ENGENHARIA DE SOFTWARE - AULA 3

INSTITUTO FEDERAL
Pernambuco
Campus
Jaboatão dos
Guararapes

Métodos Ágeis e Programação Extrema

Prof: Nilson Júnior
nilson.junior@jaboatao.ifpe.edu.br

www.professornilson.wix.com/ifpe

Agenda

- Conhecer o que é um processo ágil.
- Conhecer as **práticas ágeis** para desenvolvimento de software.
- História do Extreme Programming.
- Princípios e benefícios do Extreme Programming.

O que são os métodos ágeis?

Métodos Ágeis

A insatisfação com os **overheads** envolvidos em métodos tradicionais de desenvolvimento levou à criação dos métodos **ágeis**. Esses métodos:

- **Focam no código** ao invés de **modelos ou documentos**;
- Baseiam-se em uma abordagem **iterativa e incremental**;
- **Visam entregar software funcionando rapidamente** e evoluir esse software também rapidamente, a fim de **satisfazer requisitos em constante mudança;**

Métodos ágeis são mais apropriados para sistemas de negócios de tamanhos pequeno ou médio

Métodos Ágeis

Os métodos ágeis são uma alternativa à gestão tradicional de projetos, eles nasceram nos braços do desenvolvimento de software, mas hoje podem ser aplicados a qualquer tipo de projeto (inclusive os que não se remetem ao software). Os métodos ágeis vem ajudando muitas equipes a encarar a imprevisibilidades dentro de um projeto através de entregas incrementais e ciclos iterativos. As metodologias ágeis passaram a ser uma alternativa aos métodos tradicionais, também conhecidos como métodos pesados ou clássicos.

Manifesto Ágil

Em fevereiro de 2001, uma reunião nas montanhas nevadas do estado norte-americano de Utah no resort de inverno e verão Snowbird, marcava o surgimento e propagação do paradigma dos métodos ágeis. Essa reunião desencadeou o que conhecemos hoje como manifesto ágil, se tornando um grito de guerra para a indústria de software e para aquelas dezessete pessoas. O **manifesto ágil** possui **doze princípios** e **quatro valores**, vemos a seguir os quatro valores:

Valores do manifest ágil

- Indivíduos e interação entre eles 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.

http://www.agilemanifesto.org/iso/ptbr/

Princípios dos Métodos Ágeis

- Envolvimento do cliente
- Entrega incremental
- Pessoas, não processos
- Aceite as mudanças
- Mantenha a simplicidade

