Desenvolvendo Software Livre com Programação eXtrema

Dairton Bassi

FISL 7.0

abril/2006

IME-USP

AgilCoop

dairton@ime.usp.br

Panorama sobre o Desenvolvimento de Software

A sociedade demanda:

- Grande quantidade de sistemas/aplicações
- Sistemas complexos, software distribuído, muitas tecnologias, etc
- Requisitos mutantes

Mas, infelizmente...

- É difícil produzir software de alta qualidade
- Nem todas as equipes conseguem bons resultados

Dificuldades da produção de Software

- Gerenciamento da equipe
- Comunicação interna
- Entendimento das necessidades do usuário
- Negociação com o cliente
- Mudanças de requisitos
- Prazos curtos

Problemas com metodologias

- Supõem que é possível prever o futuro.
- Pouca interação com os clientes.
- Ênfase na burocracia (documentos, formulários, processos, controles rígidos, etc.).
- Evolução baseada em cronogramas pré-definidos.
- Demora para obter versões funcionais.

Conseqüências

- Grande quantidade de erros
- Software sem flexibilidade
- Não atendimento de todas as necessidades do cliente
- Falta de confiança no software

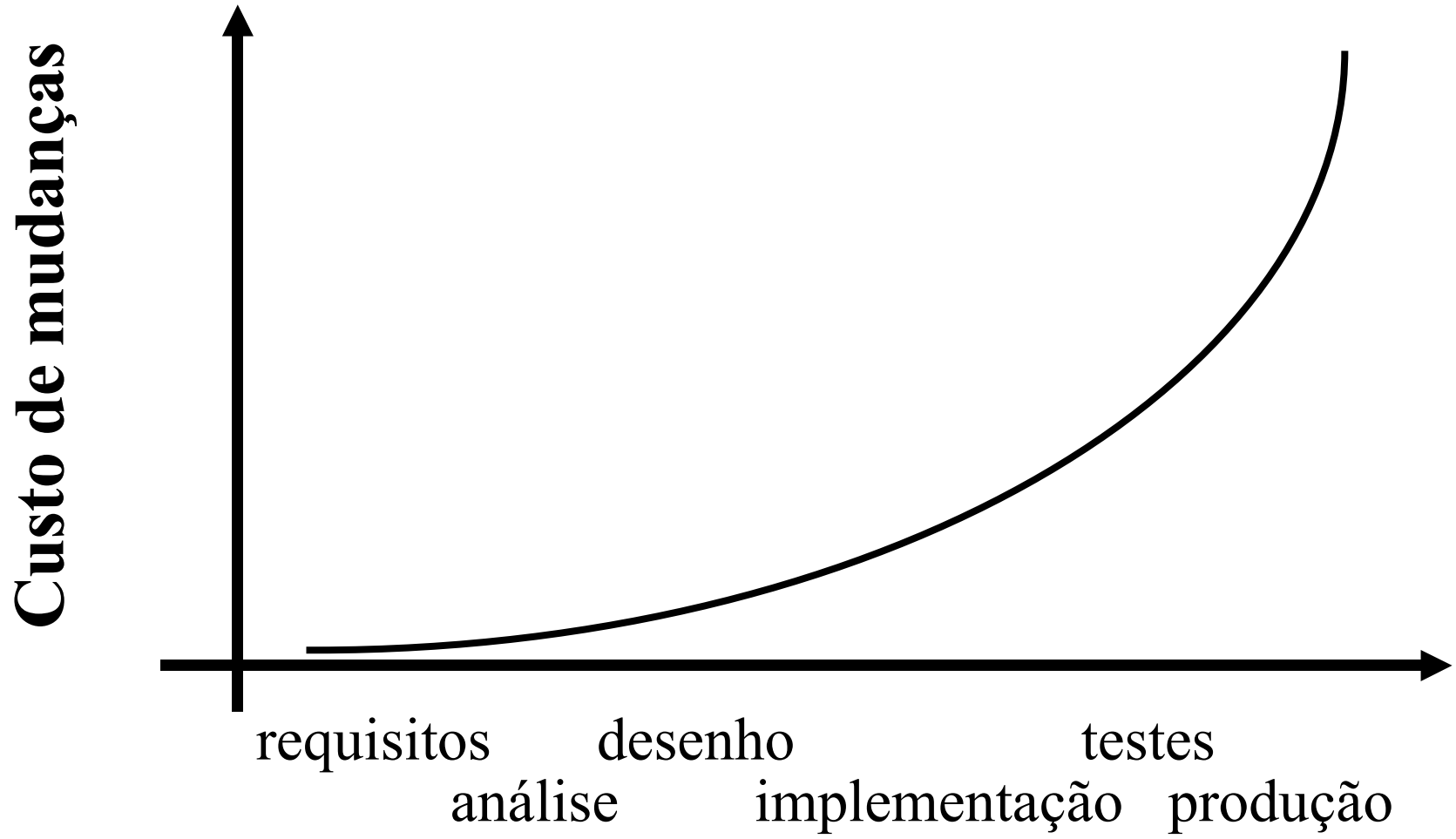
Como resolver esses problemas?

