Métodos Ágeis de desenvolvimento de software: uma abordagem humana

Alfredo Goldman

Professor do IME – USP

AgilCoop

Pequeno histórico

- Docente do IME - USP desde 1993
- Começamos uma disciplina:
  - Laboratório de Programação Extrema em 2000
- De uma forma natural começamos a ensinar no que acreditávamos ser bom
- Fizemos os primeiros eventos do Brasil
  - Encontro Ágil
- Hoje temos:
  - AgileBrazil
  - AgileTrends e uma enorme comunidade ágil

Programação hoje

- Não é mais uma atividade solitária
- Mais de uma pessoa para criar software
- Pessoas vão manter o software
  - Geralmente outras
- Atividade cooperativa
  - Melhor analogia: Alpinismo
- Fator complicador
  - Pessoas não são estáticas

Solução: virou um problema

- Métodos ágeis fizeram sucesso
- *Agile* virou *buzzword*
- Muita gente se diz ágil hoje
  - Mas, infelizmente poucos são ágeis de verdade
- Preocupação da comunidade na volta aos valores

Métodos ágeis NÃO são processos ou práticas

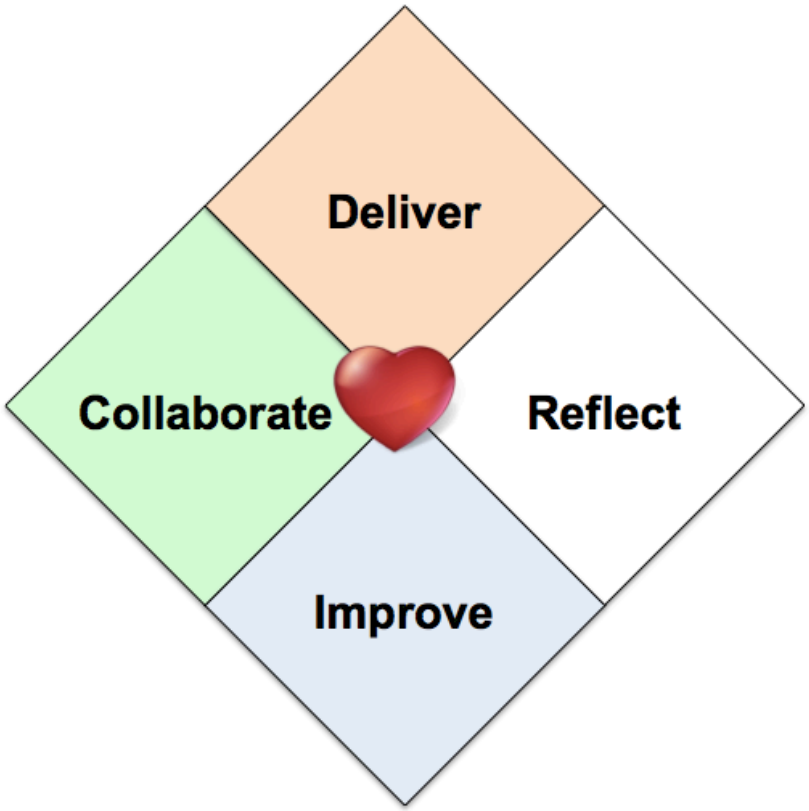
- Artigo do Highsmith
  - "Stop doing agile start being agile"
- Não adianta tirar um certificado CSM e achar que basta
- Agilidade é um estado de espírito

Insight da TW de 30/10/16

- O futuro do ágil:
  - Inovadores, imitadores e idiotas
- Não siga regras, siga os valores
- Como prosseguir evoluindo:
  - Inovação (ex: CD, DevOps)
  - Compromisso entre ideais e prática
  - Volta aos valores originais
  - Procurar a união ao invés de separar

O Coração do Ágil

- Um diagrama muito significativo
  - Do Alistair Cockburn



Agora sim: linhas gerais da apresentação

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

Um caso de outra área

Apolo 13

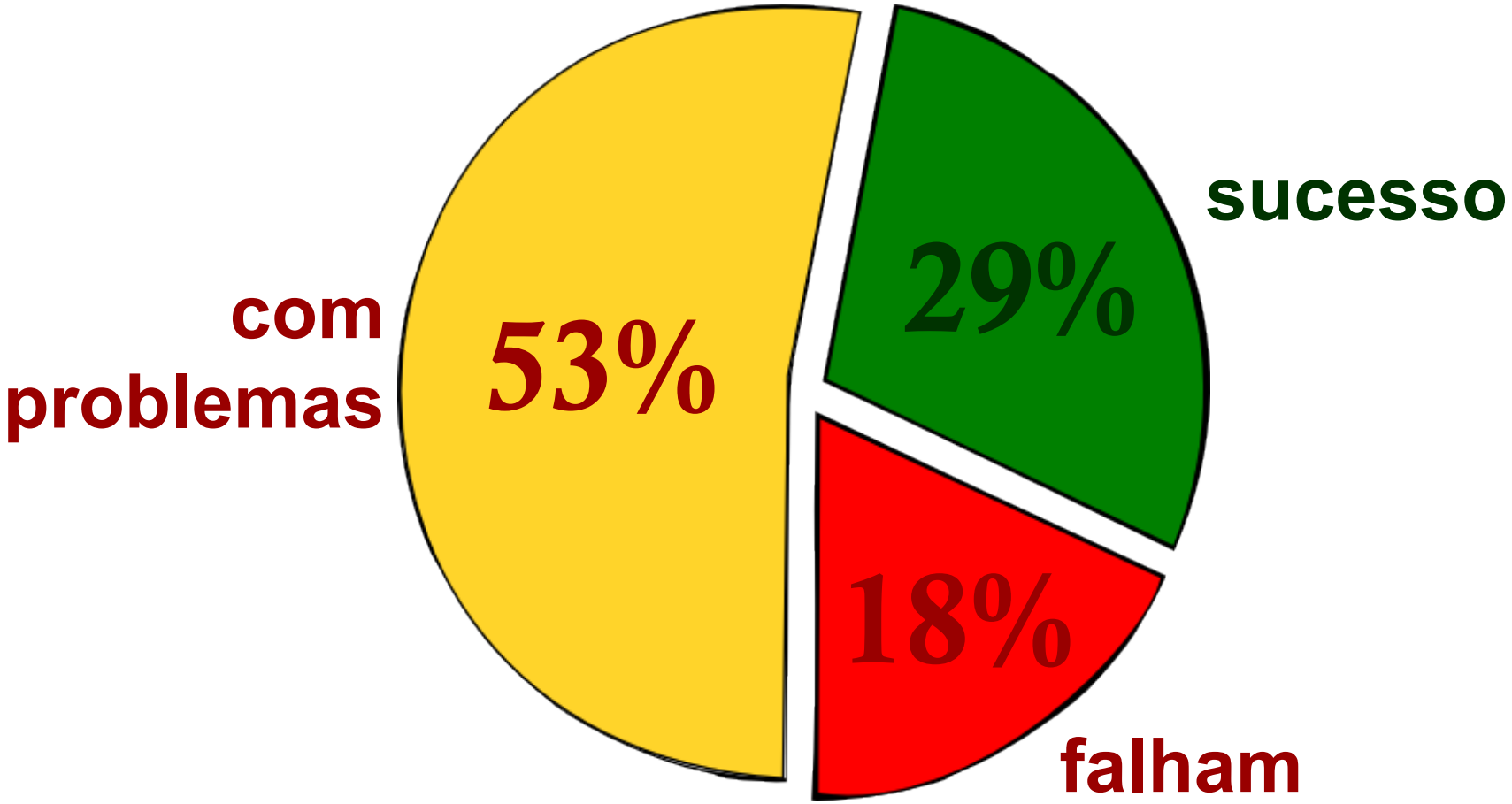
- Acidente em módulo de serviço
- 1970
- *Houston, we have a problem*
- Problema de refrigeração
- Equipe
- Entrosamento
- Improvisação

