

# Introdução a Métodos Ágeis com ênfase em XP



Alfredo Goldman  
Professor do IME - USP

# Pequeno histórico

- Docente do IME - USP desde 1993
- Interesse inicial em métodos ágeis: 2001
  - Ensino: Laboratório de Programação Extrema
  - Vários cursos pelo Brasil (de SC -> PA)
- Orientações e publicações
  - Chair do WBMA 2010
  - Comitê de programa da XP desde 2010
- Trabalhos práticos em desenvolvimento de software com métodos ágeis

# Roteiro

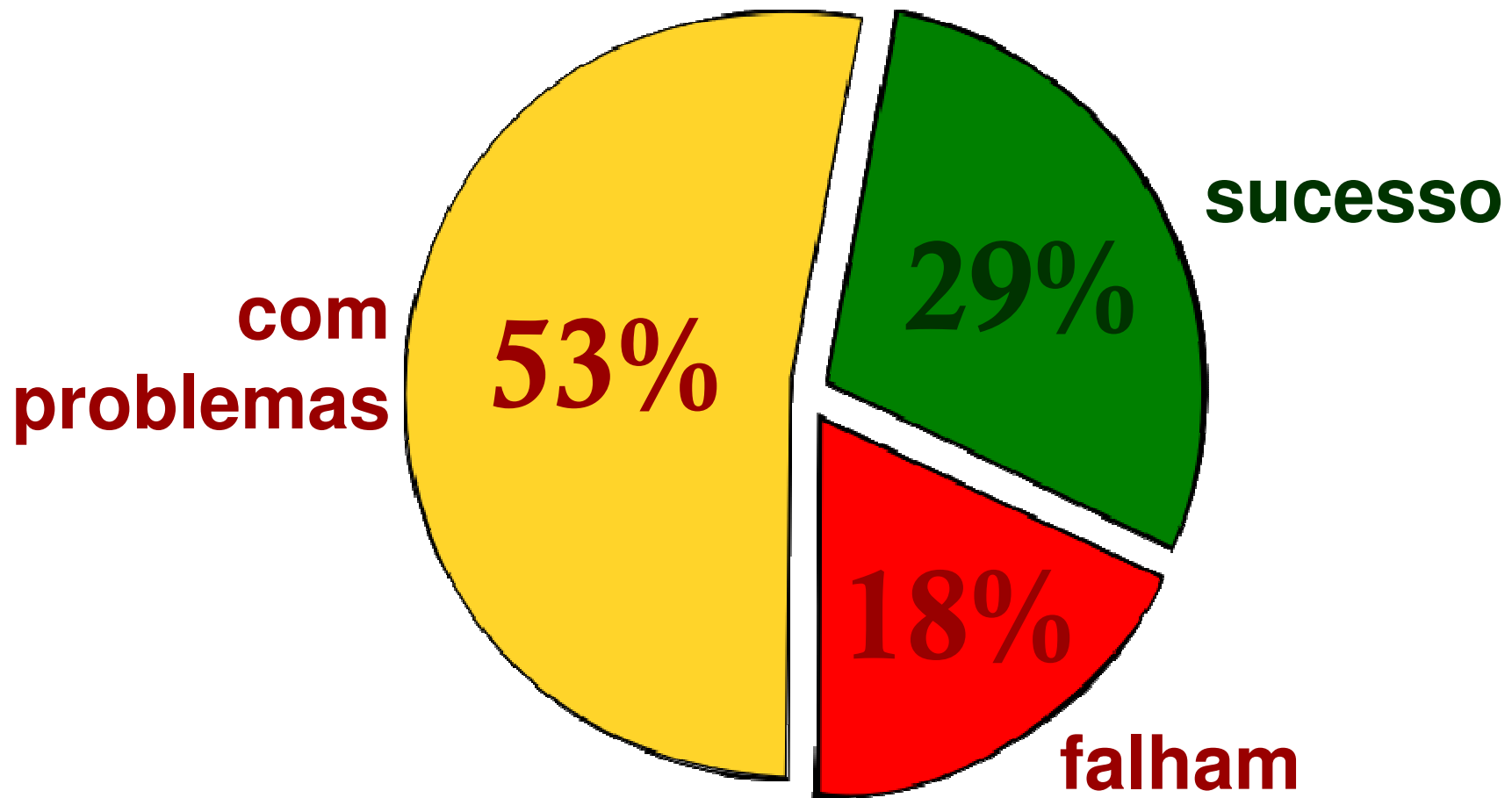
- Introdução a métodos ágeis
- Um pouco de Programação eXtrema
- Panorama no IME - USP

# Linhas gerais sobre MAs

- Motivação para o surgimento de métodos ágeis
  - Isto é: problemas 😊
- Princípios comuns a métodos ágeis
  - Manifesto ágil
- Um gostinho de Programação eXtrema
  - Mas, existem outros: Crystal, FDD, Lean, Scrum, etc.

