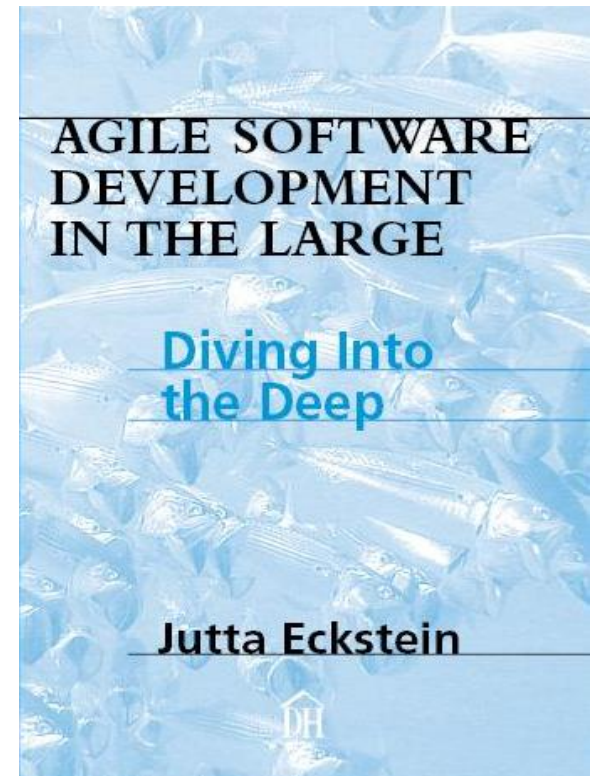


# Desenvolvimento Ágil de Software em Larga Escala

Jutta Eckstein  
Encontro Ágil 2009



# Agilidade é Quente

Gerenciamento Ágil de Projetos

Testes Ágeis

Arquitetura Ágeis

Offshore Ágil

Investimento Ágil

PLM Ágil

Desenvolvimento Web Ágil

Desenv. Ágil de Jogos

SOA Ágil

AJAX Ágil

Soluções Web Ágeis

MDA / MDD Ágil

# Agilidade é Coisa Velha

- E.g. Scrum e XP foram publicados em 1995

XP

SCRUM

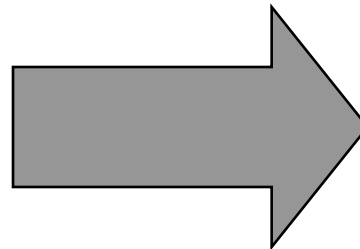
DSDM

Metodologias Crystal

FDD

Dev. Adaptativo de Software

...



**Manifesto Ágil**

**Feb. 2001**

**[agilemanifesto.org](http://agilemanifesto.org)**

# „Grande“

- **Grande em...**
  - Escopo
  - Tempo
  - Pessoas
  - Dinheiro
  - Riscos
- **Nós nos concentramos „Grandes Times“**
- **Grande é relativo**
  - 1, 2, 10, 100, 2000 Pessoas

# „Grande“ e Métodos Ágeis

## ■ XP

- Tamanho típico de times < 12

## ■ Scrum

- Scrum de Scrums

## ■ Crystal

- Cores diferentes para tamanhos diferentes:
  - Clear: para equipes < 10
  - Orange: para equipes < 40
  - Red, blue, ... (ainda não definido)

## ■ FDD

- Equipes são montadas para projetar uma funcionalidade

# Sem Método pré-fabricado

- Mas: Desenvolvimento Ágil é definido pelo valor do sistema:

Indivíduos e interações

sobre processos e ferramentas

Software funcionando

sobre documentação compreensiva

Colaboração com o cliente

sobre negociação de contratos

Responder às mudanças

sobre seguir um plano

Isto é, enquanto existe valor nos itens à direita, nós valorizamos mais os itens à esquerda

# Princípios Ágeis

- **O valor do sistema é baseado nos seguintes princípios:**
  - Entrega contínua de software de valor
  - Acolha requisitos mutantes
  - Entregue software funcionando frequentemente
  - Pessoas de negócio e desenvolvedores trabalhando junto
  - Confie em indivíduos motivados
  - Conversa Frente-a-frente
  - Software funcionando é a principal medida de progresso
  - Promova desenvolvimento sustentável
  - Excelência técnica e bom desenho
  - Simplicidade é essencial
  - Time auto-organizadores
  - Reflexão e ajuste na equipe

