AULA 1 – CONCEITOS ESSENCIAIS DE DESENVOLVIMENTO ÁGIL

OBJETIVO DA AULA

Conhecer o modelo de desenvolvimento ágil e os conceitos essenciais que fazem parte desta filosofia de desenvolvimento de software.

APRESENTAÇÃO

O desenvolvimento ágil de software tornou-se uma prática muito popular há alguns anos, isso porque alguns modelos clássicos ou que estavam em voga na época demoravam muito para entregar um produto executável para o cliente.

Não era incomum alguns projetos, mesmo já iniciados há alguns meses, não terem ainda um protótipo executável, e ao invés disso, apenas pilhas de documentação e diagramas.

Com a velocidade como as coisas mudam na atualidade e grande possibilidade de mudança de prioridades e requisitos, os processos fracassavam muitas vezes por diversos motivos oriundos do modelo de desenvolvimento escolhido.

Nesta aula vamos abordar os principais conceitos da filosofia de desenvolvimento ágil.

1. MODELOS ÁGEIS

Há muito tempo já era questionado o processo de desenvolvimento clássico de software, pois eles eram normalmente longos e inflexíveis. Havia uma demora grande entre o momento em que o projeto era definido e o cliente finalmente poder testar e usar o seu produto.

Esses questionamentos iniciaram já nos anos 1980, mas foi apenas no final dos anos 1990 que a ideia tomou forma com o desenvolvimento da noção de abordagens ágeis como a Metodologia de Desenvolvimento de Sistemas Dinâmicos (DSDM), cunhado em 1997, a *Extreme Programming* (XP) de 1999 e o *Scrum* em 2001.

O modelo de desenvolvimento ágil é uma filosofia cujo objetivo é produzir mais rapidamente projetos úteis, isto é, elaborar produtos de maneira mais rápida, realizando entregas constantes e executáveis para os clientes.

Essas entregas são incrementos que trazem novas funcionalidades a cada entrega, trazendo com isso uma série de benefícios tanto para o cliente quanto para a equipe de desenvolvimento:

- Maior interação desenvolvedor-cliente: como são realizadas várias pequenas entregas, ao invés de uma grande entrega única, o processo de validação e aceite ocorre de maneira constante, o que reduz o risco de falhas;

- Priorização definida pelo cliente: como os requisitos são entregues a cada nova entrega, o cliente pode negociar com a equipe de desenvolvimento qual é ou quais são as prioridades de entrega para a nova versão;
- Flexibilidade na mudança de requisitos: esta filosofia faz com que seja muito mais rápida e menos impactante a mudança de determinado requisito.

Com a absurda velocidade das mudanças nos negócios, impulsionadas pela alta competitividade e o suporte indispensável da tecnologia da informação, o conceito de desenvolvimento ágil trouxe diversos benefícios, como a rapidez no processo como um todo e a qualidade do produto final.

Há a necessidade, porém, de um ambiente que encoraje esse tipo de processo, além de lideranças que apoiam essa filosofia. O ambiente deve ser organizado, inspecionado constantemente e munido de objetos que facilitem a prática desses modelos, como quadros e canetas *pilot*, para que o andamento do projeto e prazos sejam percebidos por toda a equipe.

LINK

Conheça a comunidade de desenvolvimento ágil no Brasil. Disponível em: www.desenvolvimentoagil.com.br. Acesso em: 15 nov. 2022.

Os modelos ágeis normalmente utilizam como base o modelo iterativo e incremental, além de incluir no processo de desenvolvimento todos os interessados no projeto, utilizando reuniões diárias e semanais de acompanhamento do desenvolvimento do produto.

Esse tipo de modelo de desenvolvimento considera algumas características que compartilham entre si.

- Os processos de especificação, projeto e implementação são intercalados;
- A documentação do projeto é minimizada, trazendo apenas aquilo que for considerado fundamental pela equipe;
- Prefere-se a comunicação com o cliente do que a elaboração de documentação;
- São geradas diversas versões do mesmo sistema, cada uma trazendo um ou mais incrementos prontos para serem testados e usados;
- Há a participação ativa de todos os *stakeholders* do projeto;
- Os próprios usuários validam cada nova versão do sistema;
- Cada novo ciclo dura, em média, entre duas e três semanas.

Após toda a discussão sobre a efetividade dos modelos clássicos, abarrotados de documentos que, muitas vezes, nem seriam utilizados, foi instituído o chamado *Manifesto Ágil*, elaborado por desenvolvedores que visavam desburocratizar o processo.

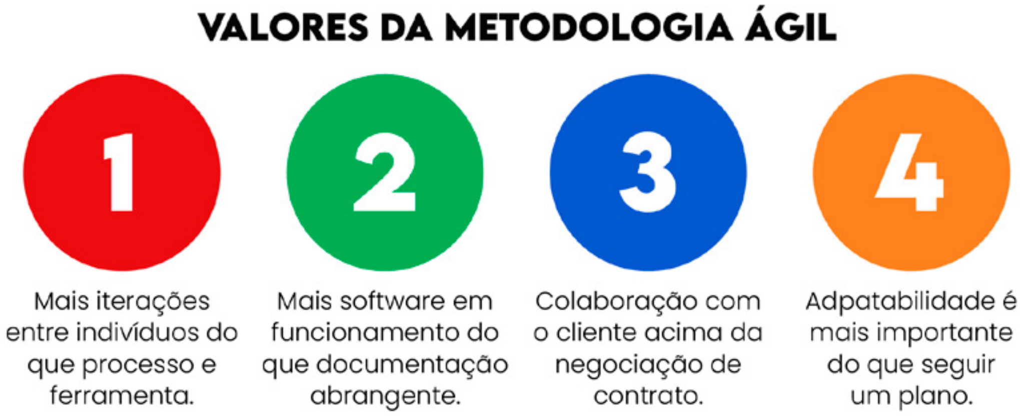
Neste manifesto, uma série de condições foram estabelecidas para que um projeto de software fosse encaixado nessa categoria. Vamos discutir mais profundamente este assunto a seguir, mas basicamente o manifesto ágil afirma que:

Estamos descobrindo maneiras melhores de desenvolver softwares, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

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

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

FIGURA 1 | **Valores da Metodologia Ágil**



Elaboração própria.

A Figura 1 mostra de maneira gráfica os valores utilizados nas metodologias ágeis.

