

Projeto Integrado III

DD - UFC - Quixadá



Prof.: Aníbal Cavalcante

Agenda

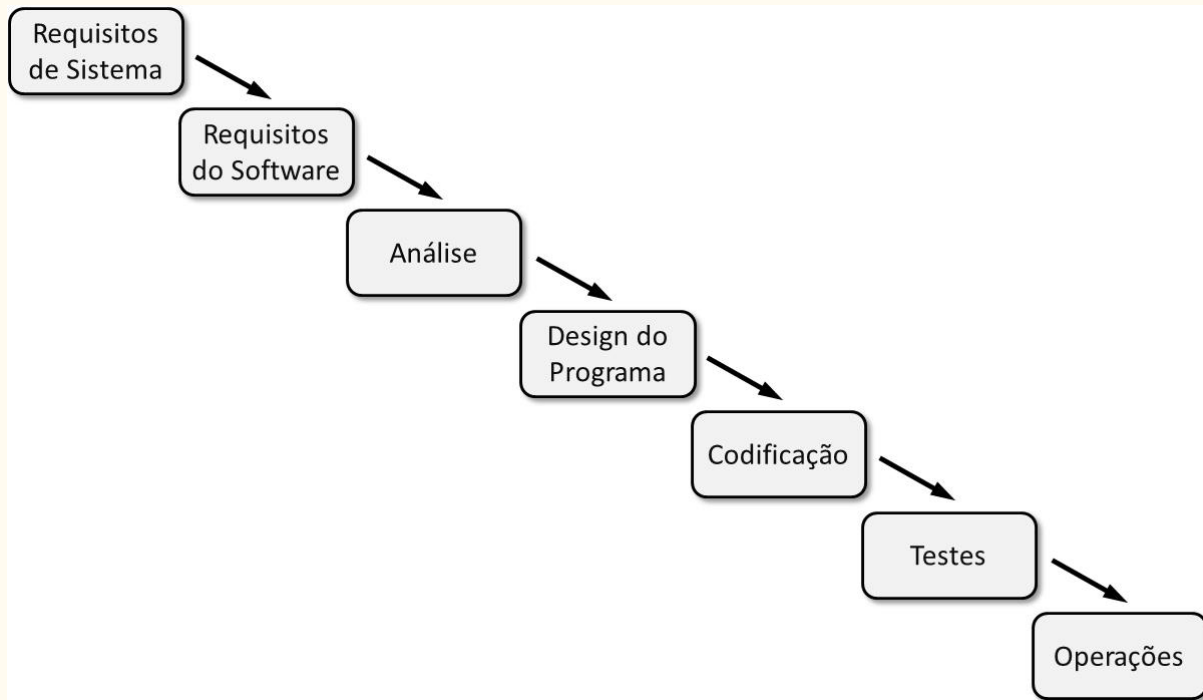
1. O que é Agilidade?
2. Os métodos tradicionais.
3. O manifesto Ágil
4. Os valores Ágeis
5. Os princípios Ágeis
6. Os principais métodos Ágeis
7. Programação Extrema
8. Kanban

O que é Agilidade?

1. Até o final dos anos 90 os processos de desenvolvimento de software eram caracterizados por:
 - a. Fortemente prescritivos. (idéia de receitas de bolo...)
 - b. Foco em planos detalhados definidos no início do projeto. (burocráticos...)
 - c. Micro Gerenciamento. (controle excessivo nos detalhes...)
 - d. Centralização de poder. (equipe engessada e desmotivada...)
 - e. Extensa documentação. (desperdício...)
 - f. As mudanças eram fortemente indesejadas. (dificuldade em lidar com mudanças...)

O que é Agilidade?

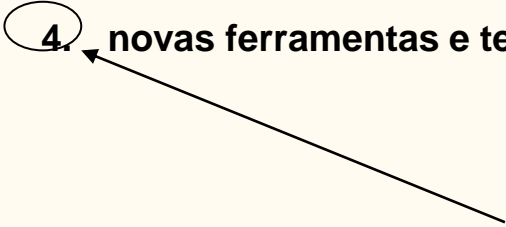
Essencialmente os modelos tradicionais eram uma evolução do modelo em cascata ou waterfall, também chamados de processos pesados.