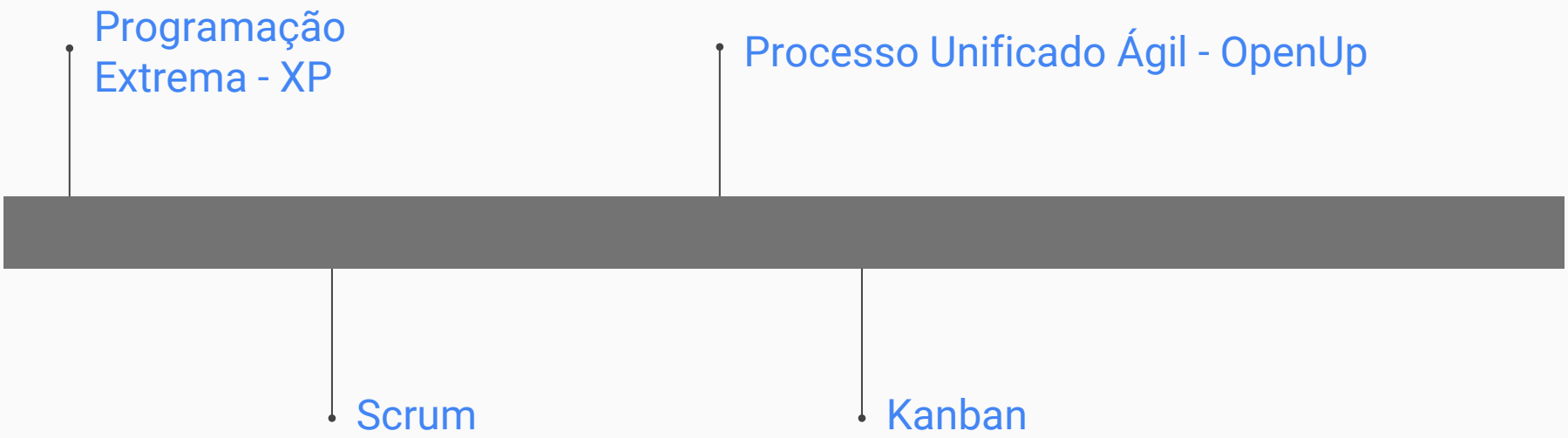
Benefícios dos Métodos Ágeis

- Clientes, quando ativamente envolvidos no desenvolvimento, experimentam uma “Síndrome de Estocolmo” benéfica.
- Lidam bem com mudanças de requisitos Em geral, a equipe de desenvolvimento
- gosta de processos mais focados no código e menos em planos e modelos
- Produzem software funcional desde as primeiras iterações

Problemas com Métodos Ágeis

- Pode ser difícil manter os clientes tão ativamente envolvidos quanto exigido pelos métodos.
- Membros da equipe podem não se prestar ao envolvimento intenso que caracteriza os métodos ágeis.
- Manter a simplicidade requer trabalho extra.
- Contratos podem ser um problema, como acontece no desenvolvimento iterativo e incremental.