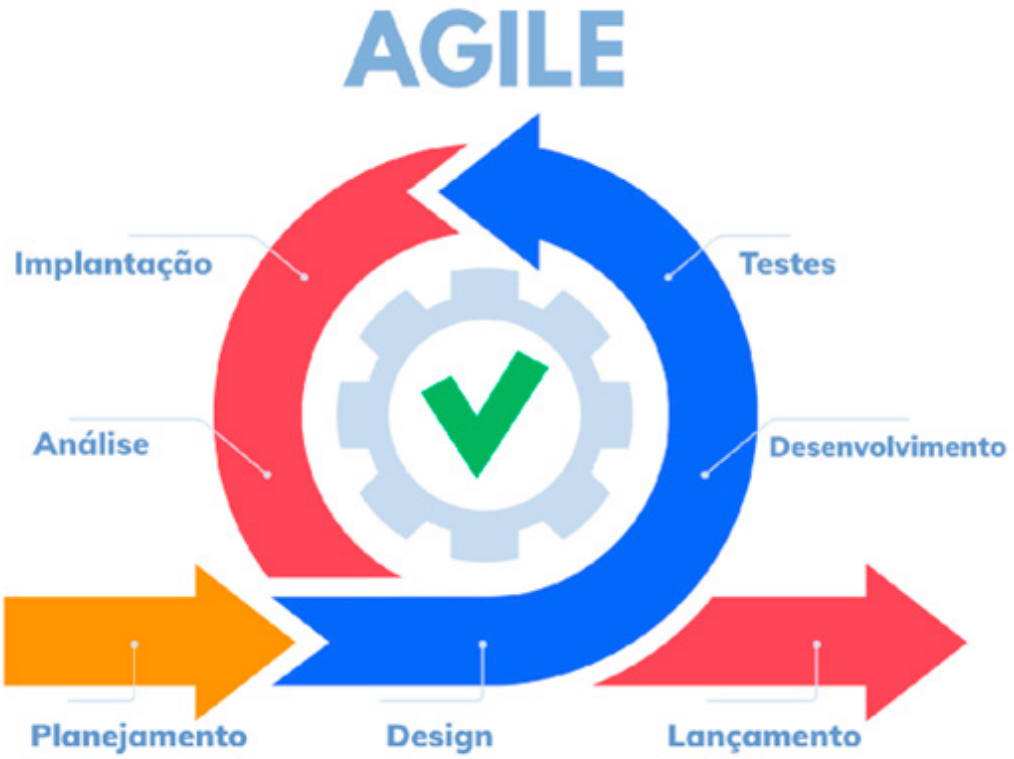
2. TIPOS DE METODOLOGIAS ÁGEIS

Dentre os vários tipos de modelos temos como mais populares os modelos *Extreme Programming* (XP) e o *Scrum*. Mas existem também outros modelos considerados como ágeis:

- FDD – Desenvolvimento dirigido à funcionalidade (1997);
- *Dynamic Systems Development Method* (1997);
- Desenvolvimento de Software Adaptativo (2000);
- *Crystal* (2001); e
- Desenvolvimento Dirigido a Características (2002).

Devido ao sucesso deste movimento, alguns modelos já existentes acabaram criando adaptações de seus modelos para a metodologia ágil, como foi o caso do modelo RUP (*Rational Unified Process*), que criou instâncias ágeis de seu modelo.

FIGURA 2 **| Ciclo de Vida Básico das Metodologias Ágeis**



Fonte: blog.cronapp.io/governanca-de-dados/ – Acesso em: 14 nov. 2022.

A Figura 2 mostra um exemplo do ciclo de vida básico dos métodos de desenvolvimento ágeis.

Os métodos ágeis são muito utilizados atualmente e têm sido bem-sucedidos principalmente para desenvolvimento de produtos de complexidade e tamanhos pequenos ou médios. No entanto, esse sucesso depende também de um amplo envolvimento e comprometimento do cliente com todo o processo de desenvolvimento.

CONSIDERAÇÕES FINAIS

Nesta aula tratamos da conceituação dos modelos ágeis, apresentando o cenário de desenvolvimento que motivou aos desenvolvedores a propor este tipo de filosofia, publicando logo a seguir o chamado manifesto ágil, que determinava quais pensamentos conduziria a abordagem.

Discutimos as características principais de um modelo ágil, que normalmente é compartilhado entre os modelos mais conhecidos no mercado como o *Extreme Programming* e *Scrum*.

As metodologias ágeis são alternativas ao modelo tradicional de desenvolvimento de software e são utilizadas com o objetivo de agilizar o trabalho dos desenvolvedores, minimizando a elaboração de documentação desnecessária, e gerando melhoria contínua para os processos que envolvem a produção de software.

Vale a pena destacar que, embora seja muito utilizada em projetos de software em todo o mundo, esta forma de desenvolver softwares não inviabiliza outros modelos e metodologias de desenvolvimento.

MATERIAIS COMPLEMENTARES

Vídeo: Decifrando o Agile 1 – Por que usar métodos ágeis? Disponível em: https://www.youtube.com/watch?v=ds_FydzsuO8. Acesso em: 14 nov. 2022.

Vídeo: Decifrando o Agile 2 – Entenda o que são os métodos ágeis em 5 minutos. Disponível em: https://www.youtube.com/watch?v=efZlpew90Nk. Acesso em: 14 nov. 2022.

Link: Modelos Ágeis: Conceitos e Princípios. Disponível em: https://www.devmedia.com.br/modelos-de-processos-ageis-conceitos-e-principios/30059. Acesso em: 14 nov. 2022.

REFERÊNCIAS

KENNETH C. LAUDON; JANE P. LAUDON. *Sistemas de informação gerenciais*. 11ª edição. Editora Pearson, 2014.

PRESSMAN, R. G. *Engenharia de Software.* 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software.* 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.

AULA 2 – O MANIFESTO ÁGIL

OBJETIVO DA AULA

Compreender o *Manifesto Ágil*, uma declaração de valores e princípios essenciais para o desenvolvimento de software usando a filosofia ágil, além de conhecer os doze princípios utilizados no modelo ágil.

APRESENTAÇÃO

O *Manifesto Ágil* foi um documento criado por 17 desenvolvedores que se uniram para criar os fundamentos, crenças e valores que, de acordo com suas experiências, apoiaram a construção de software de maneira ágil.

Nesta aula vamos apresentar os detalhes deste manifesto, bem como aprofundar questões sobre os 12 princípios dos métodos ágeis, abordando as vantagens e desvantagens da filosofia.

Ao fim, vamos diferenciar os métodos de construção tradicional e ágil, além de descrever quais são os melhores cenários para a implementação de um processo de software baseado em cada uma dessas possibilidades.

1. MANIFESTO ÁGIL

O *Manifesto Ágil* para Desenvolvimento de Software trata-se de um documento assinado em 2001 no estado de Utah, nos Estados Unidos, por 17 desenvolvedores de software que questionavam a forma clássica de desenvolvimento de software, e já utilizam alguns padrões que seriam utilizados na filosofia ágil.

LINK

Conheça os detalhes do Manifesto Ágil. Disponível em: agile-manifesto.org/iso/ptbr/manifesto.html. Acesso em: 14 nov. 2022.

Criado em fevereiro de 2001, o documento trouxe algumas experiências dos seus 17 entusiastas a partir de padrões já praticados por eles em métodos como XP, DSDM, *Scrum* e FDD. Embora estes fossem métodos distintos entre si, eles carregavam algumas características em comum, que seria utilizado na padronização desta metodologia.

Esse consenso fez com que eles escrevessem o *Manifesto Ágil*, que funcionaria como base para qualquer metodologia que se intitulasse como ágil. Assim foi gerado o documento que traria um conjunto de crenças e valores que aqueles profissionais acreditavam.

O *Manifesto Ágil* foi publicado da seguinte forma:

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

Em outro momento, foi trabalhada a lista de princípios do *Manifesto Ágil*, que aborda as questões marcantes da filosofia, deixando claro o que é, de fato, a metodologia ágil.

2. VALORES

Como vimos anteriormente, o *Manifesto Ágil* apresenta uma lista de valores que todos os profissionais envolvidos acordaram em seguir e disseminar.