# Entregue Software Funcionando Frequentemente

- **Balanceie redução de riscos com cumprimento de funcionalidade**
  - Iterações de duas semanas já foram provadas
  - Uma versão aproximadamente a cada 3 iterações
  - Mesmo ritmo para todos os times
  
- **Desenvolvimento em tempo fixo:**
  - Conhecimento sobre velocidade
  - Melhoria do planejamento do projeto como um todo

# Entrega de Software de Valor

- **Foco em completar funcionalidades de valor**
  - Frequentemente anormal para grandes times
  - Realização significa integração, teste e documentação
    - Entrega
- **Estruture os times ao redor das funcionalidades**
  - Para garantir o valor de negócio e a vantagem para o cliente
  - Times para funcionalidades
    - Abranja todos os papéis necessários e todos os conhecimentos necessários

# Envolvimento do Cliente

É raro ter um único cliente bem definido, são mais típicos:

- **Bases de clientes grandes e invisíveis**

- Típicos para software comum

- **Comunidade de clientes**

- Frequentemente heterogênea

## Portanto: **Cliente dublê**

- Product Owner
- Um product owner pode não ser suficiente
  - Equipes de product owners com um product owner principal
  - Um product owner pode direcionar 2-3 times de funcionalidade

# Agilidade significa Acolher Mudanças

- **Sucesso é definido pelo cliente, por mais ninguém.**
- **Mudanças nas funcionalidades e nas prioridades**
  - Sinais claros que o cliente entende melhor o sistema.
  - Na maioria dos casos pode-se resolver isso com iterações
- **Mas:**
  - Uma mudança significa:
    - Atraso
    - Eliminação de outra funcionalidade

# Promova Desenvolvimento Sustentável

- **Determine a velocidade de cada time**
  - Planejamento Realístico
  - Note: velocidades não são comparáveis
- **Revisões e inspeções de código**
  - Externas
  - Internas
    - Time de revisores
    - Continuamente: programação em pares

# Confie em Indivíduos Motivados

**Confiança é baseada em:**

- Comunicação
- Transparência
- Honestidade
- Contato

**Confiança é importante para todos:**

- Desenvolvedores
- Clientes
- Gerentes

# Desenvolvimento Ágil dentro da Organização

- **Desenvolvimento Ágil é um detector de problemas**
  - Consciência que más notícias também são boas notícias
  - Apoio para ações apropriadas
- **Integração transfuncional de departamentos**
  - Projetos como clientes
- **Relação próxima com o cliente**
  - Garante feedback e portanto o sucesso do projeto

# Conversa Frente-a-frente

- **Comunicação Frente-a-frente sempre é preferencial**
- **Sincronização diária é essencial**
  - ter um entendimento comum
  - lidar com papéis
  - lidar com problemas
  - obter feedback
- **Pense sobre:**
  - Scrum de Scrums diários
  - Troca de pessoas
  - Facilitador de comunicação

# Simplicidade é Essencial

## ■ Integridade Conceitual

- „Simplicidade vem de integridade conceitual“ (Parnas)
- Uma arquitetura de sistema é o principal passo em direção à integridade conceitual

## ■ Liderança na arquitetura guia

- Comunica a visão
  - Ideias principais
  - Atribuição de responsabilidade técnica
  - Decisões técnicas
- Um funcionário com coragem e experiência

# Atenção Contínua na Excelência Técnica

- **Um bom (e simples) design nunca é fácil**
  - E: Faça-o certo logo da primeira vez nunca funciona
  - A melhor arquitetura evolui
- **Testes garantem o funcionamento das funcionalidades existentes**
  - Provêm a rede de segurança para refatorar
  - Sincronizados automaticamente e sincronamente com o desenvolvimento
- **Refatoração acontece continuamente**
  - Responsabilidade compartilhada
  - Toda grande refatoração pode ser realizada através de várias pequenas refatorações

# Time de Serviços Técnicos

- **Não tente finalizar a arquitetura antes de crescer o time**
  - Permita que a arquitetura evolua
- **Times de funcionalidades fornecem um product owner**
  - Formule e priorize os requisitos
  - Direcionam as iterações do time de serviços técnicos
- **Arquitetura é um serviço**