Os métodos tradicionais

Dentre as razões pelos quais os métodos tradicionais de desenvolvimento de software não funcionam como esperado, temos que:

1. requisitos não são completamente compreendidos antes do início do projeto;
2. usuários só sabem exatamente o que querem após ver uma versão inicial do produto;
3. requisitos mudam frequentemente durante o processo de desenvolvimento;
4. novas ferramentas e tecnologias tornam as estratégias de desenvolvimento imprevisíveis.



novos requisitos podem surgir a partir de novas tecnologias....

Nascimento da Agilidade

- Início dos anos 2000....
 - **Havia a necessidade de métodos menos burocráticos para a construção de software.**
 - **Menor quantidade e maior qualidade nas documentações produzidas.**
 - **Pouca formalidade e regulamentação.**
 - **Focado na entrega contínua de software funcional e na satisfação do cliente/usuário.**
- Algumas dessas metodologias já existiam e eram chamadas de processos “**leves**”, p. ex.: **Lean, Scrum, Extreme Programming (XP), Feature Driven Development (FDD) e Kanban.**

Manifesto Ágil

- Em 2001, um grupo de 17 desenvolvedores renomados publicam o manifesto ágil.



Kent Beck
Software Engineer



Mike Beedle
Computer Scientist



Arie Van Bennekum
Project Manager



Alistair Cockburn
Computer Scientist



Ward Cunningham
Software Developer



Martin Fowler
Software Developer



James Grenning
Software Engineer



Jim Highsmith
Software Developer



Andy Hunt
Software Developer



Ron Jeffries
Software Developer



Jon Kern
Program Manager



Bob Martin
Software Engineer



Stephen J. Mellor
Computer Scientist



Jeff Sutherland
Software Developer



Ken Schwaber
Software Developer



Dave Thomas
Computer Programmer



Brian Marick
Computer Scientist

Manifesto Ágil - Valores

O MANIFESTO ÁGIL

Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:

- **Indivíduos e a interação entre eles** mais que processos e ferramentas;
- **Software em funcionamento** mais que documentação abrangente;
- **Colaboração com o cliente** mais que negociação contratual;
- **Responder a mudanças** mais que seguir um plano.

Mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Manifesto Ágil - 12 princípios

- 1. A 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. Elas trazem vantagem competitiva para o cliente.**
- 3. Entregar frequentemente software funcionando, na menor escala de tempo, através de sprints de desenvolvimento.**
- 4. Usuários e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.**
- 5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário 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 através de conversa face a face.**

