

# XP - Segunda Edição

## Revisão depois de 5 anos de *feedback*



**Curso de Verão 2007 – IME/USP**

[www.agilcoop.org.br](http://www.agilcoop.org.br)

Alexandre Freire e Mariana Bravo

# Por que mudou?

- 5 anos de retorno e experiência na comunidade
- Críticas ao 1º livro:
  - “Você quer nos forçar a programar da sua maneira!”
  - XP é muito radical
  - XP despreza o *design*
  - XP despreza a documentação
  - XP despreza conhecimentos especializados
  - XP não escala e não funciona para equipes dispersas geograficamente

# Sobre a 2ª edição

- Reconhece a responsabilidade de cada indivíduo pelas suas próprias decisões
- Mantém o conteúdo, mas muda a forma, re-fraseando a mesma mensagem de maneira mais positiva e inclusiva
- 1ª Edição focava muito mais em “Como” XP funciona
- 2ª Edição foca muito mais no “Por quê” XP funciona
- Apresenta a filosofia por trás da criação de XP e paralelos com outras áreas

# Definindo XP

- 1ª edição - 1999
  - *“XP é uma metodologia leve para times médios ou pequenos desenvolvendo software em face a requisitos vagos e que mudam rapidamente”*
- 2ª edição - 2004
  - *“XP é sobre mudança social”*

# O que faz parte deste método?

- Uma filosofia de desenvolvimento de software baseada em alguns **valores**
- Um conjunto de **práticas** comprovadamente úteis que expressam esses valores, se complementam e se amplificam
- Um conjunto de **princípios** complementares que traduzem os valores em práticas
- Uma **comunidade** que compartilha tais valores e muitas das mesmas práticas

# Nova definição

- XP é leve
- XP é focado no desenvolvimento de software
- XP funciona em times de qualquer tamanho
- XP se adapta à requisitos vagos e que mudam rapidamente

# Valores, princípios e práticas

- Valores são abstratos, práticas são concretas
- Valores trazem propósito às práticas
- Práticas evidenciam e facilitam a “prestação de contas” dos valores
- Princípios são técnicas intelectuais para traduzir valores em práticas

# Adaptando tudo isso

- Outros valores da sua organização podem ser incorporados a XP
- Práticas podem ser adaptadas
- Princípios ajudam a entender as práticas e até improvisar práticas complementares
- Não existe XP puro, o importante é o comprometimento do time com os valores

# Valores – o que mudou?

- Comunicação, simplicidade, *feedback*, coragem
- + Respeito
  - valor complementar aos 4 anteriores
  - respeito entre a equipe
  - respeito ao projeto

# Princípios – o que mudou?

- 1ª edição tem princípios básicos e alguns menos importantes
- 2ª edição explica melhor o propósito dos princípios, que ganham mais importância
- Alguns foram mantidos, como *trabalho de qualidade, mudanças incrementais e aceitação de responsabilidade*
- Outros princípios eram implícitos e agora ganharam mais destaque

# Princípios

- Humanidade
  - Balancear as necessidades pessoais com as necessidades do time
- Economia
  - Evite o risco do “Sucesso Técnico”. Tenha certeza que o sistema cria valor para o negócio
- Benefício Mútuo
  - Todas as atividades devem trazer benefício a todos os envolvidos

# Princípios

- Auto-Semelhança
  - Tente aplicar a estrutura de uma solução em outros contextos e escalas
- Melhoria
  - Valorize atividades que começam agora e se refinam ao longo do tempo
- Diversidade
  - Times devem ser formados por uma variedade de habilidades, atitudes e perspectivas

# Princípios

- Reflexão
  - Reflexão vem após a ação. O aprendizado é o resultado da reflexão sobre a ação
- Fluxo
  - Desenvolva todas atividades em um fluxo contínuo, ao invés de fases discretas
- Oportunidade
  - Enxergue os problemas como uma oportunidade para mudança, aprendizado e melhoria

# Princípios

- Redundância
  - Resolva os problemas difíceis de várias formas diferentes
- Falha
  - Quando você não sabe o que fazer, arriscar o fracasso pode ser o caminho mais curto para o sucesso
- Qualidade
  - Sacrificar a qualidade nunca é um meio efetivo de controle

# Princípios

- Passos Pequenos
  - A execução em passos pequenos diminui o risco de uma grande mudança
- Aceitação da Responsabilidade
  - Responsabilidade não pode ser imposta, deve ser aceita

# Práticas – o que mudou?

- Abordagem mais leve para adoção das práticas
  - as práticas não devem ser impostas, devem ser escolhidas
  - não precisam ser adotadas todas de uma vez
- Práticas primárias
  - independentes e seguras de adotar
  - proporcionam melhorias imediatas
- Práticas corolárias
  - difíceis de adotar sem domínio das primárias

# Práticas Primárias

- Sentar Junto
  - Desenvolva num ambiente grande o suficiente para o time todo ficar junto
- Time Completo
  - Monte um time multi-disciplinar, com todas as habilidades necessárias para o sucesso do projeto
- Área de Trabalho Informativa
  - Um observador interessado deve ser capaz de ter uma idéia do andamento do projeto andando pela área de trabalho