Columbia

- Tecnologia mais avançada
- 2003
- Ônibus espacial
- 28 missão
- Pequeno dano na asa, na decolagem
- Três pedidos de revisão
- Negados pelos gerentes

CHAOS report

- Resultado dos projetos (2004):



Chaos Report: evolução ?

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

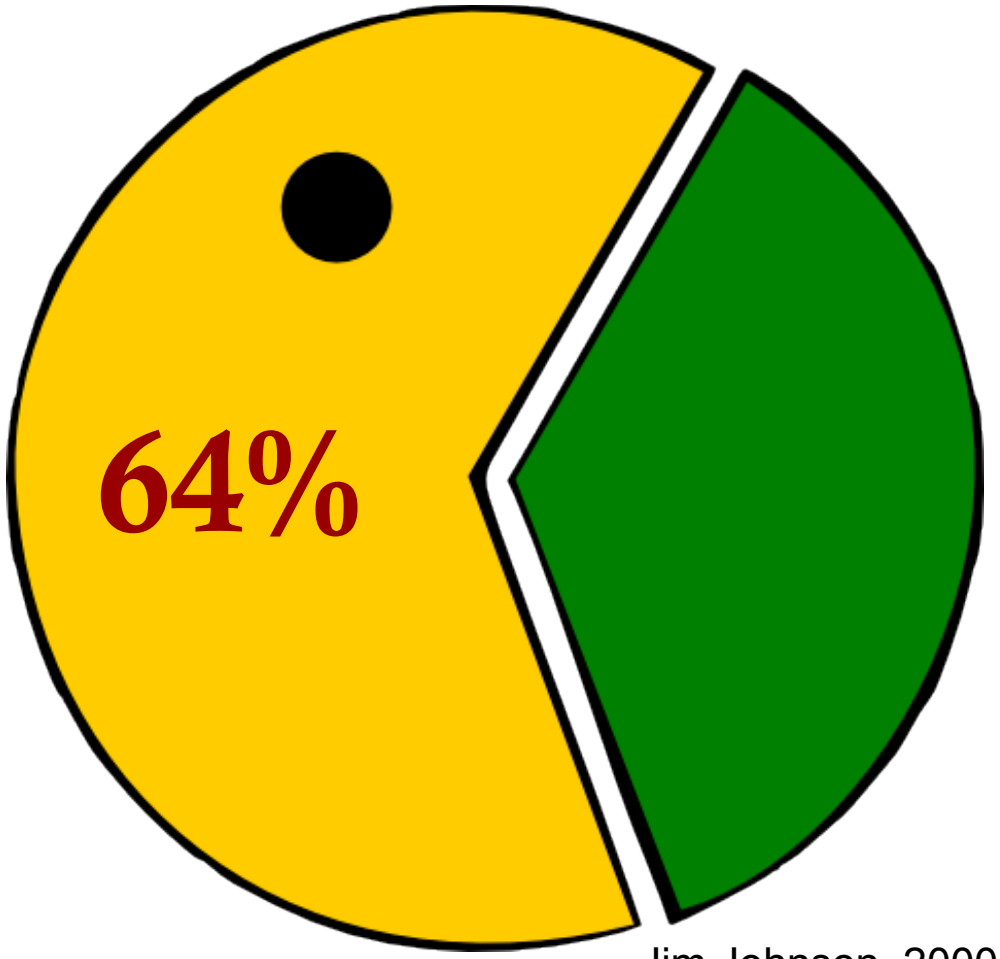
The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

https://www.infoq.com/articles/standish-chaos-2015

Ver também: The Rise and Fall of the Chaos Report - Vrije Universiteit Amsterdam
http://www.cs.vu.nl/~x/chaos/chaos.pdf

