


Programação Extrema

Marcos L. Chaim
Segundo Bimestre
2007
EACH-USP



A Solução

- A matéria-prima de XP são:
 - Quatro valores: comunicação, simplicidade, realimentação e coragem.
 - Os princípios;
 - As quatro atividades básicas -- codificação, teste, *listening* e criação de arquitetura.
- Porém, XP é feita mesmo por meio de práticas.

Práticas de XP

- Planejamento;
- Pequenas *releases*;
- Metáfora;
- Arquitetura Simples;
- Teste;
- *Refactoring*;
- Programação em pares;

Práticas de XP

- Propriedade coletiva;
- Integração contínua;
- Semana de 40-horas;
- Cliente residente;
- Padrões de codificação.

Planejamento

- Os conhecedores do negócio (clientes e usuários) devem decidir sobre:
 - Escopo;
 - Prioridade;
 - Composição de *releases*:
 - quanto deve ser feito para que o software passe a ser útil para o negócio.
 - Data de entrega.

Planejamento

- Programadores devem decidir sobre:
 - Estimativas;
 - Conseqüências;
 - Processo a ser utilizado:
 - definição do processo; ajuste à cultura, porém, este ajuste não deve ser prejudicial ao objetivo de produzir software;
 - Cronograma detalhado.

Pequenas *releases*

- Toda *release* deve ser a menor possível, contendo os requisitos de negócios mais importantes.
- *Release* deve fazer sentido como um todo.
- É melhor planejar uma *release* para ser entregue em um mês do que em um ano.

Metáfora

- Deve-se escolher uma metáfora que represente a idéia subjacente ao sistema.
- Arquitetura vai ser um reflexo da metáfora.
- Exemplo:
 - Sistema de controle de projetos realizados em unidades de negócios distribuídas.
 - Arquivo central contém projetos contidos em *folders*.

Arquitetura simples

- A arquitetura correta para o software em qualquer momento é aquela que:
 - Permite a execução de todos os casos de teste;
 - Não possui lógica duplicada;
 - Descreve tudo que é importante para os programadores;
 - Possui o menor conjunto de classes e métodos.

Teste

- Um requisito do sistema que não possui um teste automatizado *não* existe.
- Programadores devem escrever teste unitários de forma a ganhar confiança no seu código.
- Clientes devem escrever casos de teste funcionais para obterem confiança no software desenvolvido.
- Confiança no sistema aumenta com teste automatizado.

Refactoring

- Ao implementar uma funcionalidade, o programador deve se perguntar:
 - há uma maneira tornar o programa mais simples para incluir essa funcionalidade e manter os casos de teste funcionando?
- Se houver, então, ao tornar o programa mais simples, está-se realizando *refactoring*.

Programação em Pares

- Todo código é gerado com duas pessoas olhando a mesma máquina, com um teclado e um *mouse*.
- O programador que fica com o teclado e o mouse se preocupa com a melhor forma de implementação.
- O outro tem preocupações mais estratégicas: Essa idéia funciona? Estão faltando casos de teste? Há uma maneira de fazer mais simples?

Programação em Pares

- Atividade de programação é definida no início do dia com uma reunião matinal (*stand-up meeting*).
- Encontro diário, de pé, em geral em um café da manhã, no qual são discutidos:
 - as atividades de programação,
 - os problemas e soluções encontrados e
 - comunicações gerais para manter o foco do grupo.
- A reunião matinal não deve durar muito, por isso, é realizada de pé.

Propriedade coletiva

- Em XP não existe *propriedade individual* de código.
- Todos programadores têm responsabilidade sobre todo o sistema.
- Nem todo mundo conhece cada parte da mesma maneira, mas todo mundo conhece algo sobre tudo.

Integração Contínua

- O código deve ser integrado e testado depois de algumas horas.
- Uma máquina deve ficar dedicada para integração.
- Um par de programadores, depois de gerar código, deve:
 - carregar a *release* atual,
 - carregar suas modificações e
 - rodar os testes até todos (100%) passarem.

Semana de 40 horas

- As pessoas precisam de descanso para serem produtivas e criativas.
- Horas extras são um sintoma de problemas sérios no projeto.
- XP admite uma semana com horas extras;
- Duas já são uma indicação de problemas de estimativa e gerência do projeto.