# Software Funcionando requer Integração

- **Mudanças serão integradas tão frequentemente quanto possível**
  - Tornam o progresso visível e mensurável
  - Conflitos são mais fáceis de resolver
- **Cada integração resulta num sistema funcionando**
  - Provê feedback imediato
- **Integração precisa de apoio**
  - Planeje pelo menos 10% do esforço de desenvolvimento

# Reflexão e Ajuste Frequente do Time

## ■ Revisão da Iteração em times de funcionalidade

- Apresenta entregas
- Aceitação / Rejeição

## ■ Retrospectiva

- Reflita sobre:
  - Coisas que funcionaram bem,
  - que precisam ser melhoradas,
  - que ainda são difíceis
- Defina ações para os 3 primeiros
- Retrospectiva por estágios

# Times Auto-Organizadores

- **Pessoas e times são diferentes**
- **Deixe o time decidir**
  - Cada time define seu próprio processo
  - Retrospectivas ajudam a moldar o processo
- **Não especifique demais nem mande demais**
  - Use um ponto inicial e ajuste a partir dele
    - I.e. a experiência dos membros do projeto é um bom começo
- **Lembre-se:**
  - "Indivíduos e interações sobre processos e ferramentas"

# Funcionários para Grandes Equipes

## ■ Scrum Master / Coach

- Visão do processo

## ■ Product Owner

- Visão do negócio

## ■ Especialmente para projetos grandes:

### - Gerente de Projeto

- Visão Organizacional

### - Arquiteto

- Visão Técnica

# Resumo

- Times de funcionalidade e product owner(s) garantem o valor de negócio
- Iterações curtas garantem feedback frequente para reduzir os riscos
- Times técnicos como provedores de serviços
- Comunicação é ainda mais importante

# 150 Anos Atrás e Ainda Atualizado

„Não é a espécie mais forte que sobrevive, nem a mais inteligente, mas aquela que mais responde às mudanças.“

[Charles Darwin, *A Origem das Espécies*, 1859]

# Muito Obrigado!

Um novo livro chegando sobre:

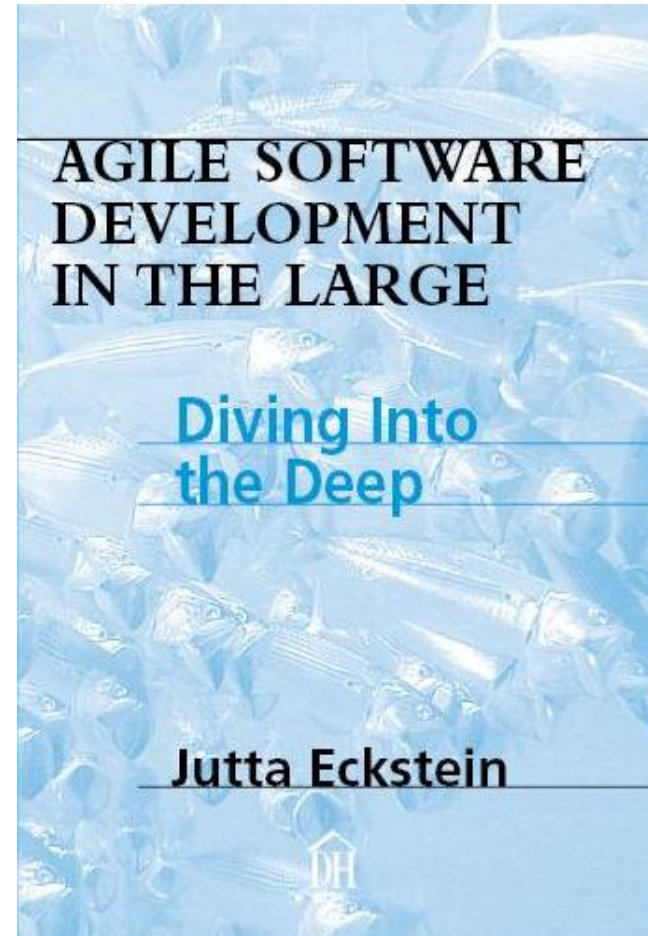
**Agile Software with  
Distributed Teams  
(Métodos Ágeis para  
Times Distribuídos)**

Contatos:

**Jutta Eckstein**

**je@it-communication.com**

**www.it-communication.com**



Ilustrações por:

josuttis | eckstein  
www.grelloelb.de