

# Introdução a Métodos Ágeis



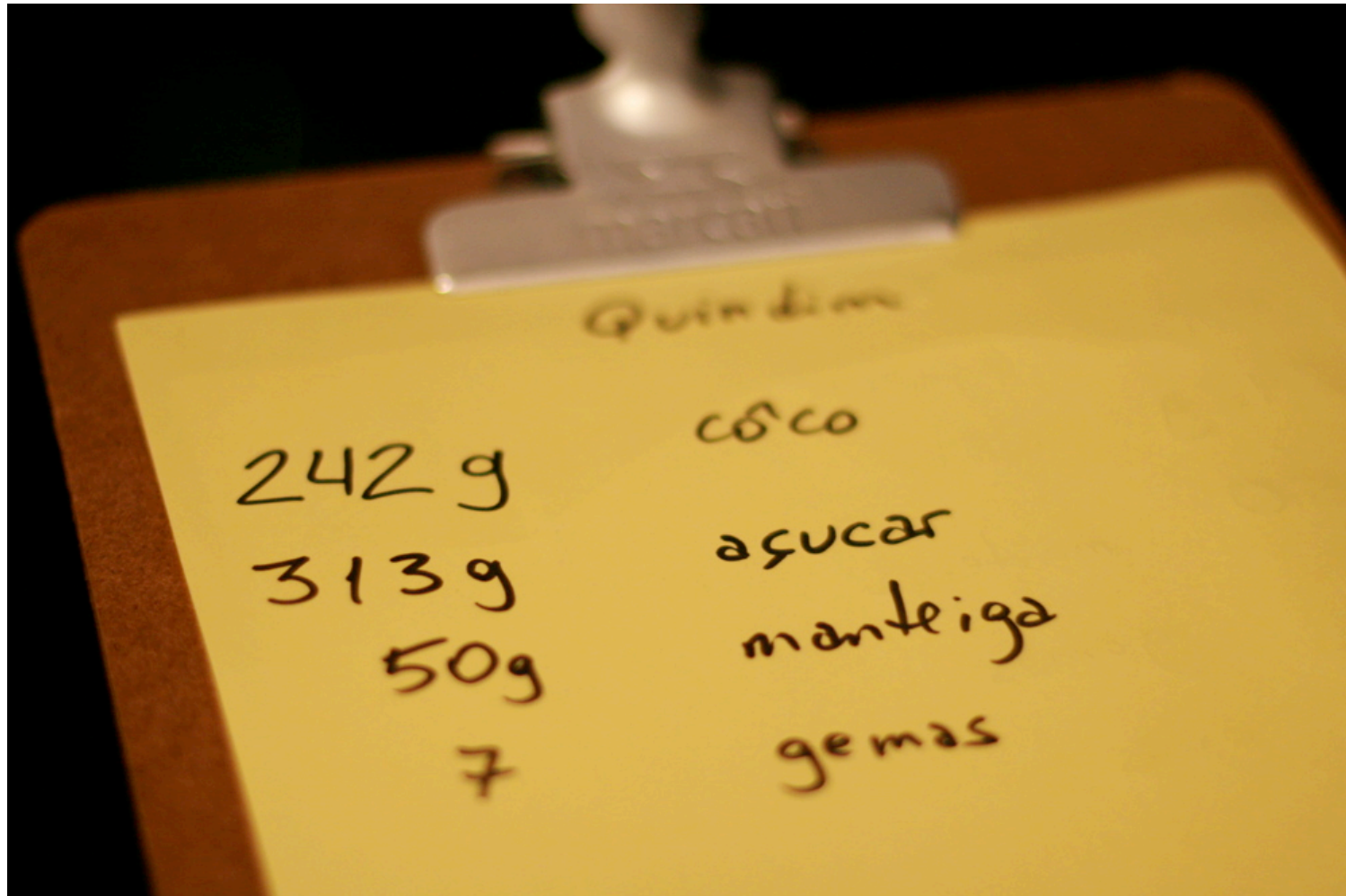
Curso de Verão 2008 - IME/USP

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

Danilo Sato

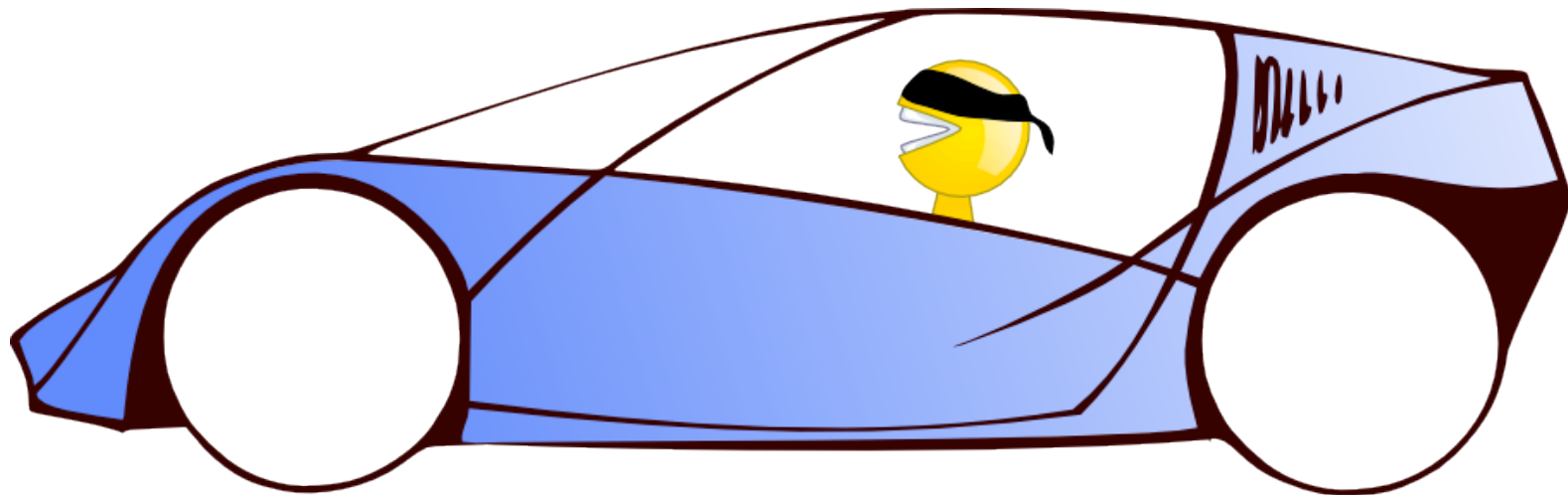
Mariana Bravo

# Tradicional ou Ágil?

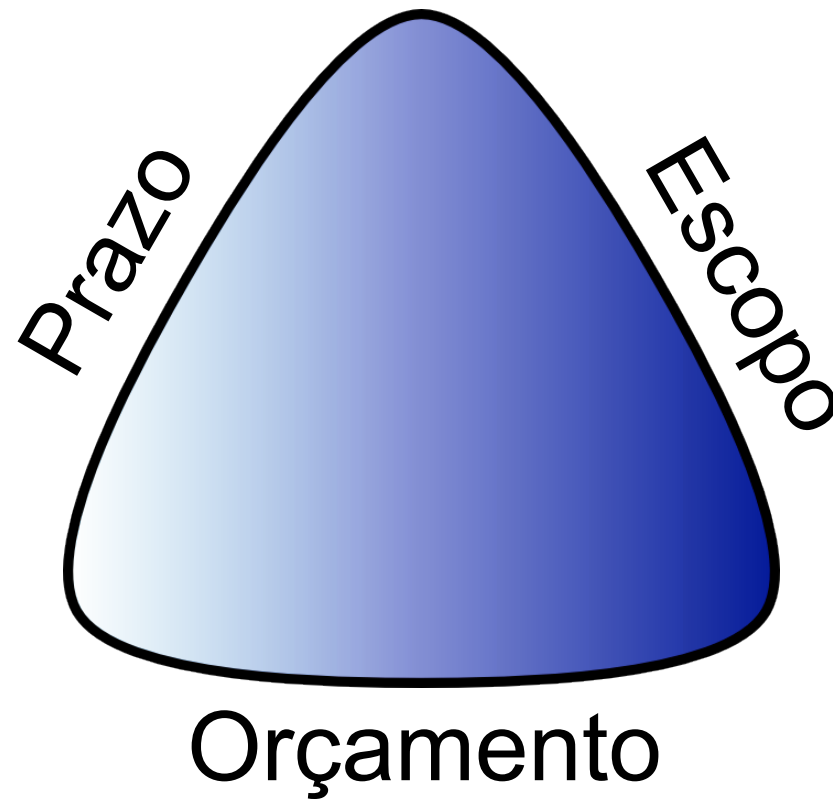