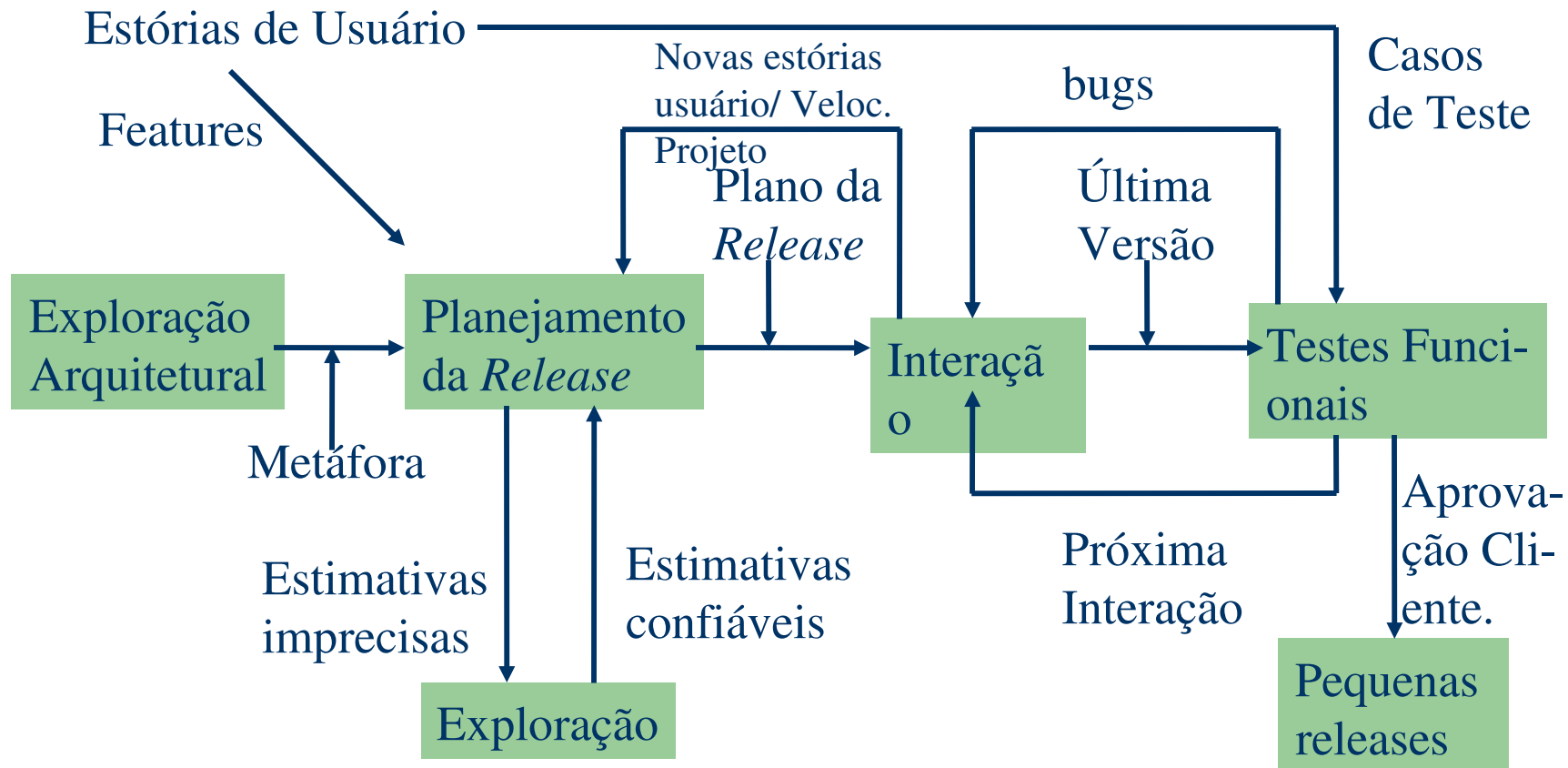
Cliente residente

- Um cliente deve estar disponível, no site, para responder dúvidas, disputas e ajustar prioridades menores.
- Este cliente deve ser um usuário real do sistema.
- Apesar de longe do seu site original, o cliente deve continuar a fazer o seu trabalho usual.