Esses valores podem ser explicados da seguinte forma:

- Indivíduos e interação entre eles mais que processos e ferramentas – Neste primeiro valor a ideia é de que as pessoas são os elementos mais importantes na construção de software, o bom convívio e relacionamento pode desimpedir processos e facilitar mais que o uso de processos engessados em ferramentas;
- Software em funcionamento mais que documentação abrangente – Aqui há uma clara referência à minimização de documentação, pois a quantidade de documentos e artefatos produzidos no modelo tradicional é uma das principais críticas. A ideia é que a documentação produzida em um projeto seja mínima e, de preferência, gerada pelas próprias ferramentas de desenvolvimento;
- Colaboração do cliente mais que negociação de contratos – Aqui é novamente estabelecida a comunicação como um fator essencial no desenvolvimento de software, assim como a participação intensa do cliente ao longo do processo de desenvolvimento, que impacta diretamente no processo de tomada de decisão;
- Responder a mudanças mais que seguir um plano – Aqui a ideia é a de que a equipe e demais envolvidos no projeto possam rapidamente estabelecer meios de reagir e implementar mudanças assim que elas ocorrerem.

Segundo o documento original *The Agile Manifesto*, mesmo havendo valor nas importantes atividades e itens à direita como processos e ferramentas, documentação abrangente, negociação de contratos e plano, os itens à esquerda, como indivíduos e interações, software em funcionamento, colaboração e resposta a mudanças, devem ser mais valorizados para que se tenha a construção de software seguindo a metodologia ágil.

3. OS 12 PRINCÍPIOS

Conforme o documento que ficou conhecido como *Manifesto Ágil*, os princípios foram descritos da seguinte forma:

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

2) Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas;

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

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

5) Construir projetos em torno de indivíduos motivados, dando a eles o ambiente e o suporte necessário e confiando neles para fazer o trabalho;

6) O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é por meio de conversa face a face;

7) Software funcionando é a medida primária de progresso;

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

9) Contínua atenção a 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 times auto-organizáveis;

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

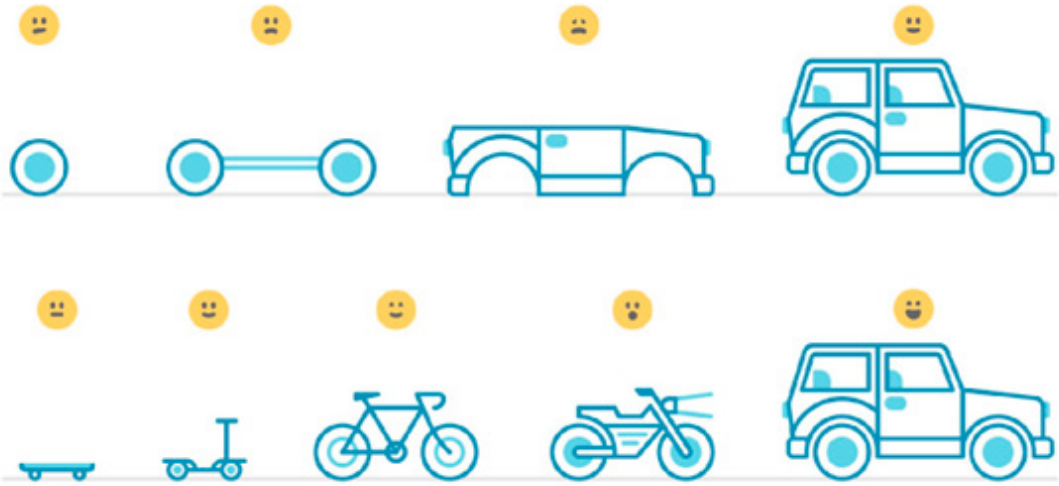
Para manter o *Manifesto Ágil*, seus valores e princípios, foi criada uma organização chamada *Agile Alliance* já no fim de 2001. O objetivo desta organização sem fins lucrativos é o de promover o conhecimento e manter as discussões sobre os métodos ágeis disponíveis no mundo, reforçando sempre o *Manifesto Ágil*.

Atualmente empresas de todo o mundo aplicam os métodos no desenvolvimento de seus produtos, incluindo poderosos nomes como Google, Yahoo, Microsoft e IBM.

4. MODELOS TRADICIONAIS X MODELO ÁGIL

Com tudo o que já foi abordado em nossas duas primeiras unidades, fica fácil estabelecer um comparativo entre os modelos tradicionais, estudados na unidade 1 e os métodos ágeis, que estamos acabando de aprender na unidade 2.

FIGURA 1 **| A Construção de um Veículo do Ponto de Vista Tradicional e Ágil**



Fonte: https://www.ieepeducacao.com.br/metodologias-tradicionais/ <Acesso em 14/11/2022)

A Figura 1 mostra um exemplo teórico da diferença entre as abordagens tradicionais e ágeis. Na parte superior vemos a evolução da construção do veículo, repare que só podemos utilizar o veículo ao final da produção. Na parte inferior, vemos que a cada entrega temos um produto completo, que já pode ser utilizado, mesmo antes do final do projeto.

De um ponto de vista minimalista, as metodologias tradicionais se baseiam em etapas mais rígidas e controladas, enquanto as metodologias ágeis se fundamentam na flexibilidade e adaptabilidade das estratégias.

O quadro 1 lista as principais diferenças entre as duas abordagens:

Quadro 1 | **Principais diferenças entre metodologias tradicionais e ágeis**

Quadro 1 – Principais diferenças entre metodologias tradicionais e ágeis		
Característica	**Tradicional**	**Ágil**
Hierarquia	Há um líder ou gerente que é uma figura central no projeto que responde pelas decisões	Equipes multidisciplinares com autonomia para certas decisões
Gestão do projeto	Rígido, com etapas bem definidas e uma entrega final	Flexível, com a participação do cliente na priorização das muitas entregas
Documentação	Pesada, com artefatos (documentos, diagramas) produzidos em cada fase	Mínima, normalmente produzidas pelas próprias ferramentas de apoio, com engenharia reversa
Orçamento	Definido a longo prazo	O custo pode ser avaliada para cada etapa
Colaboração	O cliente tem grande participação apenas no início do projeto	O cliente participa ativamente do projeto
Comunicação	Cada equipe se limita a conhecer as questões em que estão envolvidas	Todos acabam conhecendo o projeto como um todo

Quadro 1 – Principais diferenças entre metodologias tradicionais e ágeis		
Entrega	Realizada uma única vez ao final de todo o projeto	Realizada em ciclos curtos com incrementos usáveis do produto

A escolha da metodologia que será aplicada para cada projeto deve ser pensada com calma. Essa decisão deve considerar o tipo de projeto que será desenvolvido e a cultura da empresa, pois a própria organização pode ter preferência por um ou outro modelo de projeto e os seus colaboradores podem já estar acostumados a trabalhar com determinada tecnologia.

Deve ser observado o tamanho do projeto, seus requisitos, tecnologias, e a partir daí optar pela metodologia ágil ou a tradicional. A grosso modo, em um projeto em que as necessidades do cliente podem mudar a qualquer momento, favorecem o uso de métodos ágeis, por conta da flexibilidade do mesmo.

Já a metodologia tradicional pode ser uma boa opção em casos em que o projeto deve ser planejado e decidido desde o início, com os objetivos e escopo bem definidos, com poucas chances de ter mudanças e com baixo risco e longo prazo.

Na prática, as duas metodologias têm vantagens e podem ser utilizadas até mesmo de forma complementar, desde que possam realmente contribuir para um andamento de projeto otimizado.