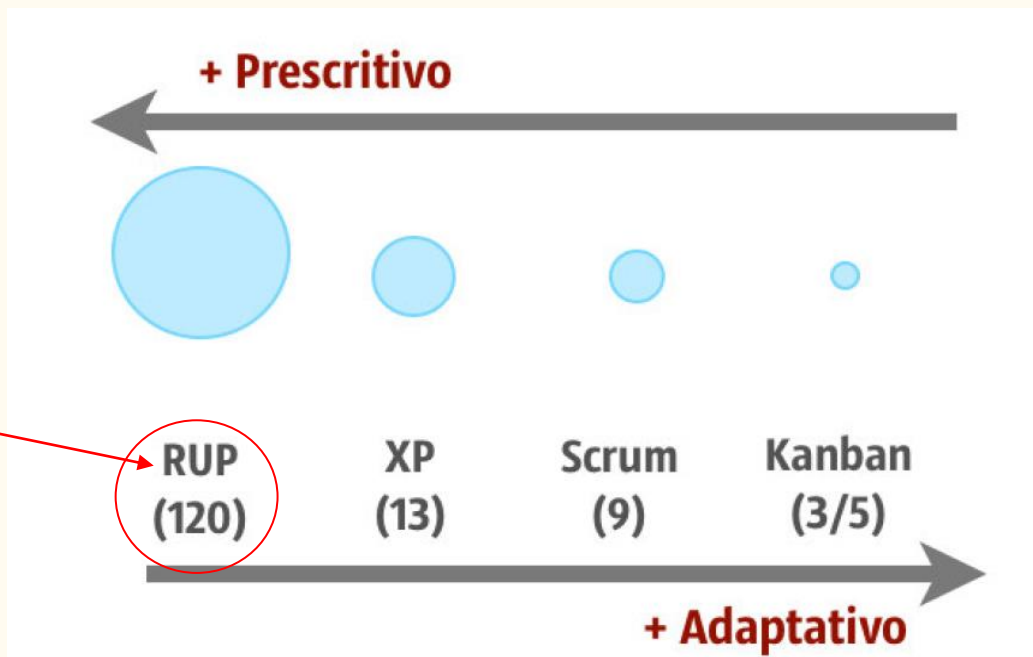
Manifesto Ágil - 12 princípios

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 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 e então refina e ajusta seu comportamento de acordo.

Adaptativos vs. Prescritivos

Métodos ágeis são adaptativos ao invés de prescritivos, por isso, incentivam a melhoria contínua através de ciclos inspeção e adaptação.

Evolução do
Waterfall
desenvolvido
pela IBM



Algumas explicações....

- O **Sprint** representa um espaço de tempo dentro do qual um conjunto de atividades deve ser executado.
- **Entrega contínua** é uma abordagem na qual os times de desenvolvimento lançam produtos de qualidade de forma frequente, previsível e automatizada. Em vez de fazer grandes entregas de uma vez, fazem várias pequenas e rápidas, antecipando a correção de erros e conquistando maior controle de qualidade.

Algumas explicações....

Equipes auto-organizáveis (Sinergia).

Características:

- 1. Autonomia:** após receberem uma missão com objetivos claramente definidos, o time está livre para definir sua própria direção. A alta gerência limita-se a dar orientação, recursos e apoio moral;

Algumas explicações....

Equipes auto-organizáveis (Sinergia).

Características:

- 2. Autotranscedência:** a equipe busca continuamente estender seus limites.
- 3. Fertilização cruzada:** uma equipe multidisciplinar e heterogênea conduz o desenvolvimento do novo produto. Num mesmo ambiente de trabalho o processo de transferência de conhecimento entre seus membros acontece naturalmente.

Alguns dos principais Métodos Ágeis

1. **Extreme Programming XP** (em português Programação Extrema), cobre aspectos técnicos do desenvolvimento de software.
2. **Kanban**, modelo para gestão evolucionária de processos, emprestado do Sistema Toyota de Produção.
3. **Scrum**, cobre aspectos gerenciais do desenvolvimento de software.
4. **TDD**, é o desenvolvimento orientado a testes (Test Driven Development). Escrevemos os testes antes de escrever o código de produção.

Extreme Programming XP

- Criado nos anos 90, o XP cobre aspectos técnicos do desenvolvimento de software como codificação, design e testes.
- É uma metodologia que se destina a melhorar a qualidade do software e a capacidade de resposta às mudanças nas necessidades dos clientes.
- Defende a freqüente liberação de novas versões, em curtos períodos de desenvolvimento.
- Adota pontos de controle nos quais as novas necessidades dos clientes podem ser adotadas.

Extreme Programming XP

- Segundo o XP, durante o processo de desenvolvimento há quatro atividades básicas a serem executadas:
 - **Ouvir.**
 - **Desenhar.**
 - **Codificar.**
 - **Testar.**

Extreme Programming XP - Ouvir

Um dos princípios do método XP é ter o Cliente Presente. Ouvir e documentar suas necessidades, entender qual a visão do projeto.

Todos os membros da equipe precisam ser contagiados com a visão do projeto, ou seja, devem saber responder:

- 1. Que objetivo o projeto deve atingir?**
- 2. Por que esse projeto agregará valor ao cliente/usuário?**
- 3. Como medir se o projeto foi bem sucedido?**

Como transmitir a visão do projeto

O discurso do elevador (Harvard Business School) - Investidor bilionário....