# Práticas Primárias

- Trabalho Energizado
  - Trabalhe apenas enquanto puder ser produtivo e o número de horas que puder agüentar
- Programação Pareada
- Histórias
  - Unidades de funcionalidade visíveis ao cliente
  - Estimativa é feita o mais cedo possível para facilitar a interação entre o cliente e a equipe

# Práticas Primárias

- Ciclo Semanal
  - Representa uma iteração
  - Planeje o trabalho uma semana de cada vez
  - Reunião no início da semana para discutir o progresso, escolher histórias e dividi-las em tarefas
- Ciclo de Estação
  - Identificar gargalos e iniciar reparos
  - Planejamento do tema da estação
  - Foco no todo: onde o projeto se encaixa na organização

# Práticas Primárias

- Folga
  - Inclua no plano algumas tarefas menores que podem ser removidas caso ocorra um atraso
  - Não se comprometa além do que é possível ser realizado
- *Build* de 10 minutos
  - Faça o *build* AUTOMÁTICO do sistema INTEIRO e rode TODOS os testes em até 10 minutos
- Desenvolvimento Orientado por Testes

# Práticas Primárias

- Integração Contínua
  - Integração pode ser síncrona ou assíncrona (no *build*)
- *Design* Incremental
  - Invista no *design* do sistema todos os dias
  - O conselho não é minimizar o investimento em *design* no curto prazo, mas manter o investimento proporcional às necessidades do sistema
  - *Design* feito mais próximo de quando é usado é mais eficiente

# Como começar?

- Práticas dependem da situação e do contexto, são um guia para onde você pode chegar
- Adote XP em *passos pequenos* (mude uma coisa de cada vez)
- Crie consciência da necessidade de uma prática usando métricas
- Avalie a experiência de adoção

# Práticas Corolárias

- **Envolvimento Real com o Cliente**
  - Faça com que as pessoas cujas vidas e negócios serão afetados pelo sistema façam parte do time
- **Implantação Incremental**
  - Ao substituir um sistema legado, troque partes da funcionalidade gradualmente
- **Continuidade do Time**
  - Mantenha times eficientes trabalhando juntos

# Práticas Corolárias

- Redução do Time
  - Mantenha a carga de trabalho constante e distribua as tarefas de modo a deixar alguém ocioso
  - Com o tempo, essa pessoa pode ser liberada para formar novos times
- Análise de Causa Inicial
  - Sempre que encontrar um defeito, elimine o defeito e sua causa
  - “Os 5 Porquês”
- Código Compartilhado

# Práticas Corolárias

- Código e Testes
  - Os únicos artefatos permanentes
- Repositório Único de Código
  - Desenvolva num repositório único. *Branches* podem existir, mas devem ser incorporados logo
- Implantação Diária
  - Coloque software novo em produção toda noite

# Práticas Corolárias

- Contrato de Escopo Variável
  - Contratos devem fixar tempo, custo e qualidade, mas a negociação de escopo deve ser aberta
- Pague-Pelo-Uso
  - Cobrar por cada vez que o sistema é usado
  - O dinheiro é o *feedback* máximo

# Resumo – Mudança nas práticas



# O time de XP – o que mudou?

- Abordagem mais inclusiva, destaca outros papéis típicos do desenvolvimento de software, além do programador (foco da 1ª versão)
- Diversidade de pessoas trás benefício ao *Time Completo*
- Cada parte do grupo deve compreender seu papel no todo, estamos todos no mesmo barco!
- Interação entre pessoas deve seguir princípios de *fluxo e benefício mútuo*

# Papéis

- Testadores
  - Trabalham com o cliente para escrever testes de aceitação
  - Treinam programadores em técnicas de teste
- Projetistas de Interface
  - Ajudam a escolher uma metáfora pro sistema
  - Trabalham com o cliente para escrever histórias
  - Avaliam o uso do sistema entregue, identificando novas histórias e melhorias na interface com os usuários

# Papéis

- Arquitectos
  - Procuram e executam refatorações de larga escala
  - Escrevem testes para mostrar falhas na arquitetura, direcionando a evolução do *design*
  - Particionam o sistema (Conquistar e Dividir)
- Gerentes de Projeto
  - Facilitam a comunicação dentro do time, com clientes e o resto da organização
  - Marcam progresso (ajudando a coletar e divulgar métricas)
  - Mantém o plano em dia com a realidade

# Papéis

- Gerentes de Produto
  - Ajudam no planejamento semanal e da estação, escolhendo e priorizando histórias e temas
  - Representa as necessidades do cliente
- Executivos
  - Patrocinam e acompanham o time XP, incentivando e facilitando melhorias
  - Avaliam e articulam objetivos do time com objetivos de larga escala da organização

