scrum inspecionar a si próprio e criar um plano para melhorias a serem aplicadas na próxima Sprint.
d) um time-boxed de 15 minutos, para que o Time de Desenvolvimento possa sincronizar as atividades e criar um plano para as próximas 24 horas.
e) um time-boxed de 60 minutos, durante o qual os produtos de uma Sprint são definidos.

19. (CESGRANRIO / Banco da Amazônia – 2021)

"O Scrum é um arcabouço que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos."

SCHWABER, K. ; SUTHERLAND, J. O Guia do Scrum, O Guia Definitivo para o Scrum: As Regras do Jogo. Nov. 2020. p 3. Adaptado.

Para cumprir seu objetivo, o Scrum se baseia em quatro eventos formais, contidos dentro de um evento de maior duração: a Sprint. Tais eventos formais implementam os três pilares empíricos do Scrum, que são:

a) compromisso, abertura e adaptação
b) respeito, coragem e foco
c) respeito, inspeção e adaptação
d) transparência, compromisso e respeito
e) transparência, inspeção e adaptação

20. (CESGRANRIO / Banco da Amazônia – 2014)

Uma prática que NÃO é adotada por Extreme Programming (XP) é

a) usar duas pessoas trabalhando juntas em um único computador para produzir todo o código que será enviado para a produção.

b) criar os testes antes do código que será testado.

c) refatorar frequentemente, e ao longo de todo o projeto, o código produzido pelos desenvolvedores.

d) integrar continuamente o código recém-produzido com o código existente no repositório.

e) variar a duração de cada iteração durante todo o projeto para acomodar eventuais mudanças de prioridade dos requisitos, definidas pelo cliente.

21. (CESGRANRIO / Banco da Amazônia – 2018)

O Manifesto Ágil se tornou um marco da Engenharia de Software, chamando a atenção de que vários processos propostos de forma independente tinham valores em comum. Além disso, foram definidos 12 princípios. Entre eles, figura o seguinte princípio:

a) cada pessoa em um projeto deve ter sua função predeterminada para acelerar o desenvolvimento em conjunto.
b) a contínua atenção à simplicidade do trabalho feito aumenta a agilidade.
c) software funcionando é a medida primária de progresso.
d) os indivíduos, clientes e desenvolvedores, são mais importantes que processos e ferramentas.
e) o software funcional emerge de times auto-organizáveis.

Gabaritos

1. A
2. B
3. E
4. C
5. B
6. B
7. A
8. D
9. B

10. A
11. B
12. B
13. E
14. B
15. A
16. C
17. A
18. D
19. E
20. E
21. C

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.