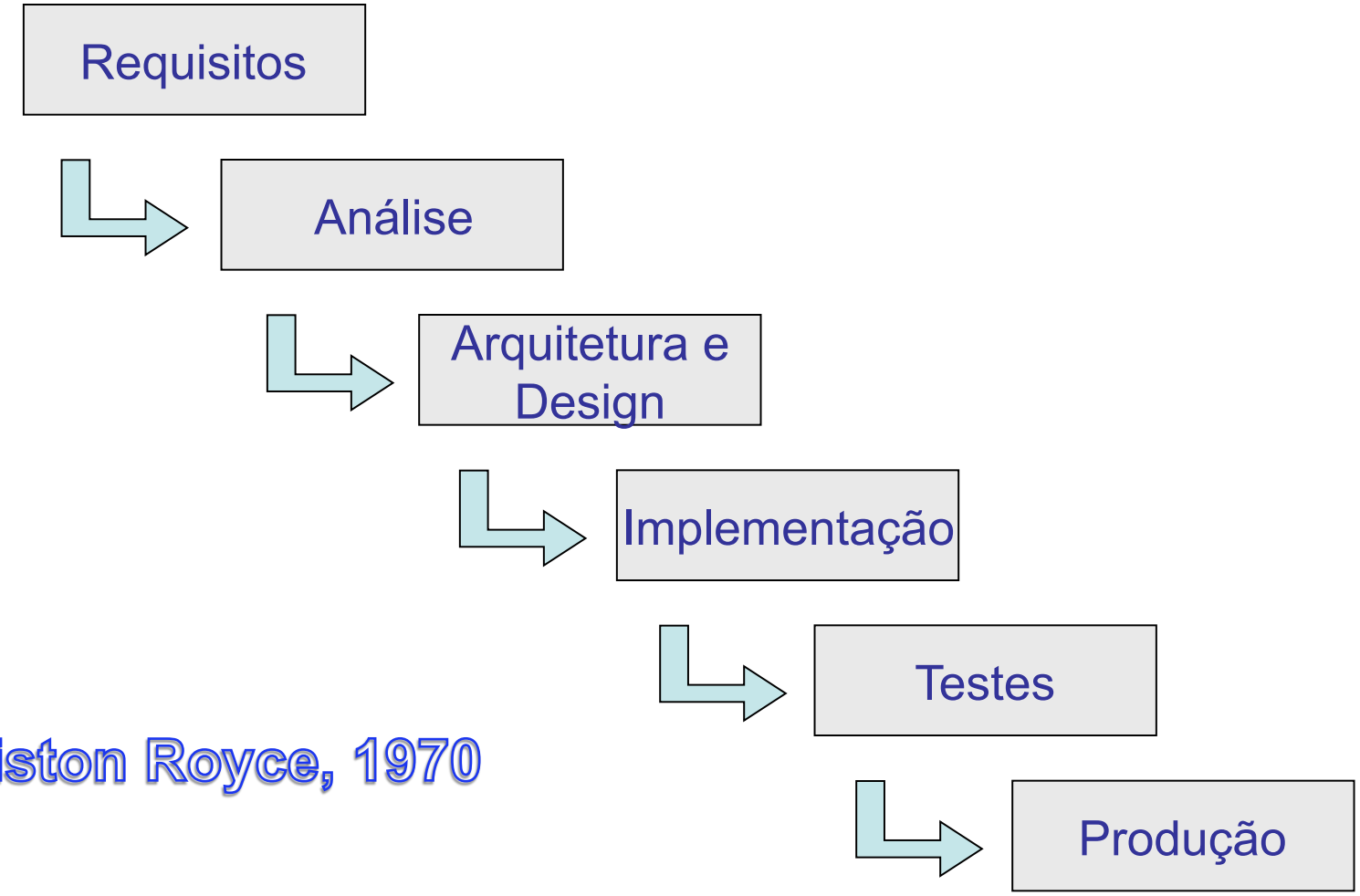
Qual software?

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



Jim Johnson, 2000

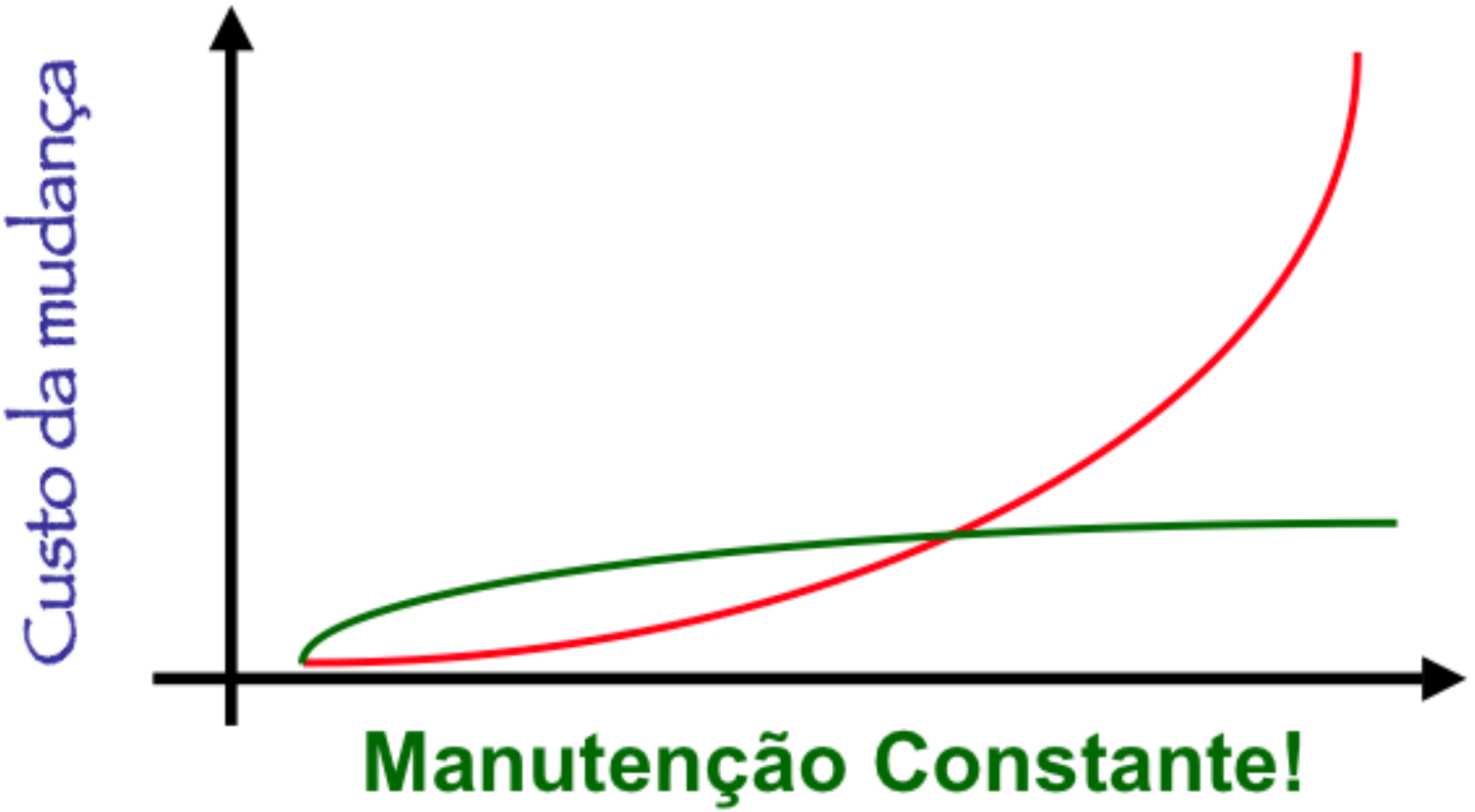
Modelo Cascata



Wiston Royce, 1970

Analogia incorreta 1

Engenharia de Software ≠ Engenharia Civil



Analogia incorreta 2

Engenharia de Software ≠ Engenharia Civil

Programar é fazer Design!