# Tradicional ou Ágil?

***Forecast-driven***  
**VS**  
***Feedback-driven***

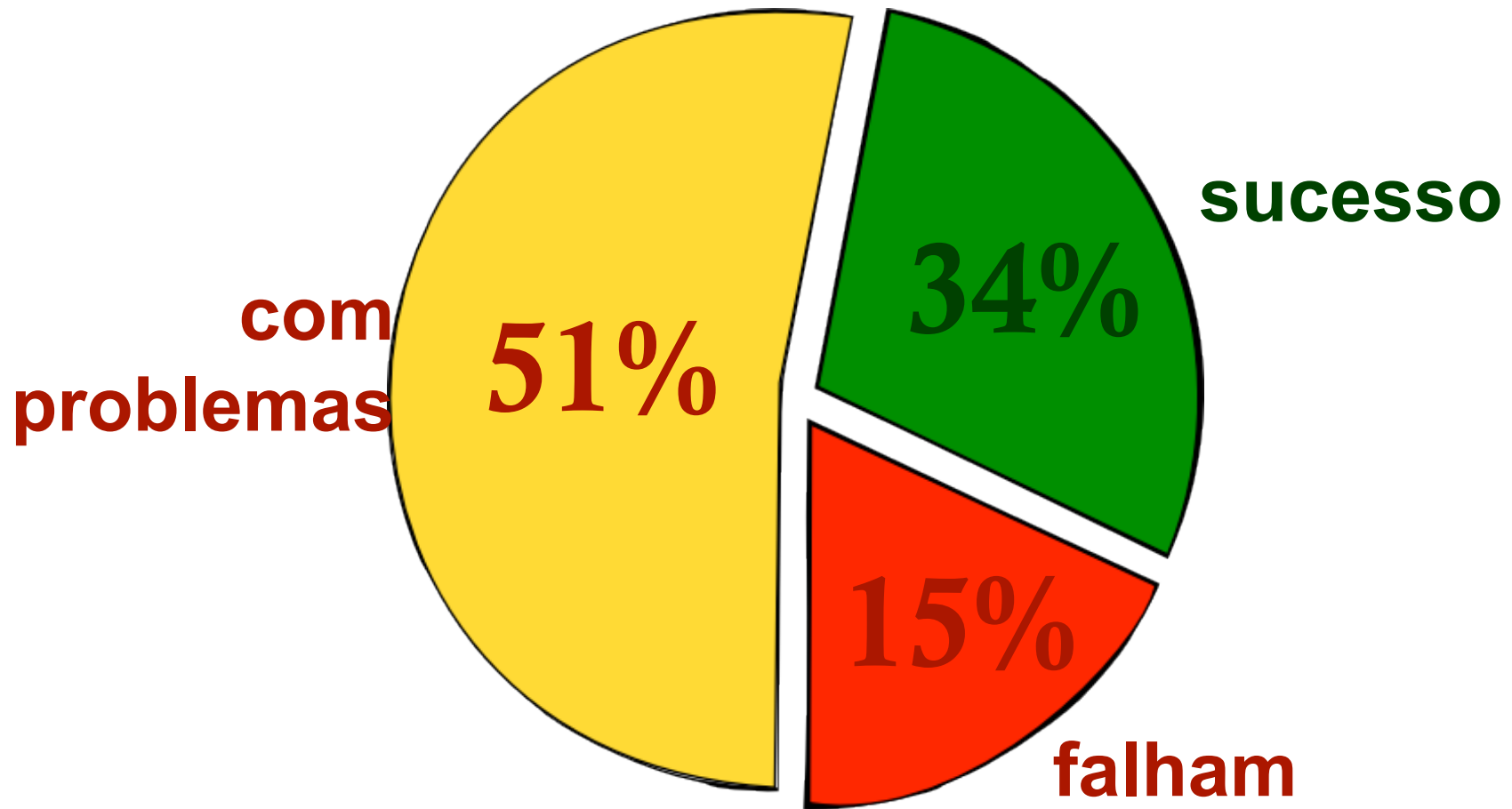


# O Que é Sucesso?



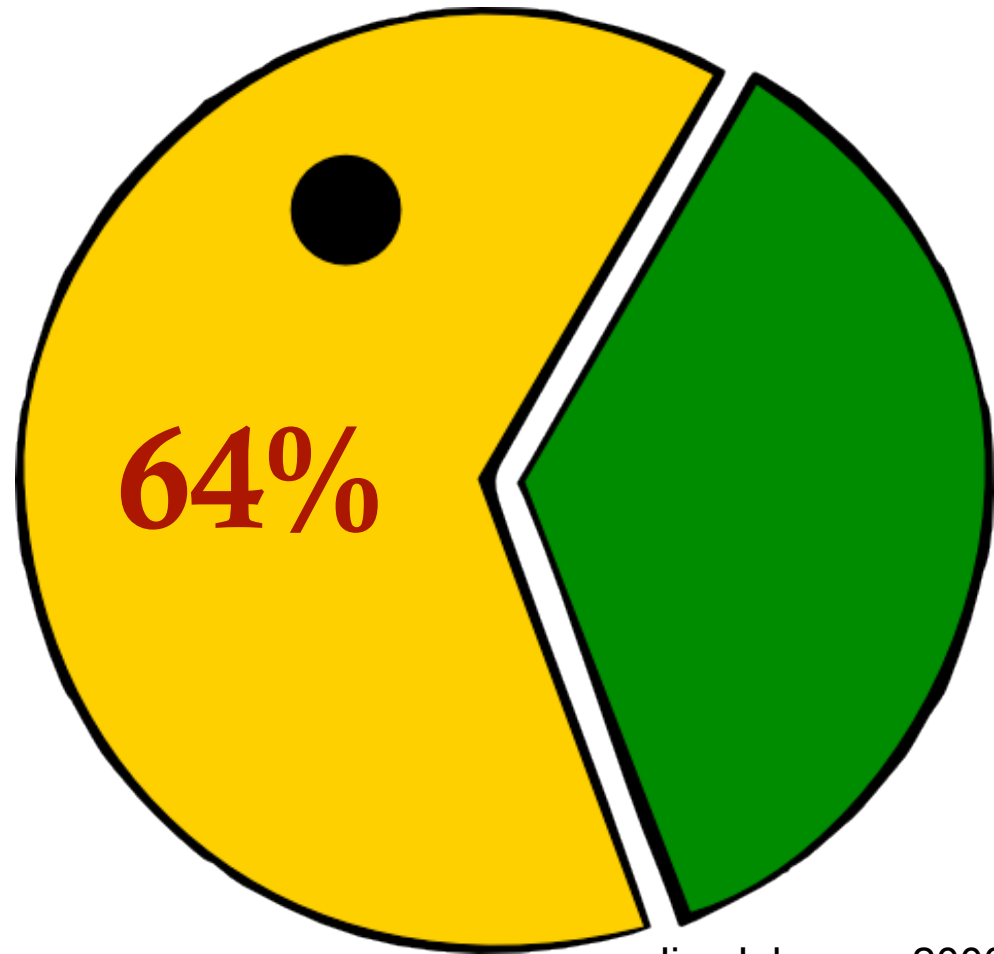
# CHAOS Report

- Resultado dos projetos (2003):