Exemplos de Métodos Ágeis

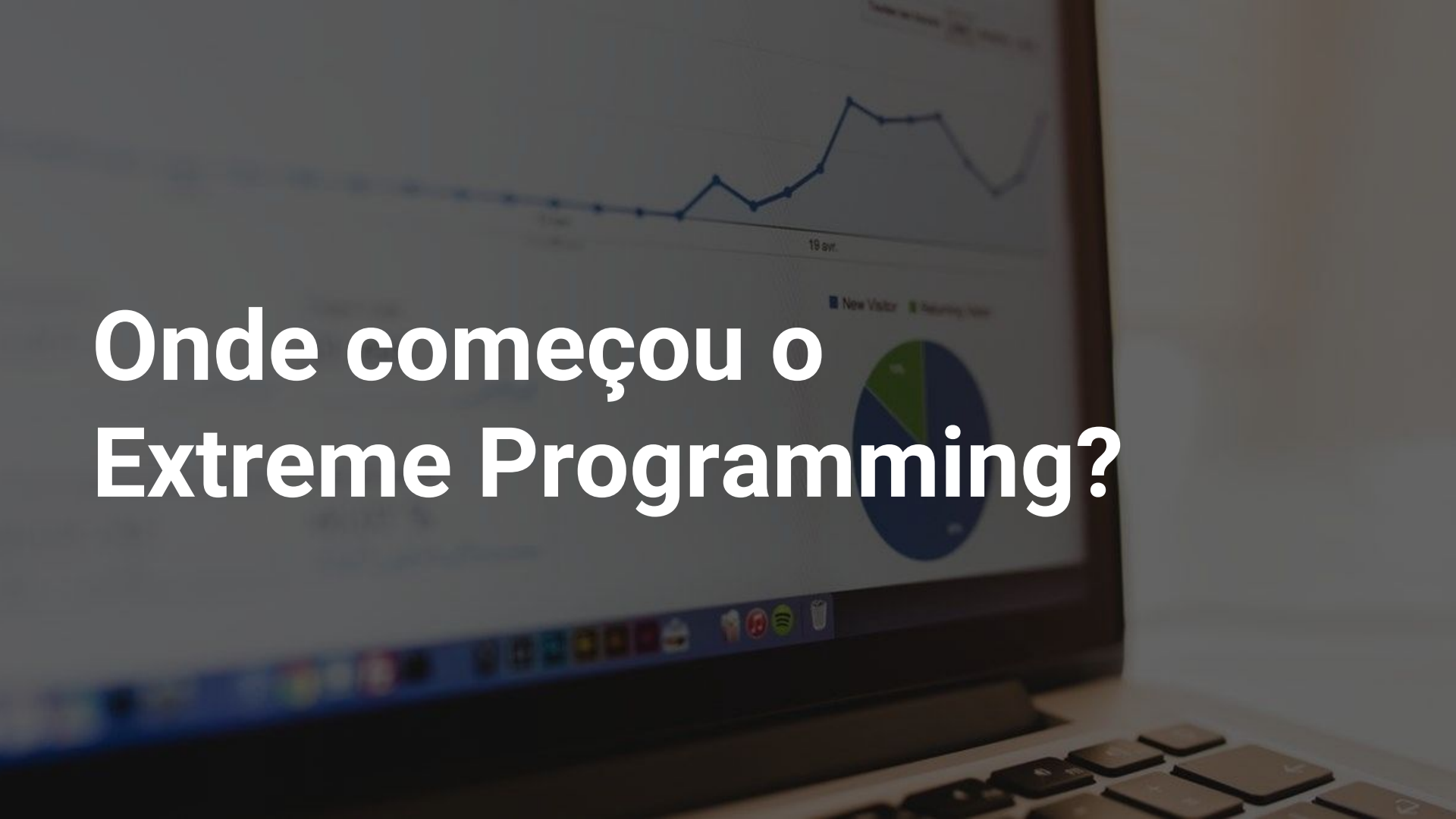


Programação Extrema (XP)

Provavelmente o mais popular e amplamente utilizado método ágil

Adota uma abordagem “extrema” para o desenvolvimento iterativo e incremental:

- Novas versões podem ser integradas várias vezes por dia;
- Incrementos são entregues aos clientes mais ou menos a cada duas • semanas;
- Todos os testes devem ser executáveis e executados para cada
- versão integrada. Um build só é aceito se os testes passarem.
- Usaremos XP nesta disciplina, mas com algumas modificações



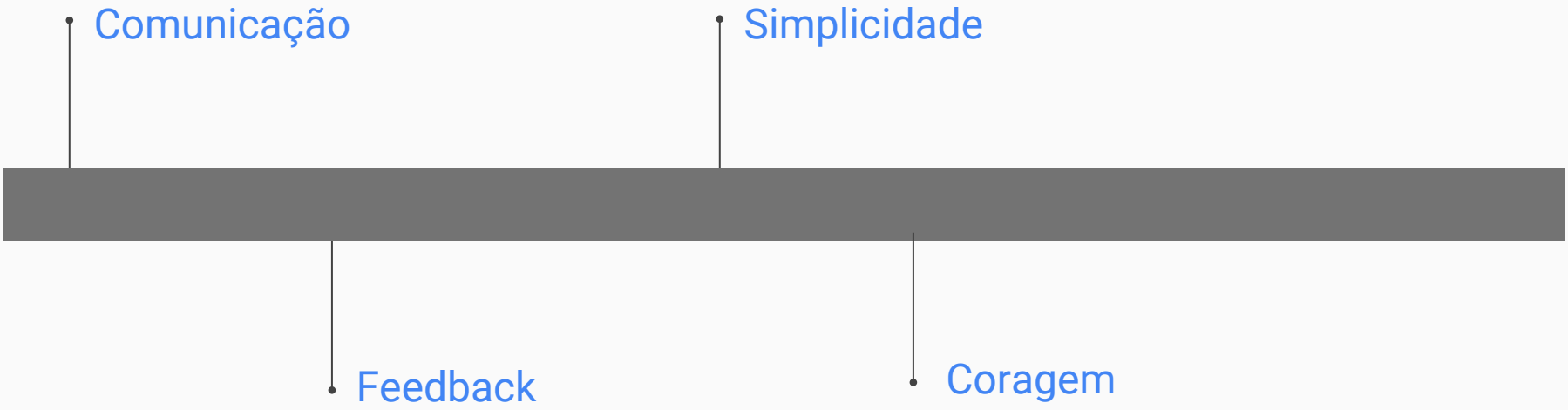
Onde começou o Extreme Programming?

Extreme Programming - XP

O Extreme Programming é um modelo de desenvolvimento de software ágil, criado em 1996, por Kent Bech, no Departamento de Computação da montadora de carros Daimler Crysler.

O XP possui muitas diferenças em relação a outros modelos, podendo ser aplicado a projetos de alto risco e com requisitos dinâmicos.