O que é desenvolvimento de software (A. Cookburn)?

What is / isn't software development?

Model Building	**(Jacobson)**
Engineering	**(Meyer)**
Discipline	**(Humphreys)**
Poetry	**(Cockburn)**
Math	**(Hoare)**
Craft	**(Knuth)**
Art	**(Gabriel)**

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.

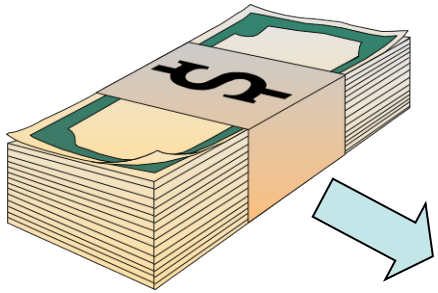
O que é valor?

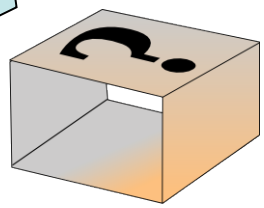
Processos tradicionais:

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

- “Telefone sem fio”

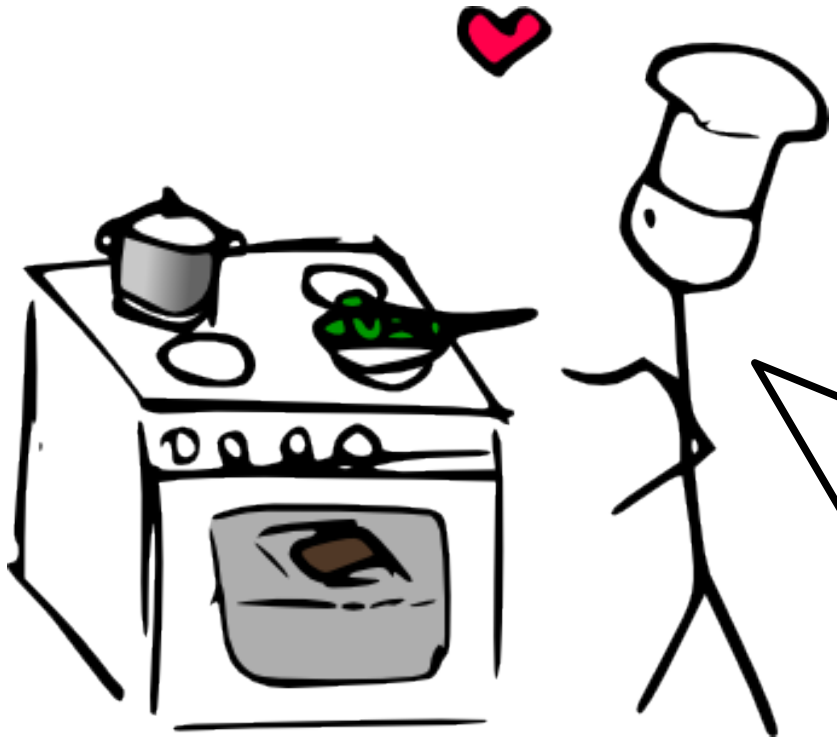
- “Documento sem fio”

Segundo princípio

Indivíduos e interações são mais importantes que processos e ferramentas

Ferramentas como controle de versões, boas IDEs e ambientes de IC continuam sendo usadas

Fazer certo o software

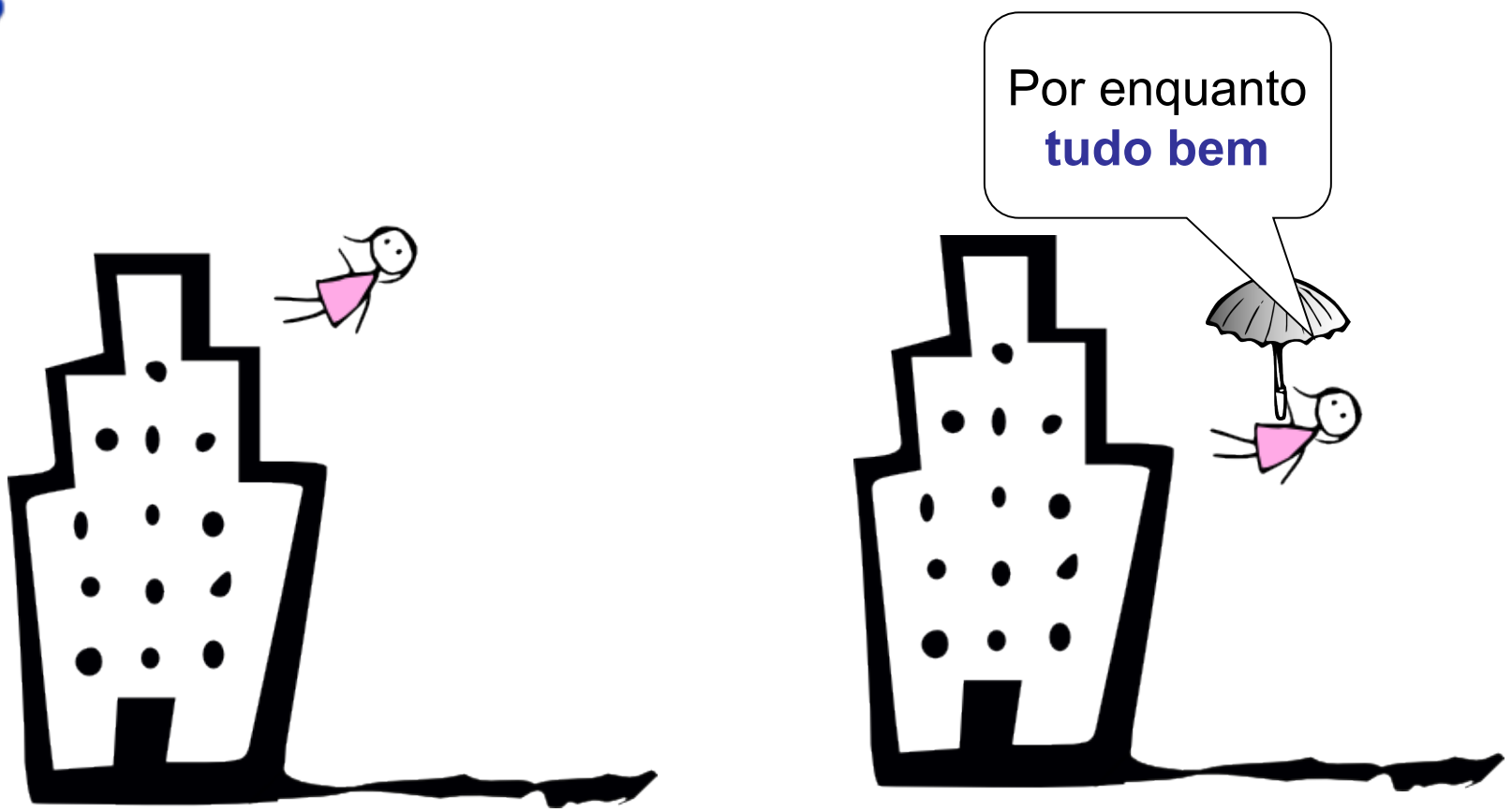


Carne assada e vagem com *bacon*, uma delícia. Seguindo a receita vai ficar muito gostoso!