# Qual software?

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

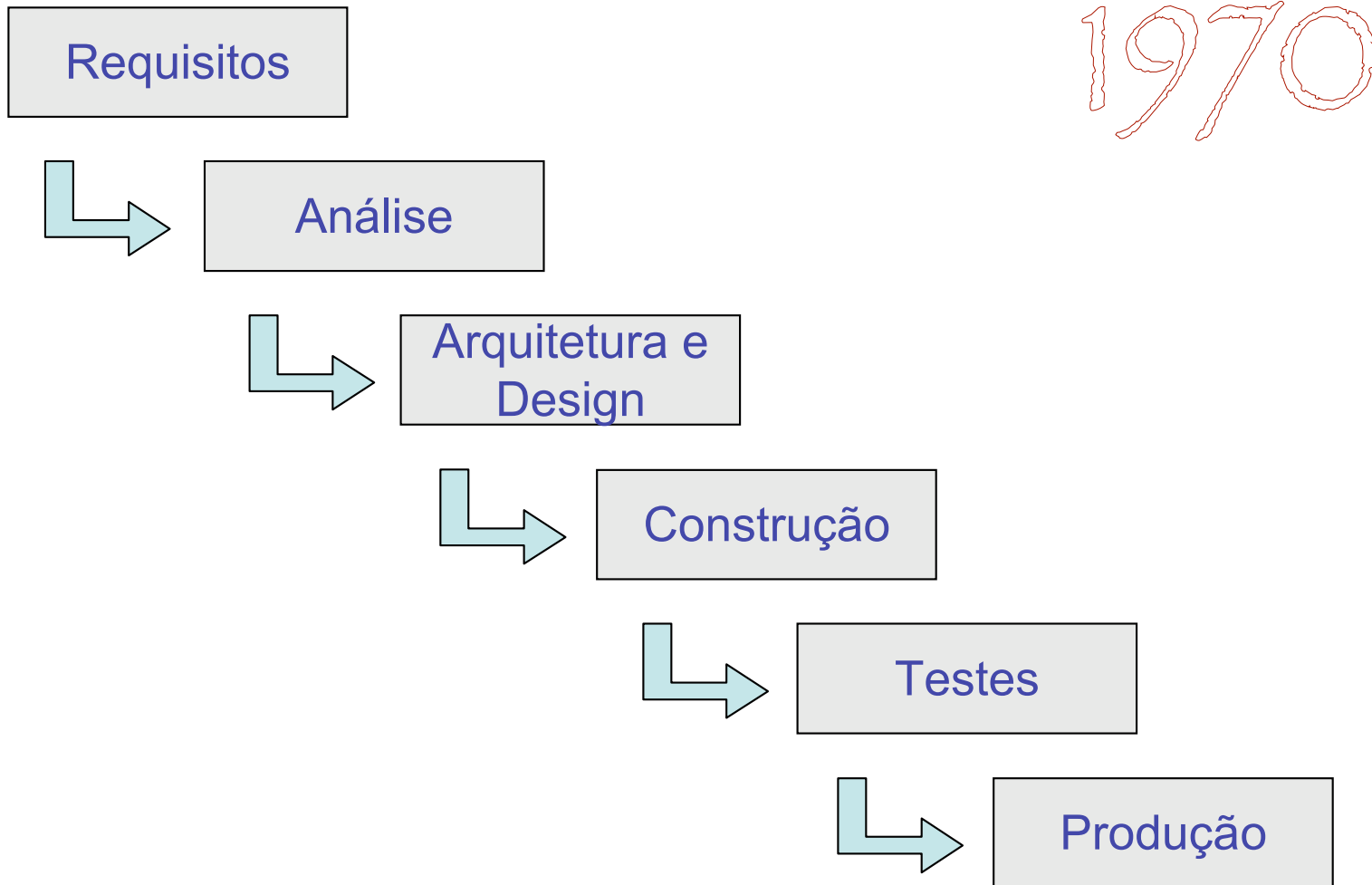


Jim Johnson, 2000

# O Que é Sucesso?

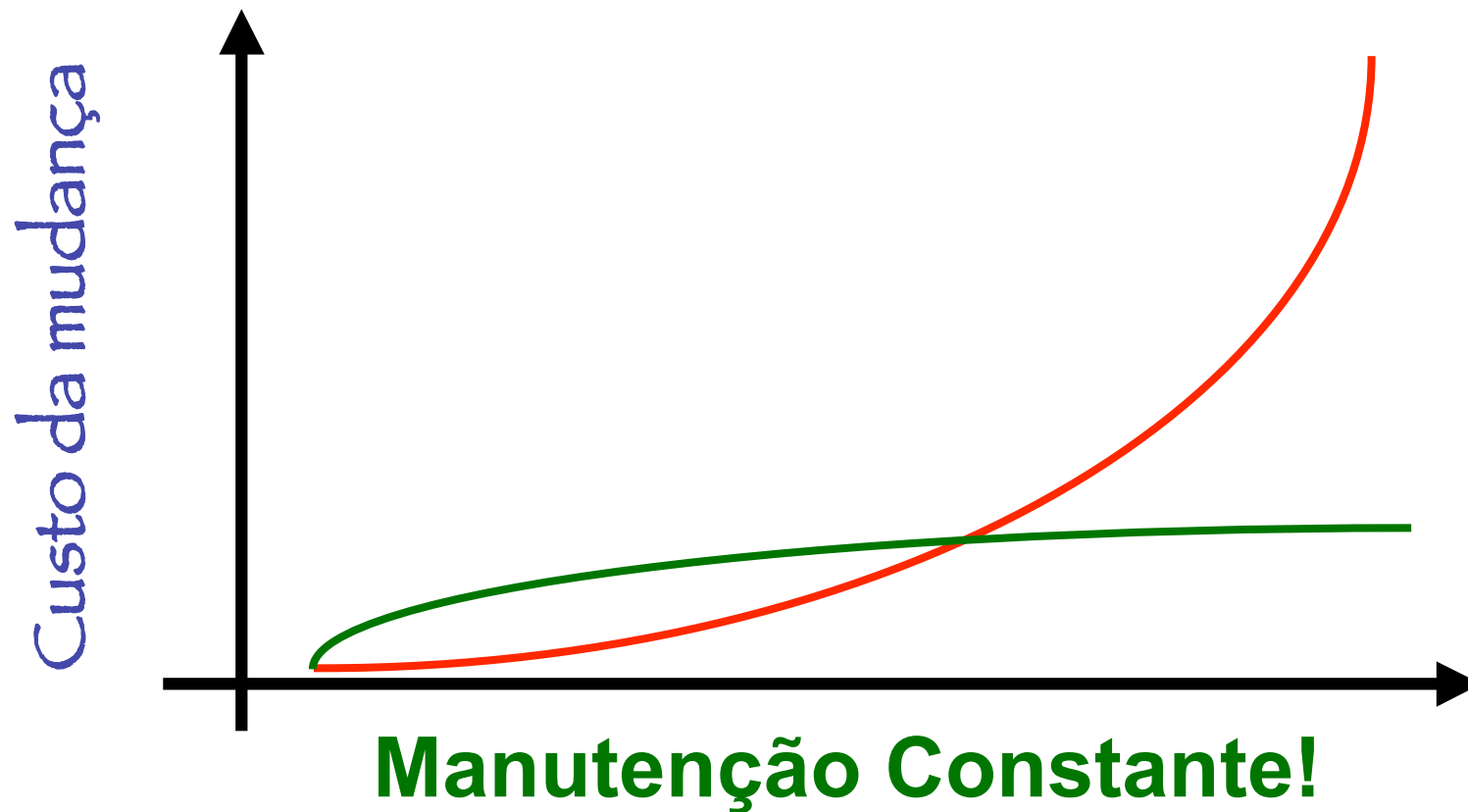


# Um pouco de história...



# Analogias...

Engenharia de Software  $\neq$  Engenharia Civil



# Analogias...

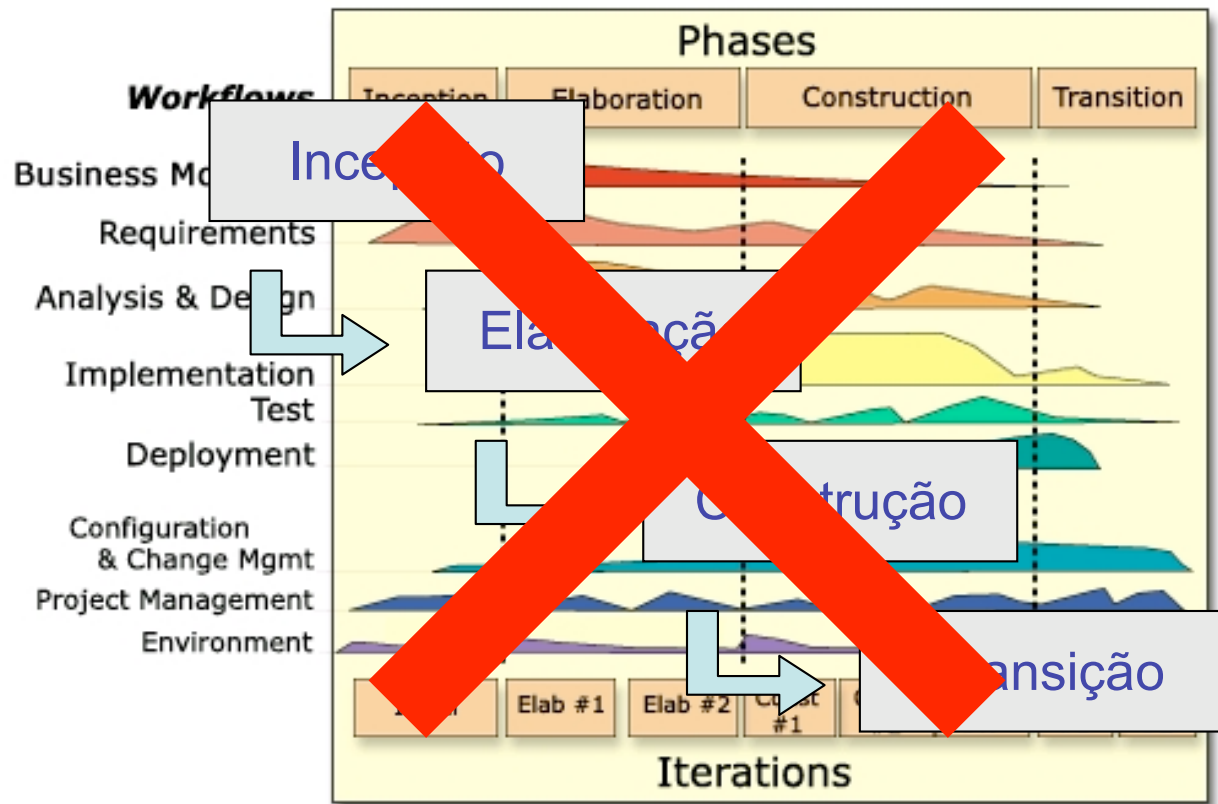
Engenharia de Software  $\neq$  Engenharia Civil



**Programar é fazer Design!**

# E o RUP?

“Ninguém segue o modelo cascata”