- Melhores Tecnologias
  - Componentes (reutilização de código)
  - Padrões de Projeto (reutilização de idéias)
- Melhores Metodologias
  - Métodos Ágeis (nosso foco)
  - outras... (RUP, relacionadas a CMMi, etc.)

Modelo Tradicional de Desenvolvimento de Software

0. Levantamento de Requisitos
    1. Análise de Requisitos
        2. Desenho da Arquitetura
            3. Implementação
                4. Testes
5. Produção / Manutenção

Conseqüência da abordagem tradicional



Métodos Ágeis de Desenvolvimento de Software

- Movimento iniciado por programadores experientes e consultores em desenvolvimento de software.

- **Manifesto Ágil**: Assinado em fevereiro/2001 por 17 desenvolvedores em Utah.

O Manifesto do *Desenvolvimento Ágil de Software*

1. ***Indivíduos e interações*** são mais importantes que *processos e ferramentas.*
2. ***Software funcionando*** é mais importante do que *documentação completa e detalhada*.
3. ***Colaboração com o cliente*** é mais importante do que *negociação de contratos*.
4. **Adaptação a mudanças** é mais importante do que *seguir o plano inicial*.

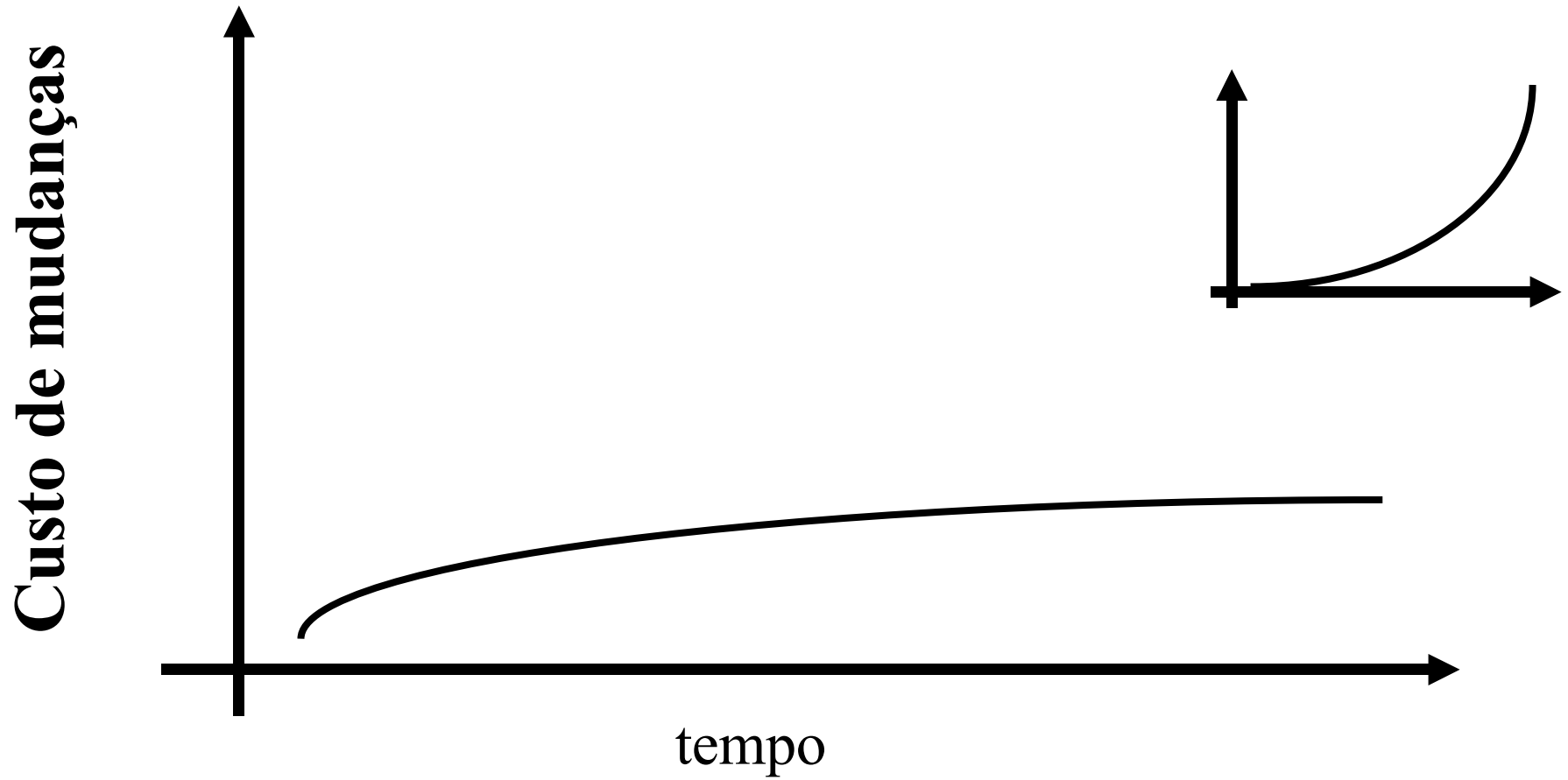
Mas... o que é XP?

