Métodos Ágeis

Edes Garcia da Costa Filho
edes_filho@dc.ufscar.br

Tópicos

- Histórico;
- Valores;
- Métodos Ágeis x Modelos Tradicionais;
- Exemplo:
  - Extreme Programming (XP).
- Referências Bibliográficas.

Histórico

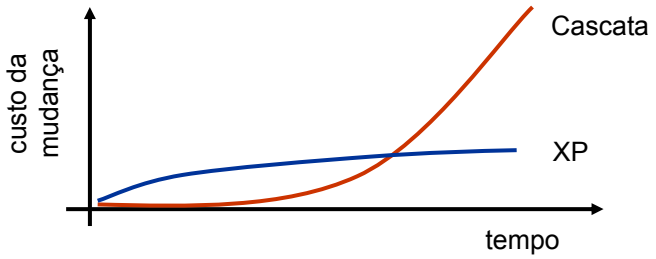
- A entrega de software em prazos e custos estabelecidos nem sempre é conseguida.
- Formalidade nos modelos de processo propostos nos últimos 30 anos.
- Métodos Ágeis:
  - Propõem desenvolvimento de software de forma mais rápida, mas com qualidade.
- Popularização:
  - O Movimento Ágil tomou forças em 2001 com a publicação do Manifesto Ágil de Desenvolvimento de Software.

Valores

- Manifesto Ágil:
  - Indivíduos e interações são mais importantes que processos e ferramentas;
  - Software funcionando é mais importante do que documentação detalhada;
  - Colaboração dos clientes é mais importante do que negociação de contratos;
  - Adaptação às mudanças é mais importante do que seguir um plano.
- Diferenças entre métodos ágeis e modelos propostos anteriormente pela Engenharia de Software (“tradicionais”)?

Métodos Ágeis x Modelos Tradicionais

- Tratamento das mudanças durante o desenvolvimento.
- Modelos Tradicionais:
  - Resistir à mudanças;
  - Planejar muito bem antes.
- Métodos Ágeis:
  - Abraçar as mudanças;
  - Planejar o tempo todo.



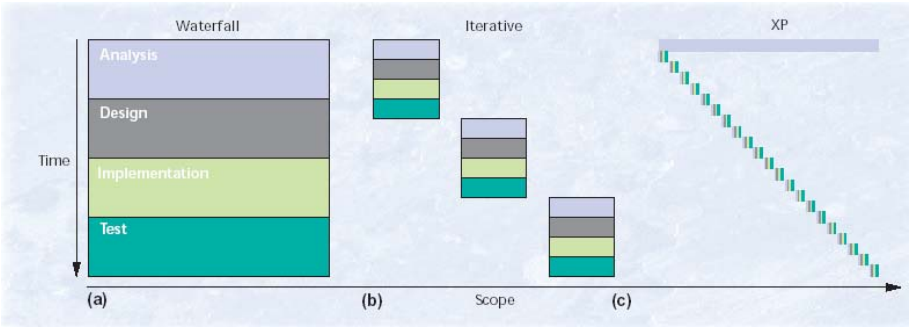
Métodos Ágeis x Modelos Tradicionais

- Características:

Modelos Tradicionais	Métodos Ágeis
• Previsibilidade; • Controlar mudanças; • Burocráticos; • Excesso de documentação; • Enfatizam os aspectos de Engenharia do desenvolvimento.	• Adaptabilidade; • Planejamento é contínuo; • Documentação essencial; • Mudanças rápidas; • Enfatizam os aspectos humanos do desenvolvimento.

Métodos Ágeis x Modelos Tradicionais

- (a) Cascata: Longo ciclo de desenvolvimento
- (b) Espiral: Iterações
- (c) XP: Iterações curtas



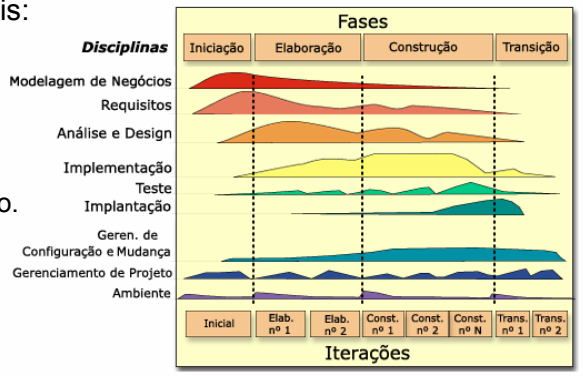
Fonte: [Beck 1999b]

Exemplos

- Modelos Tradicionais:
  - Modelo Cascata;
  - Modelo de Prototipação;
  - Modelo Espiral;
  - Modelo Concorrente.
- Métodos Ágeis:
  - Extreme Programming (XP);
  - Scrum;
  - Crystal;
  - Agile Modeling (AM).
- Rational Unified Process (RUP)?