# Papéis

- Escritores Técnicos
  - Proporcionam *feedback* sobre as funcionalidades enquanto estas são implementadas
  - Estreitam o relacionamento com os usuários, escrevendo manuais, tutoriais, etc...
  - Ajudam a escolher histórias ligadas a *feedback* dos usuários
- Usuários
  - Ajudam a escrever e escolher histórias
  - Ajudam nas decisões de domínio durante o desenvolvimento, representando a comunidade

# Papéis

- Programadores
  - Estimam histórias e tarefas, dividem histórias em tarefas, escrevem testes e código pareando, automatizam processos tediosos e melhoram o *design* gradualmente
- Recursos Humanos
  - Ajudam nas avaliações, que podem ser individuais ou focadas no desempenho do time
  - Ajudam nas contratações, dando ênfase a habilidades sociais e de trabalho em grupo
  - Entrevistas: um dia trabalhando com o time

# Resumo – Mudanças no Time

- Papéis não são fixos, uma mesma pessoa pode exercer diversos papéis
- Cada um deve contribuir tudo que pode para a equipe
- Atividades do *programador* continuam iguais
- Atividades do *coach*, *tracker* e *cliente* foram distribuídas em novos papéis

# Planejamento – o que mudou?

- Estimar usando tempo real de desenvolvimento e não sistema de pontos
- Definição mais detalhada do ciclo diário, semanal e de estação

# Testes – o que mudou?

- Em vez de testar *antes*, testar *cedo*
- Dupla verificação: princípio da redundância
  - Testes e o código lidam com uma mesma funcionalidade
  - Não necessariamente a mesma dupla faz as duas coisas
- Aumento de Custo de Defeito (*DCI*)
  - Quanto mais cedo encontrar o defeito, mais barato é a correção
  - Testes devem ser feitos próximos à implementação

# *Design* – o que mudou?

- 1ª versão foca em metáfora e refatoração
- Agora a prática de *design incremental* engloba essas atividades no dia-a-dia
- Conquistar e dividir
  - Começar com uma solução simples, que ao crescer pode ser dividida nas suas linhas de fratura naturais
- *Design* feito em vista à experiência é mais eficaz

# Escalabilidade

- 1ª versão -> times pequenos
- Valores e Princípios valem em qualquer escala
- Práticas podem ser adaptadas

## Medidas:

- Número de pessoas
  - Transforme o problema em problemas menores
  - Aplique soluções simples
  - Aplique soluções complexas somente se sobrar algum problema

# Escalabilidade

- Investimento
  - O problema é como contabilizar projetos XP
  - Achar um aliado na área de contabilidade
- Tamanho da Organização
  - O objetivo não é esconder o trabalho do time nem forçar uma mudança no resto da empresa
  - Não mude o modo de comunicação com o resto da organização

# Escalabilidade

- Tempo
  - Projetos longos funcionam bem com XP
  - Testes e *Design* Incremental são o “histórico” do projeto
- Complexidade do Problema
  - XP funciona bem em projetos que requerem cooperação entre uma equipe de especialistas
  - Incentivar a troca de conhecimento nas especialidades

# Escalabilidade

- Complexidade da Solução
  - Desafio: parar de tornar o problema mais complicado
  - Estratégia de XP: elimine a complexidade gradualmente, sem parar de entregar
- Conseqüência das Falhas
  - XP não é suficiente para projetos onde a segurança e a confiabilidade são críticos
  - O princípio de fluxo sugere que processos de auditoria sejam feitos mais cedo e com freqüência

# Equipes distribuídas

- Valores se aplicam para equipes distribuídas geograficamente
- É necessário nutrir comunicação e *feedback* devido à separação
- Adaptar as práticas, *base de código única* é a conexão entre as equipes
- Não abandonar práticas difíceis de manter à distância
  - eg.: programação pareada usando VNC

# Como aplicar XP na sua organização

- Aprendizado contínuo
- XP é um novo contexto para resolver os problemas já existentes
- Tenha ciência do perigo de reverter aos velhos padrões
- Ache um executivo para patrocinar a adoção, preste contas a ele
- Comece mudando você mesmo, depois ofereça os frutos dessa mudança aos outros

# Escolhendo um *Coach*

- Dá pra adotar XP sem um *Coach*, mas não recomendamos
- Uma perspectiva independente e experiência são importantes no aprendizado
- Deve ser uma pessoa alinhada com os valores existentes da organização, mas firme nos valores de XP
- Habilidades técnicas são importantes

# Quando não usar XP

- Quando os valores reais da sua organização não se alinham com os valores de XP

# Comunidade

- O suporte de uma comunidade é muito importante
- Reconhecimento e critica dos seus pares
- Oferece uma perspectiva de pessoas que compartilham os mesmo valores, mas estão em outras organizações
- Melhoria através de grupos de estudo
- Entre na comunidade da AgilCoop!

# Conclusão

- Depois de 5 anos de experiência, XP amadureceu
- Adotar XP deixou de ser tão extremo
- Novas práticas e princípios facilitam o aprendizado
- A comunidade cresceu, venha fazer parte
- Adote os valores, e comece você mesmo com uma prática que faça sentido na sua organização