# 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

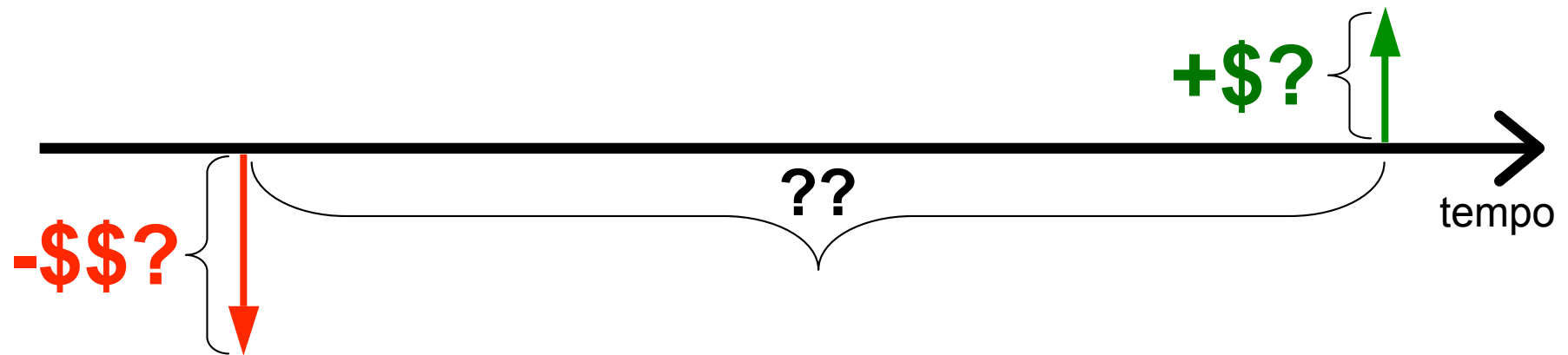
### Abstract

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.

# Como extrair valor de TI?

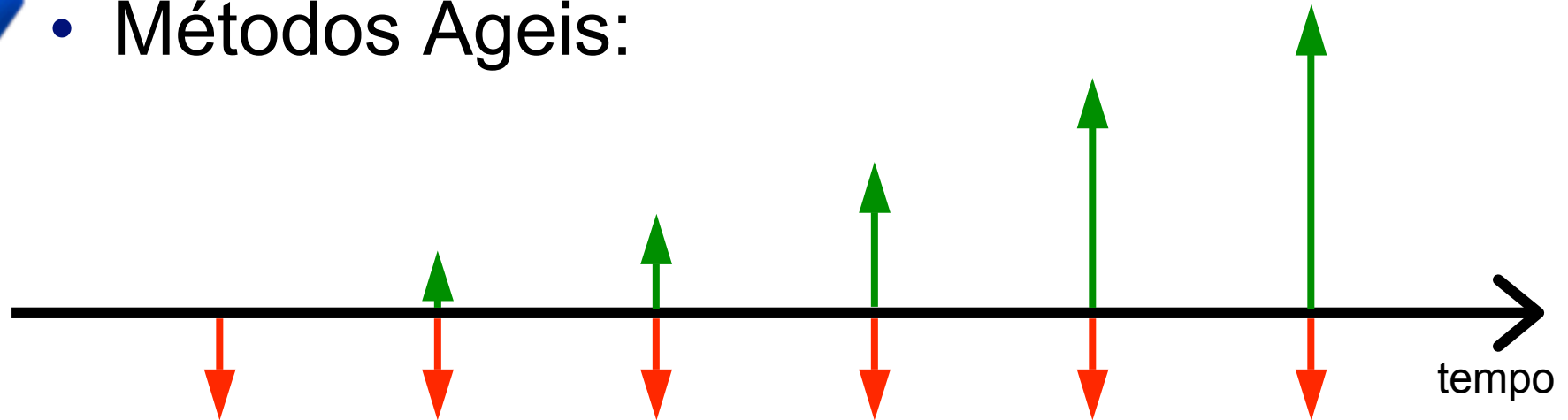
- Processos tradicionais:



Gasto:	-30\$	0\$	0\$	0\$	0\$	0\$
Lucro:	0\$	0\$	0\$	0\$	0\$	20\$
Saldo:	<b>-30\$</b>	<b>-30\$</b>	<b>-30\$</b>	<b>-30\$</b>	<b>-30\$</b>	<b>-10\$</b>