RUP

- *Framework* de processo:
  - RUP pode ser adaptado e estendido de acordo com as necessidades da organização.
- Adaptável;
- Elementos principais:
  - Disciplinas;
  - Papéis;
  - Atividades;
  - Artefatos;
  - Fluxos de trabalho.



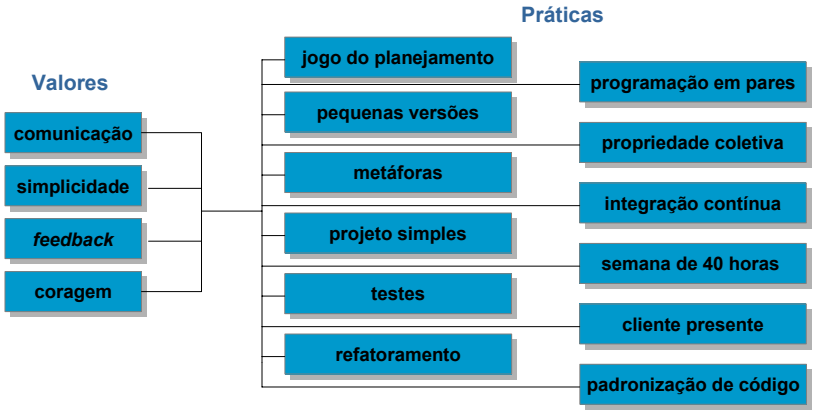
Fonte: [RUP 2002]

Extreme Programming (XP)

- **Iterações curtas:**
  - Duram de duas a quatro semanas.
- **Poucos artefatos exigidos:**
  - Artefatos devem ser simples e de valor.
    - Código é o principal artefato.
    - Existem outros: User Stories, Testes de Unidade, Testes de Aceitação, Estimativas (Stories e Tarefas), entre outros.
- **Menor quantidade de atividades no processo:**
  - Quatro atividades básicas: Codificação (*Coding*), Teste (*Testing*), Escuta (*Listening*) e Projeto (*Designing*).
  - Atividades estruturadas de acordo com as práticas.
- **Poucos papéis:**
  - Sete papéis: Programador (*Programmer*), Cliente (*Customer*), Testador (*Tester*), Investigador (*Tracker*), Orientador (*Coach*), Consultor (*Consultant*) e Gerente (*Manager*).

Extreme Programming (XP)

- Equipes pequenas e médias (dois a dez membros);
- Reúne práticas de implementação em um conjunto coerente, acrescentando idéias de processo.



Extreme Programming (XP)

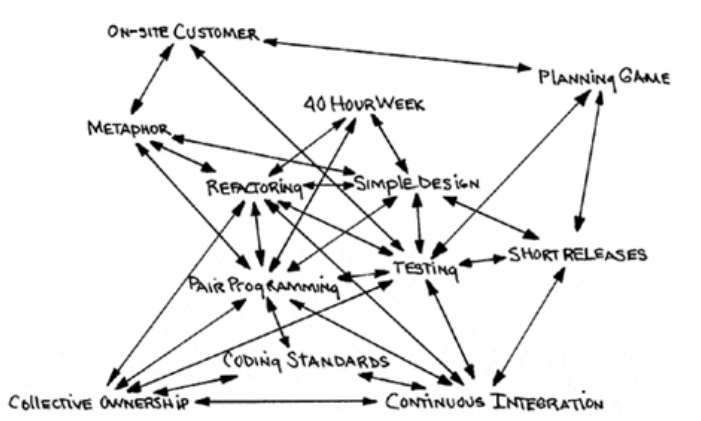
- **Jogo do Planejamento (*The Planning Game*)**
  - Determina rapidamente o escopo das próximas versões, combinando as prioridades de negócio e as estimativas técnicas.
- **Pequenas Versões (*Small releases*)**
  - A equipe deve colocar rapidamente um sistema simples em produção, uma versão pequena, e depois entregar novas versões em poucos dias ou semanas.
- **Metáfora (*Metaphor*)**
  - Uma metáfora é uma descrição simples de como o sistema funciona. Ela fornece uma visão comum do sistema e guia o seu desenvolvimento.
- **Projeto simples (*Simple design*)**
  - O sistema deve ser projetado o mais simples possível. Complexidade extra é removida assim que descoberta.
- **Testes (*Testing*)**
  - Os programadores escrevem testes de unidade continuamente. Esses testes são criados antes do código e devem ser executados perfeitamente para que o desenvolvimento continue. Os clientes também escrevem testes para validar se as funções estão finalizadas.
- **Refatoração (*Refactoring*)**
  - Os programadores reestruturam o sistema durante todo o desenvolvimento, sem modificar seu comportamento externo. Isso é feito para simplificar o sistema, adicionar flexibilidade ou melhorar o código.