1. **Quem:** O que você mais quer que seu ouvinte se lembre a respeito do seu produto?
2. **O quê:** De que forma seu produto agregará valor e trará resultados ao negócio de seu ouvinte?
3. **Por quê:** Quais são os benefícios exclusivos, os diferenciais que seu produto oferece?
4. **Objetivo:** O que você espera que o ouvinte faça?

Extreme Programming XP - Desenhar (Simplicidade)

- Um bom desenho é a ferramenta para que todos possam compreender melhor um sistema complexo, suas estruturas, dependências e regras de negócio.
- O objetivo é evitar criar funcionalidades que não sejam necessárias.
- Evita-se também dependências dentro de um sistema, o que significa que ao alterar uma parte do sistema, isto não afetará outras partes do mesmo.

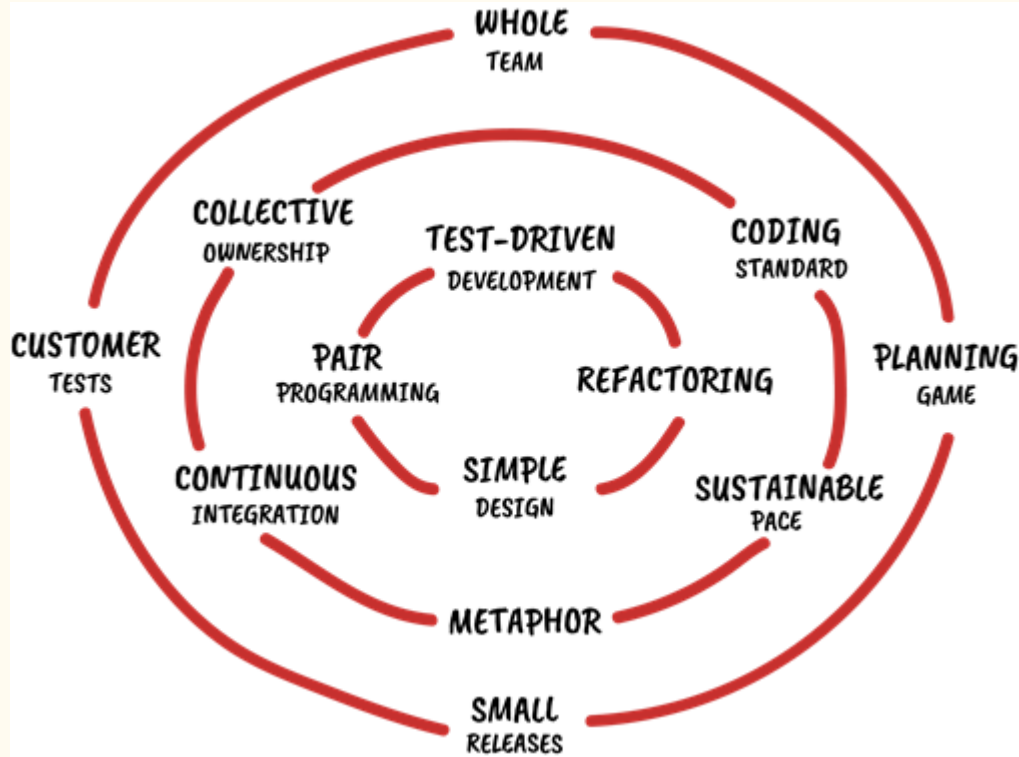
Extreme Programming XP - Desenhar (Planning Game)

- **Planejar é uma atividade constante**, para responder rapidamente às mudanças.
- O desenvolvimento é dividido em **iterações curtas** de uma ou poucas semanas de duração.
- A cada iteração é realizado uma reunião de planejamento, junto com o cliente, para definir o que será feito.
- O cliente tem a informação necessária sobre o que agrega valor ao software.
- A equipe tem a informação de quanto custa para agregar tal valor.
- **O objetivo é minimizar os custos e maximizar o valor agregado.**