Sem dúvida vai ser um **sucesso**!!

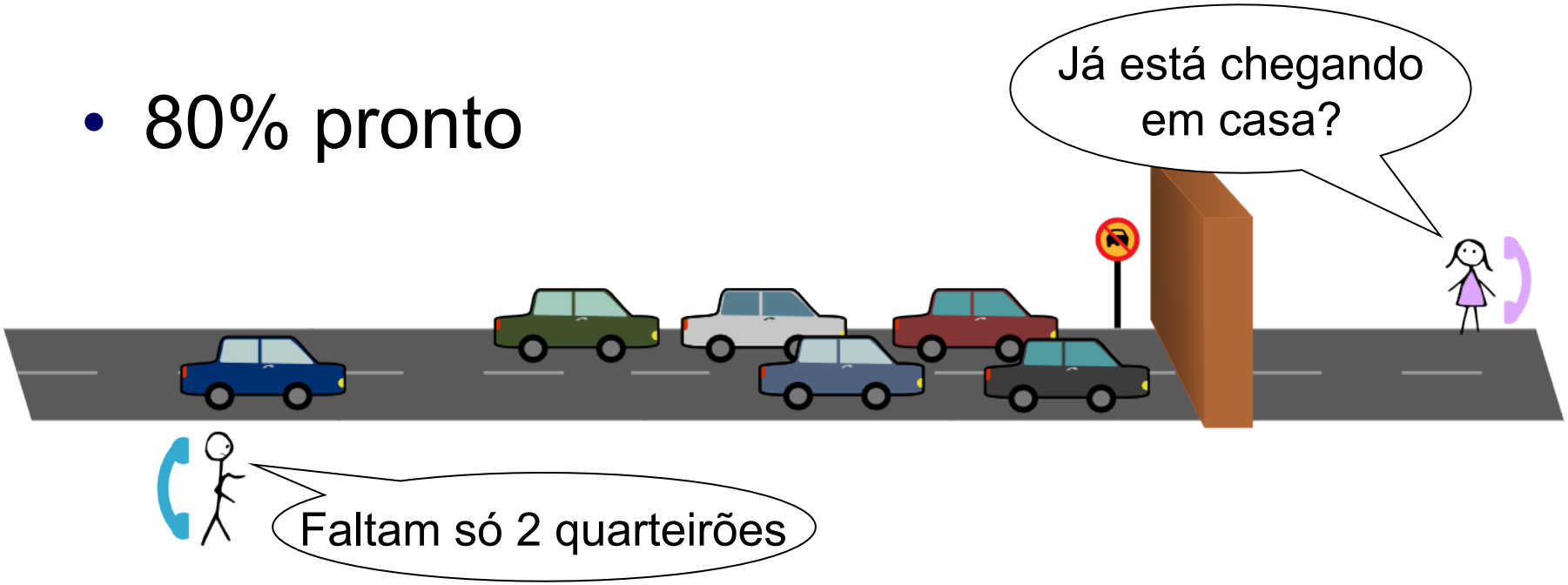
Fazer o software certo

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 do ano que vem
  - Faça sua lista de compras dessa semana
- 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

Manifesto Ágil

- Leitura obrigatória
- Não dá para transformar tudo em processo
  - Há um trabalho intelectual/criativo
- Hoje o valor ágil principal para mim é
  - **Melhoria contínua**

Algo mais concreto: Programação eXtrema

- Metodologia de desenvolvimento de software aperfeiçoada desde 1999
- 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 4 Variáveis do Desenvolvimento de Software

- Tempo
- Custo
- Qualidade
- Escopo (foco principal de XP)

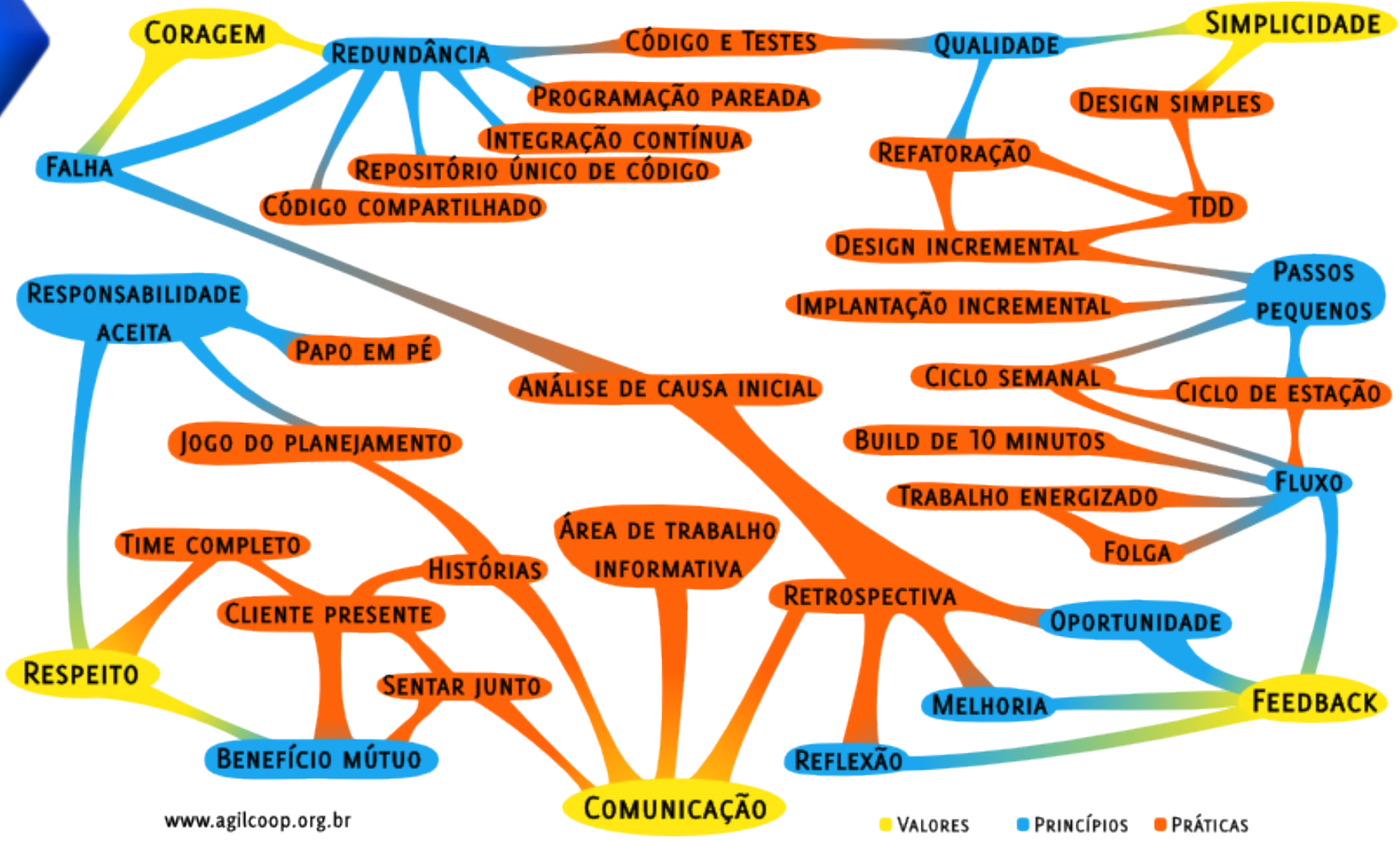
As 12 práticas de XP (versão 2000)