# CHAOS report

- Resultado dos projetos (2004):



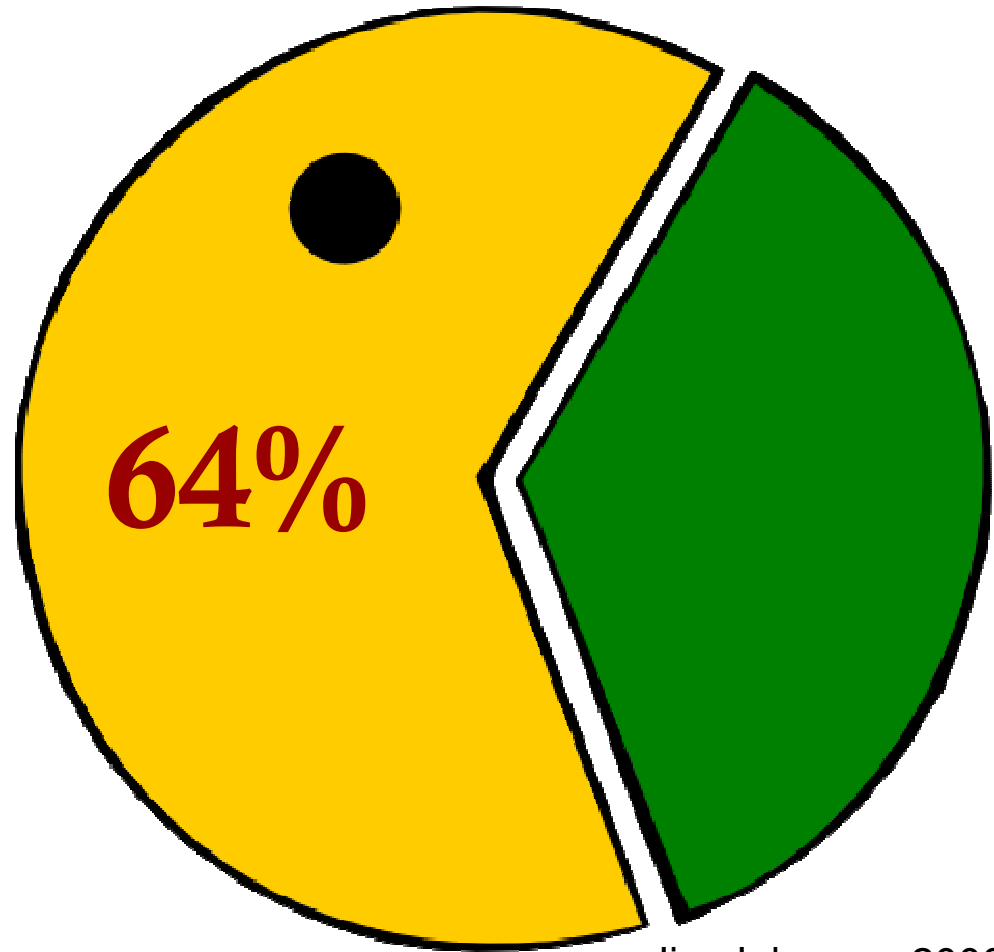
# Chaos Report: evolução ?

	1994	1996	1998	2000	2002	2004	2006	2009
Successful	16%	27%	26%	28%	34%	29%	35%	32%
Challenged	53%	33%	46%	49%	51%	53%	46%	44%
Failed	31%	40%	28%	23%	15%	18%	19%	24%

<http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.html>

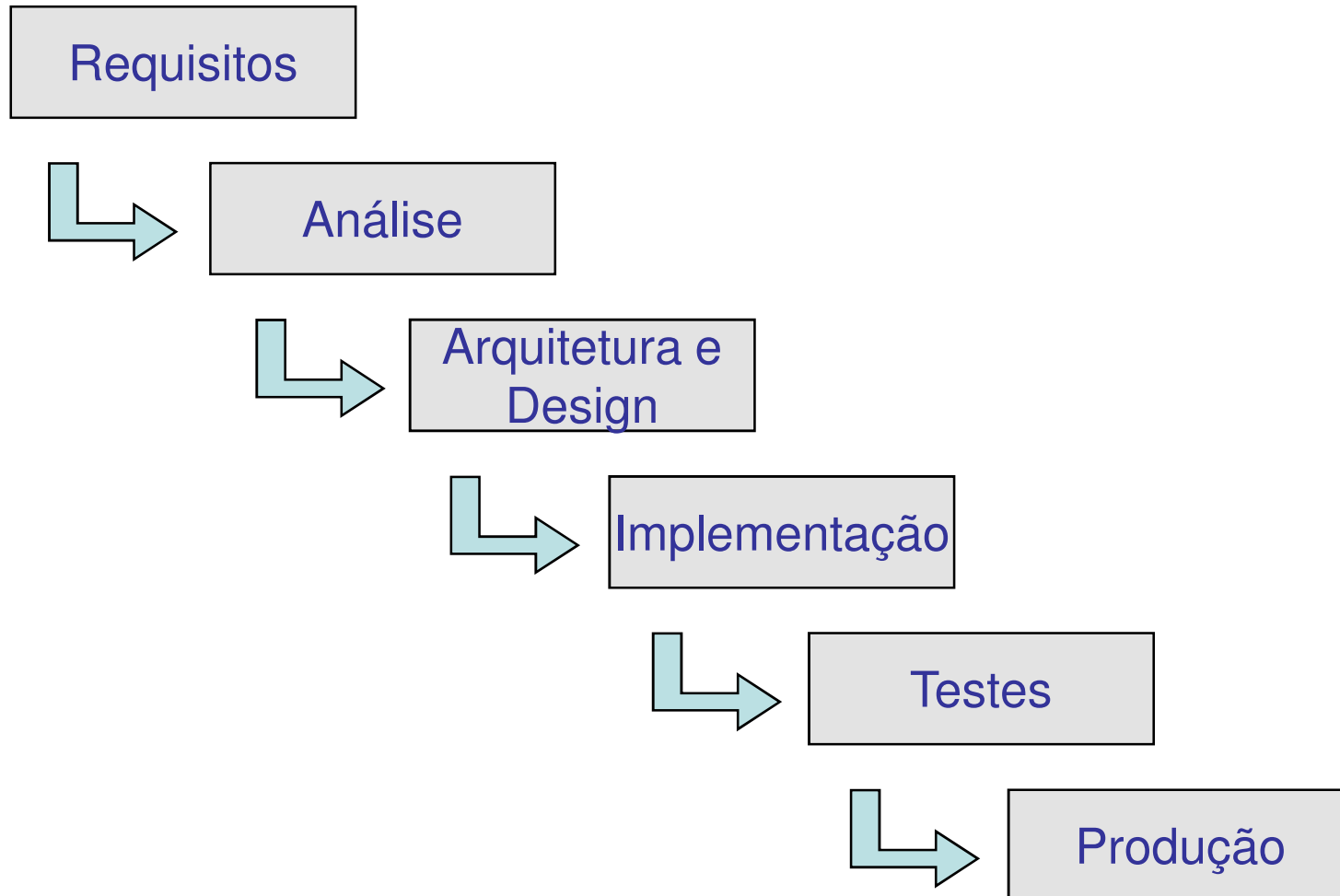
# Qual software?