Padrões de codificação

- Com todos os programadores podendo *alterar* qualquer parte do código, padrões de codificação é essencial em XP.
- Os padrões devem:
 - exigir pouco esforço, caso contrário, não serão seguidos;
 - evitar duplicação de código;
 - enfatizar comunicação;
 - ser adotados voluntariamente.

© 2013 Pearson Education, Inc. or its affiliate(s). All rights reserved.



Estórias de Usuário

- Estórias de usuário são semelhantes aos casos de uso, mas não iguais.
- São escritos pelos clientes.
- Devem ser escritos em três frases na linguagem do cliente.
- As estórias de usuário serão utilizadas para guiar a elaboração de testes funcionais automatizados.

Estórias de Usuário

- Programadores estimam tempo de implementação de estórias de usuário.
 - Em geral, duas a três semanas para cada estória;
 - Se preciso mais, subdividir a estória do usuário.
- Estórias de usuário são implementadas por ordem de prioridade definida pelo cliente.
- Estórias de usuário não são detalhadas; mais detalhes programador obtém-se com o cliente.

Exemplo de estória de usuário

- O agrônomo Francisco Mineiro da Casa da Lavoura de Muzambinho, MG, que já é aluno da UniAgro, está interessado em se matricular nos cursos de extensão em manejo de milho e sorgo que constam do catálogo da UniAgro. Para a matrícula propriamente ocorrer, Chico Mineiro conecta-se ao sistema utilizando o seu login e senha. Depois de logado, ele seleciona os cursos que estão no catálogo sobre o manejo de milho e sorgo e submete o seu *horário* para o semestre. O sistema verifica se Chico Mineiro preenche todos os requisitos para a matrícula dos cursos (pré-requisitos, adimplência, existência de vagas, etc) e apresenta o resultado para o Chico.



Planejamento da *Release*

- A essência do planejamento da *release* é *estimar* cada história de usuário em termos de semanas *ideais de programação*.
- Uma semana ideal de programação é aquela em que a única atividade do programador é implementar a história de usuário.
- O cliente então decide quais as histórias de usuário são as mais importantes e devem ser implementadas primeiro.

Planejamento da *Release*

- A *velocidade do projeto* é dada pelo número de histórias de usuário que são implementadas em cada interação.
- A estimativa inicial da *velocidade do projeto* é um chute que tende a ser refinado na medida em vão ocorrendo as interações.



Regras e práticas de XP - Resumo

- Planejamento:
 - Estórias de usuário devem ser escritas.
 - Planejamento da *release* define cronograma.
 - *Release* freqüentes e pequenas devem ser geradas
 - Meça a velocidade do projeto.
 - O projeto é dividido em interações.
 - Cada interação começa com o seu planejamento.
 - Movimento a equipe.
 - Faça reunião matinal diária (*stand-up meeting*).
 - Adapte XP se necessário.

Regras e práticas de XP - Resumo

- Codificação:
 - O cliente está sempre disponível.
 - Código deve seguir padrões.
 - Codifique os testes automatizados antes.
 - Todo código é gerado em programação por pares.
 - Apenas um par integra o código por vez.
 - Realize integração freqüentemente.
 - Utilize propriedade coletiva do código.
 - Deixe a otimização para o final.
 - Evite horas extras.

Regras e práticas de XP - Resumo

- Projeto Arquitetural:
 - Simplicidade.
 - Escolha uma metáfora para o sistema.
 - Utilize cartões CRC nas sessões de projeto arquitetural.
 - Crie soluções exploratórias para avaliar risco.
 - Não adicione funcionalidade antes.
 - Faça *refactoring* sempre e onde for possível.

Regras e práticas de XP - Resumo

- Teste:
 - Todo código deve possuir testes de unidade.
 - Todo código deve passar 100% dos testes de unidade antes de ser liberado.
 - Quando um defeito é encontrado, testes devem ser gerados para garantir que ele não reapareça.
 - Testes funcionais são executados freqüentemente e seus resultados apresentados.

Exercícios

- Com relação aos modelos de ciclo de vida, qual deles se adapta mais às práticas de XP?
- Quais são as características de XP que lhe dão *agilidade*?
- O princípio no qual XP se baseia -- custo de mudança é constante -- aplica-se em qualquer projeto? Se não, quais são os projetos nos quais XP *funciona* e quais são aqueles que em XP *não funciona*?