Extreme Programming XP - Codificar

- Codificar é a atividade principal para o desenvolvimento de software, sem código não há software.
- O cliente deve estar **sempre acessível** para feedbacks e dúvidas.
- Os testes de unidade são escritos antes do código de produção;
- Todo o código fonte que será executado em produção é desenvolvido em pares;
- A integração do código é realizada através da prática de integração contínua;
- O código fonte é **coletivo**, pertencendo a todos os membros da equipe;
- Deve ser escrito de acordo com os padrões definidos pelo próprio time.

Extreme Programming XP - Práticas

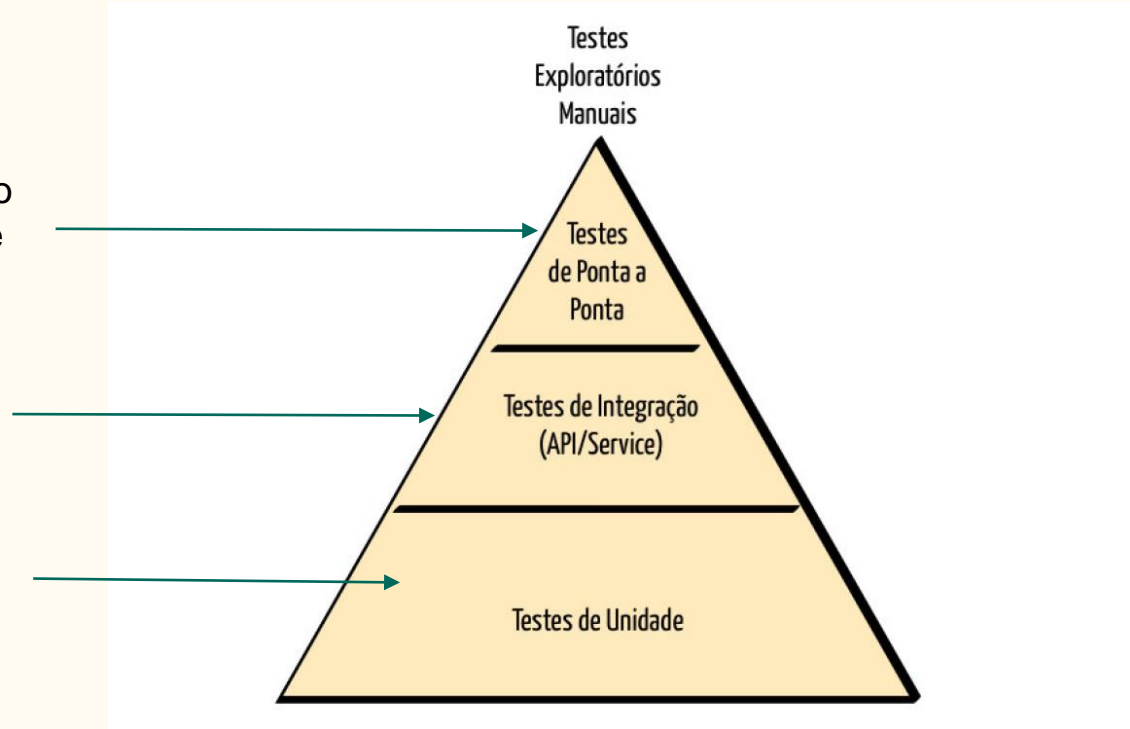


XP - Testes (client assertions)

Teste de aceitação com o usuário final.(teste caixa-preta ou alpha e beta)

Os módulos são combinados e testados em grupo.