Funcionalidades  
**nunca** ou  
**raramente**  
utilizadas



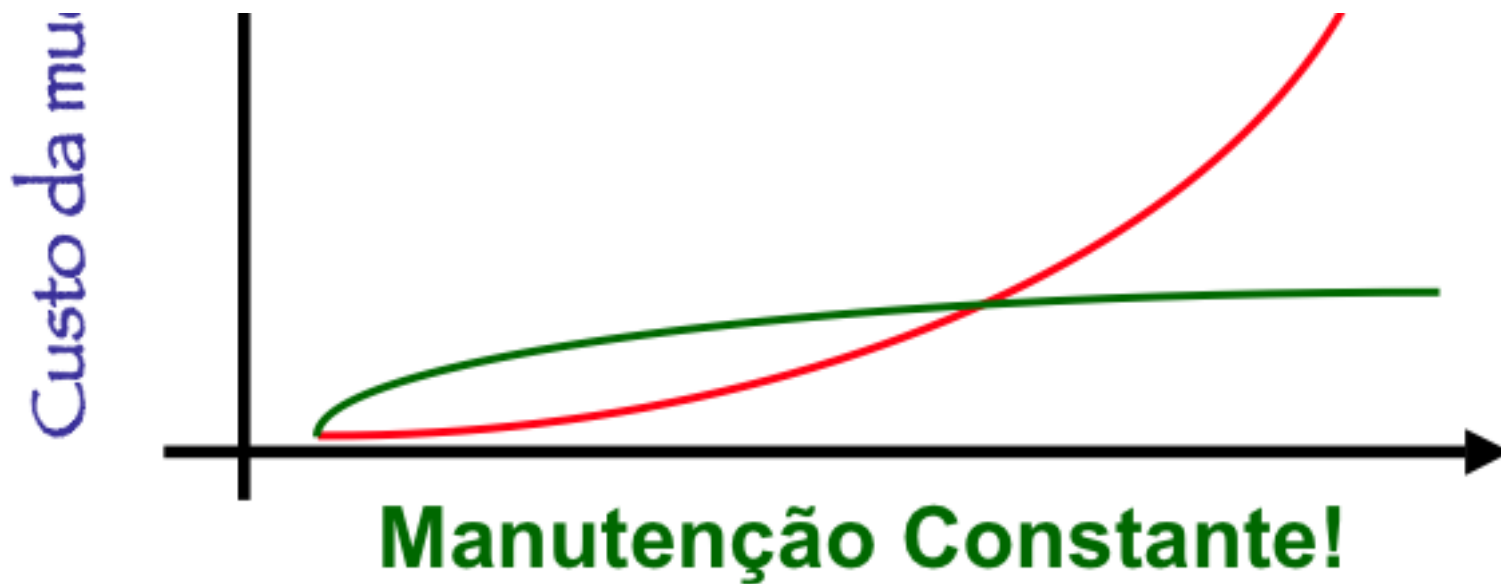
Jim Johnson, 2000

# Modelo tradicional





# Analogia incorreta 1



# Analogia incorreta 2



**Programar é fazer Design!**

# O que é desenvolvimento de software (A. Cockburn)?

## What is / isn't software development?

<b>Model Building</b>	<b>(Jacobson)</b>
<b>Engineering</b>	<b>(Meyer)</b>
<b>Discipline</b>	<b>(Humphreys)</b>
<b>Poetry</b>	<b>(Cockburn)</b>
<b>Math</b>	<b>(Hoare)</b>
<b>Craft</b>	<b>(Knuth)</b>
<b>Art</b>	<b>(Gabriel)</b>

**If you know what it is,  
you can apply known solutions.**

# Jacobson, agosto/2007

BOOK REVIEWS

PRODUCT REVIEWS

EARLIER ISSUES

SEARCH

GO!