- 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

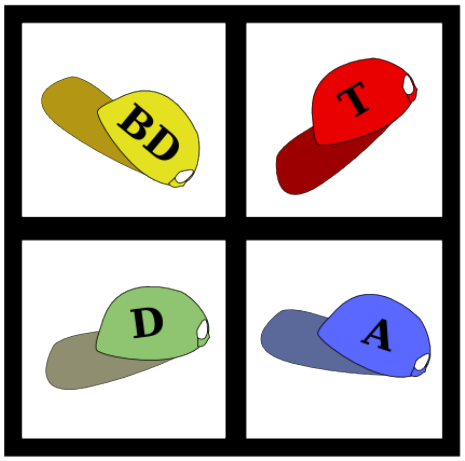
XP também se adaptou

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

Resumo: valores, princípios e práticas

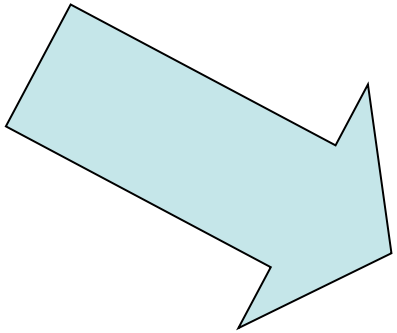


A equipe e seu ambiente

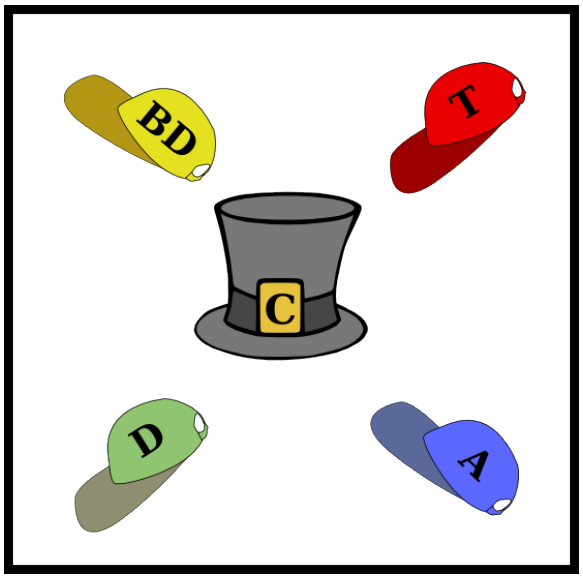


Time completo

Cliente presente



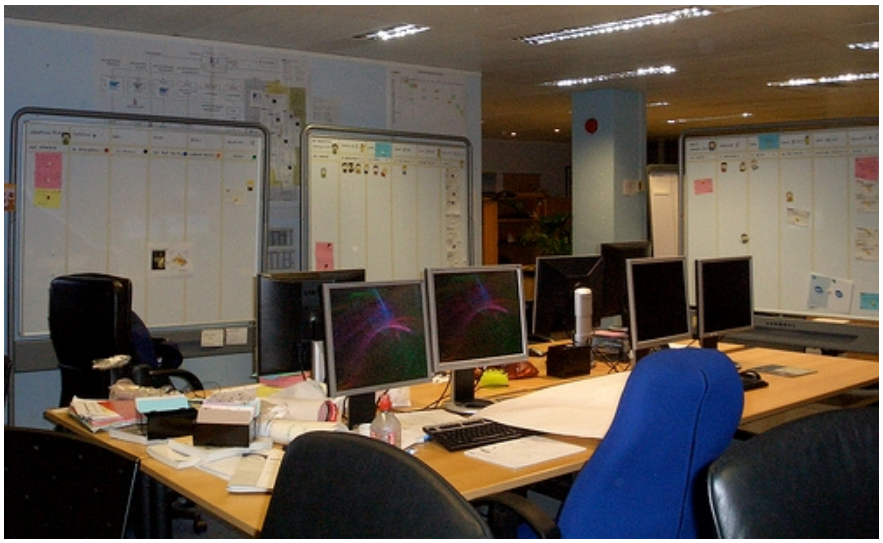
Sentar juntos



A equipe e seu ambiente

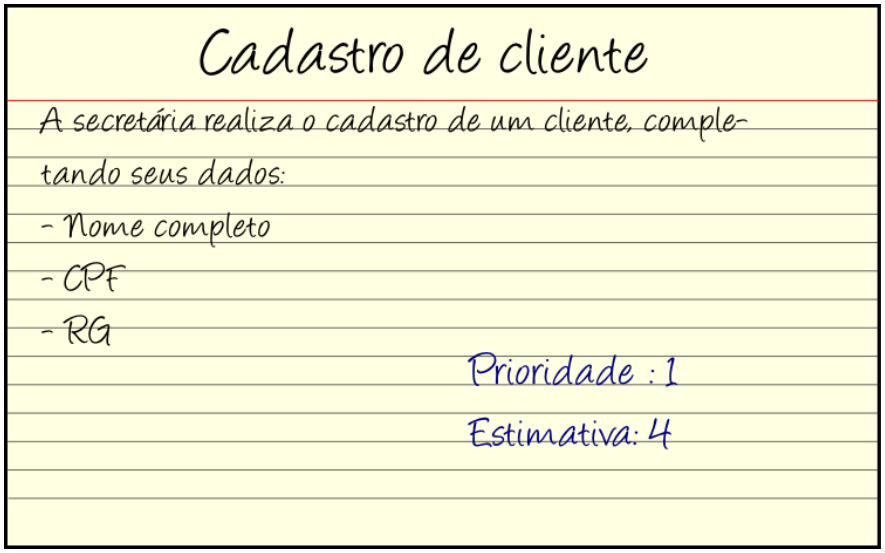
- Papéis
  - *Coach*: Lembra a todos as práticas e ajuda com dificuldades na equipe
  - *Tracker*: Mantem informações sobre o projeto e elabora gráficos que mostrem as mais importantes à equipe
  - *Cliente*: Determina o que é mais importante, responde dúvidas dos programadores e toma decisões sobre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