# Como extrair valor de TI?

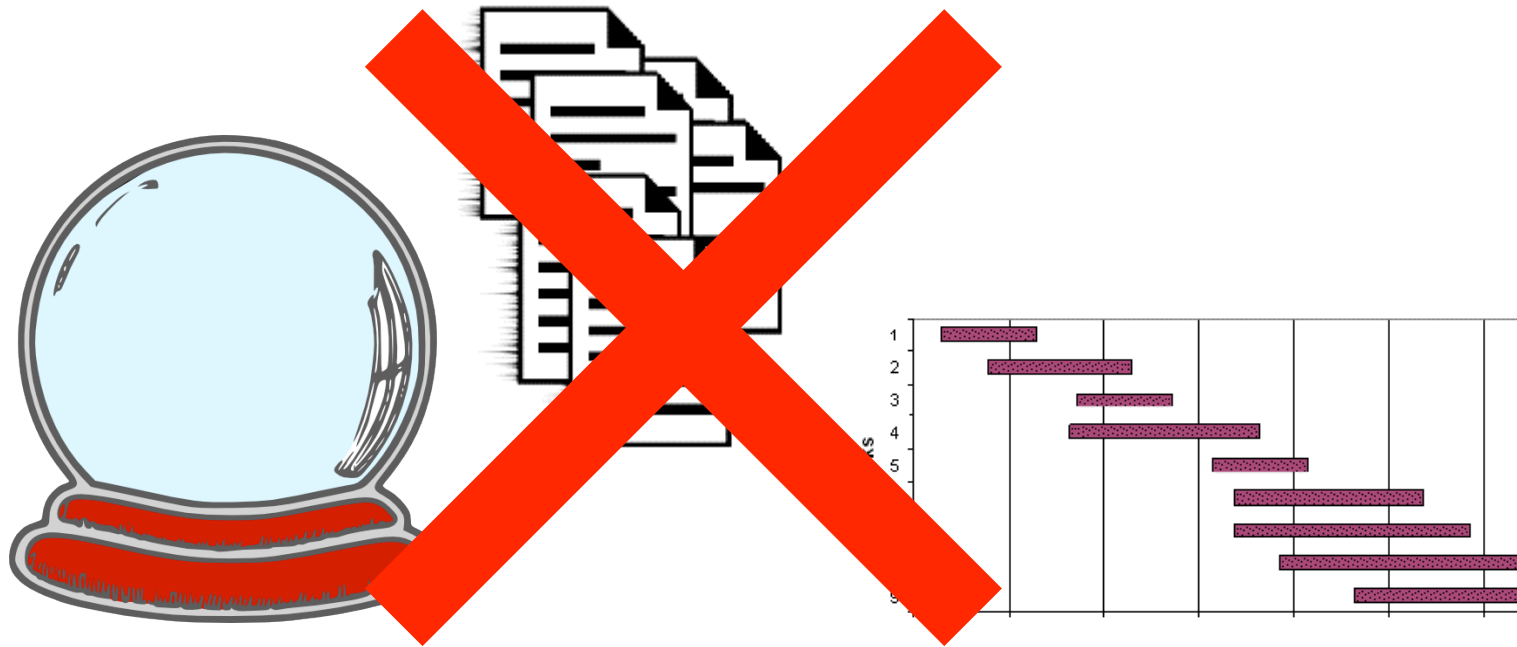
- Métodos Ágeis:



Gasto:	-5\$	-5\$	-5\$	-5\$	-5\$	-5\$
Lucro:	0\$	2\$	4\$	9\$	15\$	20\$
Saldo:	<b>-5\$</b>	<b>-8\$</b>	<b>-9\$</b>	<b>-5\$</b>	<b>5\$</b>	<b>20\$</b>

# O que é valor?

Processos tradicionais:



**Valor = software funcionando**



Ou seja...

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

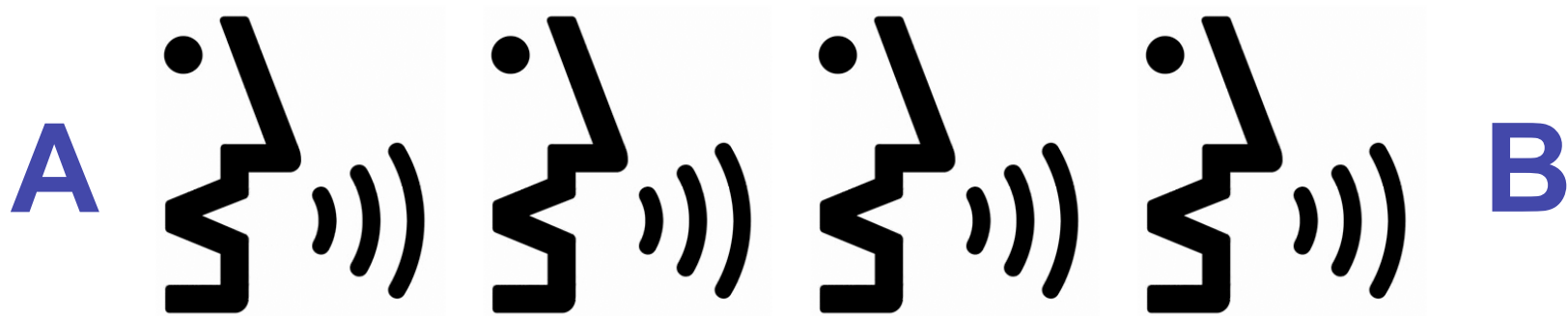
# Então não documenta?

- Documentação é uma excelente forma de armazenar o histórico de decisões de um projeto. Quando algo dá errado, é importante poder rastrear tais decisões para descobrir o que deu errado. Além disso, um documento é escrito numa linguagem muito mais formal e, portanto, mais correta, evitando ambigüidades de sentido. É importante manter toda a documentação sincronizada com o resto do projeto. Por exemplo, um documento de design ou arquitetura do sistema precisa refletir a realidade implementada. Outro tipo de documentação comumente escrita ao desenvolver software como um produto, é um manual técnico ou guia de uso para o usuário final. No entanto, é preciso lembrar que produzir documentação é uma tarefa que exige esforço e, portanto, toma tempo da equipe. A quantidade de tempo gasta produzindo documentação desnecessária é desperdício para o projeto e para o cliente. A equipe deve colaborar com o cliente para determinar o real valor da produção de um documento, levando em conta pontos como custo, benefício, precisão, manutenção e linguagem utilizada...

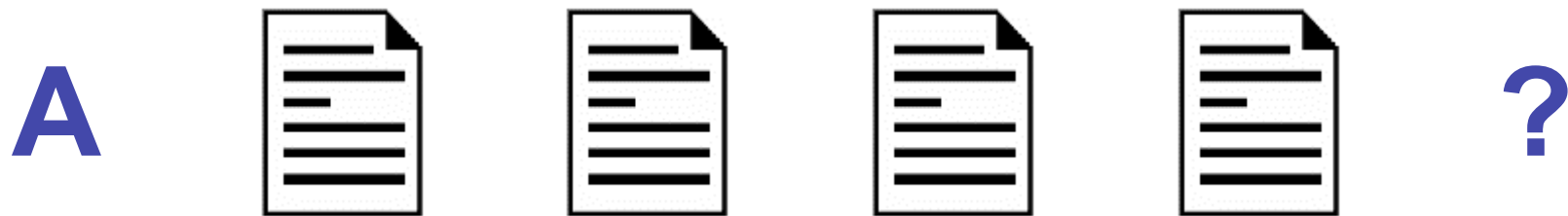
## Entenderam?

# Comunicação

- “Telefone sem fio”



- “Documento sem fio”



# Documento o necessário

- Bom senso:
  - O que a equipe julgar necessário
  - O que o “dono” do projeto exige
- Tratada como **funcionalidade**



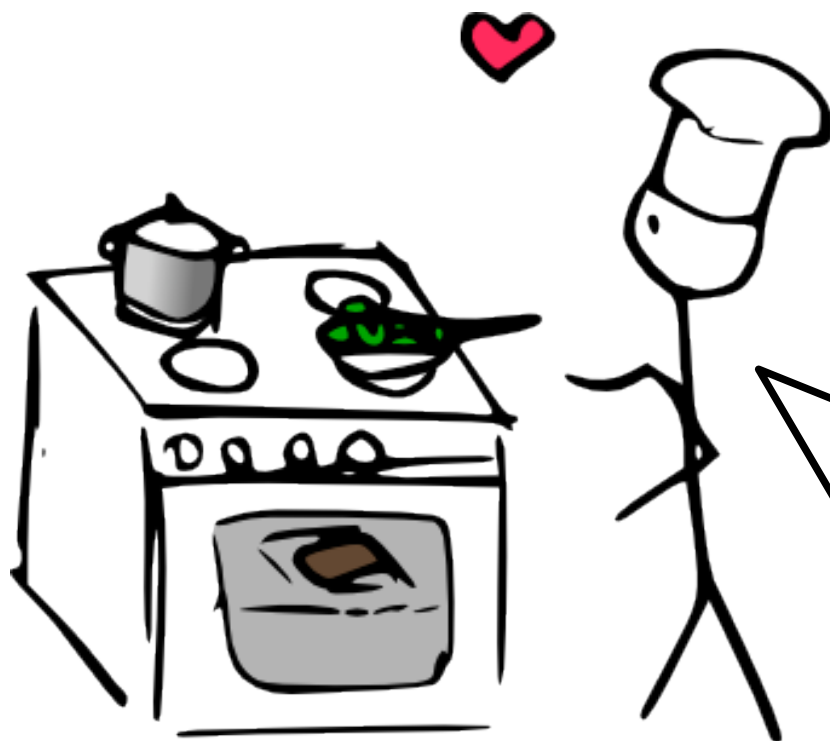
Então...