Extreme Programming (XP)

- **Programação em pares (*Pair programming*)**
  - Todo código produzido é feito em pares, duas pessoas trabalhando em conjunto na mesma máquina.
- **Propriedade coletiva (*Collective ownership*)**
  - Qualquer um pode alterar qualquer código em qualquer momento, o código é de propriedade coletiva.
- **Integração contínua (*Continuous integration*)**
  - Uma nova parte do código deve ser integrada assim que estiver pronta. Consequentemente, o sistema é integrado e construído várias vezes ao dia.
- **Semana de 40 horas (*40-hour week*)**
  - XP defende um ritmo de trabalho que possa ser mantido, sem prejudicar o bem estar da equipe. Trabalho além do horário normal pode ser necessário, mas fazer horas extras por períodos maiores que uma semana é sinal de que algo está errado com o projeto.
- **Cliente junto aos desenvolvedores (*On-site customer*)**
  - Os desenvolvedores devem ter o cliente disponível todo o tempo, para que ele possa responder às dúvidas que os desenvolvedores possam ter.
- **Padronização do Código (*Coding standards*)**
  - Os programadores escrevem o código seguindo regras comuns enfatizando a comunicação por meio do código.

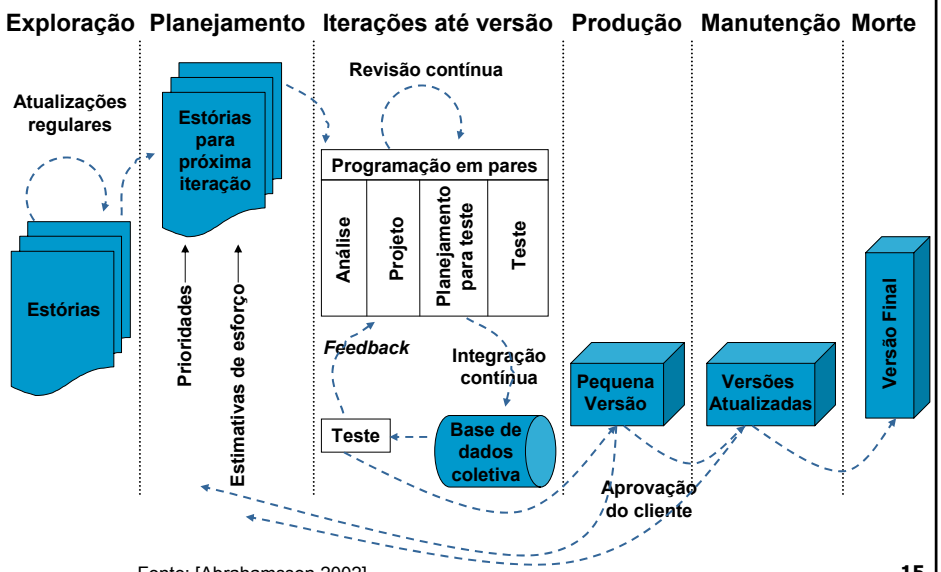
Extreme Programming (XP)

- As práticas apóiam umas as outras.
- Todas as práticas devem ser executadas?



Fonte: [Beck 1999a]

Extreme Programming (XP)



Fonte: [Abrahamsson 2002]

Referências Bibliográficas

- [Beck 1999a] Beck, K. *Extreme Programming Explained – Embrace Change*. Addison-Wesley. 1999.
- [Beck 1999b] Beck, K. *Embracing Change with Extreme Programming*. IEEE Computer, October, 1999.
- [Beck 2001] Beck, K. et al. *Manifesto for Agile Software development*. 2001. Disponível em: <http://www.agilemanifesto.org/>.
- [RUP 2002] Rational Software Corporation. RUP - Rational Unified Process: Versão 2002.05.00.
- [Abrahamsson 2002] Abrahamsson, P. et al. *Agile software development methods: reviews and analysis*. Espoo: VTT Publications, 2002. Disponível em: <http://www.inf.vtt.fi/pdf/publications/2002/ P478.pdf>.