CONSIDERAÇÕES FINAIS

Nesta aula abordamos os métodos ágeis, tratando do seu ponto de vista histórico, em que desenvolvedores se reuniram para criar o documento chamado *Manifesto Ágil*, que apresentou as diretrizes e valores para que um método seja categorizado como ágil.

Falamos sobre os valores definidos pelo *Manifesto Ágil,* e também sobre os seus 12 princípios, que ajudam a guiar toda a filosofia e a classificar um método como ágil.

Por fim, estabelecemos um comparativo entre o modelo clássico e o modelo ágil, diferenciando-os sob 7 quesitos, que podem guiar o profissional na escolha do modelo mais adequado para cada projeto.

MATERIAIS COMPLEMENTARES

Link: *Manifesto para Desenvolvimento Ágil de Software*. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html Acesso em: 14 nov. 2022.

Vídeo: *O que é ser ágil?* Disponível em: https://www.youtube.com/watch?v=60wt4OqxjtsAcesso em: 14 nov. 2022.

Vídeo: *Entenda o que são métodos ágeis em 8 minutos*. Disponível em: https://www.youtube.com/watch?v=cT_X4_n0NJ4 Acesso em: 14 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software*. 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software*. 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.

AULA 3 – EXTREME PROGRAMMING

OBJETIVO DA AULA

Conhecer o modelo de desenvolvimento ágil *Extreme Programming*, conhecido como modelo XP e saber quais são os valores e as características deste modelo.

APRESENTAÇÃO

Os métodos de desenvolvimento ágeis apresentam alguns importantes modelos que vêm sendo amplamente utilizados no mercado para o desenvolvimento de software.

Um dos mais conhecidos modelos é o chamado *Extreme Programming*, ou simplesmente XP. Este modelo tem como valores fundamentais a comunicação entre equipes de desenvolvimento, clientes e usuários.

Outra característica importante deste modelo é a chamada Programação em Pares e o refatoramento do código.

Nesta aula você vai conhecer um pouco das características, valores e detalhes de uso deste modelo de desenvolvimento ágil.

1. O MODELO XP

O modelo de desenvolvimento *Extreme Programming*, ou simplesmente XP, nasceu por volta de 1999 quando o desenvolvedor Kent Beck, um dos autores do *Manifesto Ágil*, batizou esse processo de trabalho que tinha a intenção de impulsionar boas práticas de desenvolvimento, lançando o livro *Extreme Programming Explained: Embrace Change*.

A ideia era utilizar o modelo iterativo em condições em que era possível criar várias versões do mesmo software desenvolvidas, integradas e testadas por programadores diferentes em um único dia.

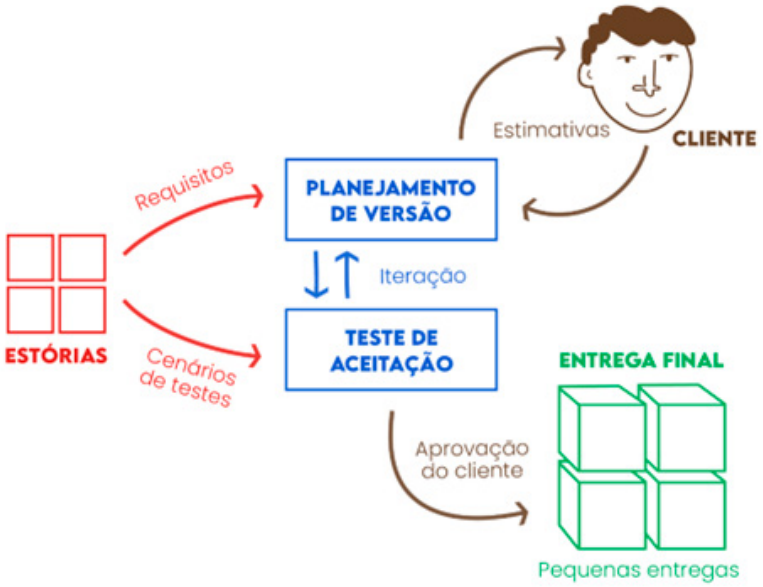
LINK

Conheça os detalhes do XP na comunidade brasileira de desenvolvimento ágil. Disponível em: www.desenvolvimentoagil.com.br/xp/. Acesso em: 15 nov. 2022.

O XP trabalha com a definição de requisitos no que chamamos de “estórias do usuário” que são implementadas a partir de uma série de tarefas. O desenvolvimento é feito a partir da programação em pares.

A estória deverá ser descrita pelo cliente, que é quem detém o conhecimento do funcionamento do negócio. Quando uma estória for muito grande, ela deve ser dividida em tarefas com duração máxima de alguns dias.

FIGURA 1 | **Ciclo de Vida do Modelo XP**



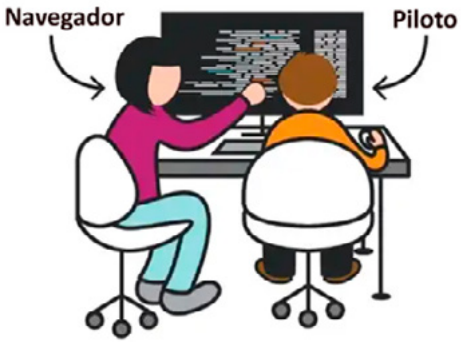
Fonte: Adaptado de www.digite.com/pt-br/agile/programacao-extrema-xp/. Acesso em: 15 nov. 2022.

Na Figura 1 vemos o ciclo de vida do XP, onde é possível ver o seu processo de funcionamento, onde os requisitos são estimados, as versões são planejadas e, a cada iteração, há um novo teste de aceitação junto ao cliente, que aprova a nova versão do sistema. As estórias do cliente se transformam em requisitos validados em cenários de testes.

2. PROGRAMAÇÃO EM PARES

A programação em pares (*Pair Programming*) é um conceito chave no XP. Nesta abordagem, todo e qualquer código implementado no projeto deverá ser feito a partir de uma dupla de programadores que desenvolvem testes para cada tarefa antes de elaborarem o código, ao finalizarem a tarefa, os códigos são novamente testados, mas desta vez com o incremento integrado ao sistema.

FIGURA 2 | **Programação em Pares**



Fonte: Adaptado de medium.com/@pedronk/kotlin-coding-dojo-e-pair-programming-d4de64da0d20. Acesso em: 15 nov. 2022.

Na Figura 2 vemos a dinâmica de trabalho da programação em pares: dois programadores, onde um é identificado como piloto e o outro como navegador.

Na técnica de desenvolvimento em pares, dois desenvolvedores trabalham no mesmo problema, ao mesmo tempo, e no mesmo computador. Um deles é responsável pela digitação, ele é chamado de piloto, e o outro acompanha o trabalho da sua dupla, ele é chamado de navegador.

Enquanto um deles codifica o problema, o outro tem a responsabilidade de revisar o código digitado, onde pequenos erros de programação que demorariam algumas horas para serem depurados são percebidos mais rapidamente pelo navegador da dupla.

Um dos grandes benefícios da programação em pares é a troca de experiências e ideias entre os desenvolvedores, permitindo que um veja um problema que o outro não conseguiu identificar. Há alguma discussão, porém, sobre a questão do custo de se manter dois profissionais envolvidos em uma mesma tarefa.