Área de trabalho informativa



Jogo do planejamento

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



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

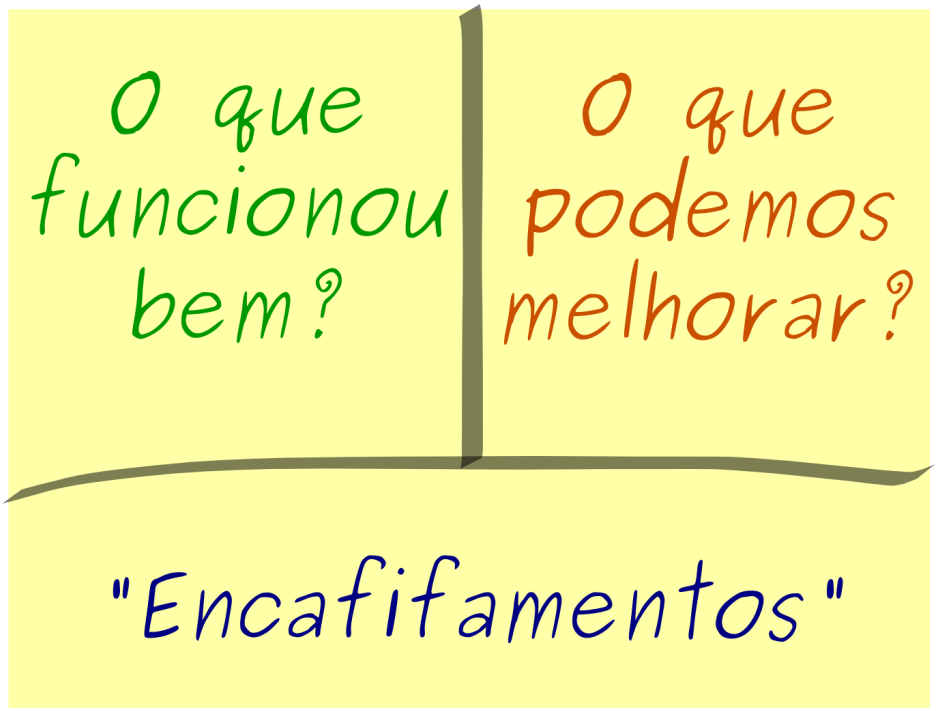
Sem exagerar

- 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



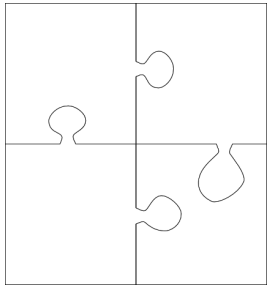
Trabalhando em equipe

- Código compartilhado:
  Eu fiz, você arruma, nós nos ajudamos

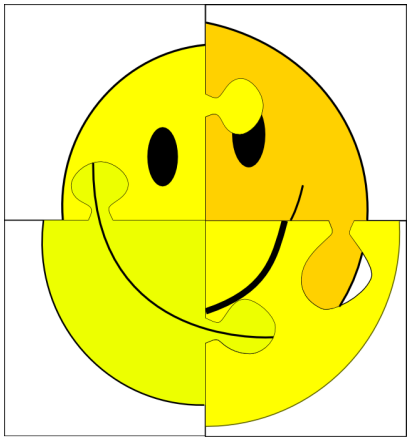
- Padronização do estilo de código:
  Seu código e meu código devem ser quase idênticos e indistinguíveis

Código unificado

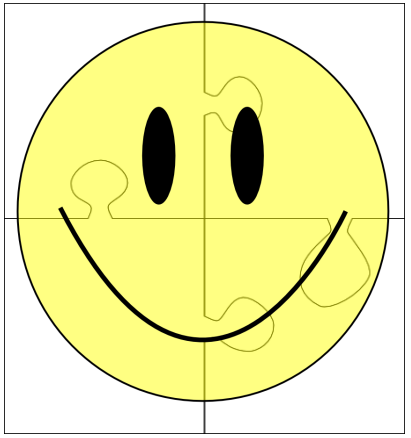
- Repositório único
- Integração contínua



SEM



COM



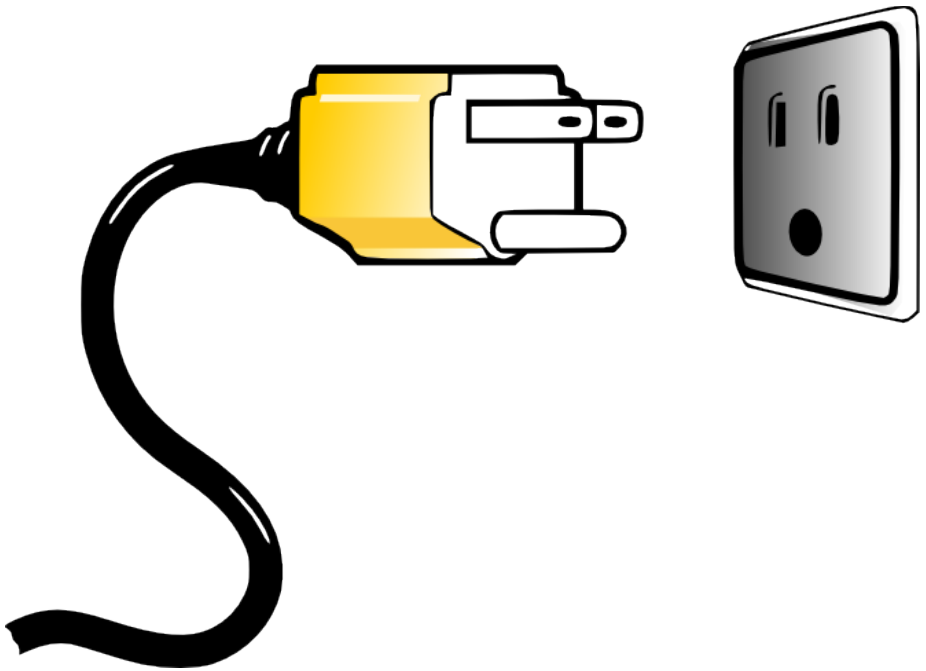
Design incremental

- 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!