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

# Então não usa processos e ferramentas?



# 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

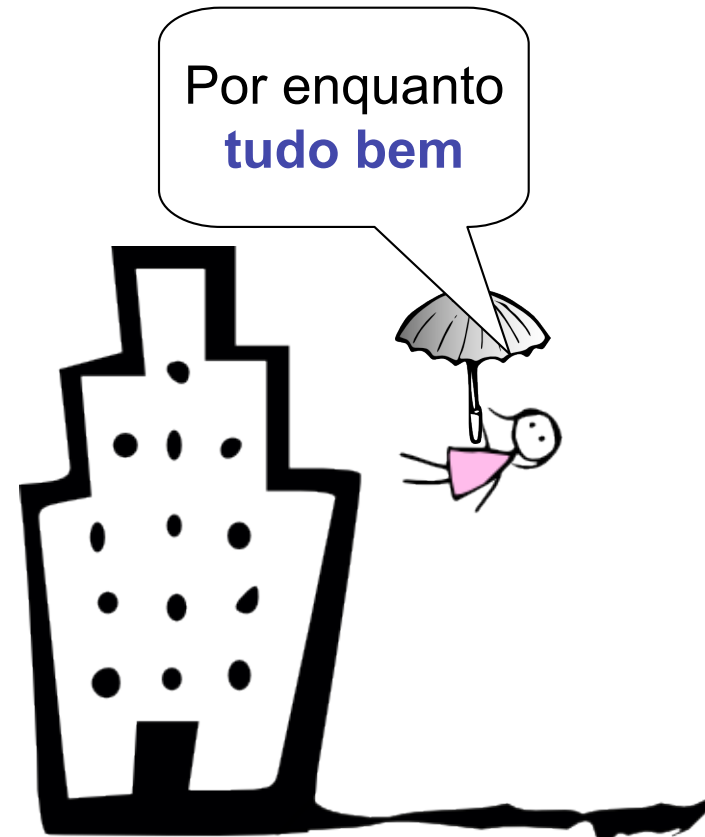
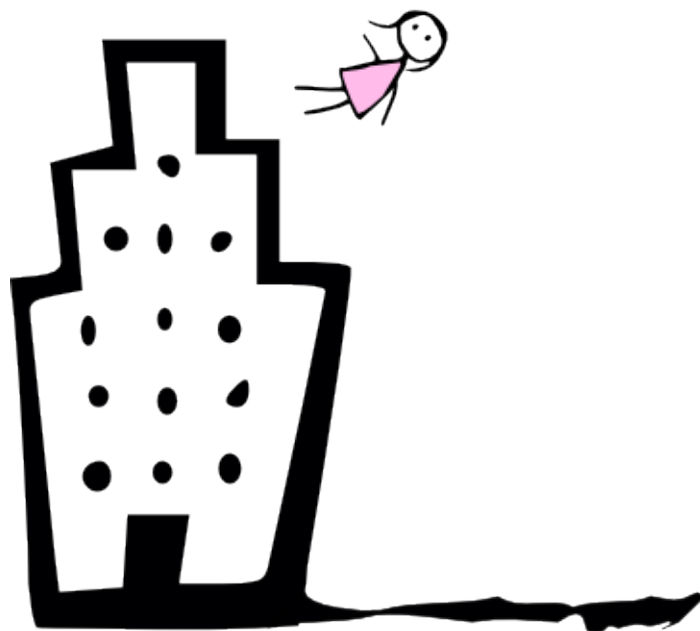


# E como fazer o software certo?



**Enxergue pelos olhos do cliente!**

# Falta de Feedback



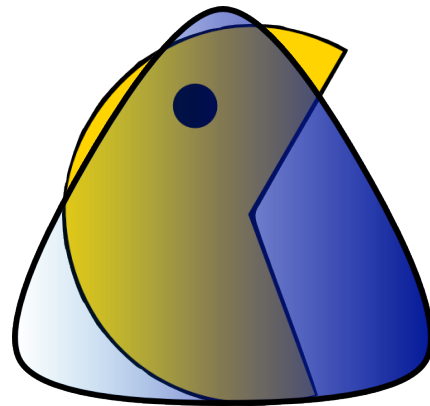


Isto é...

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

# Então não tem contrato?

- Contratos “tradicionais” incentivam comportamentos indesejados



- Faça um contrato de benefício mútuo

# Contrato de Escopo Negociável

- Cliente:
  - Tem a oportunidade de mudar de idéia
  - Pode interromper o desenvolvimento a qualquer momento
  - Custo e Prazo fixos
- Desenvolvedores:
  - Motivados a produzir o melhor sistema
  - Receita previsível

# Se não houver interação...

- Perguntas:
  - Faça sua lista de compras de 2009
  - Quem vai ganhar o campeonato de 2010?
  - Quanto tempo demora para pedalar até o RJ?
- Somos ruins para planejar a longo prazo!





Portanto...

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

# Então não planejamos?

- Mudando um pouco a história:
  - Faça a lista de compras para mês/semana que vem.
  - Quem vai ganhar o jogo do fim de semana?
  - Quanto tempo demora para pedalar de casa até o trabalho?

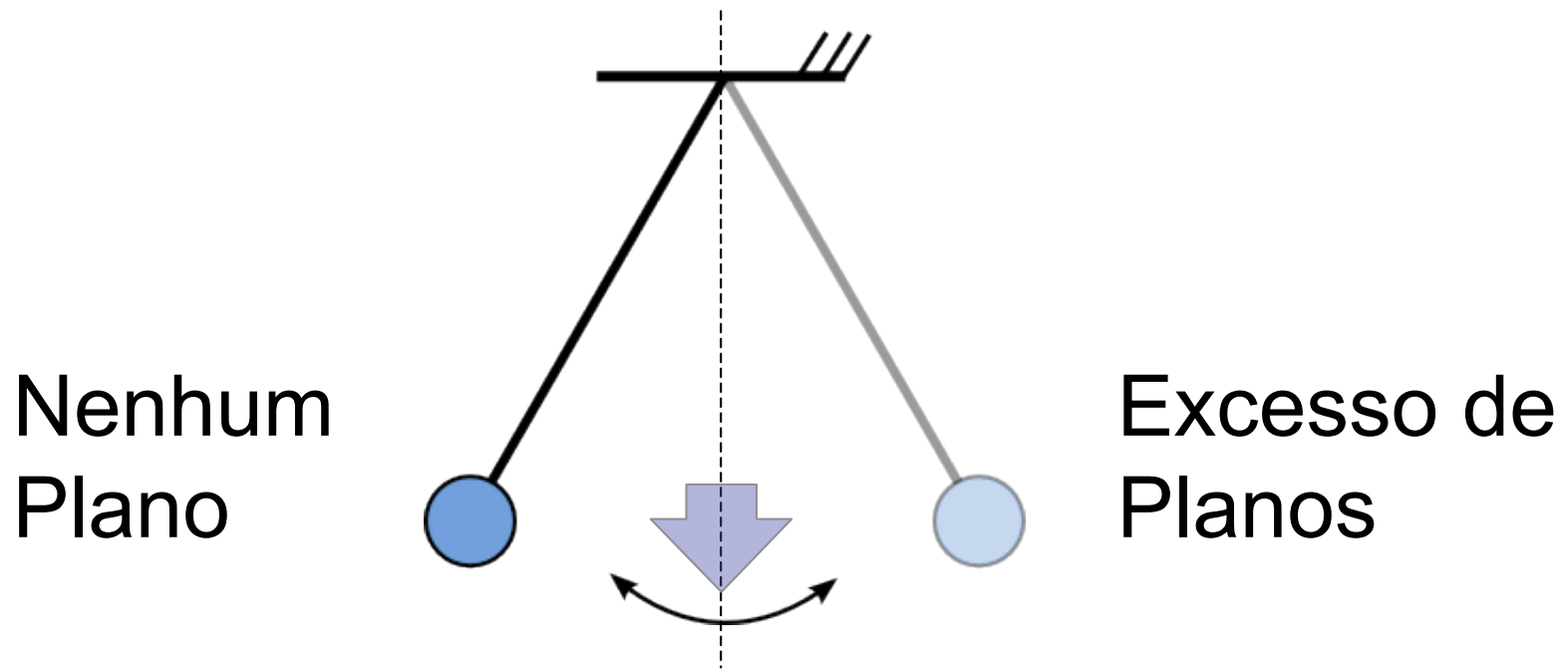


Pelo contrário...