Dimensões do XP



Características

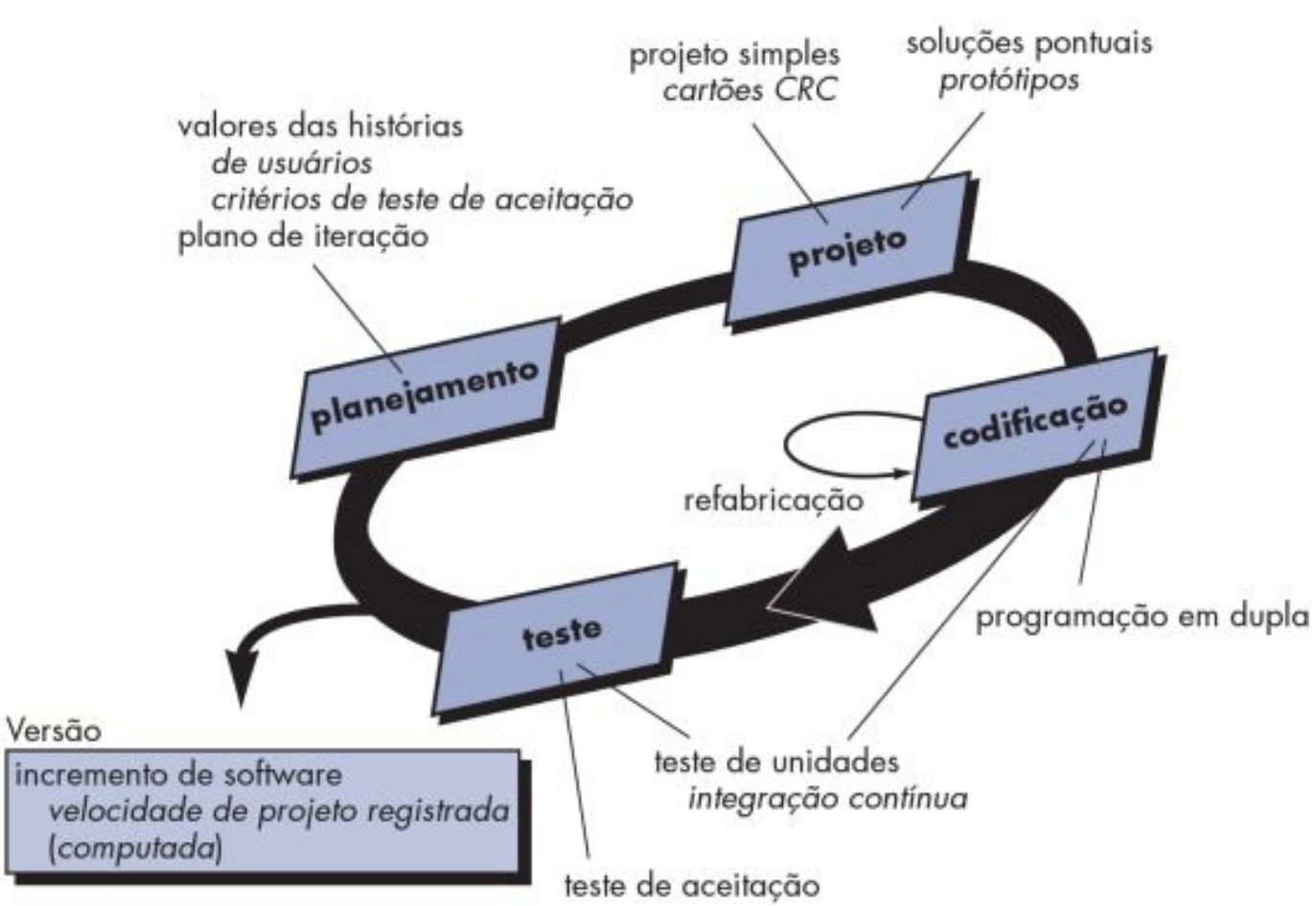
Características

- Equipe de 2 a 10 pessoas
- Desenvolvimento guiado por testes
- Equipe Proativa
- Cliente sempre disponível

