Métodos Ágeis e Programação Extrema (XP)

Fernando Castor Filho
fernando.castor@dsc.upe.br

Departamento de Sistemas e Computação
Escola Politécnica de Pernambuco
Universidade de Pernambuco

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

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

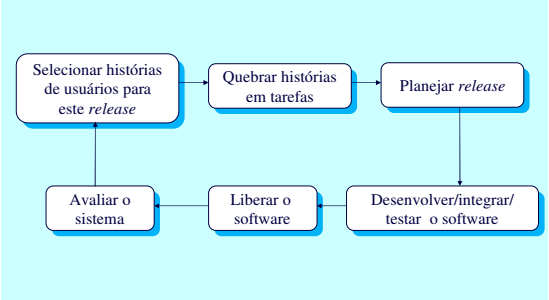
Exemplos de Métodos Ágeis

- **Programação Extrema** (XP)
- Scrum
- Processo Unificado Ágil
- Processo Whitewater

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

O Ciclo de Vida de *Releases* de XP



Práticas da Programação Extrema (1)

- Jogo do Planejamento
- *Releases* Pequenos
- Projeto Simples
- Desenvolvimento Teste-Antes
- Refatoração

Práticas da Programação Extrema (2)

- Programação em Pares
- Propriedade Coletiva do Código
- Integração Contínua
- Ritmo Sustentável
- Cliente no Local do Desenvolvimento

XP e Princípios Ágeis

- Desenvolvimento incremental é apoiado por *releases* frequentes e pequenos
- Envolvimento do cliente é completo, já que ele é considerado membro da equipe
- Programação em pares, propriedade coletiva e ritmo sustentável apoiam o princípio de Pessoas e não Processos
- Mudanças são aceitas através de várias práticas (*releases curtos*, refatoração, desenvolvimento "teste-antes", etc.)
- Simplicidade é mantidade através de projeto simples e refatoração de código

Requisitos em XP

- Em XP, requisitos são expressos por meio de histórias de usuários
- Histórias são escritas em cartões e o time de desenvolvimento as quebra em tarefas de implementação.
  - Essas tarefas são a base para determinar o cronograma e para estimativas de custo
  - Em geral, o mapeamento histórias - tarefas é 1:1
  - Uma história está mais para um lembrete do que para uma descrição detalhada dos requisitos
- O cliente escolhe as histórias que serão incluídas no próximo *release* com base em suas prioridades e nas estimativas do cronograma

Exemplo de História de Usuário para um Editor de Texto

Iniciar o Editor

Editor começa abrindo o último documento com o qual o usuário estava trabalhando.

13

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.

14

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

15

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 é responsável por checar se foi bem sucedido
- Cada teste construído funciona como teste de regressão nas iterações seguintes.
  - Se uma modificação quebra o código existente, esse problema é detectado imediatamente

16

Programação em Pares

- Em XP, programadores trabalhamem parem, sentando-se juntos para desenvolver código
- Isso auxilia na propriedade coletiva do código e espalha conhecimento por todo o time
- Funciona como um processo informal de revisão
  - Mais de uma pessoa olha para cada linha de código
- Há estudos que sugerem que a produtividade da programação em pares é similar à de duas pessoas trabalhando independentemente

17

Considerações Finais

- Métodos ágeis são métodos de desenvolvimento iterativos e incrementais
  - Visam reduzir o *overheard* de desenvolvimento, produzindo software de qualidade mais rapidamente.
- Programação extrema é o representante mais popular dos métodos ágeis
- A abordagem de XP para testes é um ponto particularmente forte desse método
  - Testes executáveis são desenvolvidos antes que o código seja escrito
- Não são uma bala de prata nem uma panacéa!
  - Podem ser úteis ou não, dependendo do contexto

18