A programação em pares resulta no chamado código coletivo, onde o código fonte não tem dono, fazendo com que ninguém precise solicitar permissão para poder modificar o mesmo. O objetivo é fazer a equipe conhecer todas as partes do sistema, o que faz com que toda a equipe domine os problemas que estão sendo resolvidos com o sistema.

3. REFATORAMENTO

Outra característica marcante no XP é o refatoramento. A filosofia por trás do XP considera que toda vez que o desenvolvedor encontrar um código duplicado, pouco legível, mal codificado, sem padronização, lento, com código legado ou uso incorreto de outras implementações, ele deverá modificar este código tornando-o mais legível e simples, porém esta alteração deve garantir que o comportamento do código em questão não mude o comportamento e o resultado final do código.

A prática do refatoramento é um processo que permite a melhoria contínua da programação, a eliminação de erros visíveis, e a compatibilidade com o código já existente. Refatorar ajuda na melhora da clareza, leitura do código e facilita a manutenção do código, garantindo uma maior longevidade ao mesmo. Além disso, o processo de refatoração garante maior coesão ao código, levando ao melhor aproveitamento e evitando duplicação no código-fonte.

4. OUTRAS PRÁTICAS

Além das marcantes práticas de Programação em Pares e Refatoramento, o XP possui outras práticas importantes em sua gestão diária.

A participação ativa do cliente é uma importante questão na metodologia, pois ele poderá acompanhar o progresso do desenvolvimento, além de aplicar diversas validações do que vem sendo construído pela equipe de desenvolvimento.

O desenvolvimento é iterativo, com ciclos realizados em períodos de 1 ou 2 semanas, onde tarefas são agrupadas a partir de conversas e negociações com o cliente, que é quem vai determinar as suas prioridades.

As reuniões são realizadas a partir de *Stand-Up Meetings*, reuniões de curta duração feitas em pé, com o objetivo de saber o que cada um está fazendo, em que ponto está o projeto e se alguém está tendo problemas para executar suas tarefas. Mesmo que surja algum indício de problema, essa reunião não tem o objetivo de pensar em soluções naquele momento.

Sobre testes, o XP usa o conceito de Desenvolvimento Orientado a Testes (*Test Driven Development*), conhecido como TDD. Em um primeiro momento são realizados testes unitários, e depois é criado o código para que o teste funcione. Os testes unitários são essenciais para que a qualidade do projeto seja mantida durante todo o ciclo de vida do desenvolvimento.

5. VALORES DO XP

Assim como toda metodologia, o XP possui um conjunto de valores e princípios. Os valores do XP são definidos por comunicação, simplicidade, feedback, coragem e respeito.

- Comunicação – é o valor básico do XP, a comunicação deve ocorrer o tempo todo, entre equipes, usuários e patrocinador do projeto;
- Simplicidade – se existe um modo mais simples de se fazer algo, por que não seguir por esse caminho?
- Feedback – o *feedback* deve ser rápido e constante. Se há algo para ser corrigido, isso deve ser feito o mais breve possível;
- Coragem – é preciso coragem para falar a verdade, mesmo que essa verdade signifique trazer notícias ruins, como estimativas e custos honestos;
- Respeito – o respeito deve imperar entre todos os stakeholders internos e externos ao projeto.

Já os princípios são definidos por: *feedback* rápido, presumir simplicidade, abraçar mudanças, trabalho de alta qualidade, pequenos passos, melhoria, diversidade, reflexão.

6. PAPÉIS NO XP

Em uma equipe organizada para trabalhar como o método ágil XP, normalmente teremos a seguinte configuração:

- Cliente – principal interessado no resultado final do projeto;
- Gerente de Projeto – responsável por liderar o projeto e resolver problemas administrativos;
- *Coach* – responsável por questões técnicas do projeto;

- Analista de testes – responsável por garantir a qualidade do sistema a partir da realização de testes;
- Redator técnico – responsável por documentar o sistema;
- Desenvolvedores – responsáveis por analisar, projetar e implementar o sistema.

Kent Beck, criador da filosofia XP, afirma que uma equipe madura de XP não deve confiar em papéis rígidos, mas reconhecer que os papéis podem ser úteis para as equipes iniciantes, até que elas comecem a atrapalhar a colaboração, ou seja, a ideia é não burocratizar o processo por conta das definições de papéis fixas.

CONSIDERAÇÕES FINAIS

Nesta aula falamos sobre a história da metodologia ágil *Extreme Programming*, mais conhecida como metodologia XP. A partir desta aula podemos conhecer a filosofia, os valores e os princípios desta que é uma das principais metodologias ágeis existentes no mercado.

Conhecemos um pouco também sobre as suas principais características como a programação em pares e a refatoração de código, e como essas características agregam valor ao XP.

As práticas do XP são importantes e construídas sobre valores, princípios e práticas, visando como objetivo permitir que equipes pequenas e médias produzam software de alta qualidade, de maneira rápida, e se adaptem a requisitos em evolução e mudança, como o mercado requer que estejamos preparados hoje.

MATERIAIS COMPLEMENTARES

Vídeo: Pair Programming – Programação em Pares. Disponível em: https://www.youtube.com/watch?v=5M8yNQSFBPg. Acesso em: 15 nov. 2022.

Vídeo: O que é XP? Disponível em: https://www.youtube.com/watch?v=bWTTPMlrQ-8. Acesso em: 15 nov. 2022.

Link: O que é Programação Extrema (XP) e seus valores, princípios e práticas? Disponível em: https://www.digite.com/pt-br/agile/programacao-extrema-xp/. Acesso em: 15 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software.* 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software.* 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.

AULA 4 – SCRUM

OBJETIVO DA AULA

Conhecer a metodologia de desenvolvimento ágil *Scrum*.

APRESENTAÇÃO

O *Scrum* é uma das metodologias de desenvolvimento ágeis mais populares do mundo atualmente, possui um processo de trabalho relativamente simples e possibilita que a equipe produza resultados incríveis em um curto espaço de tempo.

Nesta aula estudaremos um pouco mais sobre este modelo, entendendo o seu ciclo de vida, seus princípios, valores e características.

Também vamos conhecer os termos e papéis aplicados neste modelo de desenvolvimento, tratando de seus objetivos no projeto.

1. SCRUM

O *Scrum* é uma metodologia de desenvolvimento ágil que utiliza o processo de desenvolvimento iterativo e incremental para gerenciamento de projetos e desenvolvimento ágil de software.

O nome *Scrum* é derivado de uma jogada de Rúgbi, onde todo o mesmo time avança como apenas uma unidade, trabalhando com os mesmos jogadores e em conjunto, passando sempre a bola para um e para outro.

A ideia do *Scrum* é justamente essa, definir papéis bem específicos para as pessoas envolvidas no projeto e alinhar como cada pessoa atuará, ou seja, o que cada um terá que fazer para o time seguir em frente no desenvolvimento do software.

FIGURA 1 | **O *Scrum* no Rúgbi**

Fonte: https://pt.wikipedia.org/wiki/Scrum_(rugby). Acesso em: 15 nov. 2022.

A Figura 1 mostra o *Scrum*, uma típica jogada de Rúgbi que se refere a como o jogo é reiniciado após uma falta ou quando a bola sai de jogo.