[Subscribe to  
JOT's newsletter](#)

[O-O NEWS &  
EVENTS](#)

[Previous column](#) [next article](#)

## Enough of Processes - Lets do Practices

REFEREED  
COLUMN



PDF Version

**Ivar Jacobson, Pan Wei Ng and Ian Spence** Ivar Jacobson  
Consulting

### Abstract

All modern software development processes try to help project teams conduct their work. While there are some important differences between them, the commonalities are far greater - and understandably, since the end goal of them all is to produce working software quickly and effectively. Thus, it doesn't matter which process you adopt as long as it is adaptable, extensible, and capable of absorbing good ideas, even if they arise from other processes.

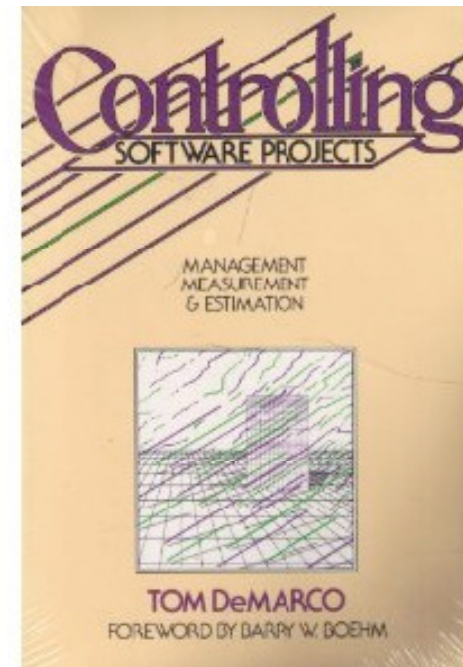
To achieve this kind of flexibility things need to change. The focus needs to shift from the definition of complete processes to the capture of reusable practices. Teams should be able to mix-and-match practices and ideas from many different sources to create effective ways of working, ones that suit them and address their risks.

# DeMarco, IEEE Software 2009

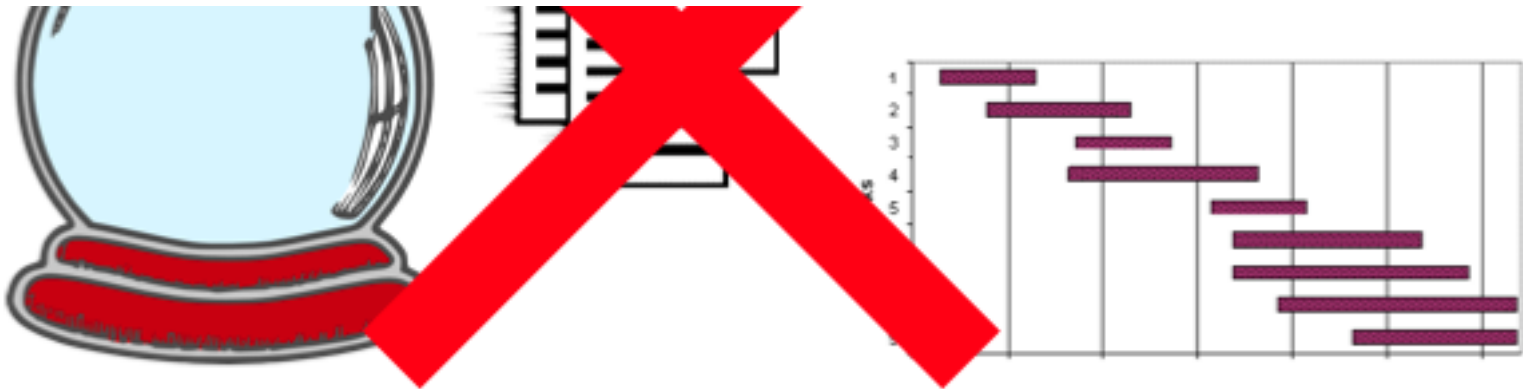
## Software Engineering: An Idea Whose Time Has Come and Gone?

**Tom DeMarco**

Rather, I'm advocating a management approach, one that might well steer the team toward agile methods, at least toward the incremental aspects of the agile school.



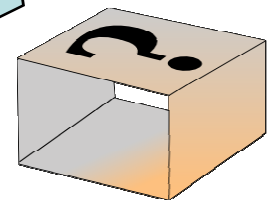
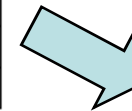
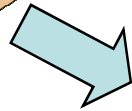
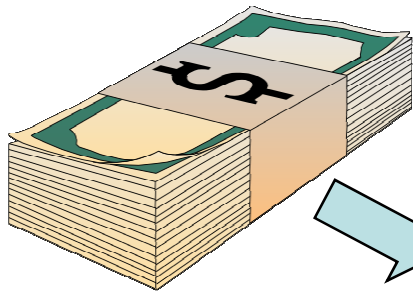
# O que é valor?



**Valor = software funcionando**

# E como fazer o software certo?

- Investindo na bolsa:



# Primeiro princípio

**Software funcionando** é  
mais importante que documentação  
abrangente

Documentação é uma funcionalidade



# Comunicação

” ” ” ” ” ” ” ”

- “Documento sem fio”

A



?