Planejamos  
**SEMPRE**

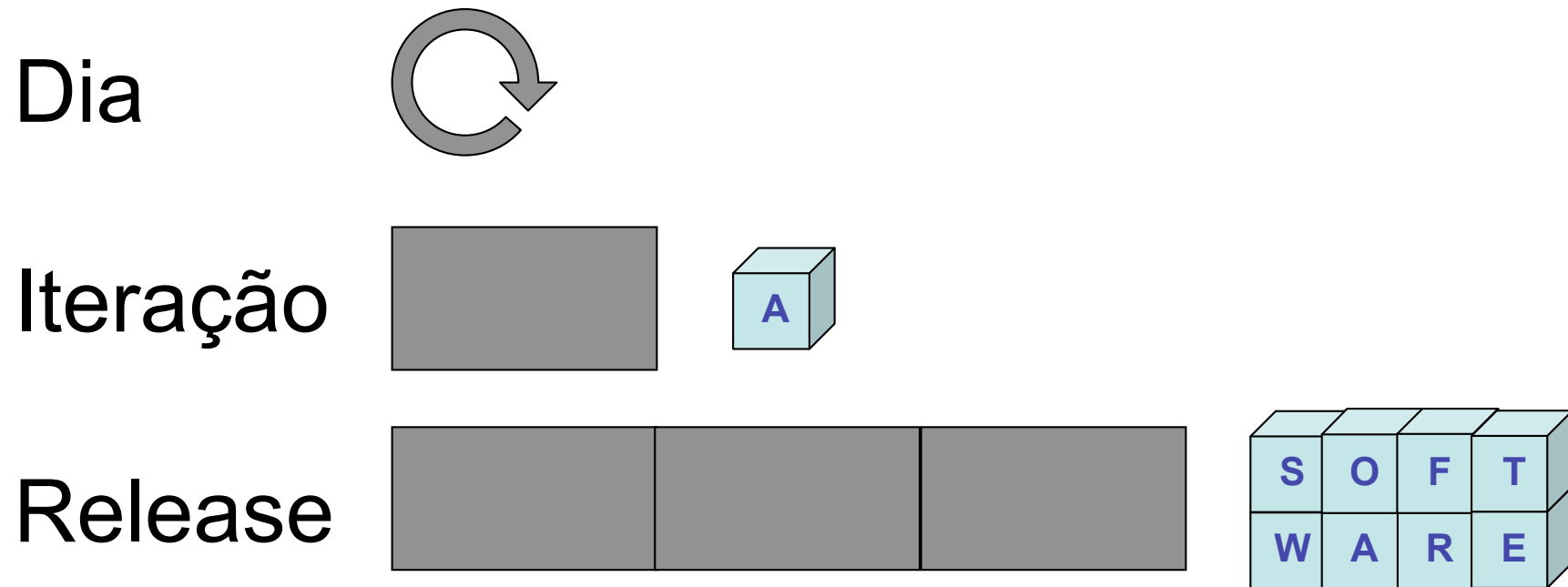
# Pêndulo do Planejamento

- No mundo não-Ágil:



**Planejar sempre, em ciclos pequenos**

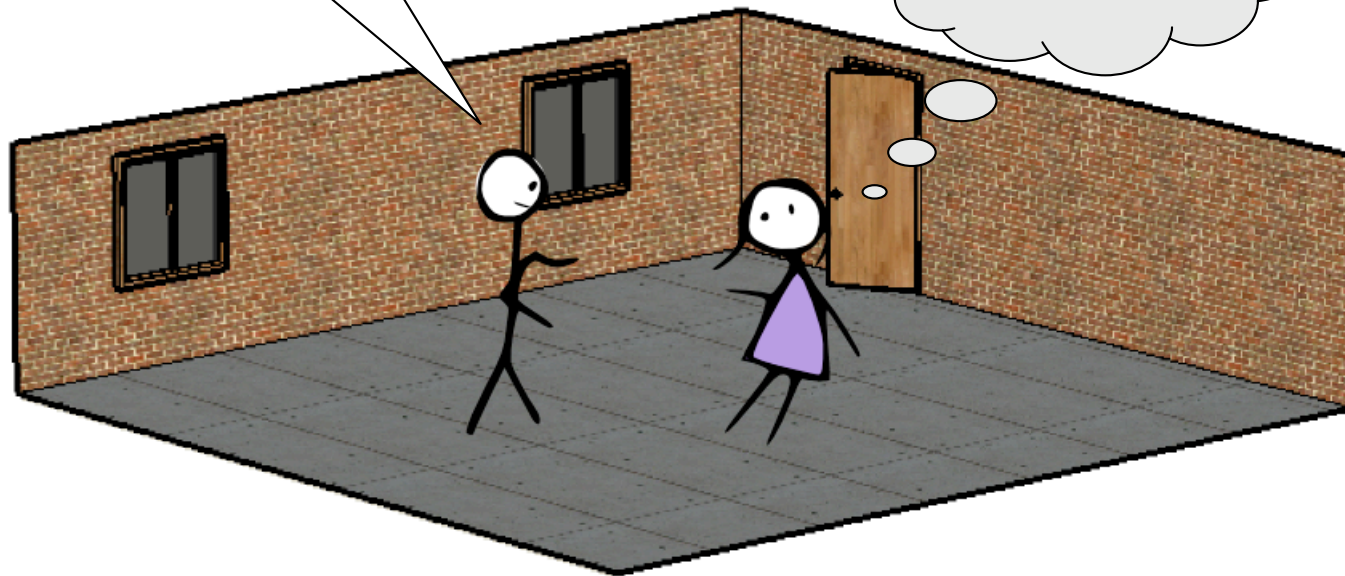
# Planejamento em Ciclos



# Novo conceito: Pronto

Seu quarto  
está **pronto!**

???



# Guiado por Feedback



# Retrospectivas

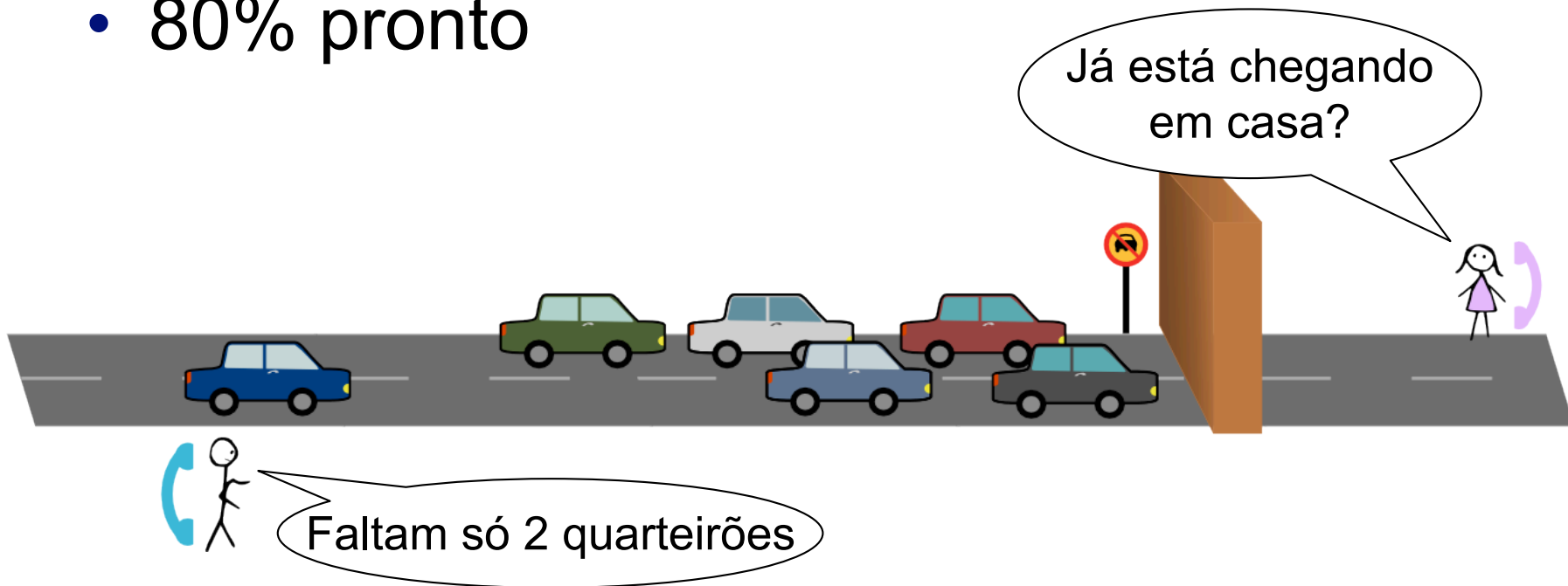
O que  
funcionou  
bem?