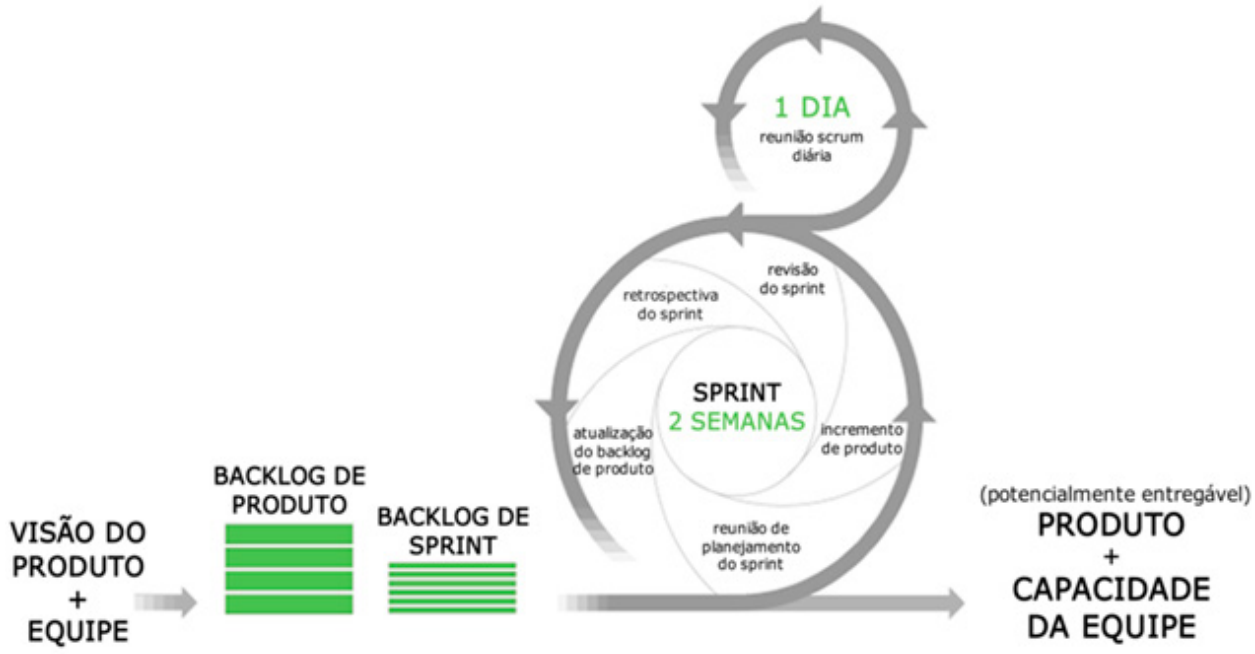
O *Scrum* nasceu em oficialmente em 1993, quando Jeff Sutherland e a sua equipe de trabalho na Easel Corporation adaptaram o artigo *The New Product Development Game* (Hirotaka Takeuchi e Ikujiro Nonaka), escrito em 1986 na Harvard Business Review (HBR), para ser utilizado em desenvolvimento de software.

A metodologia é baseada em um conjunto de práticas e papéis que devem ser envolvidos durante o processo de desenvolvimento de software. É bastante flexível e se baseia na aplicação dos 12 princípios ágeis que estudamos no início desta unidade.

2. CICLO DE VIDA

O ciclo de vida do *Scrum* se baseia em ciclos de 2 a 4 semanas resultantes em novos incrementos ao software prontos para serem utilizados. Esta nova versão produzida é validada pelo cliente e um novo ciclo inicia a partir daí.

FIGURA 2 | **O Ciclo de Vida de Projetos Desenvolvidos com *Scrum***



Fonte: Disponível em: www.semeru.com.br/blog/o-ciclo-de-vida-do-framework-scrum/. Acesso em <15/11/2022>.

A Figura 2 mostra o ciclo de vida dos projetos desenvolvidos com o método ágil *Scrum*. Inicialmente é desenvolvido o *backlog* do produto, ou seja, a lista de requisitos a serem implementadas, logo depois é definido o *backlog* do *Sprint*, isto é, a lista de requisitos que serão desenvolvidos na próxima iteração. Durante o *Sprint* haverá reuniões diárias para avaliação do projeto e, ao fim do *Sprint,* é entregue uma nova versão do sistema, com os requisitos integrados ao produto. O ciclo se repete até que o projeto esteja pronto.

No *Scrum* o passo inicial é a definição do *Backlog* do produto, feita pelo *Product Owner*, pessoa que define os requisitos, e a sua equipe, tendo como base os requisitos desejados na versão final do sistema.

O backlog é dividido em pequenos requisitos, originando vários *Backlogs* de *Sprint*. O *Product Owner* é o responsável por escolher quais requisitos serão priorizados para o próximo *Sprint*.

Com a definição do *Backlog* do *Sprint,* a equipe se reúne para planejar e estabelecer as metas para o *Sprint*, em um momento conhecido como *Sprint Plane Meeting* (Reunião de Planejamento do *Sprint*).

O desenvolvimento do produto inicia-se a seguir com a implementação do *Sprint*. Todos os dias há uma reunião de 15 minutos (*Daily Scrum*), realizada de pé pela equipe sempre no mesmo horário e local. Nela cada membro conta o que fez no dia de trabalho anterior, o que pretende fazer no dia, e se está tendo algum impedimento ao realizar o seu trabalho.

A cada fim de *Sprint*, as novas funcionalidades são testadas e integradas ao sistema, cabendo à equipe realizar uma revisão do *Sprint*. Nesta parte, chamada de *Sprint* review, a equipe apresenta o que foi realizado durante o *Sprint* e demonstra as novas funcionalidades adicionadas ao sistema. O *Product Ownwer* testa e verifica se o item atende suas expectativas e determina se o estabelecido no *Sprint* foi ou não atingido.

Depois disso há a retrospectiva do *Sprint*, neste momento são levantados os prós e contras do *Sprint* recém-finalizado, também é verificado que aspectos podem melhorar no próximo sprint.

Assim, o *Scrum* permite que haja um processo de melhoria contínua, já que todas as questões do projeto, boas e ruins, são abordadas em cada ciclo. No final do *Sprint*, o backlog deve ser atualizado e um novo ciclo iniciado.

3. TERMOS E PAPÉIS DO SCRUM

O *Scrum* talvez seja a metodologia que mais possua termos papéis particulares em seu ciclo de vida. Vamos destacar algumas destas definições a seguir, considerando as definições da comunidade brasileira de desenvolvimento ágil:

LINK

Conheça os papéis do Scrum na comunidade brasileira de desenvolvimento ágil. Disponível em: www.desenvolvimentoagil.com.br/scrum/. Acesso em: 15 nov. 2022.