# Segundo princípio

**Indivíduos e interações** são  
mais importantes que processos e  
ferramentas

Ferramentas como controle de versões  
continuam sendo usadas

# Fazer certo o software



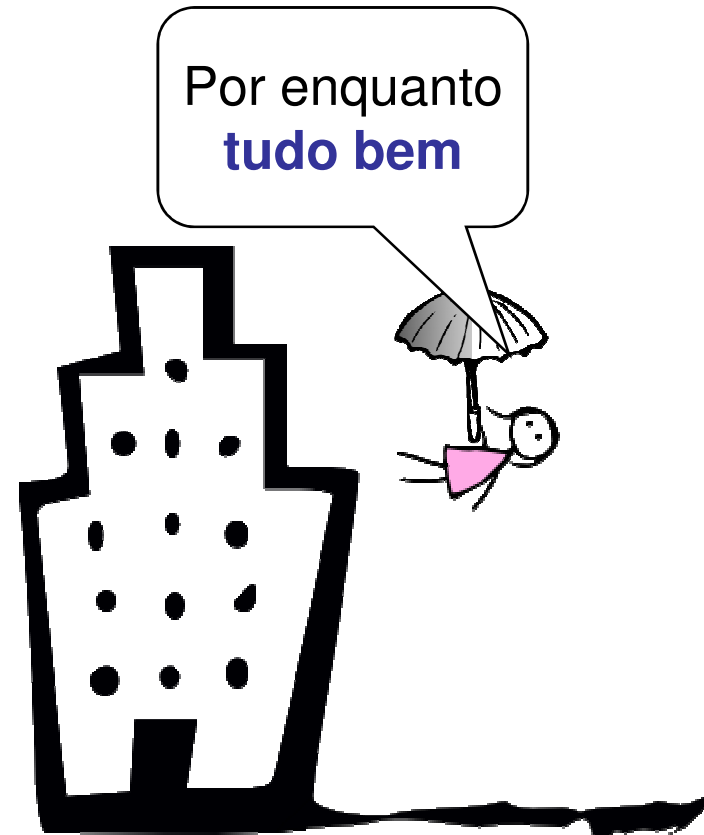
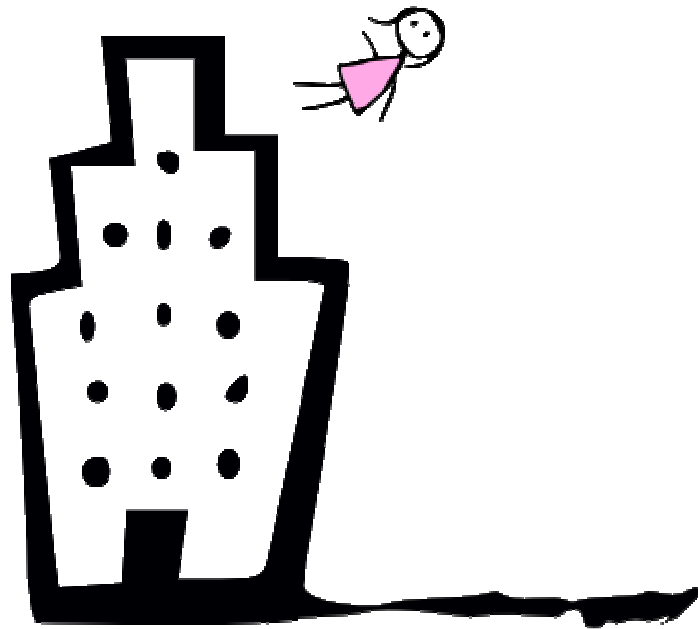
*Carne assada e vagem com bacon, uma delícia. Seguindo a receita vai ficar muito gostoso!*

Sem dúvida vai ser um **sucesso!!**

# Fazer o software certo



# Feedback



# Estimar é difícil

- Como um projeto atrasa 2 anos?
- 80% pronto



**Estimativas não são compromissos!**

# Terceiro princípio

**Colaboração com o cliente** é mais importante que negociação de contratos

Solução: contrato de benefício mútuo

# Estimando para planejar

- Perguntas:
  - Faça sua lista de compras de 2013
  - Faça sua lista de compras dessa semana
  - Compre os ingredientes para o seu jantar
- Somos ruins para planejar a longo prazo!





## Quarto princípio

**Adaptação a mudanças** é  
mais importante que seguir um plano

Vantagens competitivas

# Agora tudo junto

- Manifesto ágil:
  - **Indivíduos e interações** são mais importantes que processos e ferramentas
  - **Software funcionando** é mais importante que documentação completa e detalhada
  - **Colaboração com o cliente** é mais importante que negociação de contratos
  - **Adaptação a mudanças** é mais importante que seguir um plano

# Algo mais concreto: Programação eXtrema

- Metodologia de desenvolvimento de software aperfeiçoada nos últimos 10 anos
- Ganhou notoriedade a partir da OOPSLA'2000
- Nome principal: Kent Beck

# O que é programação extrema?

- Conjunto de práticas a serem adotadas no cotidiano da equipe

## As práticas:

- são adaptáveis a diferentes contextos
- se suportam e complementam
- permitem adoção em pequenos passos
- apoiam os valores essenciais por trás do método

# Princípios básicos de XP

- *Feedback* rápido
- Simplicidade é o melhor negócio
- Mudança incrementais
- Carregue a bandeira das mudanças / não valorize o medo (*Embrace chance*)
- Alta qualidade do código