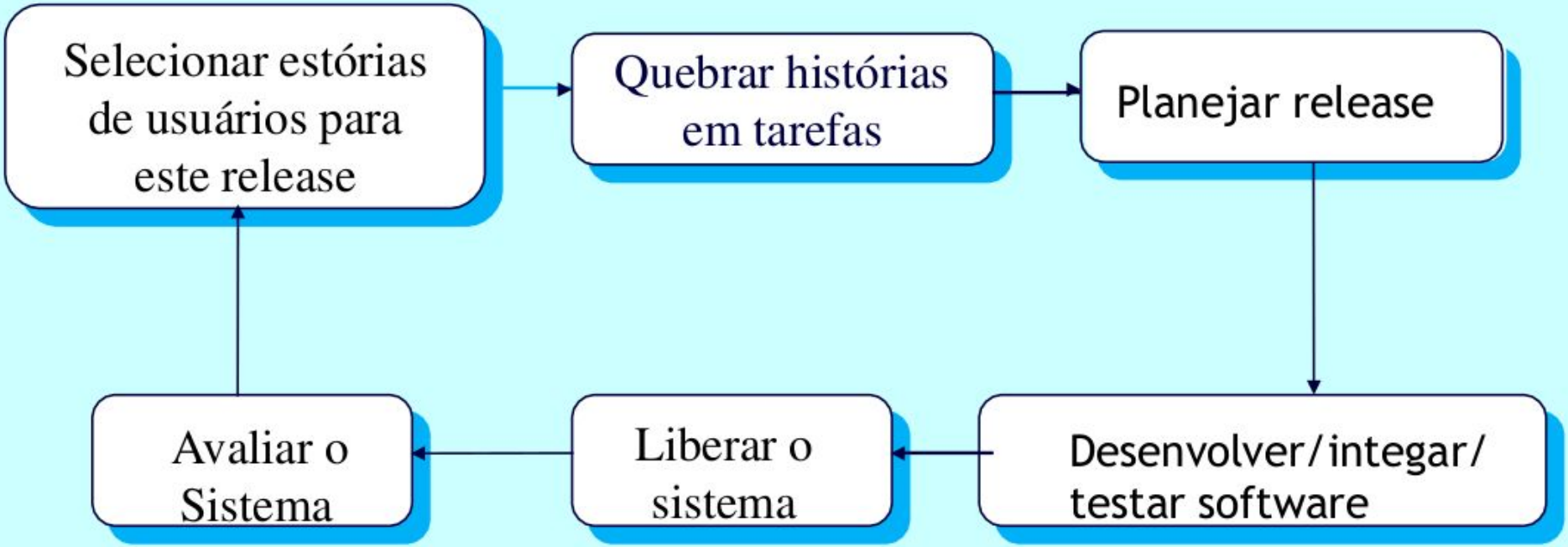
Benefícios

- Desenvolvimento de software ágil.
- Agilidade no planejamento (Planning game).
- Não foca em uma especificação completa e formal dos requisitos.
- Comunicação intensa.
- Equipe Integrada