- *Product Backlog*: lista contendo todas as funcionalidades desejadas para um produto. O conteúdo desta lista não precisa estar completo no início de um projeto. Pode-se começar com tudo aquilo que é mais óbvio em um primeiro momento. Com o tempo, o *Product Backlog* cresce e muda à medida que se aprende mais sobre o produto e seus usuários;
- *Product Owner*: é a pessoa que define os itens que compõem o *Product Backlog* e os prioriza nas *Sprint Planning Meetings*;
- *Release Burndown*: gráfico que mostra o progresso do projeto atualizado;
- *Scrum Master*: gerente do projeto ou líder técnico. Atua como facilitador do *Daily Scrum* e torna-se responsável por remover quaisquer obstáculos levantados pela equipe durante essas reuniões;
- *Scrum Team*: é a equipe de desenvolvimento. Nela, não existe necessariamente uma divisão funcional através de papéis tradicionais, tais como programador, designer, analista de testes ou arquiteto. Todos no projeto trabalham juntos para completar o conjunto de trabalho com o qual se comprometem conjuntamente para um *Sprint*. Normalmente é composta por 6 a 10 pessoas;
- *Sprint Backlog*: é uma lista de tarefas que o *Scrum* Team se compromete a fazer em um *Sprint*;
- *Sprint Planning Meeting*: é uma reunião na qual estão presentes o *Product Owner*, o *Scrum* Master e todo o *Scrum* Team, bem como qualquer pessoa interessada que esteja representando a gerência ou o cliente;
- *Sprint Retrospective*: reunião que ocorre ao final de um *Sprint* e serve para identificar o que funcionou bem, o que pode ser melhorado e que ações serão tomadas para melhorar;
- *Sprint Review Meeting*: ao final de cada *Sprint* é feito uma reunião, o *Scrum* Team mostra o que foi alcançado durante o *Sprint*. O projeto é avaliado em relação aos objetivos do *Sprint*, determinados durante o *Sprint Planning Meeting*.

CONSIDERAÇÕES FINAIS

Nesta aula vimos com detalhes o processo de desenvolvimento ágil *Scrum*. O *Scrum* é um dos modelos mais utilizados nos dias de hoje, sendo aplicado para vários tipos e tamanhos de projeto.

Abordamos o funcionamento de seu ciclo de vida, identificando cada fase e os seus resultados, destacando a sua natureza iterativa e incremental.

Apresentamos também os papéis e termos utilizados no *Scrum*, entendendo qual a sua importância em um projeto organizado gerenciado com o *Scrum*.

Vimos, por fim, que esta metodologia permite que haja sucesso no projeto, mas para isso a integração e colaboração entre a equipe interna e externa é essencial. É fundamental que todos os participantes estejam alinhados e conhecendo as expectativas do projeto por parte do cliente.

MATERIAIS COMPLEMENTARES

Vídeo: Scrum ou XP. Qual é o melhor para a sua equipe? Disponível em: https://www.youtube.com/watch?v=H30ºAN1QsPA. Acesso em: 15 nov. 2022.

Vídeo: Scrum, o que é (de um jeito bem prático). Disponível em: https://www.youtube.com/watch?v=HlmiVz0SqNQ. Acesso em: 15 nov. 2022.

Link: O que é Metodologia Scrum? Disponível em: https://www.digite.com/pt-br/agile/metodologia-scrum/. Acesso em: 15 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software.* 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software.* 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.

AULA 5 – GERENCIAMENTO E ESCALONAMENTO DE MÉTODOS ÁGEIS

OBJETIVO DA AULA

Conhecer outros métodos ágeis importantes como ferramentas de apoio para a gestão de projetos.

APRESENTAÇÃO

Esta aula final tem o objetivo de apresentar o método ágil Kanban e seu quadro utilizado para organizar e gerenciar o projeto. O Kanban será apresentado a partir de seus valores, princípios e as suas 6 práticas fundamentais.

Além disso, também será tratado um assunto importante, que é o gerenciamento de projetos a partir de métodos ágeis, discutindo quais os problemas podem ser encontrados ao aplicarmos a filosofia ágil a projetos de grande porte.

Vamos começar?

1. KANBAN

O Kanban é uma metodologia que descreve métodos para proporcionar a melhoria de processos e fluxos de trabalho. O Kanban tem como foco na gestão de mudanças e entrega de serviços.

A gestão de mudanças determina a forma como uma solicitação de alteração é tratada e integrada a um sistema. Já a entrega de serviços tem o foco no entendimento das necessidades e expectativas do cliente.

Embora estejamos falando de sistemas, o Kanban nasceu na Toyota na década de 1950, pelas mãos de Taiichi Ohno, engenheiro mecânico da indústria, e era composta por uma série de boas práticas no ambiente industrial, tendo sido adaptada para o desenvolvimento de Software por David Anderson no ano de 2007.

1.1. PRÁTICAS

De acordo com Pressman (2021), o Kanban possui 6 práticas fundamentais:

1) Visualizar o fluxo de trabalho usando um quadro Kanban. O quadro Kanban possui colunas que representam o estágio de desenvolvimento de cada elemento de funcionalidade do Software. As divisões normalmente mostram o que há para fazer, o que está sendo feito, o que já foi feito, além do *backlog*, e também pode trazer alguns problemas descobertos ao longo do processo;

2) Limitar a quantidade de estoque em processo (WIP – *Work in Progress*). Os desenvolvedores devem ser estimulados a completar o trabalho em progresso antes de iniciarem outra. A ideia é melhorar a qualidade do trabalho e aumentar a capacidade da equipe em fornecer funcionalidades de software com maior frequência para os clientes;

3) Gerenciar o fluxo de trabalho. O ponto principal aqui é reduzir os desperdícios por meio do entendimento do fluxo atual, pela análise dos pontos em que sofre interrupções, pela definição de mudanças e pela implementação destas mudanças;

4) Explicitar políticas de processo. Significa anotar os motivos pelos quais uma tarefa foi definida como "feita" e os critérios utilizados para isso;

5) Enfocar na melhoria contínua com a criação de ciclos de feedbacks. É preciso introduzir alterações com bases de dados de processos, medindo os resultados após essas alterações;

6) Alterar o processo colaborativamente e engajar todos os membros de equipe e outros envolvidos quando necessário.

1.2. O QUADRO KANBAN

O quadro Kanban talvez seja o elemento mais conhecido quando abordamos o Kanban como metodologia. Algumas ferramentas como o Trello, se baseiam nesta prática para apoiar o gerenciamento de projetos por meio de software.

LINK

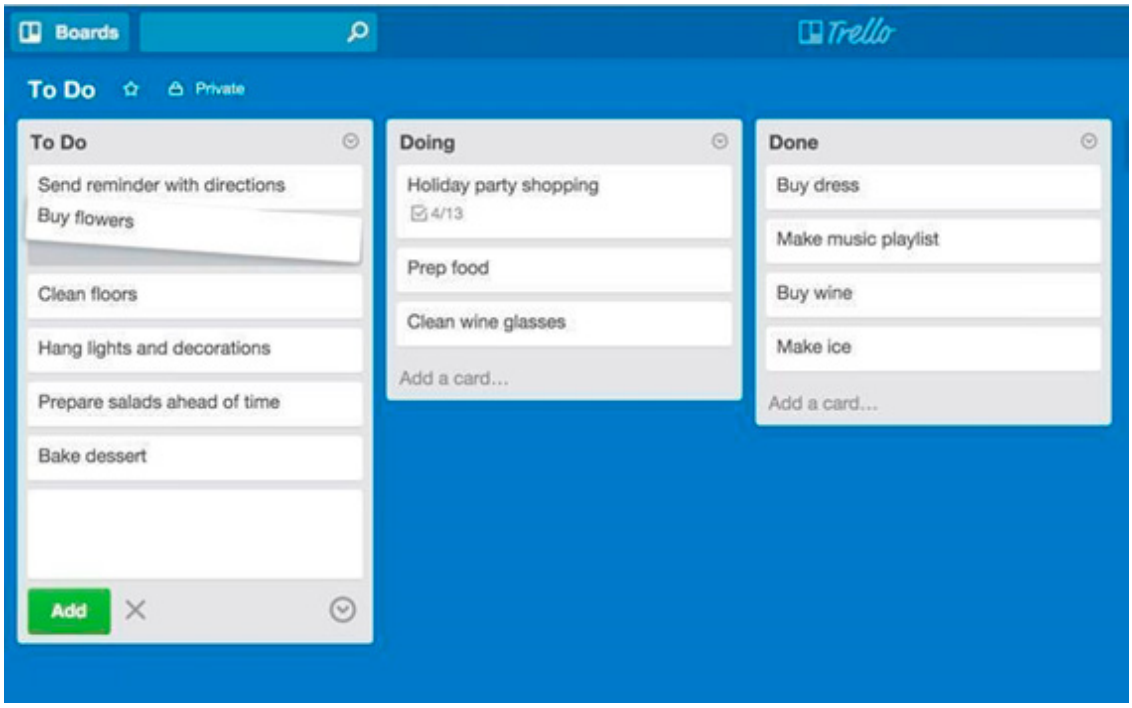
Método Kanban: *Guia Detalhado e 5 Modelos Prontos para Usar*. Disponível em: blog.trello.com/br/metodo-kanban. Acesso em: 15 nov. 2022.