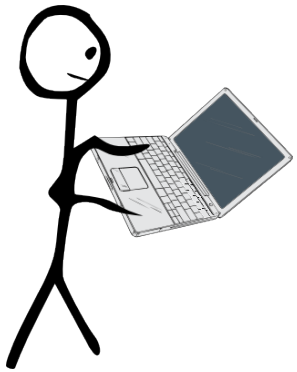
Testes

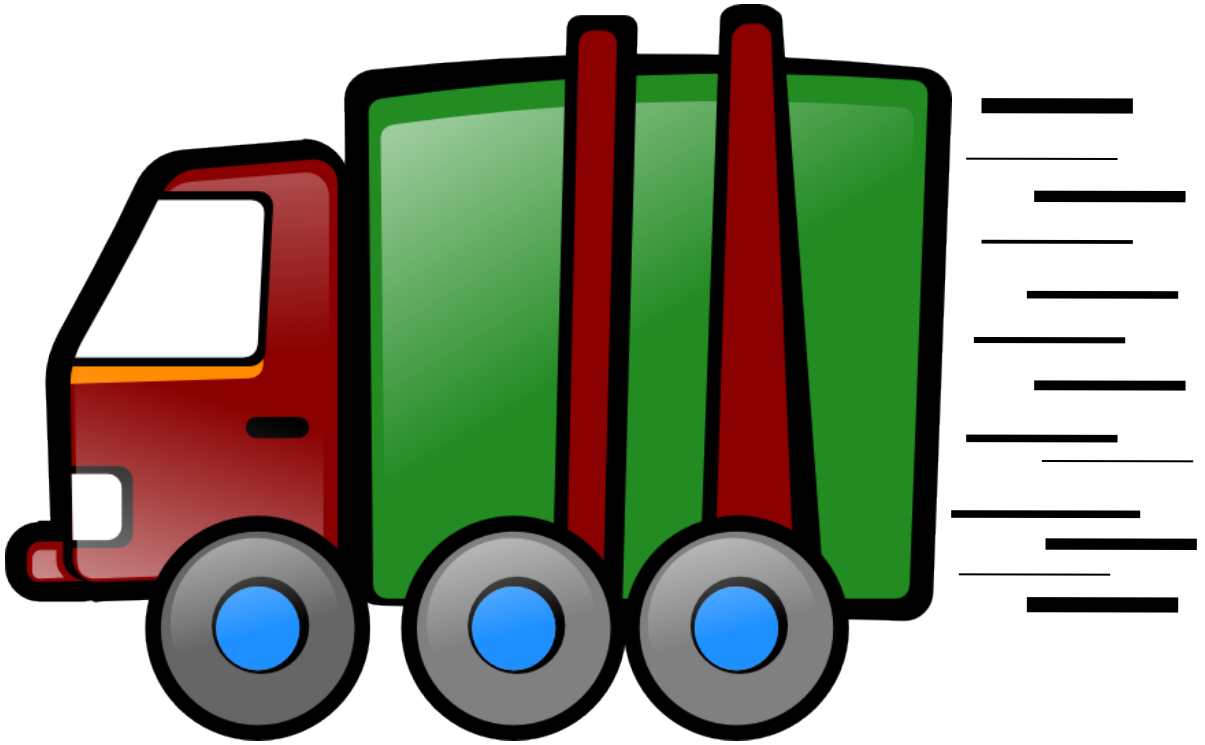
- Desenvolvimento dirigido por testes

- Testes automatizados
  - Unidade: Verifica o código
  - Aceitação: Verifica a funcionalidade
  - Interface: Verifica a interação com usuário

Redundância

- Programação pareada:
  Distribuindo conhecimento na equipe





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

Mais informação

- AgileBrazil
- AgileTrends
  - Participe, discuta e aprenda com os outros
- Agilcoop (deprecated)
- Página da ThoughtWorks
- AgileAlliance

Leia mais

- Livros
  - XP, edição 1 e 2. Kent Beck
  - Refactoring, Martin Fowler
  - TDD, Kent Beck
  - Livros da casa do código
- Livros não técnicos
  - Think Fast Think Slow, Daniel Kahneman
  - Drive, Daniel Pink (tem TED)
  - Management 3.0, Jurgen Appelo
- Vídeos do Eduardo Guerra - INPE

Dúvidas?

Obrigado !

www.agilcoop.org.br

www.ime.usp.br/~gold

Agilcoop

- Agilcast
- Aprenda mais
- Artigos
- Contato
- Cursos e Palestras
- Dissertações
- Eventos
- Membros
- Projetos de Software

área de membros

Usuário *

goldman

Senha *

••••••

- Recuperar senha

Entrar

Compartilhamento de Conhecimento entre Equipes Ágeis: Fatores Influenciadores

Enviado por viviane.almeida em ter, 14/08/2012 - 08:34

Após pesquisas preliminares realizadas pelo nosso grupo em empresas ágeis no período de 2010 a 2011 (Disponível em: http://bit.ly/technical-report-2012), identificamos que empresas ágeis reconhecem a necessidade de compartilhar conhecimento entre equipes, portanto aplicam diversas práticas com propósito, de baixo custo e focadas em gerar interação entre seus profissionais. Além disto, identificamos que fatores organizacionais influenciam na efetividade deste processo.

Os grupos de pesquisa em métodos ágeis da Universidade de São Paulo e da Universidade Estadual do Ceará convidaram empresas ágeis do ranking das 95 melhores empresas de TI para trabalhar de 2011 da revista ComputerWorld (disponível em: http://computerworld.uol.com.br/gptw/2011/ranking_geral) para responder um questionário sobre a influência das condições organizacionais no processo de compartilhamento de conhecimento entre equipes ágeis.

Leia mais

Slides de "Uma introdução ao Desenvolvimento de Software Lean"