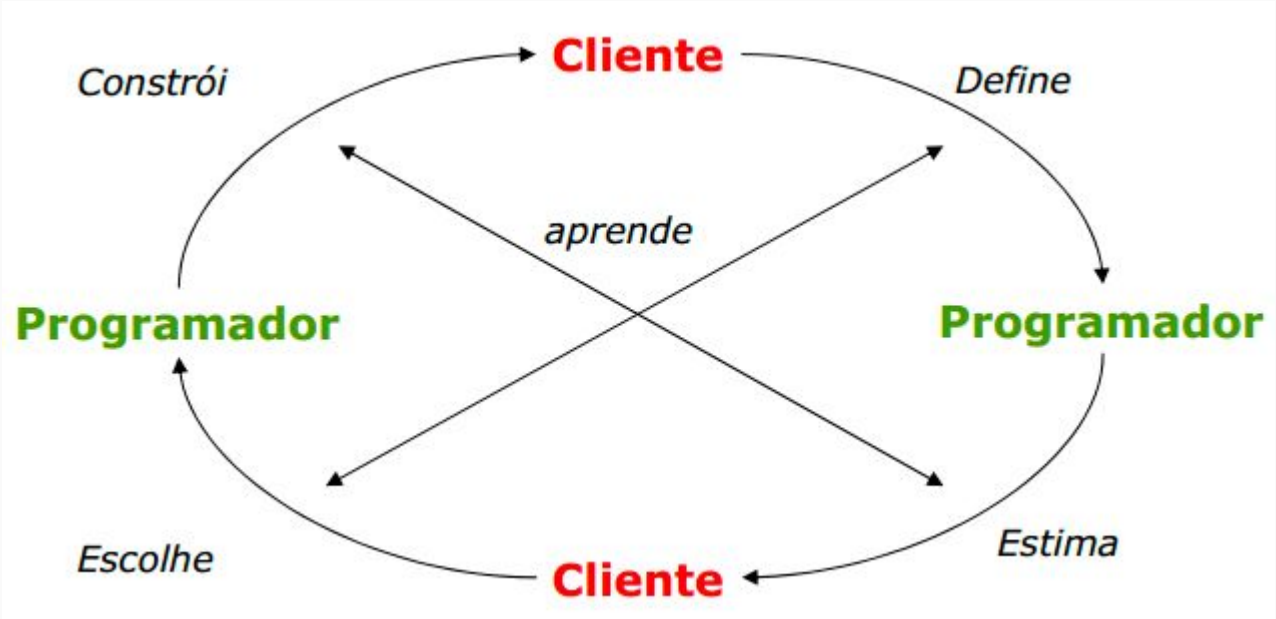
Fases do XP


projeto simples
cartões CRC
soluções pontuais
protótipos
valores das histórias
de usuários
critérios de teste de aceitação
plano de iteração
projeto
planejamento
codificação
refabricação
teste
programação em dupla
Versão
incremento de software
velocidade de projeto registrada
(computada)
teste de unidades
integração contínua
teste de aceitação

Fases do XP


Selecionar estórias de usuários para este release
Quebrar histórias em tarefas
Planejar release
Desenvolver/integar/ testar software
Liberar o sistema
Avaliar o Sistema

Fases do XP


Cliente
Constrói
Define
aprende
Programador
Programador
Escolhe
Estima
Cliente

Práticas do XP

Práticas do XP

- Jogo de **Planeamento** (Planning Game)
- **Pequenas** Versões (Small Releases)
- Projeto **Simples** (Simple Design)
- Time **Coeso** (Whole Team)
- Testes de **Aceitação** (Customer Tests)
- Ritmo **Sustentável** (Sustainable Pace)
- Reuniões em **pé** (Stand-up Meeting)
- Posse **Coletiva** (Collective Ownership)
- Programação em **Pares** (Pair Programming)
- **Padrões** de Codificação (Coding Standards)
- Desenvolvimento Orientado a **Testes** (Test Driven Development)
- **Refatoração** (Refactoring)
- Integração **Contínua** (Continuous Integration)

Jogo do planejamento (Planning Game)

O desenvolvimento é feito em iterações semanais. No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades. Essa reunião recebe o nome de Jogo do Planeamento. Nela, o cliente identifica prioridades e os desenvolvedores as estimam. O cliente é essencial neste processo e, assim, ele fica sabendo o que está acontecendo e o que vai acontecer no projeto.

Pequenas versões (Small releases)

A liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando. As versões chegam a ser ainda menores que as produzidas por outras metodologias incrementais, como o RUP.

Metáfora (Metaphor)

Procurar facilitar a comunicação com o cliente, entendendo a realidade dele. O conceito de rápido para um cliente de um sistema jurídico é diferente para um programador experiente em controlar comunicação em sistemas de tempo real, como controle de tráfego aéreo. É preciso traduzir as palavras do cliente para o significado que ele espera dentro do projeto.

Time Coeso (Whole Team)

A equipe de desenvolvimento é formada pelo cliente e pela equipe de desenvolvimento.

Coesão está, na verdade, ligado ao princípio da responsabilidade única, que foi introduzido por Robert C. Martin no inicio dos anos 2000 e diz que uma classe deve ter apenas uma única responsabilidade e realizá-la de maneira satisfatória.

Testes de Aceitação (Customer Tests)

São testes construídos pelo cliente em conjunto com analistas e testadores, para aceitar um determinado requisito do sistema.

Ritmo Sustentável (Sustainable Pace)

Trabalhar com qualidade, buscando ter ritmo de trabalho saudável (40 horas/semana, 8 horas/dia), sem horas extras. Horas extras são permitidas quando trouxerem produtividade para a execução do projeto.

Reuniões em pé (Stand-up Meeting)

Reuniões em pé para não se perder o foco nos assuntos de modo a efetuar reuniões rápidas, apenas abordando tarefas realizadas e tarefas a realizar pela equipe.