# Os valores de XP

- Comunicação
- *Feedback*
- Coragem
- Simplicidade
- Respeito

# As 12 práticas de XP (versão 2000)

Planejamento

Fases Pequenas

Metáfora

Design Simples

Testes

Refatoração

Programação

Pareada

Propriedade Coletiva

Integração Contínua

Semana de 40 horas

Cliente junto aos  
desenvolvedores

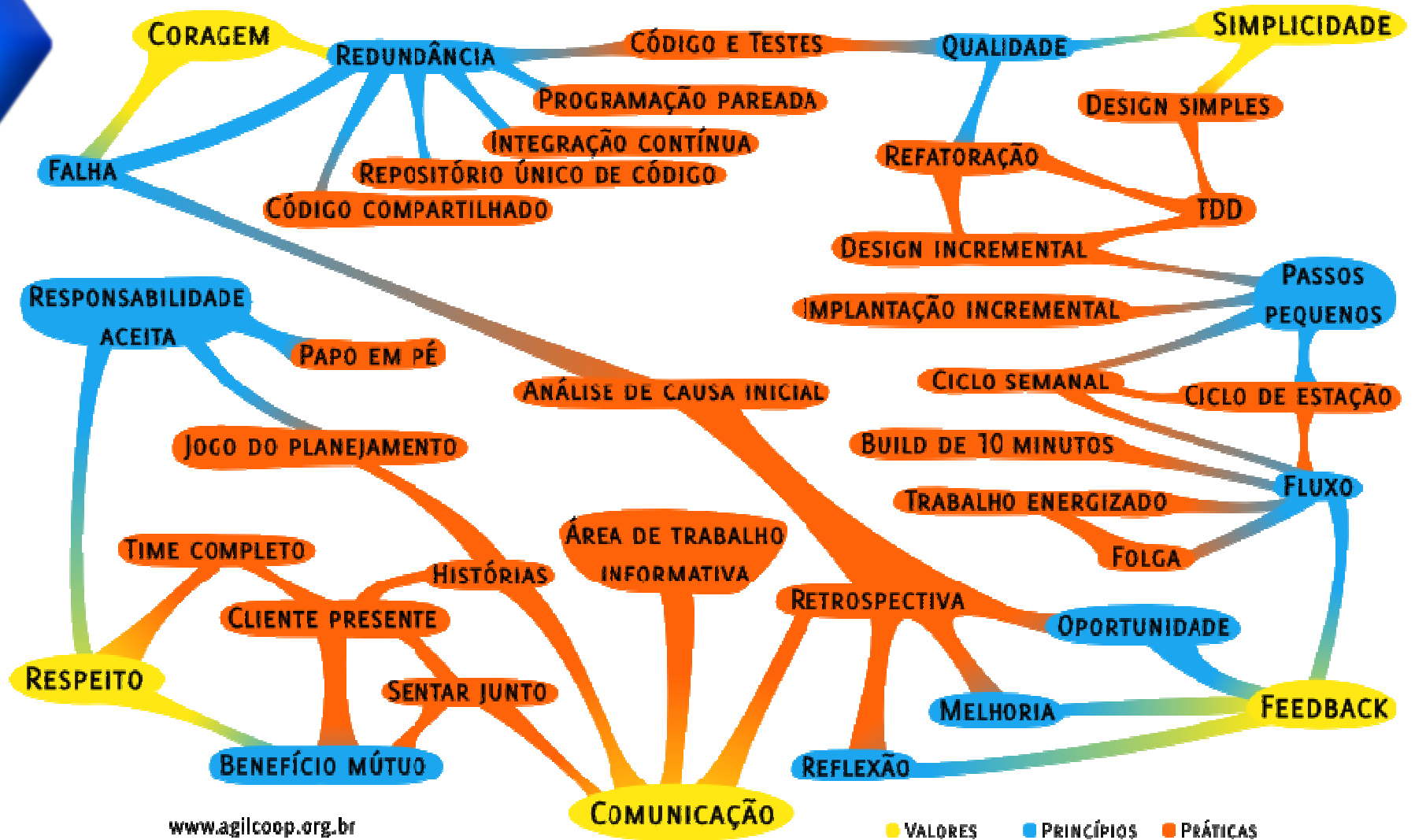
Padronização do  
código

# XP também se adaptou

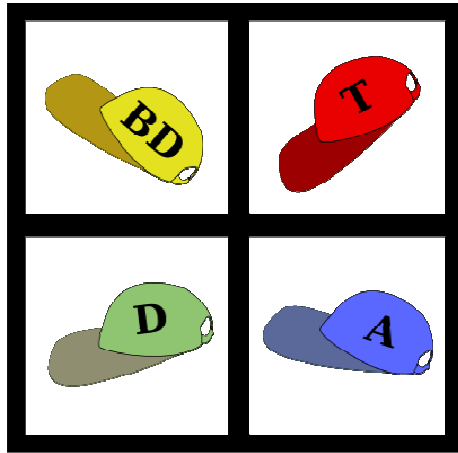
- Novas práticas
  - Principais
    - trazem benefícios imediatos
  - Corolárias
- Adapta-se a diferentes ambientes
- Visão muito mais humana
  - Pessoas são o centro