XP é uma metodologia ágil de desenvolvimento de software que reúne boas práticas de programação e de gerenciamento de projetos com o objetivo de produzir software de alta qualidade.

Princípios Básicos de XP

- *Feedback* rápido
- Simplicidade é o melhor negócio
- Carregue a bandeira das mudanças / Não valorize o medo
- Alta qualidade do código
- Comunicação intensa

E se o mundo fosse assim?



Reações a XP

- Alguns odeiam, outros amam. Quem gosta de programar ama!
- Deixa o bom programador livre para fazer o que ele faria se não houvesse regras.
- Força o [mau] programador a se comportar de uma forma similar ao bom programador.

As 12 Práticas de XP

1 - Planejamento

2 - Fases Pequenas

3 - Metáfora

4 - Design Simples

5 - Testes

6 - Refatoração

7 - Programação Pareada

8 - Propriedade Coletiva

9 - Integração Contínua

10 - Semana de 40 horas

11 - Cliente junto aos desenvolvedores

12 - Padronização do código

Equipe

- Grupos de 2 a 20 programadores
- Membros da equipe:
  - Programadores (sem hierarquia)
  - Cliente (Contratante)
  - Coach (Técnico)
  - Tracker (Acompanhador)

Testes

- É o que dá segurança e coragem à equipe
- Testes de unidades *(Unit tests)*
  - escritos pelos programadores para testar cada elemento do sistema.
- Testes de funcionalidades *(Functional tests)*
  - escritos pelos clientes para testar a funcionalidade do sistema.

Programação Pareada

- Erro de um detectado imediatamente pelo outro.
- Maior diversidade de idéias, técnicas, algoritmos.
- Enquanto um escreve, o outro pensa em contra-exemplos, problemas de eficiência, etc.
- É mais difícil produzir código feio (*gambiarras*) na frente do parceiro.
- Pareamento de acordo com especialidades.

O Código

- Padrões de estilo adotados pelo grupo inteiro.
- O mais claro possível.
  - XP não se baseia em documentações detalhadas e extensas (perde-se sincronia).
- Comentários sempre que necessários.

Refatoração

- Uma [pequena] modificação no sistema que não altera o seu comportamento funcional.
- Melhora alguma qualidade não-funcional:
  - simplicidade
  - flexibilidade
  - clareza
  - desempenho

Exemplos de Refatoração

- Mudança do nome de variáveis
- Mudanças nas interfaces dos objetos
- Pequenas mudanças arquiteturais
- Encapsular código repetido em um novo método
- Generalização de métodos
  - `raizQuadrada(float x)⇒ raiz(float x, int n)`

Propriedade Coletiva do Código

- Propriedade do código:
  - Modelo tradicional: só autor de uma função pode modificá-la.
  - XP: o código pertence a todos.
- Se alguém identifica uma oportunidade para simplificar, consertar ou melhorar código escrito por outra pessoa, que o faça. Mas rode os testes!

Um dia na vida de um programador XP (1/2)

- Escolhe uma história
- Procura um par livre
- Escolhe um computador para programação pareada
- Discute modificações recentes no sistema
- Discute história com o parceiro

Um dia na vida de um programador XP (2/2)

- Pensa em novos testes
- Discute a melhor maneira de implementar
- Implementa
- Executa os testes
- Integra o novo código

No que um programador XP pensa

- Executar testes antigos
- Encontrar oportunidades para simplificação
- Modificar a implementação para suportar uma nova funcionalidade
- Escrever novos testes
- Compreender as necessidades do cliente
- Enquanto o código não passa por todos os testes o trabalho não está terminado
- Integrar código novo ao repositório

E se eu não me encaixo nesse perfil?

- Você pode aprender muito sobre desenvolvimento de software com XP.

- No início se dizia:
  - “Ou você é 100% eXtremo ou não é eXtremo. Não dá prá ser 80% XP.”
  - Hoje não é mais assim.

Para fazer XP, é preciso lembrar...

- Supere o medo de mudanças
- Siga em frente sem saber tudo sobre o futuro
- Confie no trabalho da sua equipe
- Jogue código fora se necessário
- Negocie com o cliente
- Escreva testes sempre

As 12 Práticas

1 - Planejamento

2 - Fases Pequenas

3 - Metáfora

4 - Design Simples

5 - Testes

6 - Refatoramento

7 - Programação Pareada

8 - Propriedade Coletiva

9 - Integração Contínua

10 - Semana de 40 horas

11 - Cliente junto aos desenvolvedores

12 - Padronização do código

Mais uma prática...

Conserte XP quando a metodologia não for adequada

Mais Informações

www.agilealliance.org

htt://agilcoop.incubadora.fapesp.br

www.extremeprogramming.org

Dúvidas

dairton@ime.usp.br - agilcoop@gmail.com