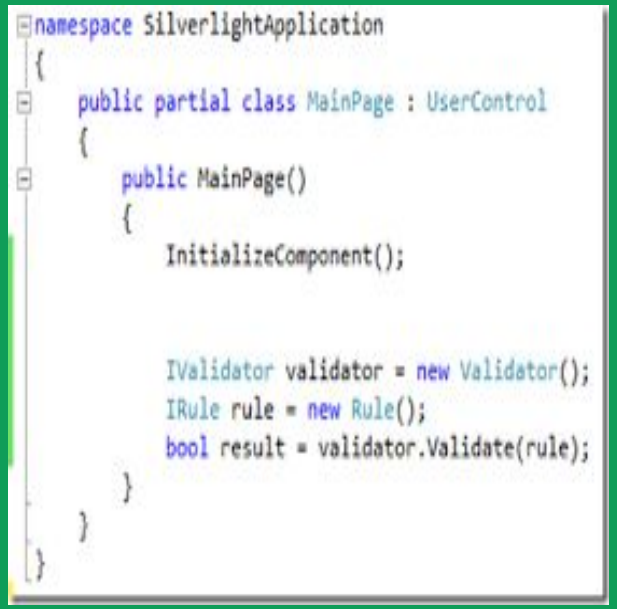
Posse Coletiva (Collective Ownership)

O código fonte não tem dono e ninguém precisa ter permissão concedida para poder modificar o mesmo. O objetivo com isto é fazer a equipe conhecer todas as partes do sistema.

Programação em Pares (Pair Programming)

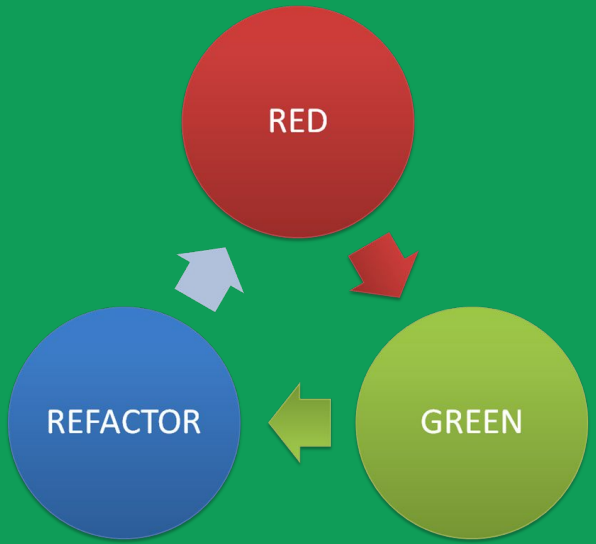
É a programação em par/dupla num único computador. Geralmente a dupla é criada com alguém sendo iniciado na linguagem e a outra pessoa funcionando como um instrutor. Como é apenas um computador, o novato é que fica à frente fazendo a codificação, e o instrutor acompanha ajudando a desenvolver suas habilidades. Dessa forma o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de erros (bugs). Com isto, procura-se sempre a evolução da equipe, melhorando a qualidade do código fonte gerado.

Padrões de Codificação (Coding Standards)



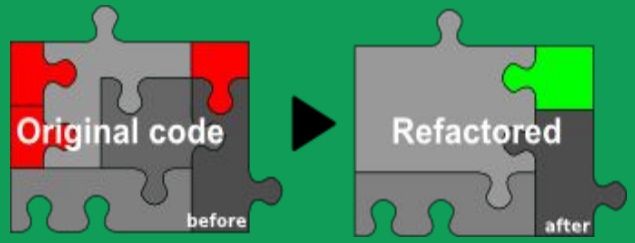
A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras. Dessa forma parecerá que todo o código fonte foi editado pela mesma pessoa, mesmo quando a equipe possui 10 ou 100 membros.

Desenvolvimento Orientado a Testes (Test Driven Development)



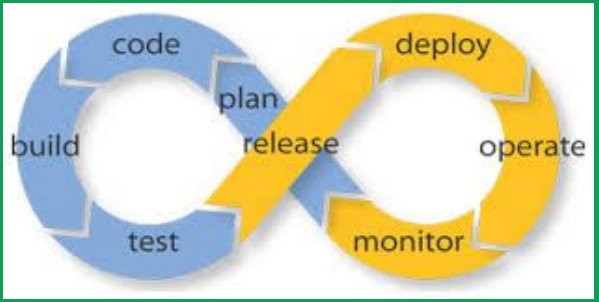
Primeiro crie os testes unitários (unit tests) e depois crie o código para que os testes funcionem. Esta abordagem é complexa no início, pois vai contra o processo de desenvolvimento de muitos anos. Só que os testes unitários são essenciais para que a qualidade do projeto seja mantida.