O quadro Kanban pode ser montado fisicamente, através de quadro branco com as divisões, conforme mostra a Figura 1. Ou então ser organizado através de softwares de gerenciamento de projetos como o Trello, conforme mostra a Figura 2.

FIGURA 1 | **Quadro Kanban para Colocar Post-It**

Fonte: Elaboração própria.

FIGURA 2 | **Quadro Kanban feito no Trello**



Fonte: Disponível em: https://www.luiztools.com.br/post/o-que-e-kanban-e-como-aplicar-em-projetos/. Acesso em: 15 nov. 2022.

Normalmente o Kanban especifica as seguintes colunas:

- TODO: aqui ficam as tarefas que devem ser feitas, o chamado backlog;
- *DOING*: esta coluna guarda as atividades que estão sendo feitas, ou seja, que já foram iniciadas. Pode-se também colocar a informação sobre quem está executando a atividade e se ela está com algum impedimento;
- *DONE*: aqui ficam as atividades concluídas. Para que ela seja definida como concluída, deve haver um artefato chamado Definição de Pronto e se estiver trabalhando com Estórias de Usuários, deverá atender também os Critérios de Aceitação para ser considerada finalizada;
- OUTRAS: não é necessário se limitar a essas colunas, podem ser adicionadas outras, com indicações que sejam úteis ao time de desenvolvimento.

2. GERENCIAMENTO ÁGIL DE PROJETOS

Os métodos ágeis hoje são amplamente utilizados para o gerenciamento e escalonamento de projetos de software. O grande desafio da gerência é garantir que os projetos sejam entregues dentro do prazo e do orçamento estipulado.

O *Scrum*, por exemplo, possui sprints, que são unidades de planejamento, onde o trabalho a ser feito é avaliado, os recursos para desenvolvimento são selecionados e o produto é implementado.

Observamos nesse processo todas as características marcantes presentes em qualquer projeto:

*Sprint*s possuem comprimento fixo;

- O ponto de partida é o backlog do produto, que traz a lista de trabalho a ser feito;
- A fase de definição de sprints tem a participação de todos os envolvidos no projeto, entre clientes, usuários e desenvolvedores;
- É necessária organização e comunicação durante todo o desenvolvimento;
- No fim o trabalho é revisto e apresentado para os stakeholders.

3. ESCALONAMENTO DE MÉTODOS ÁGEIS

Os métodos ágeis foram definidos para sistemas de pequeno ou médio porte, mas é claro que há a necessidade de também aplicar esses conceitos para sistemas maiores.

Por conta disso, tem havido muito interesse no escalonamento dos métodos ágeis para também atender esse tipo de necessidade. Escalonar, neste caso, significa aumentar o leque de possibilidades para o atendimento a um número mais amplo de problemas.

Pressman (2011) define 6 pontos que diferem o desenvolvimento de sistemas grande porte perante os de pequeno porte:

1) Sistemas de grande porte geralmente são compostas por sistemas separados que se comunicam, nos quais equipes separadas desenvolvem cada um dos sistemas. Por isso é muito difícil que toda a equipe tenha a visão do projeto como um todo;

2) Sistemas de grande porte incluem e interagem com inúmeros sistemas existentes, o que o e diminui a possibilidade de flexibilidade e desenvolvimento incremental;

3) Sempre que vários sistemas estão integrados para criar um único, uma parte significativa do desenvolvimento preocupa-se com a configuração do sistema e não com o desenvolvimento do código original. Isso não é tão compatível com o desenvolvimento incremental, conforme prevê a metodologia ágil;

4) Sistemas de grande porte e seus processos de desenvolvimento são normalmente restringidos por regras externas e regulamentos que limitam o desenvolvimento, por incluir regras e filosofias de desenvolvimento não padronizadas;

5) Sistemas de grande porte frequentemente têm um longo tempo de aquisição e desenvolvimento, com isso torna-se difícil manter equipes coerentes por um longo período de tempo no mesmo projeto;

6) Sistemas de grande porte geralmente têm um conjunto distinto de stakeholders, o que dificulta o processo como um todo, pois eles normalmente têm níveis, interesses e visões diferentes sobre o software.

Em suma, os métodos ágeis precisam de adaptação para lidar com sistemas de grande porte, mas é necessário que os fundamentos como comunicação e flexibilidade sejam mantidos, para que o método não seja descaracterizado.

CONSIDERAÇÕES FINAIS

Nesta aula abordamos o uso do método ágil Kanban, apresentando as suas características e a forma na qual ele pode ser aplicado para o gerenciamento de projetos de software.

Apresentamos uma ferramenta fundamental no Kanban chamada “Quadro Kanban”, que pode ser utilizada em quadros físicos ou ainda organizada em sistemas computacionais de gerenciamento de projetos como o Trello.

Tratamos do método ágil como uma ferramenta para se organizar e gerenciar projetos de software, traçando um comparativo entre as fases do *Scrum* com os elementos presentes em qualquer projeto.

Finalizamos com uma abordagem um pouco mais gerencial, confrontando os métodos ágeis aplicados a sistemas de grande porte, abordando as dificuldades de adaptação, de forma que os métodos não fujam de sua essência.

MATERIAIS COMPLEMENTARES

Link: O que é Kanban e como aplicar em projetos. Disponível em: https://www.luiztools.com.br/post/o-que-e-kanban-e-como-aplicar-em-projetos/. Acesso em: 15 nov. 2022.

Vídeo: Kanban, o que é? Disponível em: https://www.youtube.com/watch?v=K9b4JC5CsQs. Acesso em: 15 nov. 2022.

Vídeo: Como usar o Kanban e o Trello para organizar os projetos. Disponível em: https://www.youtube.com/watch?v=6BNWiFhqcjc. Acesso em: 15 nov. 2022.

REFERÊNCIAS

PRESSMAN, R.G. *Engenharia de Software.* 9ª ed. Rio de Janeiro: McGraw-Hill, 2021.

SOMMERVILLE, I. *Engenharia de Software.* 10ª ed. São Paulo: Pearson Addison. Wesley, 2019.