O que  
podemos  
melhorar?

"Encafitamentos"

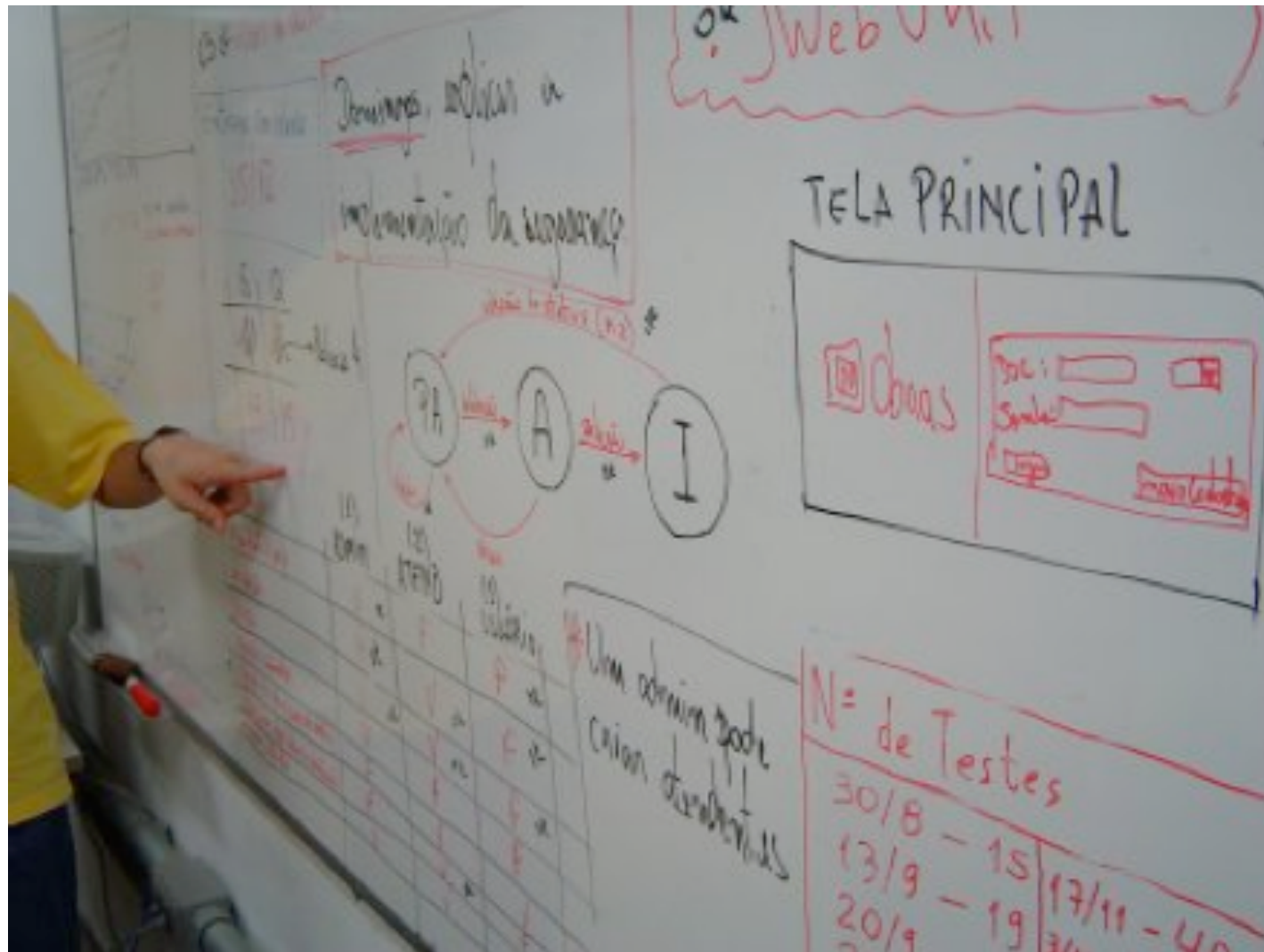
# Como o projeto atrasou 2 anos?

- 80% pronto



**Estimativas não são compromissos!**

# Área de Trabalho Informativa





# Qualidade leva à agilidade

*“Work smarter, not harder”*

-- Tom de Marco, “Slack”

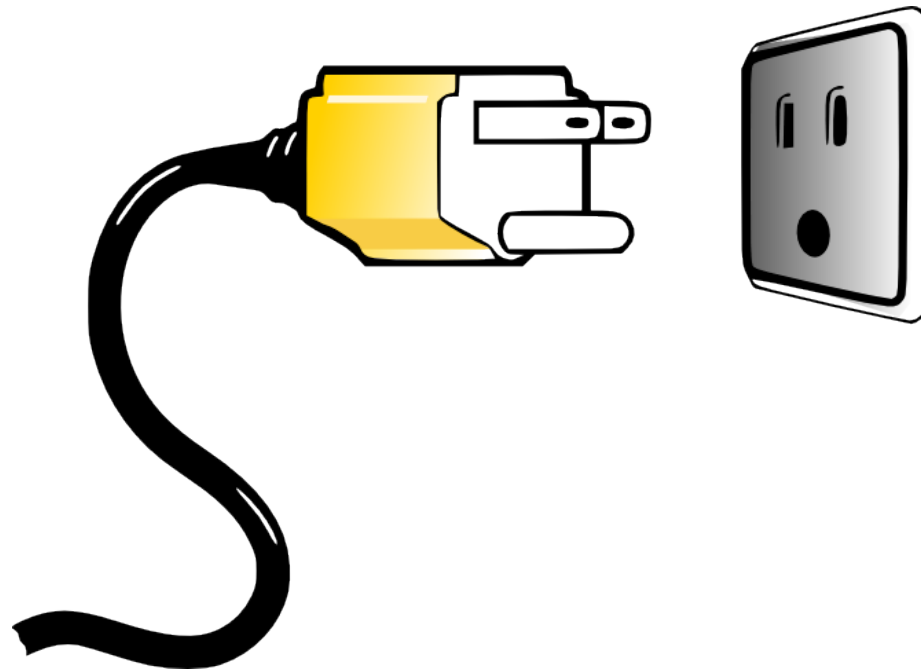
*“Inspeccionar para prevenir defeitos é bom;  
Inspeccionar para encontrar defeitos é  
desperdício”*

-- Shigeo Shingo, “The Toyota Production System”

## Qualidade é sempre alta!

# Prevenindo defeitos

- Auto-inspeção (*mistake proof*)



**Testes são a especificação!**

# Constante manutenção

- Evitando débito técnico:
  - Refatoração e Design Simples



**Complexidade é o inimigo invisível!**

# Quando não refatora....



# Onde começar?



- Comece onde está
- Use Retrospectivas
- Procure desperdícios

# Recapitulando...

- 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

# Metodologias

- XP
- Scrum
- Lean
- Crystal
- FDD
- ...



# Metodologias

A melhor metodologia é a  
**sua metodologia**

# Conclusão



**Não seja um Lemming!**

# Dúvidas?

[agilcoop@agilcoop.org.br](mailto:agilcoop@agilcoop.org.br)



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