# Resumo: valores, princípios e práticas

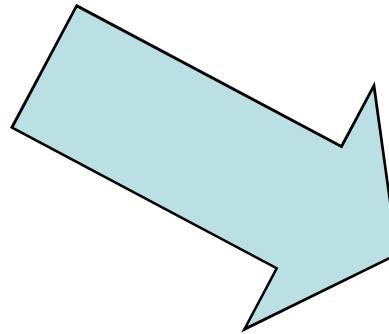


# A equipe e seu ambiente

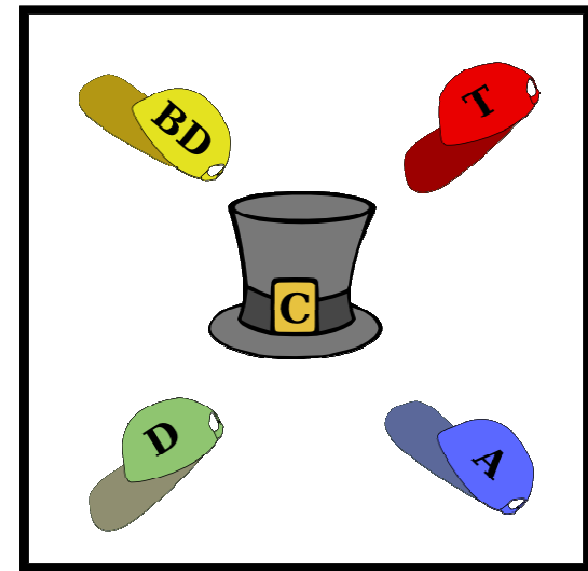


Time completo

Cliente presente



Sentar juntos



# A equipe e seu ambiente

- Papéis

- *Coach*: Lembra a todos as práticas e ajuda com dificuldades na equipe
- *Tracker*: Mantem informações sobre o projeto e elabora gráficos que mostrem as mais importantes à equipe
- *Cliente*: Determina o que é mais importante, responde dúvidas dos programadores e toma decisões sobre funcionalidades

# Área de trabalho informativa



# Jogo do planejamento

- Clientes escrevem e priorizam histórias
- Desenvolvedores estimam as mais prioritárias

Cadastro de cliente	
A secretária realiza o cadastro de um cliente, completando seus dados:	
- Nome completo	
- CPF	
- RG	
	Prioridade : 1
	Estimativa: 4

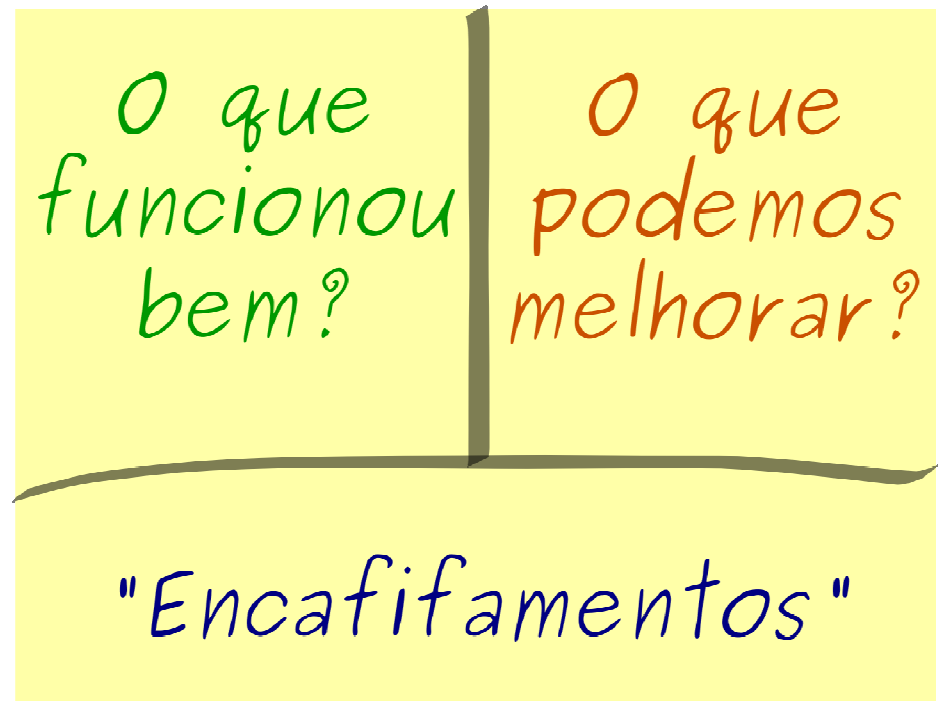
- Suporte do *coach* e do *tracker* para evitar otimismo ou pessimismo excessivo

# Sem exagerar

- Trabalho energizado:  
Balancear a intensidade do trabalho para não desgastar a equipe
- Folga:  
Não planeje até o último minuto. Deixe uma folga para imprevistos porque eles sempre surgem

# Melhorando sempre

- Análise de causa inicial
- Retrospectiva: Nada é perfeito. Tudo sempre pode melhorar e, para isso, precisa entender o que deu certo e errado



# Trabalhando em equipe

- Código compartilhado:  
Eu fiz, você arruma, nós nos ajudamos
- Padronização do estilo de código:  
Seu código e meu código devem ser quase idênticos e indistinguíveis