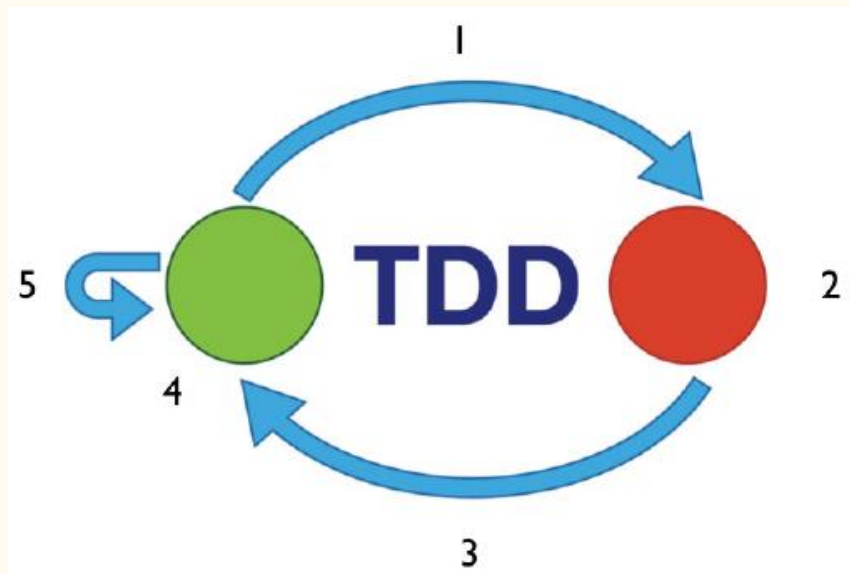
Testa os métodos e classes individualmente.



XP - Test Driven Development (TDD)

Testando antes com Desenvolvimento Guiado por Testes.

1. Escreva o teste de uma nova funcionalidade.
2. Executa-se todos os testes e observa o novo teste falhar.
3. Implemente a funcionalidade para fazer o seu teste passar.
4. Execute todos os teste e observe que foram bem sucedidos.
5. Simplifique e melhore a funcionalidade se necessário.



XP - Codificar - O Código deve ser limpo e padronizado

Características de um código limpo.

- 1. Elegante, fácil de se compreender, e agradável de se ler.**
- 2. Simples e direto.**
- 3. Sem duplicidade e bem testado.**
- 4. Parâmetros, Métodos, Funções, Classes e Variáveis com nomes significativos.**
- 5. Código é auto explicativo.**
- 6. Código bem formatado (ler sem a barra de rolagem).**
- 7. Métodos e Classes devem ter uma única responsabilidade.**

XP - Código de Propriedade Coletiva



1. **Consiste na ideia de que o time é responsável por todo o repositório de código.**
2. **Os indivíduos são responsáveis por todo o código, não apenas pelo código que eles mesmos escreveram.**
3. **Manter e garantir a qualidade do código seja responsabilidade de todos os membros do time.**
4. **Código Coletivo é uma boa vacina contra ilhas de conhecimento, e reduz o risco se alguém, por alguma razão, deixar o projeto.**
5. **Um por todos e todos por um. (Erros e Acertos)**

XP - Linguagem ubíqua (“falar” a língua do usuário/cliente)

Desenvolvedor: - “Nós estamos com um problema no NFBusinessDelegate quando enviamos o objeto para o Service, parece que a transação não é iniciada e o DAO e, então, não funciona bem.”

Cliente: - ?

A mesma linguagem de negócio deve ser utilizada no próprio código fonte, na hora de dar nomes a parâmetros, métodos, variáveis, classes, testes, etc.

XP - Codificar - Programação em Par

- Dois programadores trabalham em um mesmo problema, ao mesmo tempo e em um mesmo computador.
- O Trabalho é dividido em dois papéis:
 - **Condutor:** assume o teclado e digita os comandos que farão parte do programa.
 - **Navegador:** acompanha fazendo um trabalho de estratégia.
- Os dois programadores trocam de papel com frequência. (p.ex.: a cada hora de trabalho)
- O trabalho do par termina quando a funcionalidade estiver pronta.