Refatoração (Refactoring)



É um processo que permite a melhoria contínua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. Refatorar melhora a clareza (leitura) do código, divide em módulos mais coesos e de maior reaproveitamento, evitando a duplicação de código-fonte.

Integração Contínua (Continuous Integration)



Sempre que realizar uma nova funcionalidade, nunca esperar uma semana para integrar na versão atual do sistema. Isto só aumenta a possibilidade de conflitos e a possibilidade de erros no código fonte. Integrar de forma contínua permite saber o status real da programação.

Papéis

Responsáveis

Coach

- Diferente de um arquiteto ou gerente
- Basicamente, um desenvolvedor experiente
- Ajuda novos desenvolvedores
- Interage com gerentes

Tracker

- Medir e comunicar o progresso da equipe
- Feito através de métricas coletadas regularmente

XP

Métodos ágeis costumam ter uma forte ênfase no monitoramento.

- Feito através de métricas coletadas regularmente
- Coleta com overhead mínimo
- Comparação regular entre tempo estimado para uma tarefa e o tempo real necessário
- Big Visible Charts

Histórias de usuário

Requisitos em XP

Em XP, **requisitos** são expressos por meio de **histórias de usuários.**

Histórias são escritas em cartões e quebradas em **tarefas** de implementação.

- Essas **tarefas** são a base para determinar o cronograma e para estimativas de custo
- Uma **história** está mais para um lembrete do que para uma d**escrição detalhada dos requisitos.**

O **cliente** escolhe as histórias que serão incluídas no próximo release com base em suas prioridades e nas estimativas do cronograma.

No projeto da disciplina, usaremos histórias de usuários combinadas com um **backlog**.

História de Usuário

Uma História de Usuário representa uma pequena parte da funcionalidade do sistema a ser construído. Não se trata de uma especificação completa de uma funcionalidade.

Uma história de **usuário** pode ser caracterizada como uma curta e simples descrição da necessidade do cliente.

Apresentar Mapa Animado

O sistema deve apresentar ao usuário um mapa com animações. O mapa será constituído por cruzamentos, semáforos, vias e veículos. O usuário poderá escolher entre uma lista de configurações de mapas pré-definidas.

Prioridade: *Alta*

Exemplo de História de Usuário para um Simulador de Tráfego

Representação do Grafo

A representação computacional do mapa deverá ser implementada utilizando- se um grafo. Uma via será representada através de uma aresta. Logo, a aresta guardará algumas informações relevantes como direção e velocidade máxima de uma via. Vértices representarão esquinas, junções e confluências entre duas ou mais ruas.

XP e Mudançcas

XP e Mudanças

A “**sabedoria convencional**” da engenharia de software é que se deve projetar para mudanças.

- Supõe que antecipar mudanças antes que ocorram, reduz custos em estágios posteriores do desenvolvimento

XP, porém, é calçada na idéia de que esse esforço não vale a pena, já que é muito difícil antecipar as mudanças.

Ao invés disso, o método propõe o **uso de refatoração** para facilitar a incorporação de mudanças, quando elas forem necessárias.

Testes em XP

- Desenvolvimento “teste-antes”.
- Desenvolvimento incremental de testes a partir das histórias de usuários.
- O usuário está envolvido no desenvolvimento de testes de aceitação.
- Conjuntos de testes automáticos são executados para todo o sistema cada vez que um novo release é produzido.

Benefícios do Desenvolvimento “Teste-Antes”

Escrever os testes antes clarifica os requisitos a ser implementados

- Funciona, ao mesmo tempo, como uma especificação da funcionalidade e um projeto detalhado

Os testes são programas ao invés de especificações e podem ser executados automaticamente.
Cada teste construído funciona como teste de regressão nas iterações seguintes.

- Se uma modificação quebra o código existente, esse problema é detectado imediatamente

Considerações Finais

Métodos Ágeis são métodos de desenvolvimento **iterativos e incrementais.**

- Visam **reduzir o overhead** de desenvolvimento, produzindo software de qualidade mais rapidamente.

XP é o **método ágil mais popular.**

A abordagem de XP para testes é um ponto particularmente forte desse método.

- Testes executáveis são escritos **antes do código a ser testado.**

Não são uma panacéa!

- Podem ser úteis ou não, dependendo do contexto.


Dúvidas ?