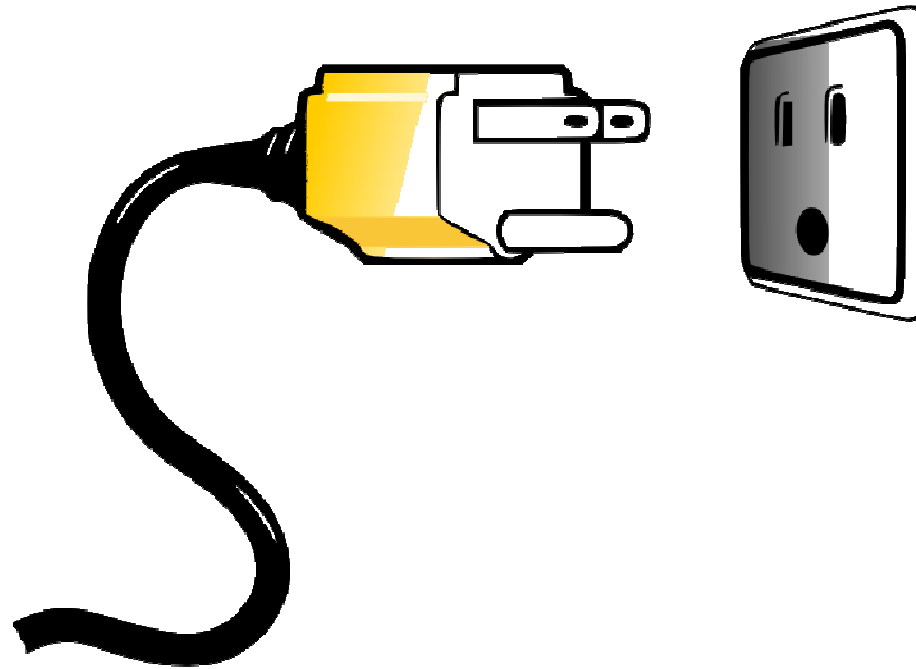


# Design incremental

- Simples  
YAGNI: You Aint Gonna Need It  
“Você não vai precisar disso”
- Refatoração  
DRY: Dont Repeat Yourself  
“Não se repita”

# Prevenindo defeitos

- Auto-inspeção (*mistake proof*)



**Testes são a especificação!**

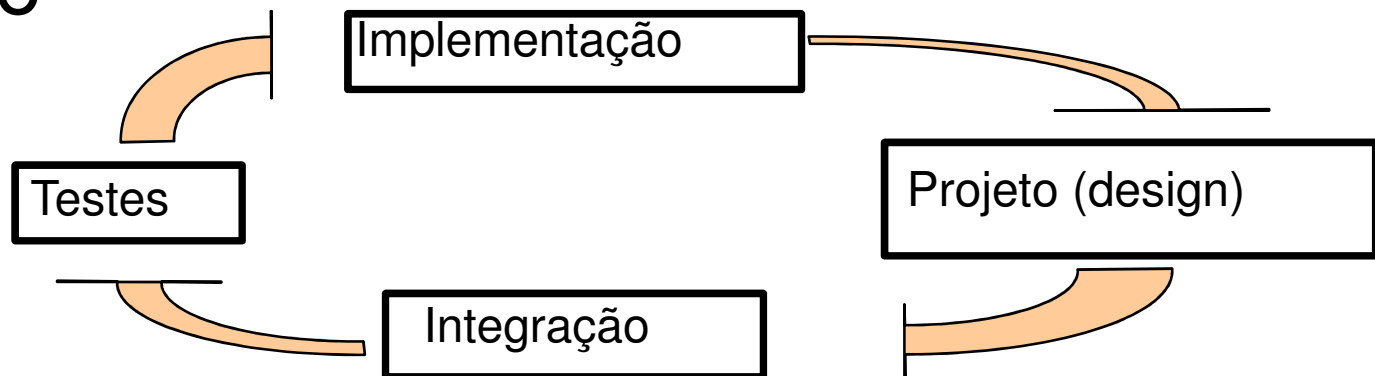
# Redundância

- Programação pareada:  
Distribuindo conhecimento na equipe



# Um dia de XP

- Escolhe uma história do cliente
- Procura um par livre e um computador
- Seleciona um “cartão de história”
- Discute modificações recentes no sistema
- Discute história do cliente
- Entra no ciclo:





# Metodologias

**A melhor metodologia é a  
sua metodologia**

**Com retrospectiva e melhorias contínuas**

# Ser Ágil = Vencer medos

- Escrever código
- Mudar de idéia
- Ir em frente sem saber tudo sobre o futuro
- Confiar em outras pessoas
- Mudar a arquitetura de um sistema em funcionamento
- Escrever testes

# Mestres pelo IME - USP

- Alexandre Freire da Silva
  - Reflexões sobre o ensino de metodologias ágeis na academia, na indústria e no governo 2007
- Danilo Sato
  - Uso eficaz de métricas em métodos ágeis de desenvolvimento de software 2007
- Dairton Bassi
  - Experiências com desenvolvimento ágil 2008
- Daniel Cukier
  - Padrões para introduzir novas ideias 2009

# Mestres pelo IME - USP

- Mariana Bravo
  - Aprendizado de métodos ágeis, 2011
- Hugo Corbucci
  - Métodos ágeis e Software livre, 2011
- Eduardo Katayama
  - Métodos de otimização global e métodos ágeis, 2011
- Renan Melo
  - Análise empírica de técnicas de acompanhamento, 2012



# Em formação

- Mestrandos
  - Maurício Aniche
    - An empirical evaluation of the effects of TDD in internal code quality
  - Thiago Ferreira
    - impacto de estratégias de coordenação inter-times na produtividade de times ágeis

# Em formação

- Doutorandas
  - Cláudia Melo
    - Produtividade de equipes ágeis
      - Qual o impacto das práticas nas equipes ?
      - Como medir esta produtividade ?
  - Viviane Almeida
    - Uso de aprendizagem organizacional na melhoria de processos de desenvolvimento ágil
      - Capturar e transmitir o conhecimento de melhoria de processo
  - Graziela Tonin
    - Métodos ágeis e dívida técnica

# Dúvidas?



[www.ime.usp.br/~gold](http://www.ime.usp.br/~gold)

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