XP - Codificar - Programação em Par

- A pirâmide de pares:

	CAROL	ROBSON	LUIZ	ALINE
CAROL				
ROBSON	XXX			
LUIZ	X			
ALINE	XX	XXX	X	

Pirâmide dos Pares

XP - Codificar - Entregas Pequenas (Small Releases)

- Entrega é algo que é implantado no cliente, ou seja, está pronto para ser utilizado.
- As Entregas normalmente são pequenas e frequentes (a cada mês por exemplo).
- As funcionalidades prioritárias são desenvolvidas mais cedo para serem entregues mais rapidamente ao cliente.
- O feedback do cliente sobre a entrega deve ser avaliado e as possíveis correções entram na próxima entrega.

XP - Codificar - Integração Contínua

- **É uma prática em que os desenvolvedores juntam suas alterações de código, a cada lançamento de um nova entrega.**
- **A ideia é testar constantemente se as partes(unidades) funcionam quando postas em conjunto.**
- **Para o XP o código deve estar sempre integrado e pronto para ser entregue, a cada nova entrega.**
- **Chamamos de build (construção) o processo de gerar um pacote executável de software.**

XP - Codificar - Ritmo sustentável

- **Para o XP, desenvolvedores de software não devem trabalhar mais de 40 horas por semana.**
- **Adota o conceito de que as pessoas executam melhor e mais criativamente se eles estiverem descansadas.**
- **A equipe deve adotar um ritmo sustentável de produtividade e mante-lo até o fim do projeto.**
- **O líder da equipe é o responsável por descobrir e manter esse ritmo.**

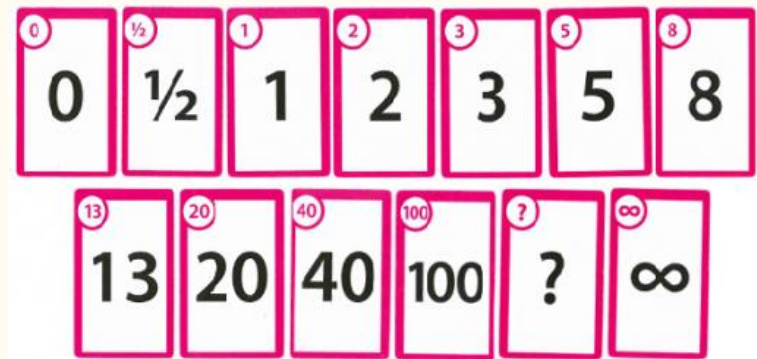
XP - Codificar – Jogo do Planejamento

- 1 - Seu objetivo é determinar rapidamente o escopo da próxima entrega.
- 2 - Combina prioridades do negócio e estimativas técnicas.
- 3 - O cliente deve estar presente e ele define as funcionalidade para a próxima entrega.
- 4 - A equipe de desenvolvimento, baseada em sua experiência, irá fornecer as estimativas de desenvolvimento.
- 5 - O cliente e a equipe devem chegar a um consenso sobre o que deve ser entregue, ambos têm que ceder.

XP - Codificar – Jogo do Planejamento

Ferramenta: Planning Poker

Resumidamente, em um jogo Planning Poker, cada membro da equipe de desenvolvimento recebe um conjunto de cartas com os valores de uma certa sequência, que irá determinar, ao final do jogo, uma estimativa para uma determinada funcionalidade.



Com o Planning Poker você pode priorizar as tarefas e fazer estimativas do esforço que é exigido para executá-las.

XP - Definindo o significado de Pronto (checklist)

1. **Código refatorado se houver necessidade.**
2. **Código dentro dos padrões de codificação.**
3. **Código revisado e feito em par.**
4. **Código integrado no sistema de controle de versão.**
5. **Documentação de arquitetura atualizada.**
6. **Testes de unidade realizados.**
7. **Testes de aceitação realizados.**
8. **Testes exploratórios realizados.**
9. **Nenhum defeito conhecido pendente.**
10. **Cliente/Usuário aceitou o software.**
11. **Manual do Usuário atualizado.**

Exercícios

Explique com suas palavras cada prática do XP.

- Programação em Par
- Desenvolvimento Orientado a Testes.
- Jogo de Planejamento.
- Integração Contínua.
- Entregas Curtas.